• Введение в машинное обучение •

Искусственные нейронные сети

Воронцов Константин Вячеславович
k.v.vorontsov@phystech.edu
http://www.MachineLearning.ru/wiki?title=User:Vokov

Этот курс доступен на странице вики-ресурса
http://www.MachineLearning.ru/wiki
«Введение в машинное обучение (курс лекций, К.В.Воронцов)»

МФТИ.ФПМИ.ИС.ИАД • 26 февраля 2026

Содержание

Напоминание: модели классификации и регрессии

Дано: выборка $X^\ell = (x_i, y_i)_{i=1}^\ell$, объекты $x_i \in \mathbb{R}^n$, ответы y_i

Задача регрессии: $y_i \in \mathbb{R}$

Найти: модель регрессии $a(x, w)$, вектор параметров w

Критерий: $Q(w; X^\ell) = \sum_{i=1}^{\ell} L\big(\underbrace{a(x_i, w) - y_i}_{\varepsilon_i(w)\ \text{— error, ошибка}}\big) \to \min_w,$

где L унимодальная: $L(\varepsilon) = \varepsilon^2$, $|\varepsilon|$, $\big(|\varepsilon| - c\big)_+$, и др.

Задача классификации с двумя классами: $y_i \in \{\pm 1\}$

Найти: модель классификации $a(x, w) = \operatorname{sign} g(x, w)$

Критерий: $Q(w; X^\ell) = \sum_{i=1}^{\ell} L\big(\underbrace{g(x_i, w) y_i}_{M_i(w)\ \text{— margin, отступ}}\big) \to \min_w,$

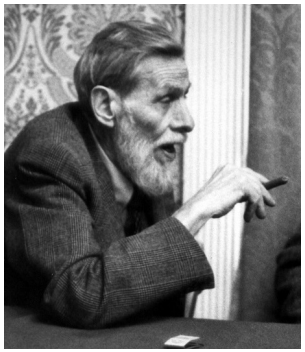
где L невозрастающая: $L(M) = \ln(1 + e^{-M})$, $(1 - M)_+$, и др.

Искусственный нейрон — линейная модель классификации

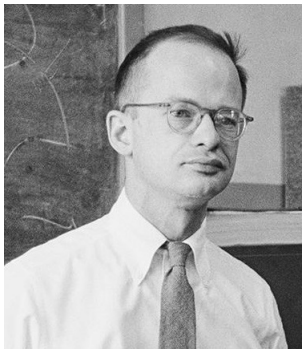
Линейная модель нейрона (1943):

$$a(x, w) = \sigma\left(\sum_{j=1}^{n} w_j f_j(x) - w_0\right)$$

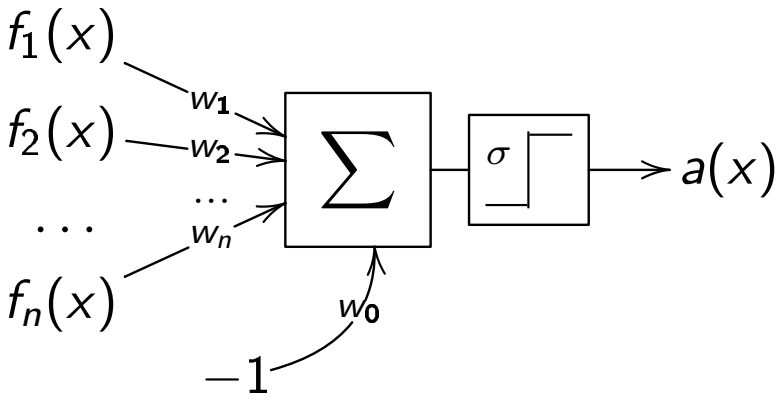
$f_j(x)$ — признаки объекта x
w_j — веса признаков
w_0 — порог активации
$\sigma(z)$ — функция активации

