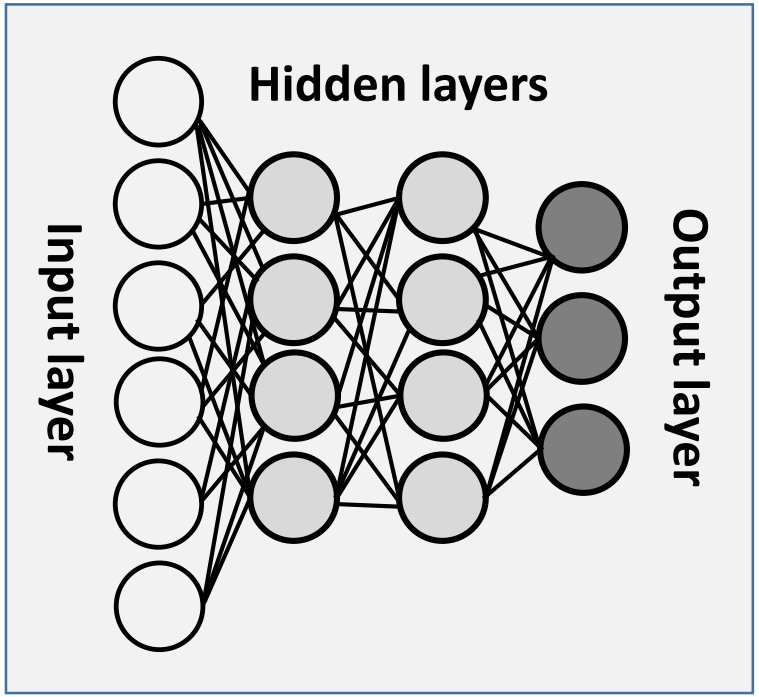
Обучение нейронных сетей

Принцип работы нейронных сетей

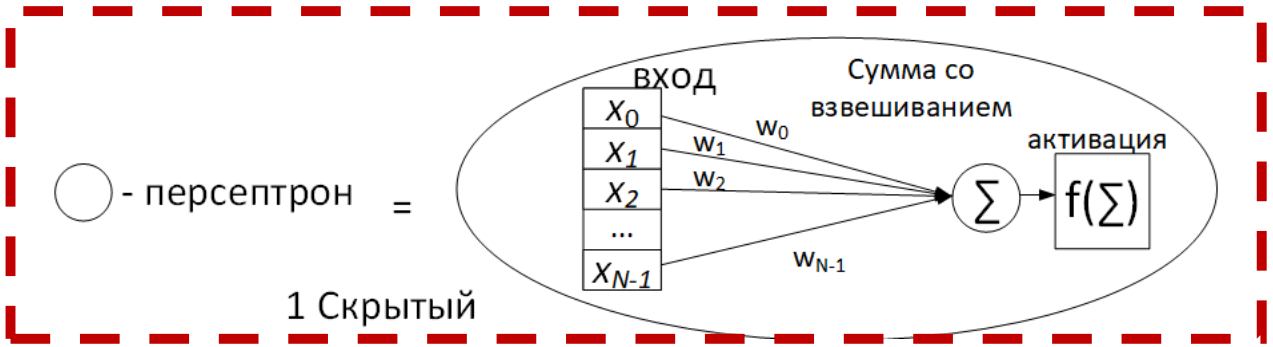
- Нейроны представляют структуру типа граф, имеющий в качестве узлов нелинейные функции – функции активации
- Функции преобразуют взвешенную сумму поступающих на них данных

$$y = f(\sum wx) = f(xw^T),$$

где x, w –вектора входных данных и весовых коэффициентов, f – нелинейная функция

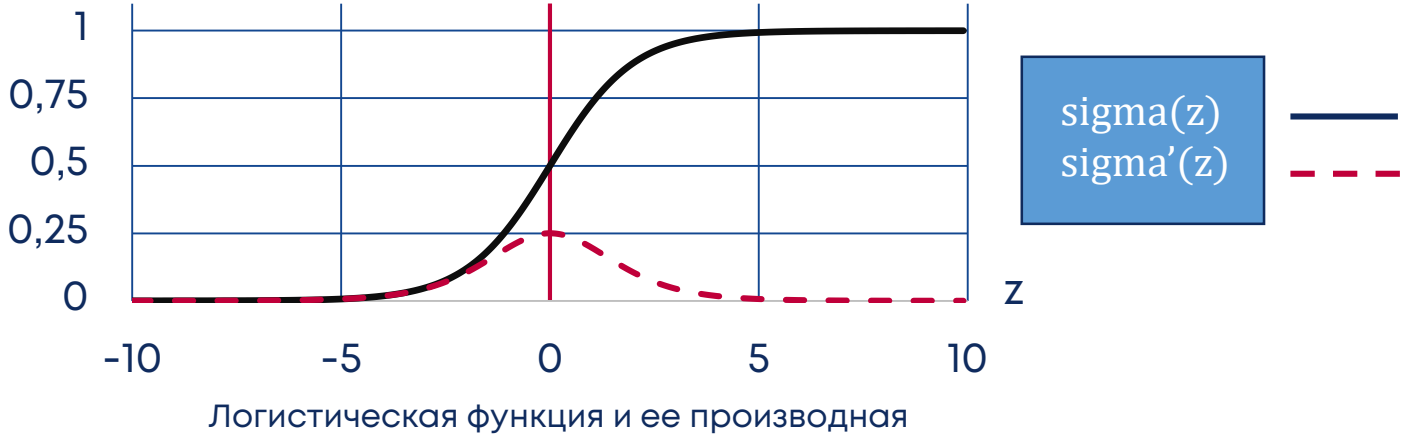


- Таким образом, каждый узел (персептрон) – это нелинейная регрессия.



Функция активации

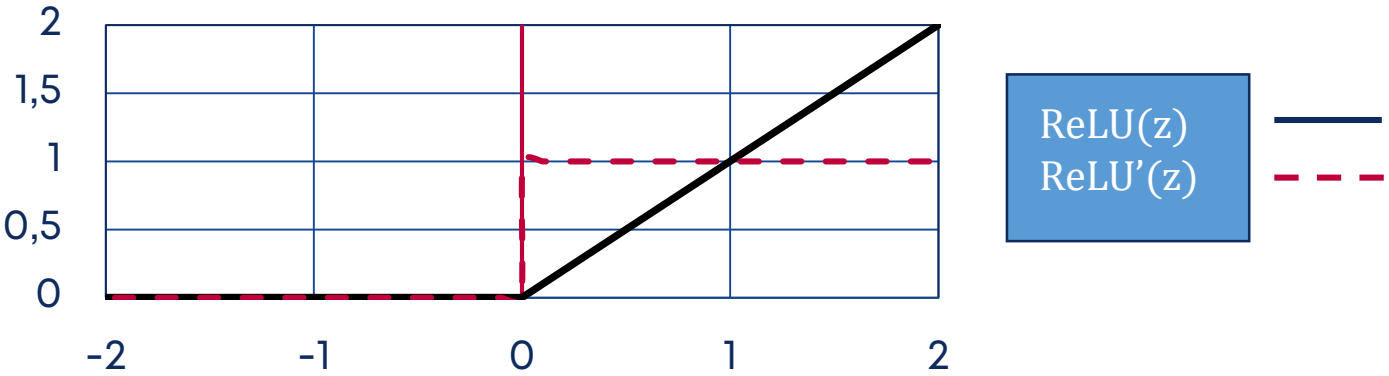
- Функция активации — это способ симуляции нелинейного поведения персептрона



Логистическая функция и ее производная

Сигмоид (логистическая функция):

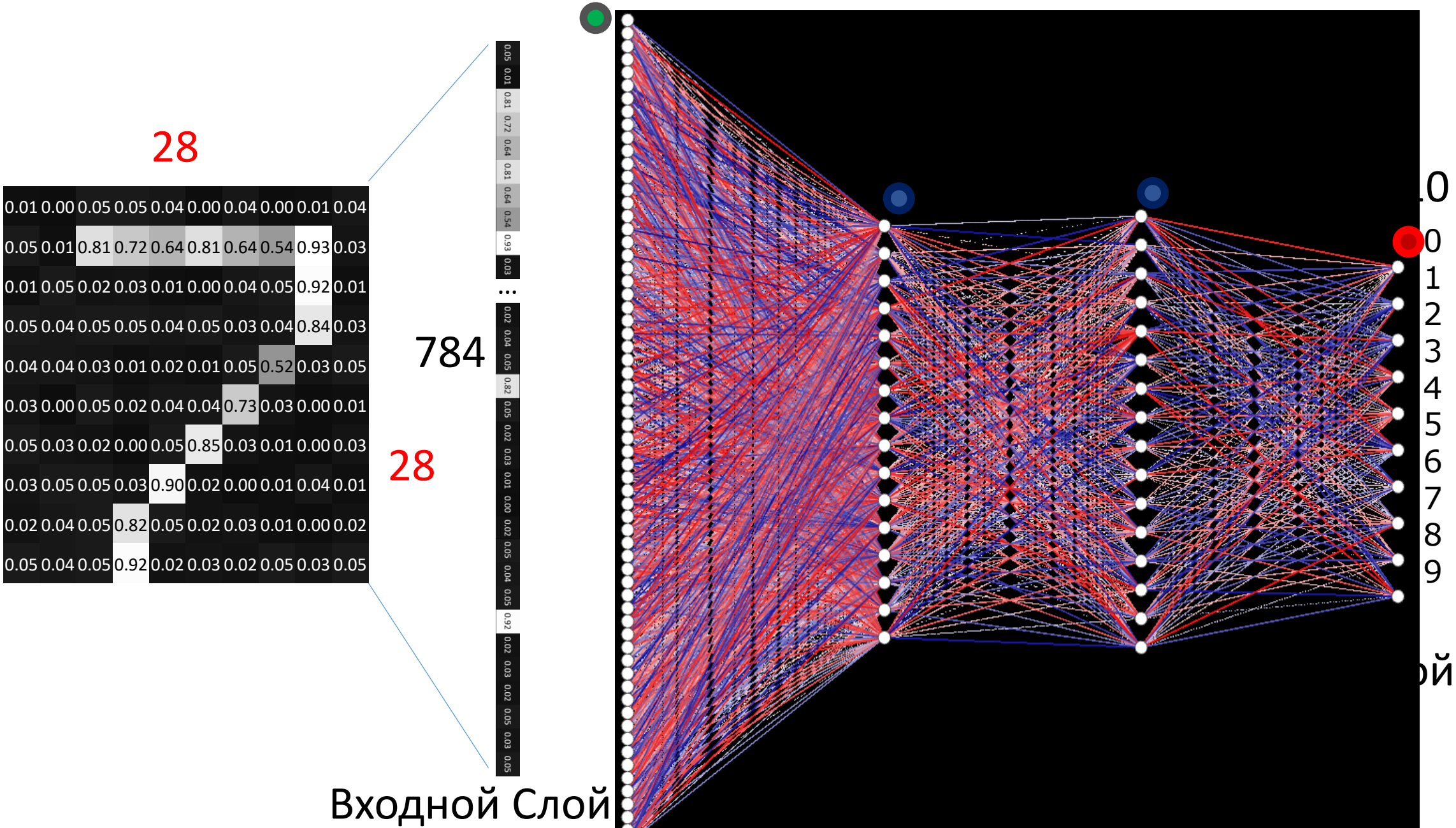
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Полулинейная функция (ReLU):