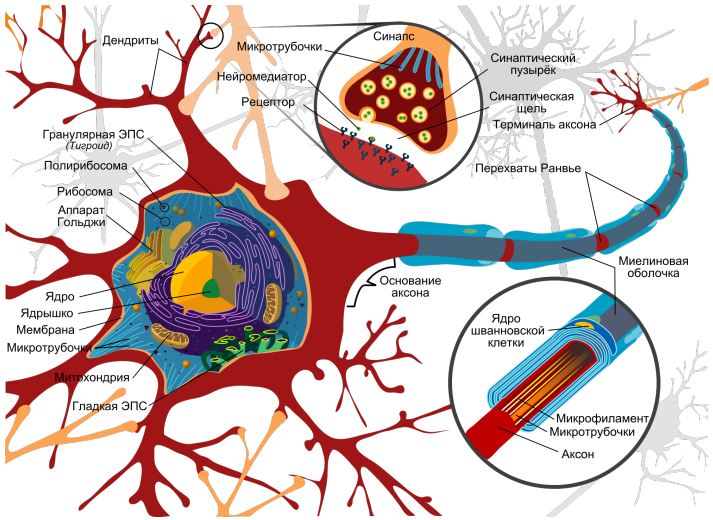


Уоррен МакКаллок (1898–1969)



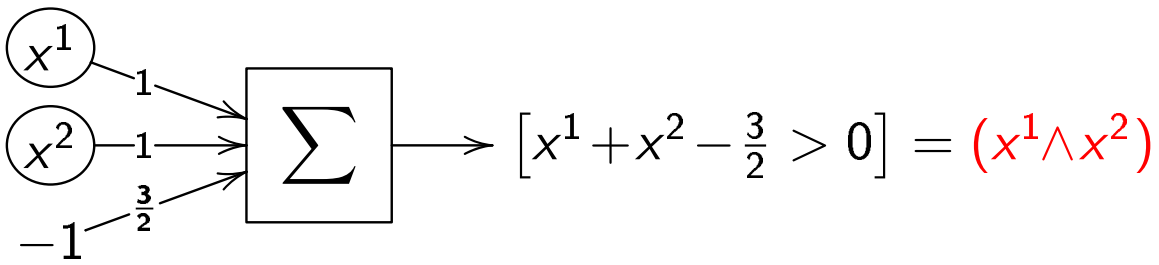
Вальтер Питтс (1923–1969)

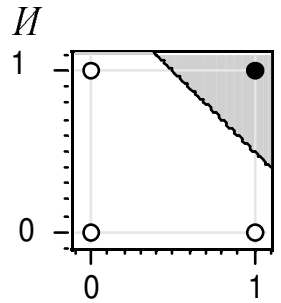


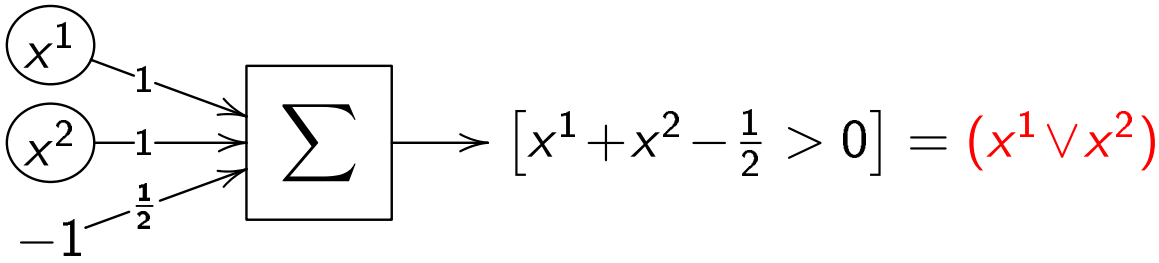


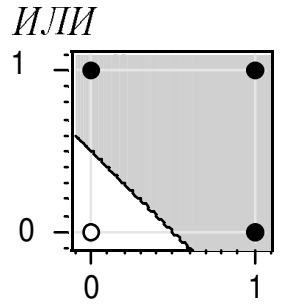
Нейронная реализация булевых функций

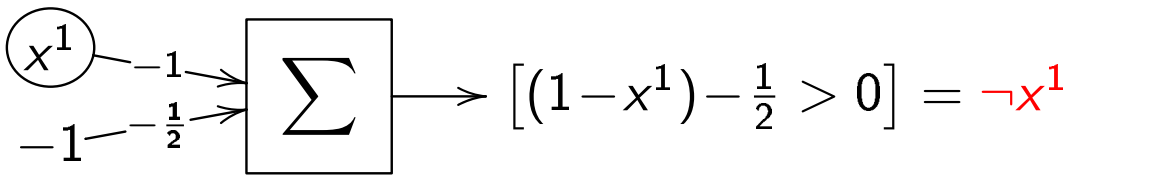
Функции И, ИЛИ, НЕ бинарных признаков x^1, x^2:

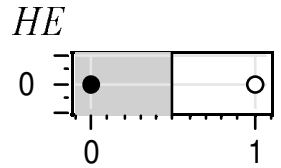












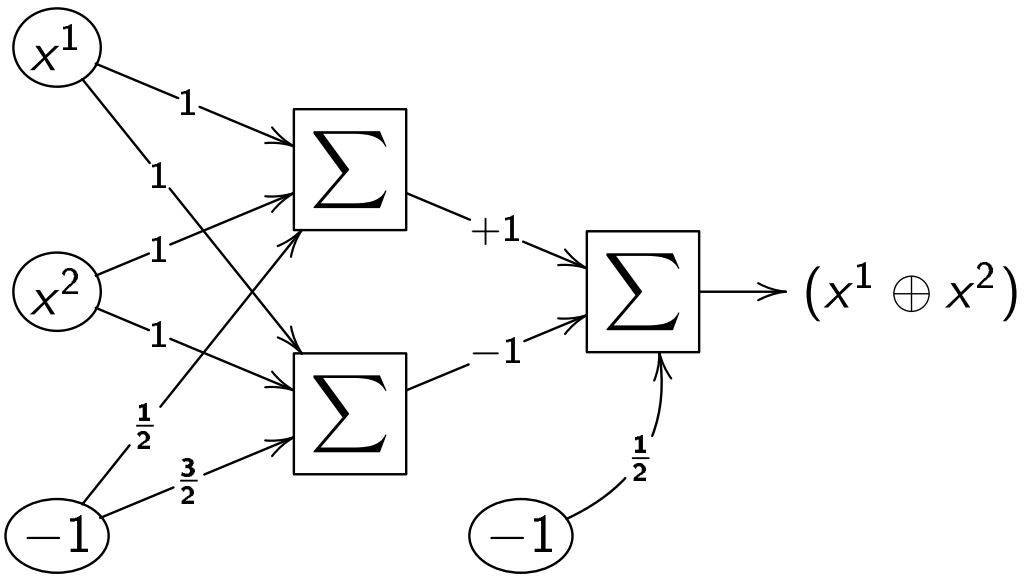
Вопрос: любую ли булеву функцию можно реализовать нейроном? нейросетью? сколько потребуется слоёв?

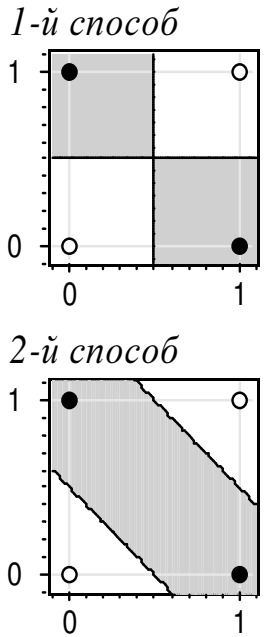
Ограниченность линейных моделей: функция XOR

Функция $x^1 \oplus x^2 = [x^1 \neq x^2]$ не реализуема одним нейроном.

Два способа реализации:

- Добавлением нелинейного признака (feature generation):
  $x^1 \oplus x^2 = \left[x^1 + x^2 - 2x^1x^2 - \frac{1}{2} > 0\right]$;
- Сетью (двухслойной суперпозицией) функций И, ИЛИ, НЕ:
  $x^1 \oplus x^2 = \left[(x^1 \vee x^2) - (x^1 \wedge x^2) - \frac{1}{2} > 0\right]$.