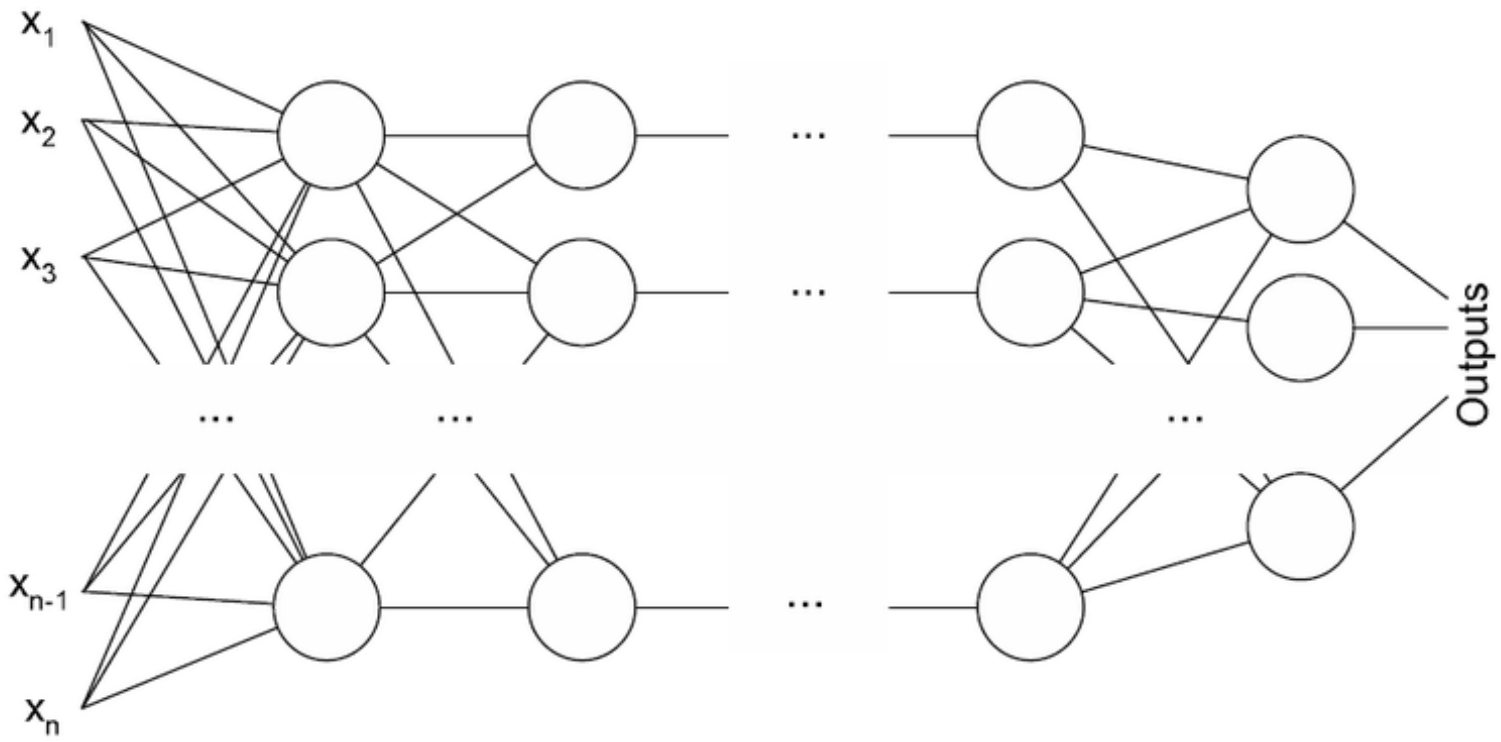
$$\mathrm{ReLU}(z) = \max(0, z)$$

Структура Нейронной Сети



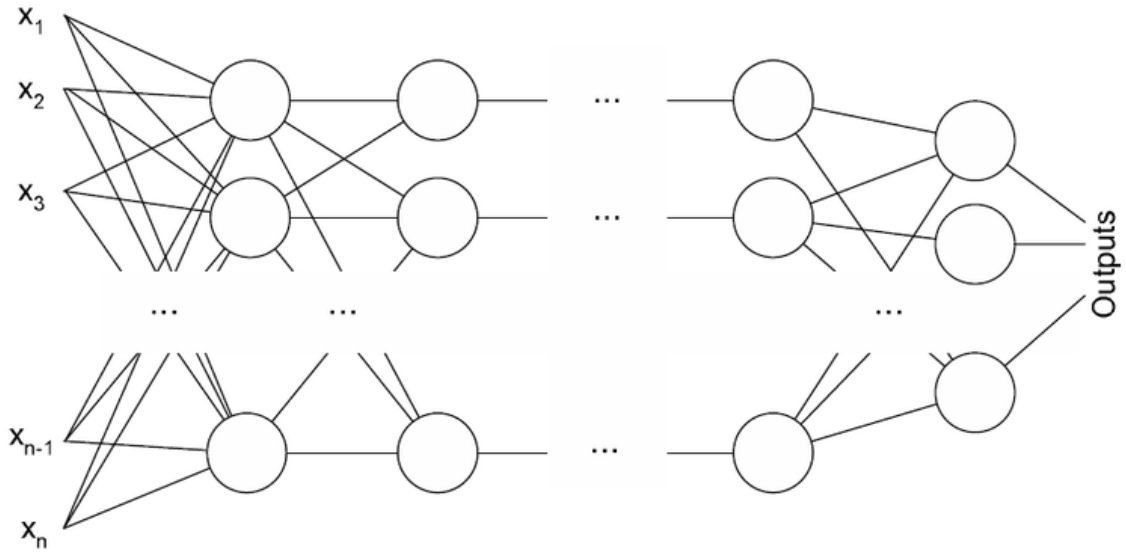
Многослойным перцептрон

- Перцептрон Румельхарта
  - Все веса обучаются алгоритмом обратного распространения ошибки,
  - увеличение слоев (больше трех) – для уменьшения числа параметров в каждом слое.
  - Использование нелинейной функции активации.
  - произвольная архитектура связей (в т.ч., и полносвязные сети).



Многослойным перцептрон по Румельхарту (MLP)

- Перцептрон Румельхарта
  - Функция ошибки - некоторая статистическая мера невязки между нужным и получаемым значением.
  - Обучение до стабилизации весовых коэффициентов при обучении или прерывается ранее, чтобы избежать переобучения.
  - преимущество улучшится способность к обобщению, то есть к правильным реакциям на стимулы которым перцептрон не обучался.

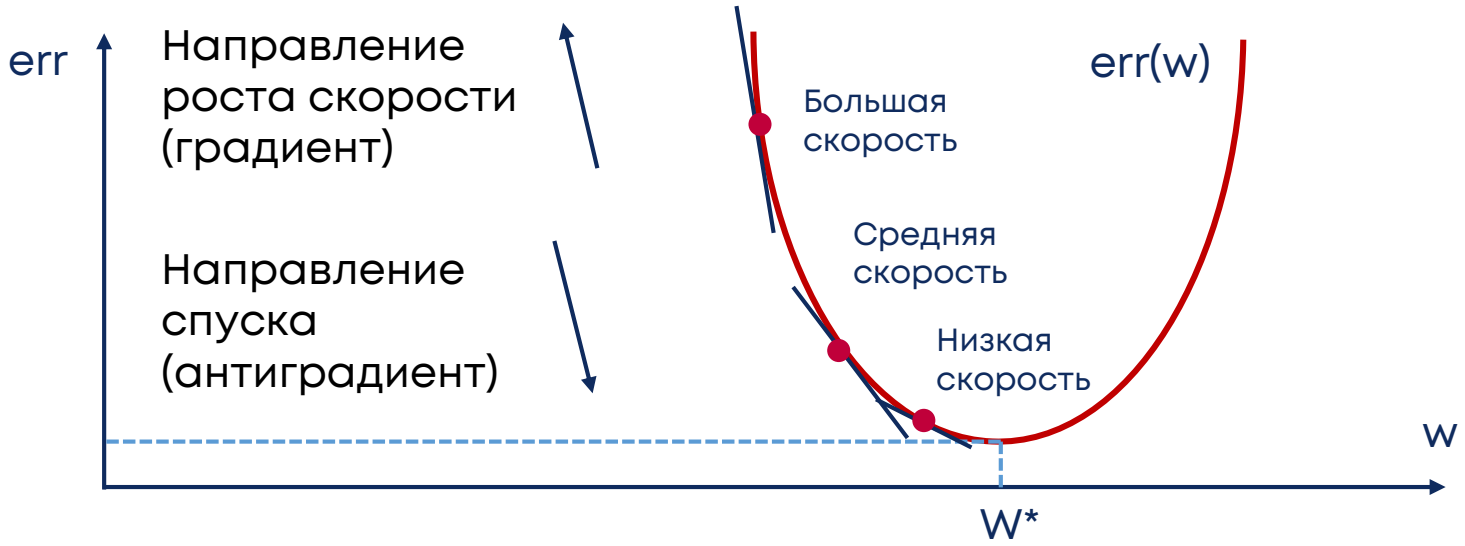


Метод обратного распространения ошибки

Практически все алгоритмы обучения современных нейронных сетей – модификации градиентного спуска.

Метод обратного распространения ошибки – это многоступенчатая (послойная) реализация метода градиентного спуска.

Правило обновления весов в нейронных сетях

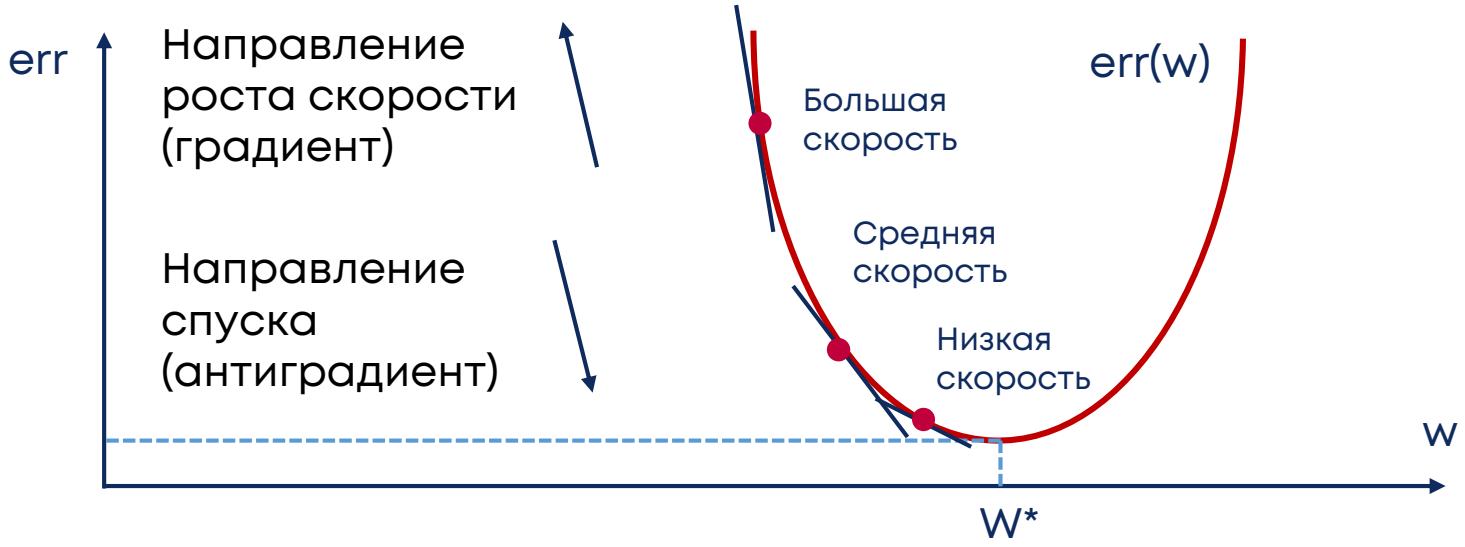


$$w_t = w_{t-1} - \eta \sum_{i=0}^{N-1} \nabla L_i(w_{t-1})$$

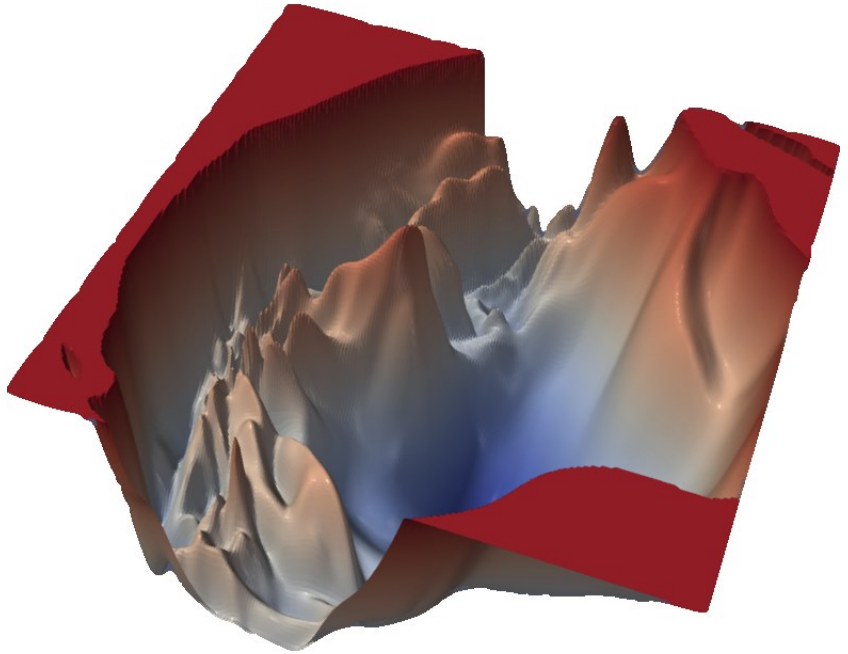
Метод обратного распространения ошибки

На практике проблема обучения заключается в том, что оптимизация проводится по большему числу параметров, ошибка ведет себя не линейно

Ожидание



Реальность



https://www.cs.umd.edu/~tomg/projects/landscapes/

Методы градиентного спуска

- Стохастический градиентный спуск:

$$W^t = W^{t-1} - \eta \nabla_W L$$

- Пакетный градиентный спуск:

$$W^t = W^{t-1} - \frac{\eta}{N_p} \sum_{i=1}^{N_p} \nabla_W L,\ N_p < N,$$

- $N_p < N$, подвыборка выбрана случайно.
- Градиентный спуск с импульсом (SGD with momentum):

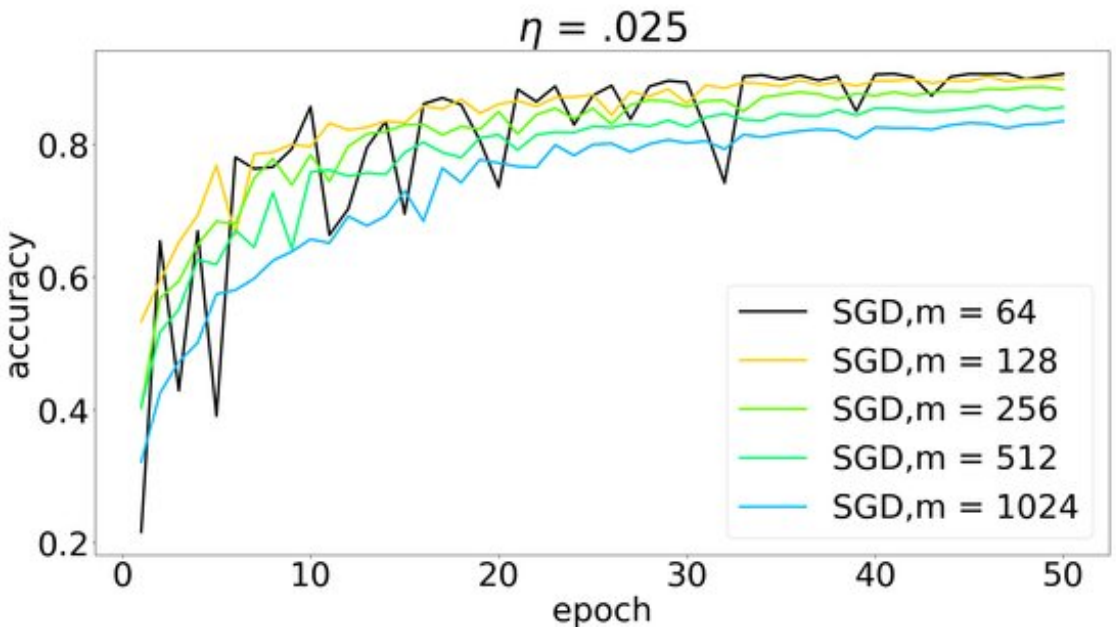
$$W^t = \beta W^{t-1} - \eta \nabla_W L + (1 - \beta) W^{t-2},$$

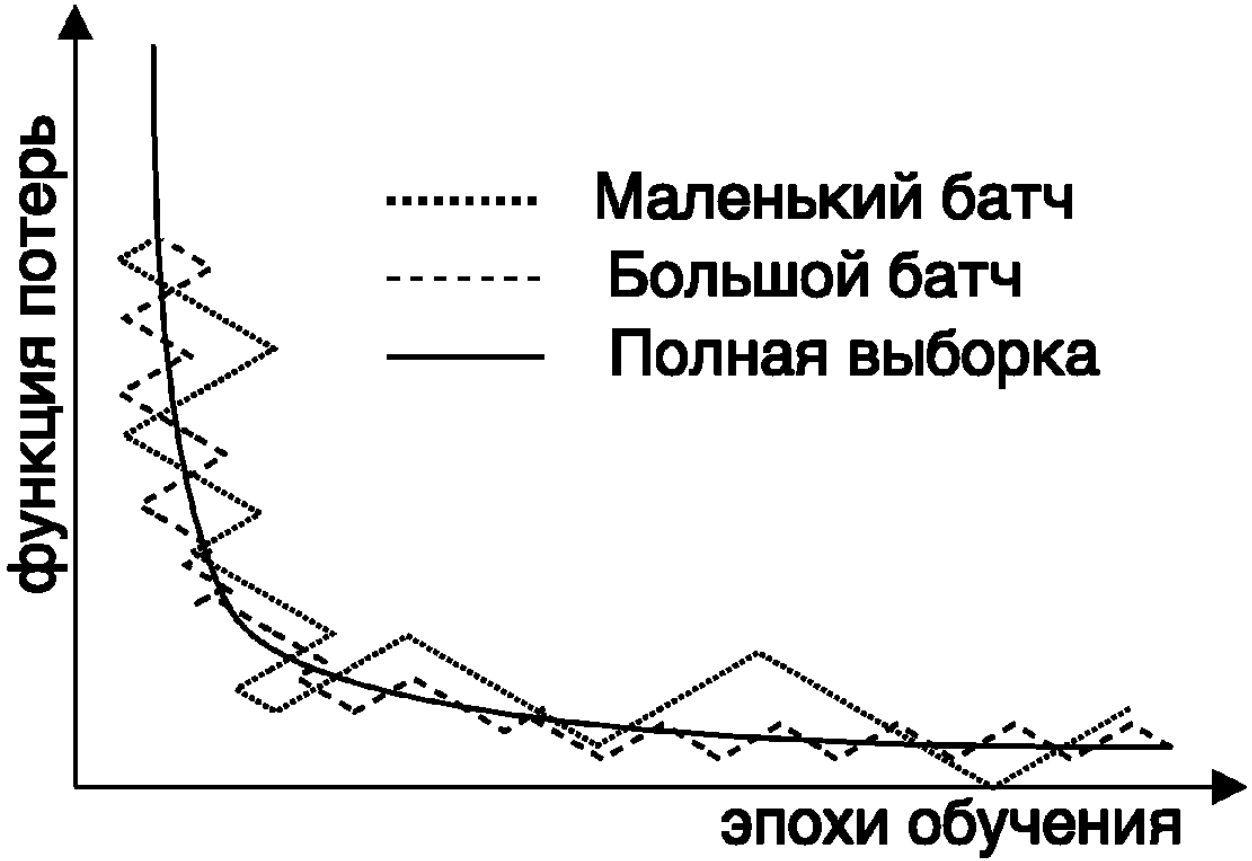
- как правило, $\beta \approx 0{,}9$.

https://artemoppermann.com/optimization-in-deep-learning-adagrad-rmsprop-adam/

Методы градиентного спуска

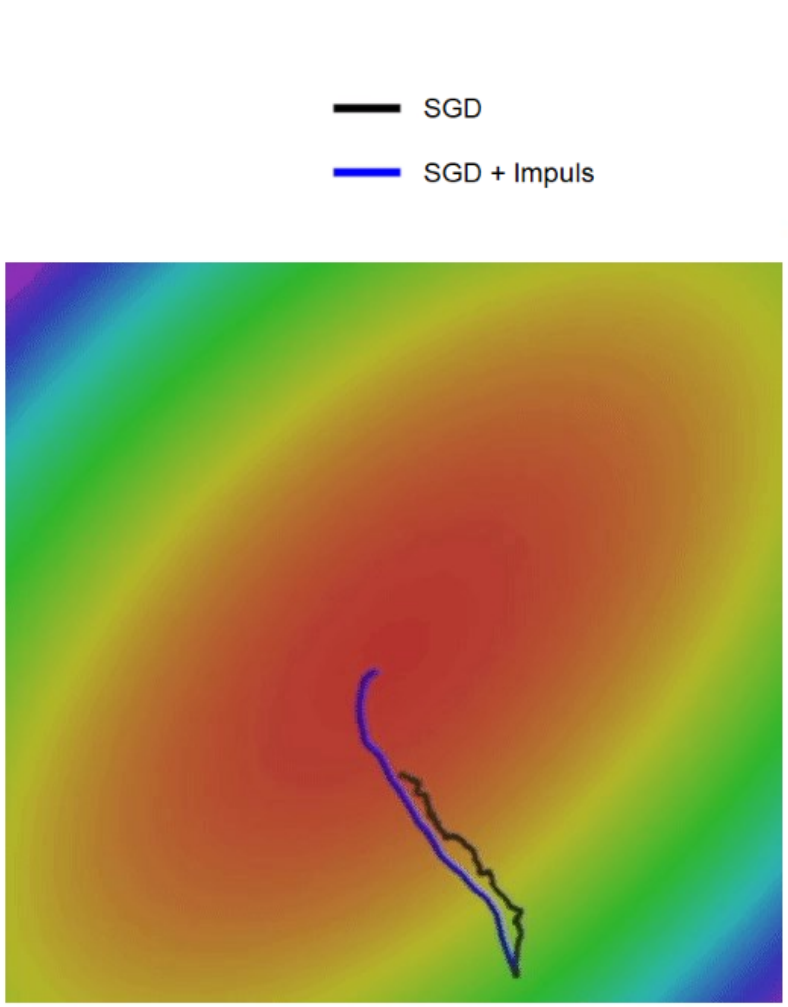
- Почему мини-пакеты?