Персептрон Розенблатта (1957)

Mark-1 — первый нейрокомпьютер (1960)
для распознавания цифр и фигур
Обучение — метод коррекции ошибки
Архитектура — двухслойная сеть

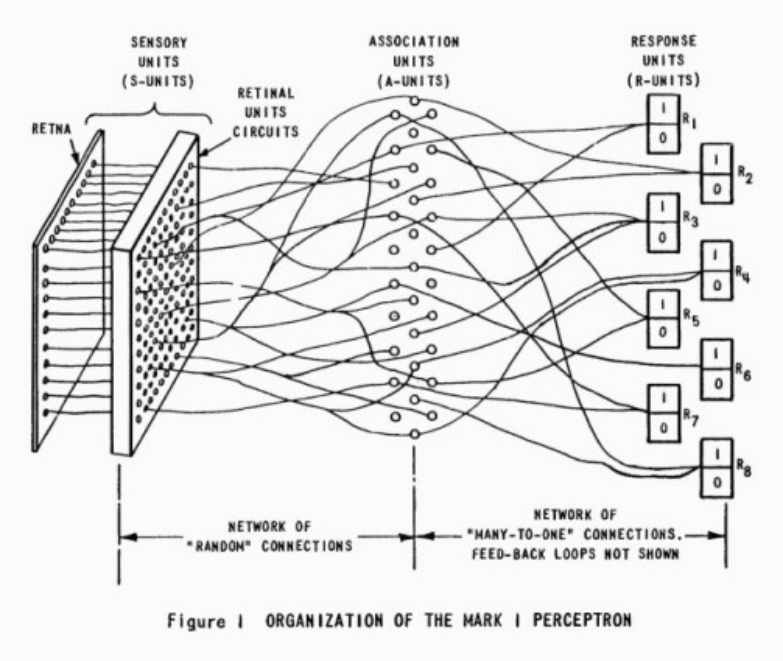


Figure 1 ORGANIZATION OF THE MARK I PERCEPTRON

Фрэнк Розенблатт
(1928–1971)

Розенблатт Ф. Принципы нейродинамики. Перцептроны и теория механизмов мозга. 1965 (1962)

Любую ли функцию можно представить нейросетью?

Решение тринадцатой (из 23) проблем Гильберта (1900):

Теорема [Колмогоров, 1956; Арнольд, 1957]

Любая непрерывная функция n аргументов на единичном кубе $[0,1]^n$ представима в виде суперпозиции непрерывных функций одного аргумента и операции сложения:

$$f(x_1,\ldots,x_n) = \sum_{k=1}^{2n+1} \Phi_k\left(\sum_{j=1}^{n} \varphi_{jk}(x_j)\right),$$

где Φ_k, φ_{jk} — непрерывные функции, и φ_{jk} не зависят от f.

Вопрос: можно ли считать это двухслойной нейросетью?

А.Н.Колмогоров. О представлении непрерывных функций нескольких переменных суперпозициями непрерывных функций меньшего числа переменных. 1956.

В.И.Арнольд. О функции трех переменных. 1957.

Имеет ли теорема Колмогорова отношение к нейросетям?

Вроде да:

- структура суперпозиции соответствует двухслойной сети
- имеются универсальные аппроксимационные свойства

На самом деле — нет:

- это точное представление; нам достаточно аппроксимации
- функции Φ_k, φ_{jk} не гладкие и сложно строятся
- нет ни весов W, ни оптимизационной задачи обучения
- число слоёв 2 и число нейронов $[2n+1, n]$ фиксированы

Но можно обобщить конструкцию эвристически (KAN):

- любое число слоёв, любая ширина слоёв
- вместо функций Φ_k, φ_{jk} — одномерные сплайны (с весами)
- использовать стандартные методы обучения (BackProp)

Ziming Liu et al. KAN: Kolmogorov–Arnold Networks. 2024/04/30

Двухслойные сети — универсальные аппроксиматоры функций

Функция $\sigma(z)$ — *сигмоида*, если $\lim\limits_{z\to-\infty}\sigma(z)=0$ и $\lim\limits_{z\to+\infty}\sigma(z)=1$.