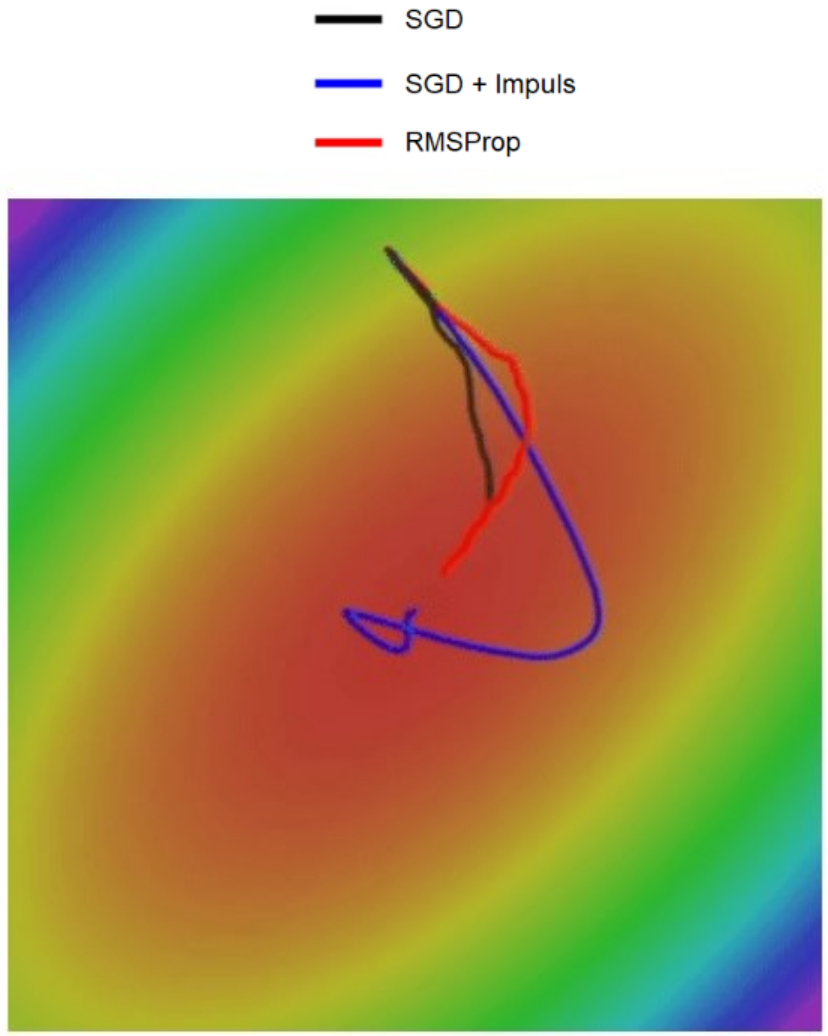




Методы градиентного спуска

- Почему надо адаптировать обновления параметров





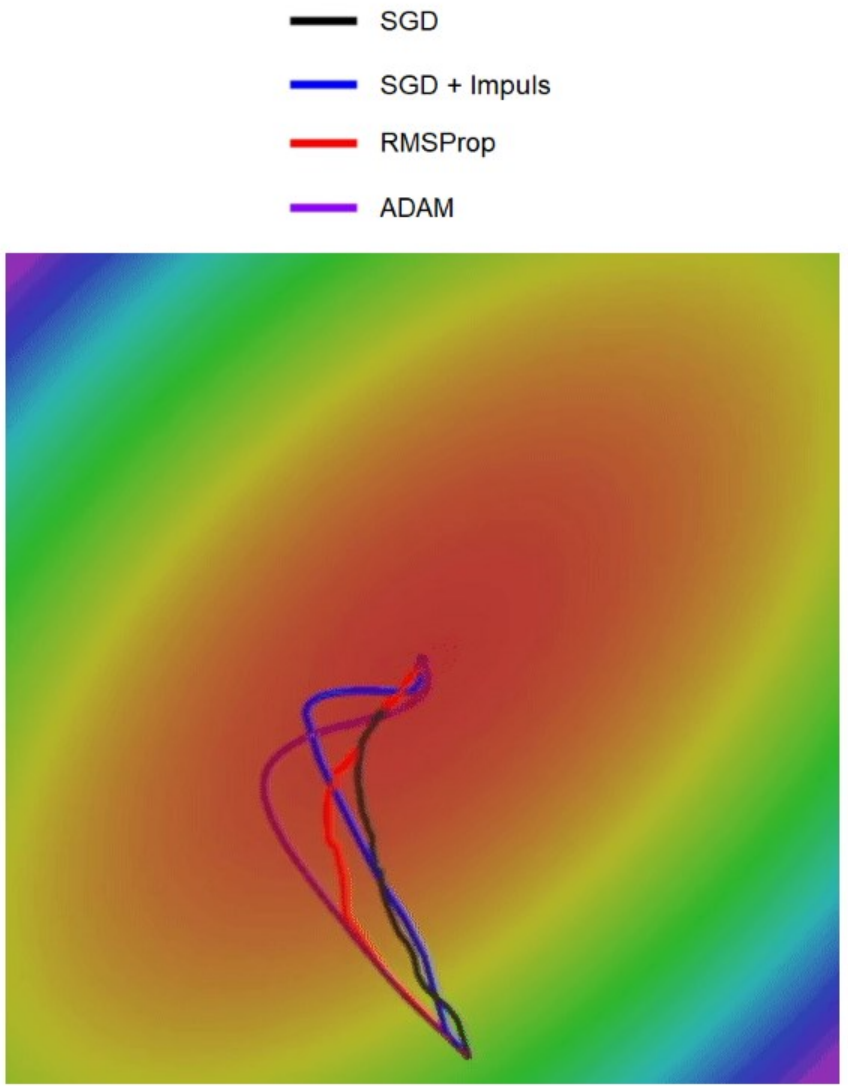


Fig. 6: Comparison of all optimization algorithms

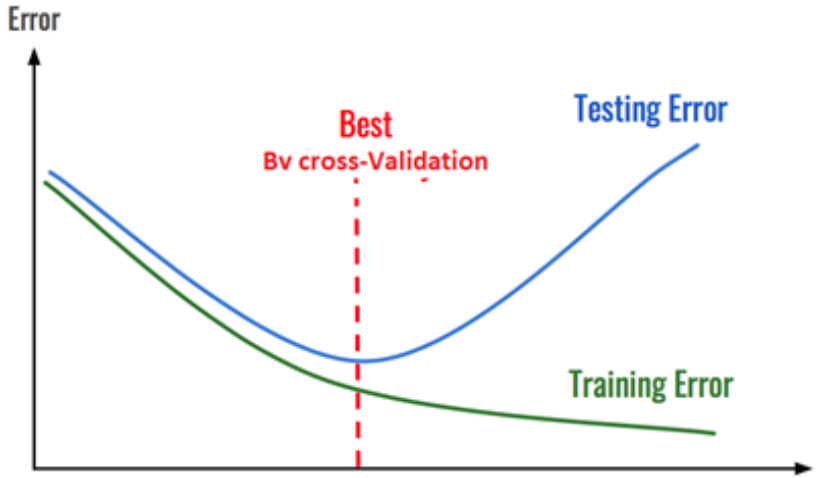
Особенности валидации

- В зависимости от валидационных данных мы можем получить разную точность, остановиться на разной эпохе обучения.
- Метод Holdout Cross Validation
  - Случайное разбиение данных на тренировочные и валидационные.
    - Как правило 30% для валидации.
    - Для небольших наборов данных можно делить 50%/50%.
    - Самый популярный метод валидации.
  - Основное достоинство:
    - простота, отсутствие дополнительных требован
  - Недостатки:
    - хорошо подходит только для больших наборов
    - Плохо подходит для несбалансированных данн
    - Плохо подходит для обоснования выбора моде
    - Результаты могут быть смещенными,
      - результаты зависят от разбиения.

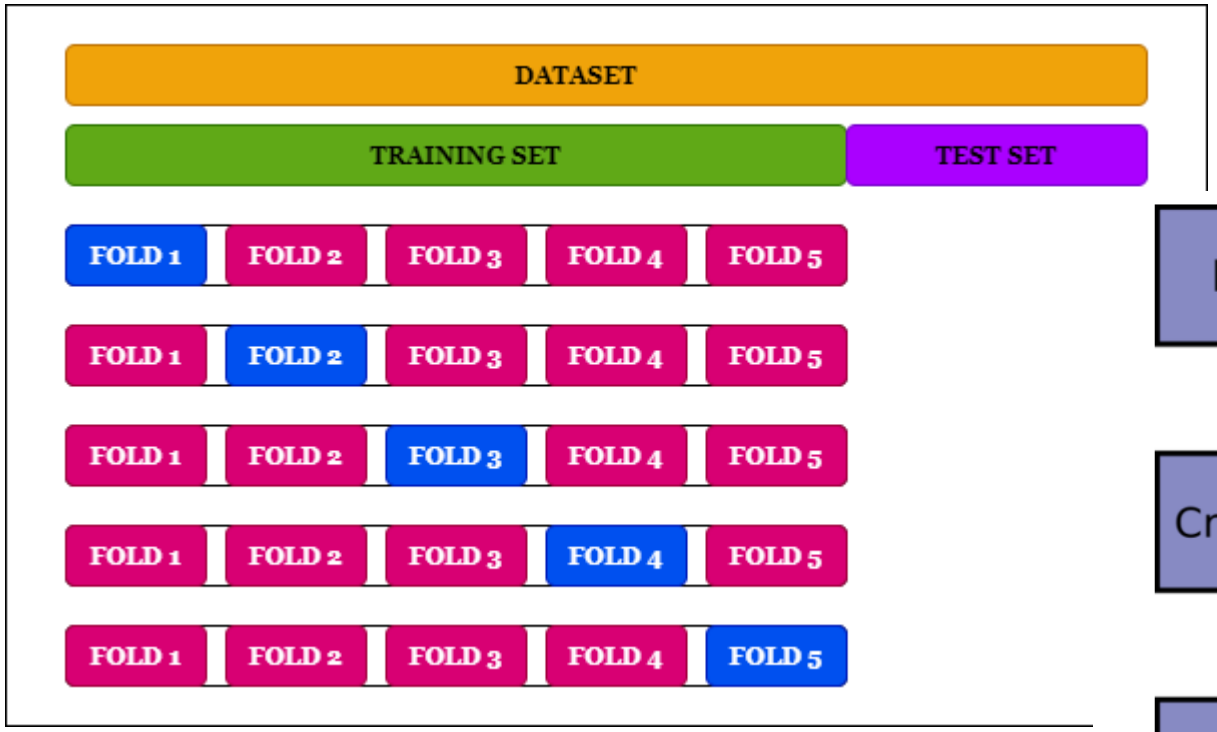
Процедура ранней остановки

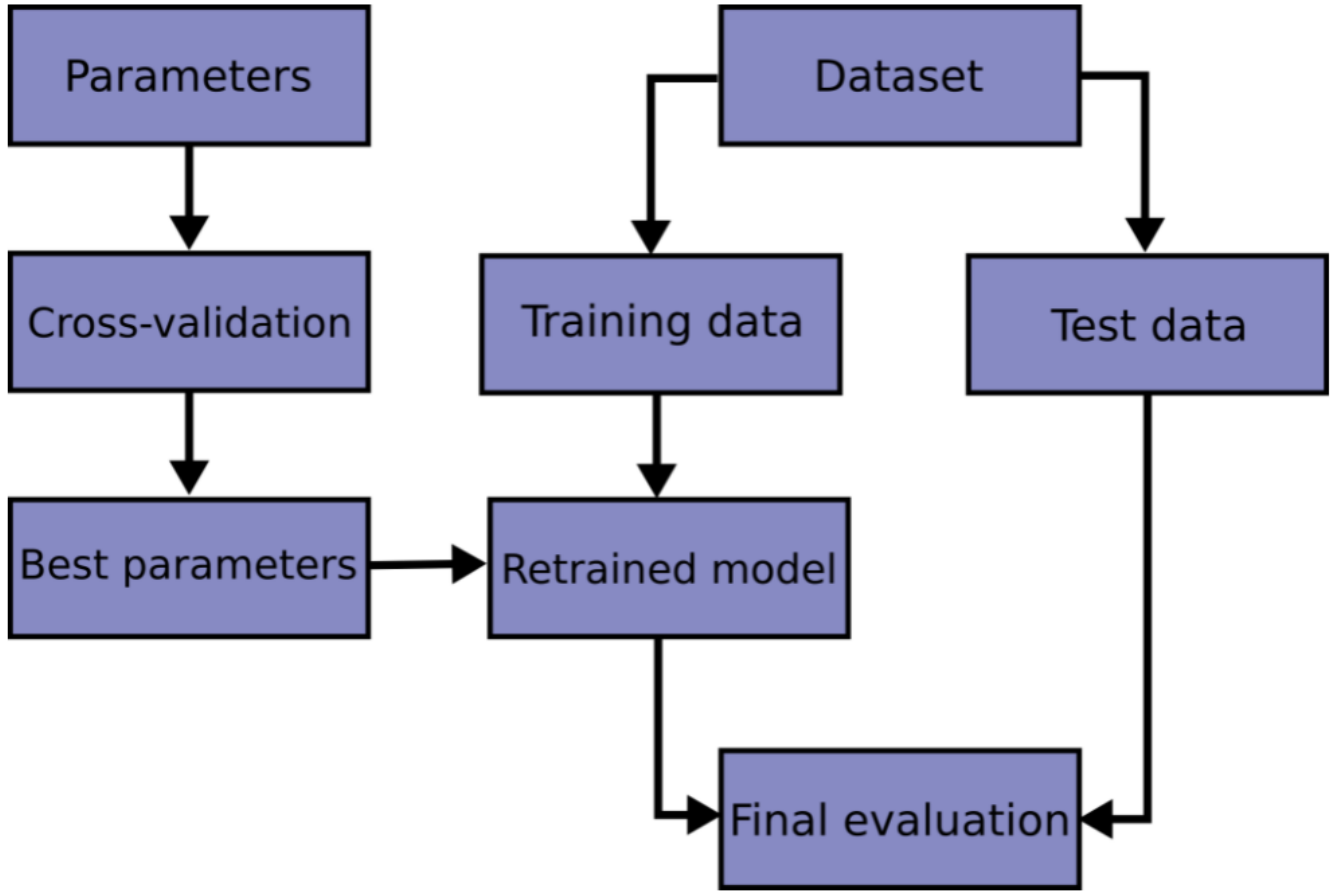
Особенности валидации K-fold

- **Данные разбиваются на k равных частей (fold).**
- **Тренировочная часть выбирается как k-1 групп, и оставшаяся одну часть как валидация.**
- Обучение проводится для каждой из k частей (обучение k раз).
- Средняя ошибка показатель качества работы модели.
- Чем выше число частей, тем менее смещенные результаты оценки.
  - Как правило k=10 (10% на валидацию за раз).
- **Основные достоинства**
  - Хорошо работает для небольших наборов данных.
  - Все данные участвуют как в тренировке, как и в тесте.
  - Более точная оценка качества работы модели, чем для **HoldOut.**
  - Метод может быть использован для обоснованного выбора моделей или гиперпараметров модели.
- **Основной недостаток**
  - большое время работы
  - и плохо подходит для несбалансированных данных.



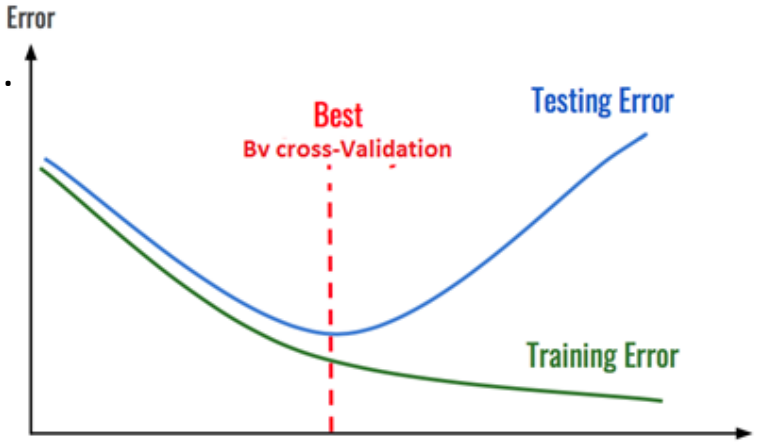
Особенности валидации





Особенности валидации модификаций K-fold

- **Случайный K-fold** (**повторяющаяся случайная выборка**)
  - В отличии от традиционного **K-fold** данные в выборках формируются случайно **K-** раз.
  - Достоинство – можно выбрать размер тренировки и валидации произвольно, при том, что число попыток не ограничено.
  - Подходит для очень маленьких наборов данных (если выбирать с возвратом значений).
- **Сбалансированный K-fold** (**стратифицированный k-fold**)
  - В каждую из к групп данные выбираются независимо по каждому классу.
  - Таким образом, можно создать сбалансированные выборки.
- **Leave-p-out cross validation, (LpOCV)** (**Leave-one-out, LOO)**
  - Обучение происходит на всей – p выборке, и p экземпляров для теста.
  - В пределе p = 1;
  - Достоинство: наименьшее смещение оценки результатов производительности модели.
  - Подходит для выбора и строгого обоснования моделей.
  - Подходит для очень маленьких выборок.
  - Недостаток – требует много машин-часов на обучение.



Адаптивный градиентный спуск

$$\begin{cases} EG^t = \beta[\nabla_W L]^2 + (1-\beta)EG^{t-1} \\ W^t = W^{t-1} - \eta \frac{1}{\sqrt{EG^t + \epsilon}} \nabla_W L \end{cases}$$

RMSProp, $\beta \approx 0.9;$ $\eta = 10^{-3}$

$$\begin{cases} EG_1^t = \beta_1[\nabla_W L] + (1-\beta_1)EG_1^{t-1} \\ EG_2^t = \beta_2[\nabla_W L]^2 + (1-\beta_2)EG_2^{t-1} \\ W^t = W^{t-1} - \eta \frac{EG_1^t}{\sqrt{EG_2^t + \epsilon}} \nabla_W L \end{cases}$$

ADAM

$\beta_1 = 0.9,$
$\beta_2 = 0.99,$
$\eta = 3 \cdot 10^{-4}.$

Соадоптация слоев нейронной сети

- Одна из основных проблем нейронных сетей – соадоптация весовых параметров нейронной сети.
- Соадоптация - это ситуация, когда каждый слой нейронной сети работает как коррекция результатов работы предыдущего слоя.
- В случае переобучения – каждый слой становится переадоптированным так чтобы корректировать работу предыдущего слоя –то есть подстраивается, а должен быть независимым – то есть должен выделять из него полезные признаки.
- Другими словами при соадоптации слой ищет возможность скорректировать шумы и нерегулярные особенности данных, которые поступают с предыдущего слоя.

Соадоптация слоев нейронной сети

- Соадоптация приводит к снижению обобщающей способности нейронной сети.
- Методы снижения эффекта:
  - Использование проброса данных через слой.
  - Ансамбль нейронных сетей.
  - Добавление шумов к слою или добавление перемешанных весов к значениям весов.
  - Исключение случайно выбранной части весовых параметров или данных на каждом проходе – чтобы на них нельзя было обучиться.

Метод Дропаут (Dropout)

- Метод Дропаут (Dropout) – это метод случайного исключения части весов или входных значений из сети при каждом проходе.
  - Выборка исключаемых значения выбирается случайно для каждого прохода.