Теорема Цыбенко (universal approximation theorem, 1989)

Если $\sigma(z)$ — непрерывная сигмоида, то для любой непрерывной на $[0,1]^n$ функции $f(x)$ существуют такие значения параметров H, $\alpha_h\in\mathbb{R}$, $w_h\in\mathbb{R}^n$, $w_0\in\mathbb{R}$, что двухслойная сеть

$$a(x)=\sum_{h=1}^{H}\alpha_h\sigma\big(\langle x,w_h\rangle-w_0\big)$$

равномерно приближает $f(x)$ с любой точностью ε:

$$\big|a(x)-f(x)\big|<\varepsilon,\ \text{для всех } x\in[0,1]^n.$$

George Cybenko. Approximation by Superpositions of a Sigmoidal function. Mathematics of Control, Signals, and Systems. 1989.

Многослойный персептрон (Multilayer Perceptron, MLP)

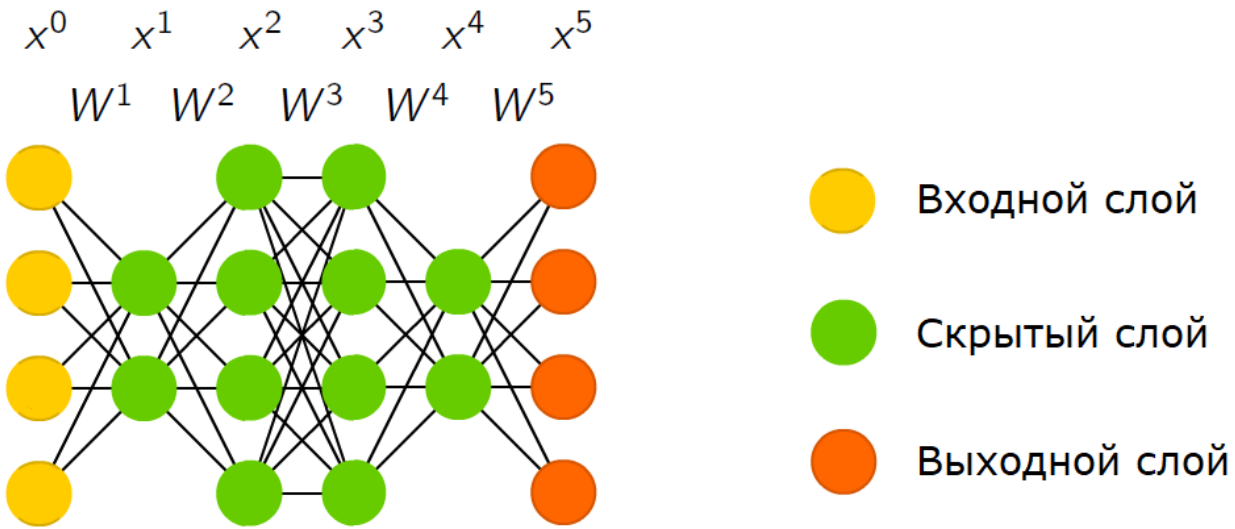
Полносвязная сеть, H_l — число нейронов в слое $l = 1, \ldots, L$

$x^0 = x = \big(f_j(x)\big)_{j=0}^{n}$ — вектор признаков на входе сети, $H_0 = n$

$x^l = \big(x_h^l\big)_{h=0}^{H_l}$ — вектор признаков на выходе l-го слоя, $x_0^l = -1$

$x^L = a(x) = \big(a_m(x)\big)_{m=1}^{M}$ — выходной вектор сети, $H_L = M$

$W^l = (w_{kh}^l)$ — матрица весов l-го слоя, размера $(H_{l-1}+1) \times H_l$



Вопрос: зачем нужно много (M) выходных нейронов?