Метод Дропаут (Dropout)

- Дропаут это техника регуляризации, защищающая от переобучения и соадпотации слоев нейронной сети.
- Стандартный дропаут полносвязной сети:

$$y = f(W^T X) \odot m, \text{ где } m \sim \text{Bernoulli}(p) = \begin{cases} 1 \text{ с вероятностью } p \\ 0 \text{ с вероятностью } 1-p \end{cases} \quad \text{На этапе обучения}$$

$$y = f(W^T X) \odot (1-p), \quad \text{На этапе тестирования}$$

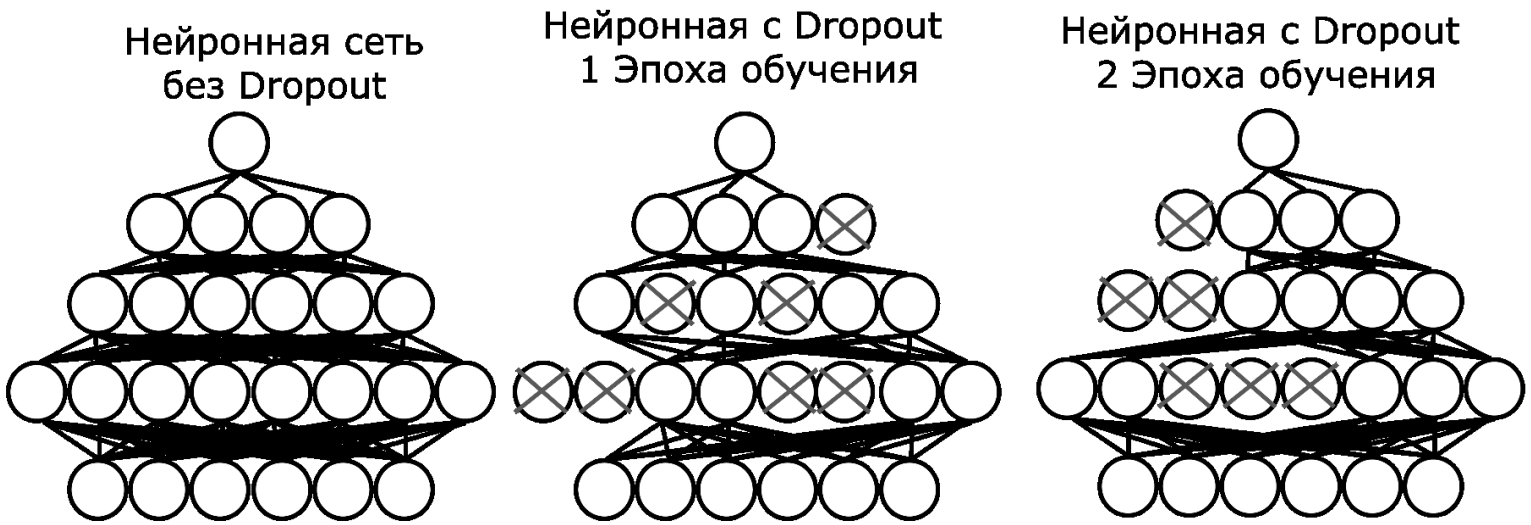
- **Дропаут это техника регуляризации, защищающая от переобучения и соадпотации слоев нейронной сети.**
- **Стандартный дропаут полносвязной сети:**

$$y = f(W^T X) \odot m, \text{ где } m \sim \text{Bernoulli}(p) = \begin{cases} 1 \text{ с вероятностью } p \\ 0 \text{ с вероятностью } 1-p \end{cases} \quad \text{На этапе обучения}$$

$$y = f(W^T X) \odot (1-p), \quad \text{На этапе тестирования}$$

Метод Дропаут (Dropout)

- Дропапут работает для выхода функции активации.
- На этапе тестирования компенсируется недостаток значения функции активации.
- Метод дропаута работает как ансамбль нейронных сетей, только во времени а не параллельно.

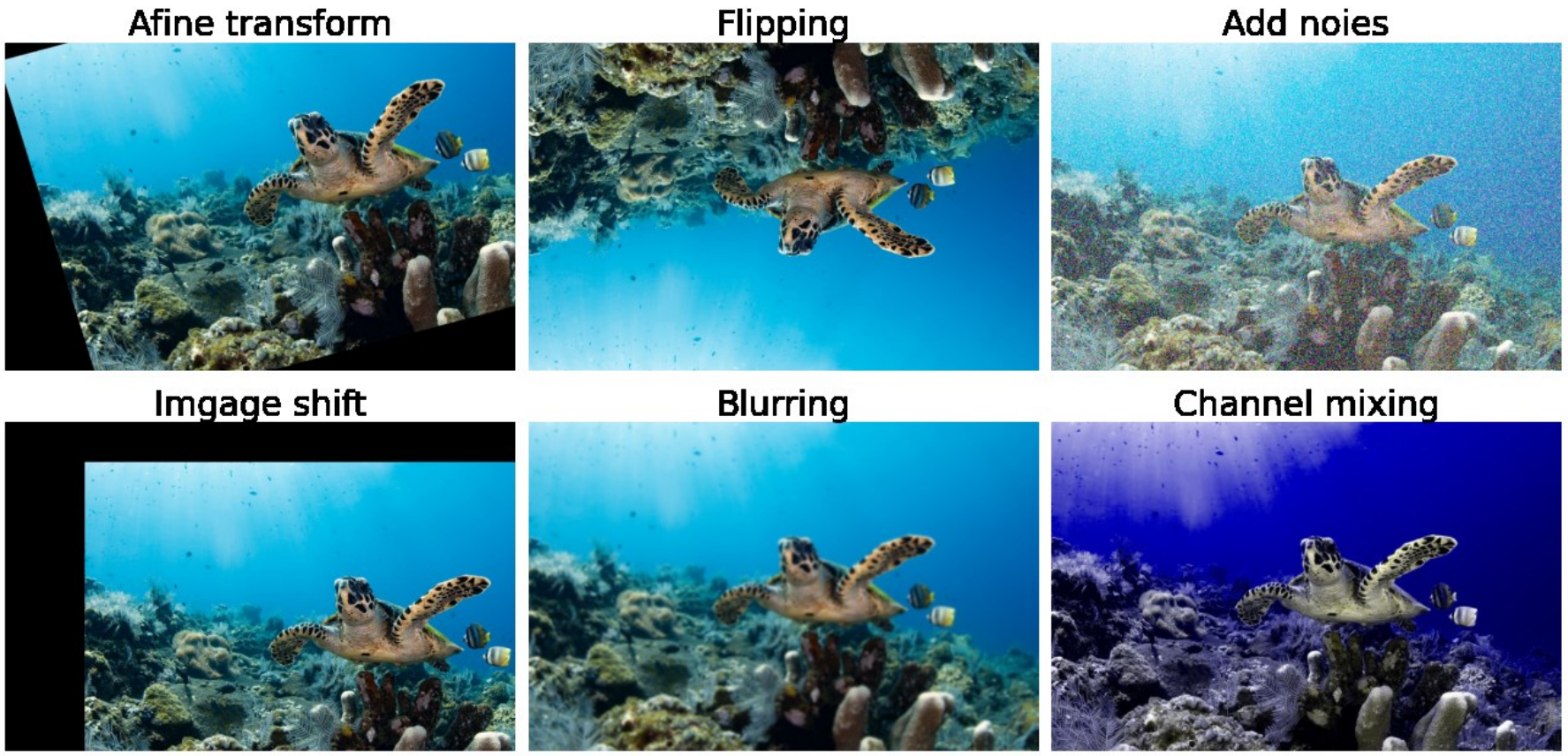


Особенности метода Дропаут

- Часто выбирается вероятность p=0.2 – 0.5.
- Можно выбрать дропаут на входном слое – но чем больше данных, тем выше вероятность. Для входного слоя дропаут – это аналог аугментации.
- Дропаут увеличивает дисперсию результатов.
- Дропаут не используется на тестирование сети.
- Недостаток метода – необходимость увеличения числа параметров, снижение скоросит обучения, более высокие требования к выбору скорости обучения и других гиперпараметров.
- Дропаут плохо работает с другими методами регуляризации.

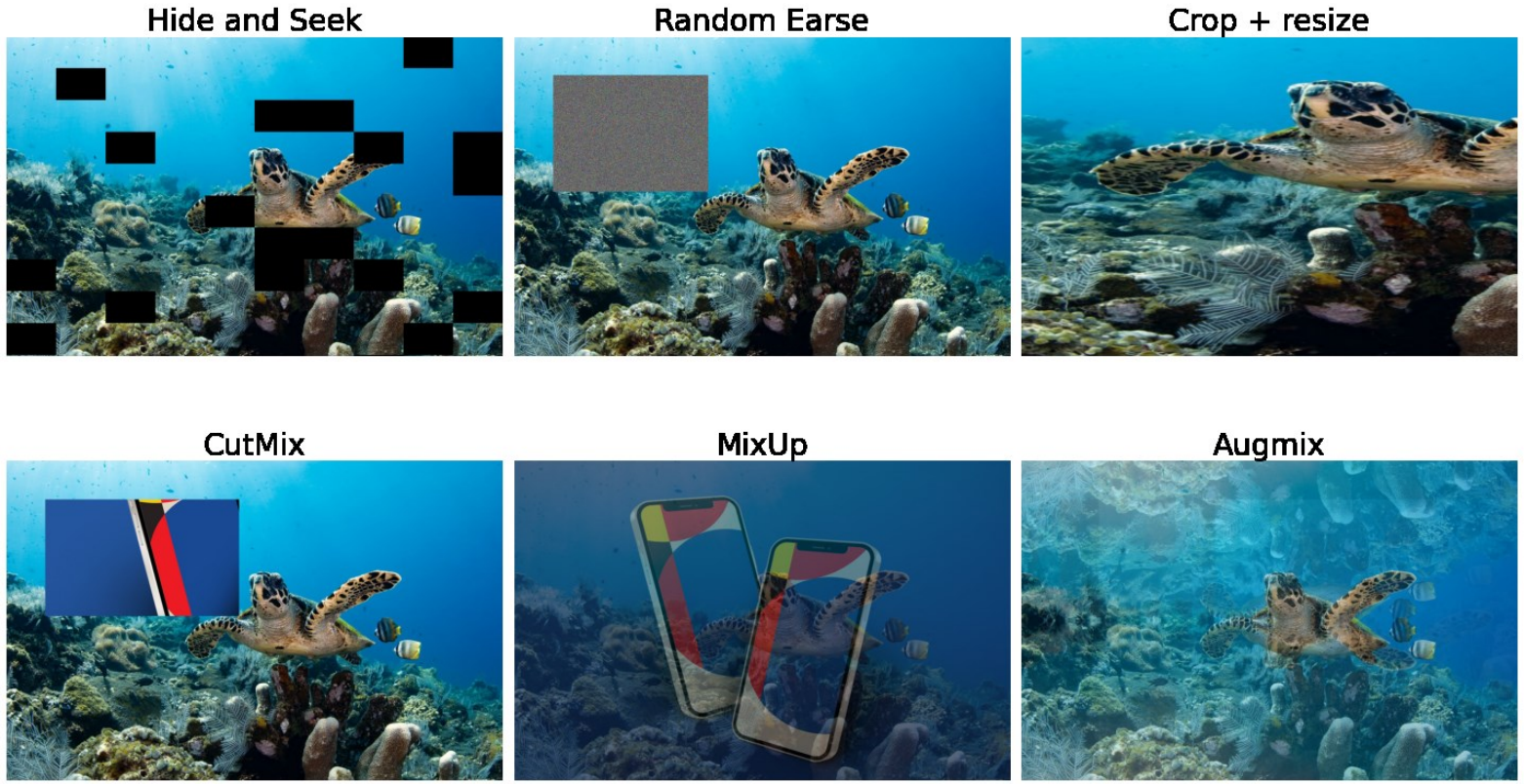
Примеры аугментации

- Аффинные преобразования
  - Вращение
  - Масштабирование
  - Случайная обрезка
  - Отражение
- Цветовы преобразования
  - Контрастный сдвиг
  - Изменение яркости
  - Размытие
  - Перемешивание каналов
- Искажающие преобрзования:
  - Добавление шумов
  - Добавление бликов
  - Добавление узоров



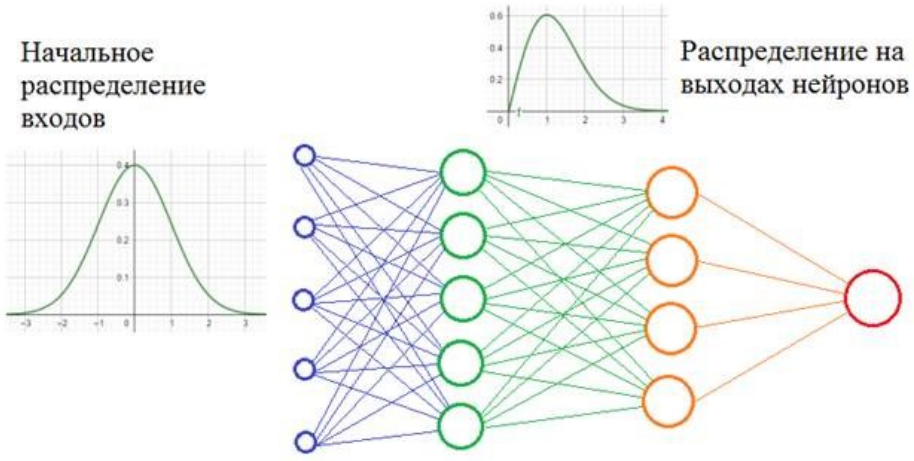
Современные примеры аугментации

- Расширенные преобразования
  - Случайное стирание
  - Добавление эффектов дождя, солнечных бликов...
  - Смешивание изображений
- Нейронные преобразования
  - соревновательные шум
  - Перенос стиля
  - Генеративно-состязательные сети
- Комбинации подходов



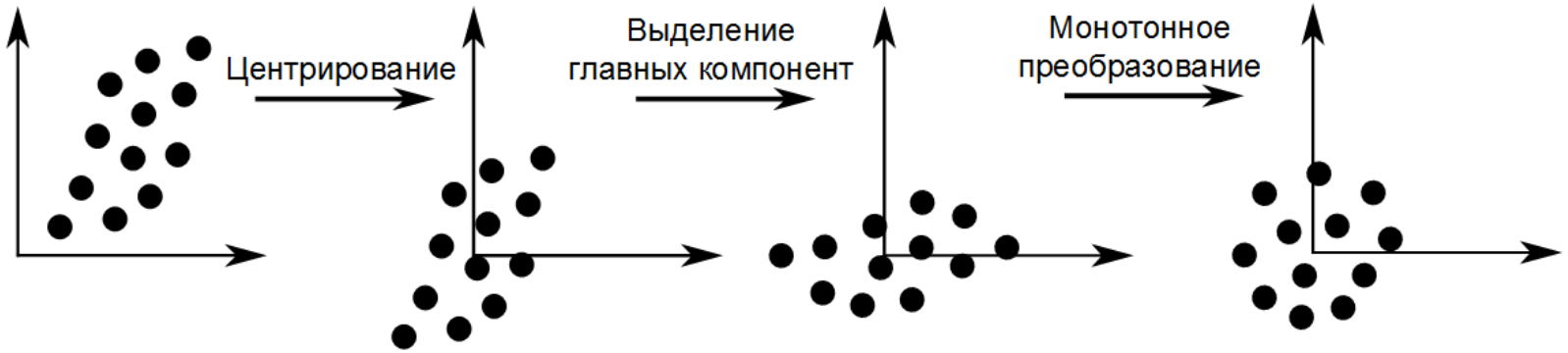
Необходимость нормализации, ковариационный сдвиг?

- Гипотеза: каждый слой нейронной сети пытается аппроксимировать распределение своих входных данных – то есть данных всех батчей.
  - Например, выделение регулярных признаков – это значит выделить среднее значение численной характеристики такого признака и выделить его дисперсию.



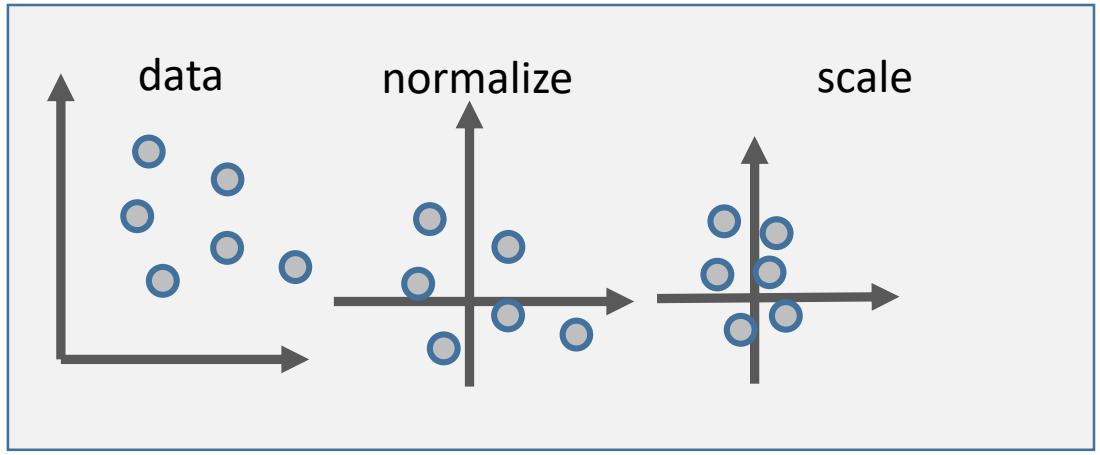
Необходимость нормализации, ковариационный сдвиг?

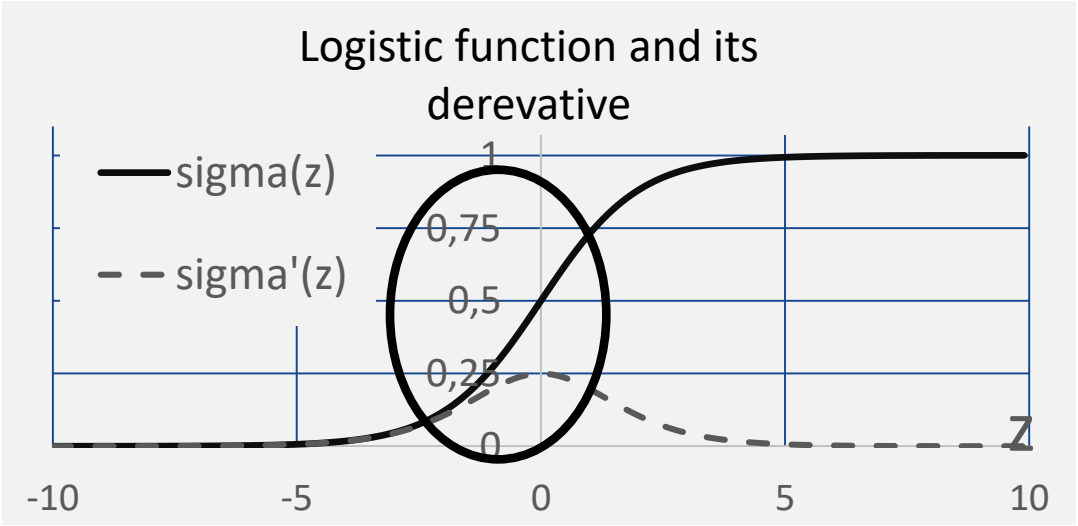
- Если каждый батч будет иметь существенно разные параметры распределения (среднее, дисперсия) – это приведет к потери стабильности результатов - этот феномен предложено назвать ковариационный сдвиг.
  - Ковариационный сдвиг можно преодолеть, если приводить все батчи к одному и тому же виду – то есть диапазону параметров.
  - На самом деле это эвристическая гипотеза, наличие и виляние ковариационного сдвига не доказано.



Метод батч-нормализации

- Одна из реализаций идеи компенсации разниц параметров батчей – это батч нормализация (BatchNorm).
  - BatchNorm: Все данные должны иметь всегда нулевое среднее и дисперсию 1 (нормализация).
    - Нормализированные можно отмасштабировать данные так, чтобы попасть в оптимальную зону значений функции активации.
      - Тогда еще и получится избежать вымывания градиента





Метод батч-нормализации

- Метод Батч-нормализации (BatchNorm).

$$y_i \leftarrow \gamma \frac{y_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta,$$

- Где
  - m размер батча.
  - $\mu_B \leftarrow \sum_{i=0}^{m-1} y_i$, среднее значение по батчу,
  - $\sigma_B^2 \leftarrow \sum_{i=0}^{m-1} (y_i - \mu_B)^2$ - дисперсия значений по батчу,
  - γ, β – параметры масштабирования, обучаются методом обратного распространения ошибки;
  - ϵ - небольшое число, предотвращающее деление на 0.

Метод батч-нормализации

- Для тестов используются значения μ_B и σ_B^2 получаются входе тренировки как EMA:

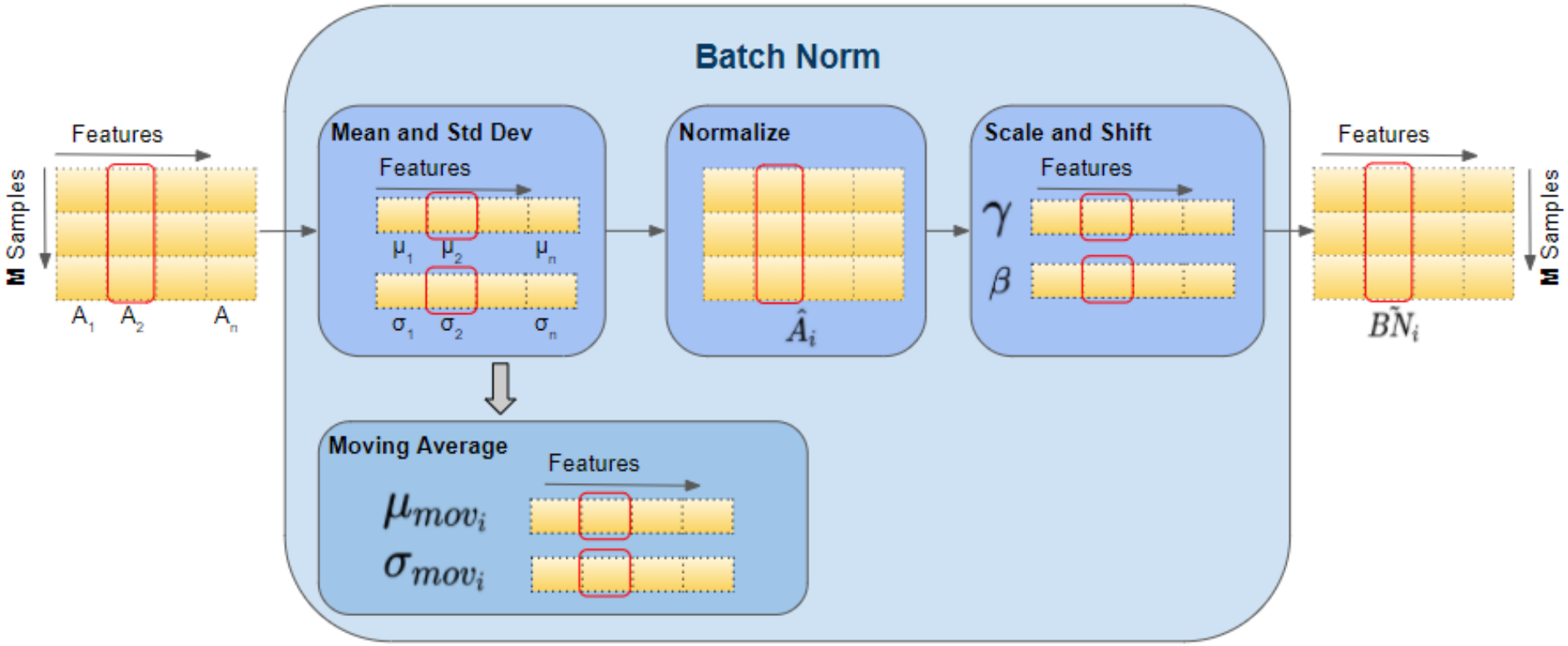
$$\mu_{B_{NEW}} = (1-\alpha)\cdot \mu_{B_{CURRENT}} + \alpha \cdot \frac{1}{m}\sum_{i=0}^{m-1} y_i,$$

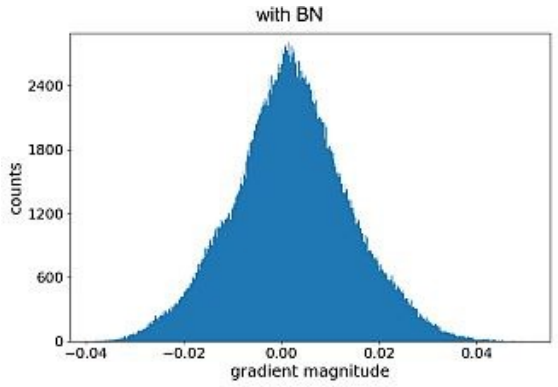
$$\sigma^2_{B_{NEW}} = (1-\alpha)\cdot \sigma^2_{B_{CURRENT}} + \alpha \cdot \frac{1}{m}\sum_{i=0}^{m-1} (y_i - \mu_B)^2,$$

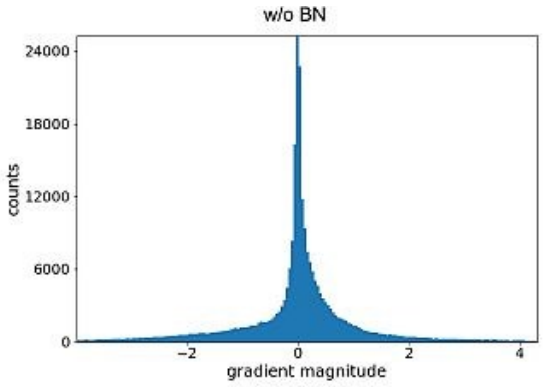
где

  - α веса экспоненциального сглаживания;
  - $\mu_{B_{CURRENT}}$ и $\sigma^2_{B_{CURRENT}}$ текущее значение среднего для тестовой выборки.
- То есть на тестовой выборке параметры не обучаются!

Метод батч-нормализации







Особенности метода батч-нормализации

- Достоинства батч-нормализации:
  - Регуляризация различий в статистических параметрах батчей их усреднение.
  - Снижение зависимости значений градиента от изменений масштаба для разных батчей.
  - Увеличение скорости обучения за счет регуляризации – ускорение тренировки.
    - В т.ч. Показано, что сети сходятся быстрее если данные выбелены (имеют нормальное распределение) и имеют низкую дисперсию.

Особенности метода батч-нормализации

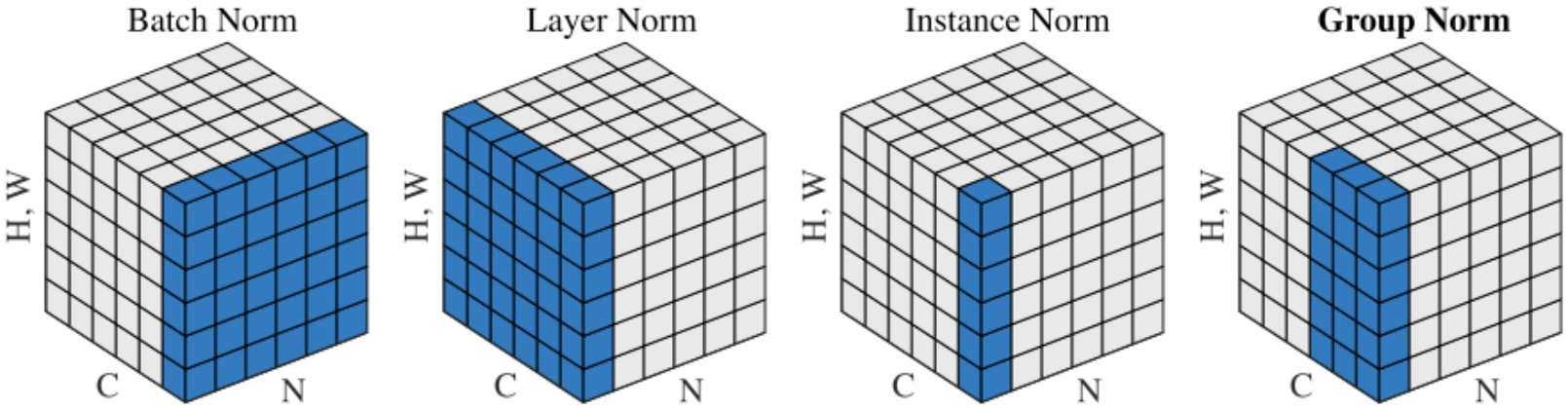
Недостатки батч-нормализации:

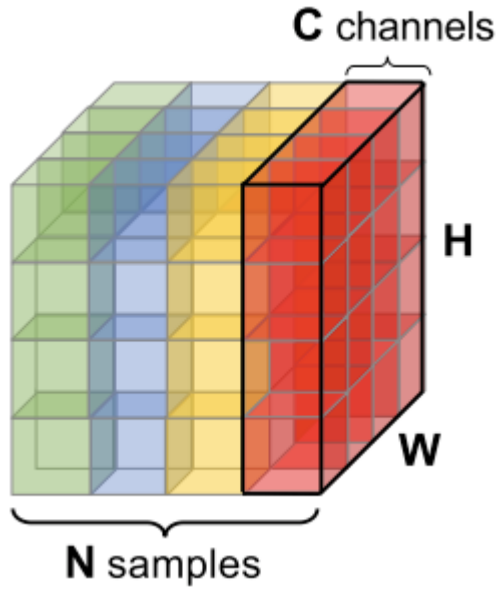
- Снижение точности вывода для переменного размера батчей.
- Может дать разную точность на предобучение, обучении и в работе, проблемы с переносом обучения.
- Хорошо регуляризует только большие размеры батчей (рекомендуют 50-100 экземпляров в батче).
- Не известно есть ковар. сдвиг, и нет математических доказательств работы метода – это эвристика.

- Метод как правило не работает с другими типами регуляризации (L1,L2, особенно дропаут).
- Как правили батч. Норм используется перед функцией активации – но иногда ставят после.

Другие методы нормализации

- **Нормализация слоя (Layer Norm) –** усреднение по слою, популярно в трансформерах, среднее и дисперсия считаются как в тренировке, так и в тесте.
- **Экземплярная нормализация** (**Instance Norm**): **–** используют в некоторых сетях переноса стиля**,** нормализация вдоль осей изображения (H,W), на тесте как батч-норм. смысл такой же как от автоконтраста.
- **Групповая нормализация** (**Group Norm**) : нормализация по несколько экземпляров каждого слоя, в остальном как нормализация слоя. Предложена как альтернатива батч-норме для маленьких выборок.
- **Переключаемая норма (Switch Norm**): комбинация других типов (напр, батч и слой).





Методы нормализации весовых параметров

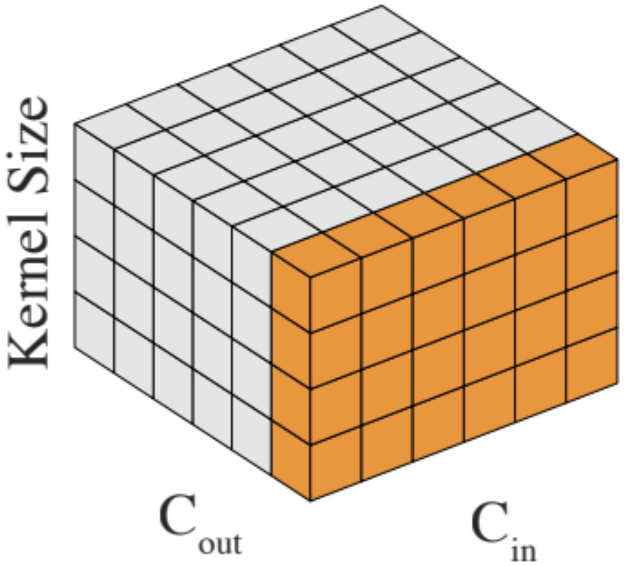
- **Иногда лучше нормализовать весовые параметры, чем данные.**

- Могут быть такие вариант как для данных:
  - батч норм, слой, группа и экземпляр, но по весовым параметрам.

- **Стандартизация весов:**

$$W = \gamma \frac{W - \min(W)}{\max(W) - \min(W)} + \beta$$

- **Спектральная нормализация – denoisng,**
  - **Разложение весовых параметров по SVD или EV и их фильтрац**
    - Это работает как метод восстановление после метода главных компонент (PCA)



Проблема деградации глубокой сети

- **Проблема деградации глубокой сети –** на практике увеличение числа параметров и числа слоев выше некоторого порога может приводить к снижению точности работы сети.
  - Даже если использовать регуляризацию и прочие методы.
    - из за того, что слои могут иметь вымытый градиент и прочие проблемы обучения.

Проблема деградации глубокой сети

- Проблему деградации глубокой сети можно решить если добавляемые слои, в худшем случае просто скопируют поведение предыдущих слоев.
  - То есть использованием тождественных слоев вида
    - $f(x) = x$
      - Однако, реализация впрямую в не даст роста точности.
- Решение этой проблемы – использование параллельно тождественного и сверточного слоев.

Остаточные связи

- Решение этой проблемы – использование параллельно тождественного и сверточного слоев.