Напоминание: алгоритм SG (Stochastic Gradient)

Минимизация средних потерь на обучающей выборке:

$$Q(w) = \frac{1}{\ell}\sum_{i=1}^{\ell}\mathscr{L}(w, x_i) \to \min_w.$$

Вход: выборка $(x_i, y_i)_{i=1}^{\ell}$; темп обучения η; параметр λ;
Выход: вектор весов всех слоёв $w = (W^1, \ldots, W^L)$;

инициализировать веса w и текущую оценку $Q(w)$;
повторять
  выбрать объект x_i из X^ℓ (например, случайно);
  вычислить потерю $\mathscr{L}_i := \mathscr{L}(w, x_i)$;
  градиентный шаг: $w := w - \eta\nabla\mathscr{L}(w, x_i)$;
  оценить значение функционала: $Q := (1-\lambda)Q + \lambda\mathscr{L}_i$;
пока значение Q и/или веса w не стабилизируются;

H.Robbins, S.Monro. A stochastic approximation method // Annals of Math. Stat., 1951.

Задача дифференцирования суперпозиции функций

Вычисление сети по входному вектору x, рекуррентно по слоям: $x^l = \sigma^l\left(W^l x^{l-1}\right)$ — в матричной записи, и покоординатно:

$$x_h^l = \sigma_h^l\left(S_h^l\right), \quad S_h^l = \sum_{k=0}^{H_{l-1}} w_{kh}^l x_k^{l-1}, \quad h = 1, \ldots, H_l, \quad l = 1, \ldots, L,$$

Функция потерь на объекте x_i (например, квадратичная):

$$\mathscr{L}(w, x_i) \equiv \mathscr{L}_i(w) = \frac{1}{2} \sum_{m=1}^{M} \left(a_m(x_i, w) - y_{im}\right)^2$$

По формуле дифференцирования суперпозиции функций:

$$\frac{\partial \mathscr{L}_i(w)}{\partial w_{kh}^l} = \frac{\partial \mathscr{L}_i(w)}{\partial x_h^l} \frac{\partial x_h^l}{\partial w_{kh}^l}, \quad k = 0, \ldots, H_{l-1}, \quad h = 1, \ldots, H_l$$

Вопрос: что изменится, если сеть будет неполносвязной?

Рекуррентное вычисление частных производных

Найдём сначала частные производные $\mathscr{L}_i(w)$ по $x_m^L \equiv a_m(x_i, w)$:

$$\frac{\partial \mathscr{L}_i(w)}{\partial x_m^L} = a_m(x_i, w) - y_{im} \equiv \varepsilon_{im}^L;$$

для квадратичной функции потерь это *ошибка выходного слоя*.

Частные производные по x_h^l будем вычислять рекуррентно, по уровням справа налево, $l = L, \ldots, 2$:

$$\frac{\partial \mathscr{L}_i(w)}{\partial x_k^{l-1}} = \sum_{h=0}^{H_l} \frac{\partial \mathscr{L}_i(w)}{\partial x_h^l} \underbrace{(\sigma_h^l)'(S_{ih}^l)}_{z_{ih}^l} w_{kh}^l = \sum_{h=0}^{H_l} \varepsilon_{ih}^l z_{ih}^l w_{kh}^l = \varepsilon_{ik}^{l-1}$$

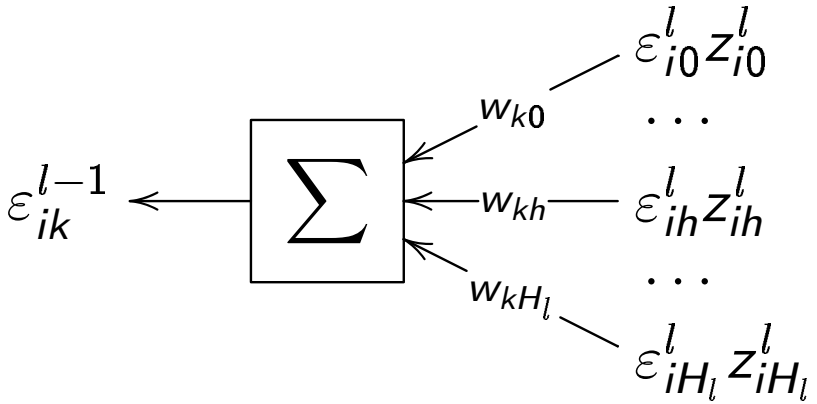
— формально назовём это *ошибкой скрытого слоя*.

Замечание: σ_h^l и $(\sigma_h^l)'$ вычисляются в точке $S_{ih}^l = \sum_{k=0}^{H_{l-1}} w_{kh}^l x_{ik}^{l-1}$

Вопрос: что изменится, если взять другую функцию потерь?

Быстрое вычисление градиента

Рекуррентная формула записана так, будто сеть запускается «задом наперёд», чтобы вычислять ε_{ik}^{l-1} по ε_{ih}^{l}:



Теперь, имея частные производные $\mathscr{L}_i(w)$ по всем x_h^l, легко найти градиент $\mathscr{L}_i(w)$ по вектору весов w:

$$\frac{\partial \mathscr{L}_i(w)}{\partial w_{kh}^l} = \frac{\partial \mathscr{L}_i(w)}{\partial x_h^l} \frac{\partial x_h^l}{\partial w_{kh}^l} = \varepsilon_{ih}^l z_{ih}^l x_{ik}^{l-1}$$

Вопрос (на внимание): чем отличается x_{ih}^l от x_h^l?

Алгоритм обратного распространения ошибки BackProp

Вход: выборка $(x_i, y_i)_{i=1}^{\ell}$, архитектура $(H_l)_{l=1}^{L}$, параметры η, λ;
Выход: вектор весов всех слоёв $w = (W^1, \ldots, W^L)$;
инициализировать веса w;
повторять
  выбрать объект x_i из X^ℓ (например, случайно);
  прямой ход: для всех $l = 1..L$, $h = 1..H_l$
    $S_{ih}^l := \sum_{k=0}^{H_{l-1}} w_{kh}^l x_{ik}^{l-1}$; $x_{ih}^l := \sigma_h^l(S_{ih}^l)$; $z_{ih}^l := (\sigma_h^l)'(S_{ih}^l)$;
  вычисление ошибок: $\varepsilon_{hi}^L := \frac{\partial \mathscr{L}_i(w)}{\partial x_h^L}$, для всех $h = 1..H_L$;
  обратный ход: для всех $l = L..2$, $k = 0..H_{l-1}$
    $\varepsilon_{ik}^{l-1} = \sum_{h=0}^{H_l} \varepsilon_{ih}^l z_{ih}^l w_{kh}^l$;
  градиентный шаг: для всех $l = 1..L$, $k = 0..H_{l-1}$, $h = 1..H_l$
    $w_{kh}^l := w_{kh}^l - \eta\, \varepsilon_{ih}^l z_{ih}^l x_{ik}^{l-1}$;
пока значения Q и/или веса w не стабилизируются;

Алгоритм BackProp: преимущества и недостатки

Преимущества:

- время вычисления градиента $O(\dim w)$ вместо $O(\dim^2 w)$
- обобщение на любые архитектуры и любые σ, $\mathscr{L}$
- возможность динамического (потокового) обучения
- сублинейное обучение на сверхбольших данных
- возможность распараллеливания

Недостатки — все те же, свойственные SG:

- медленная сходимость
- застревание в локальных экстремумах
- затухание градиентов из-за горизонтальных асимптот σ
- взрывы градиента из-за овражного ландшафта критерия
- мультиколлинеарность линейных нейронов $\Rightarrow$ переобучение
- подбор комплекса эвристик является искусством

Алгоритм BackProp: эвристики

Те же, что для любого градиентного методы:

1. подбор инициализации
2. адаптивные градиентные шаги для ускорения сходимости (Momentum, NAG, PMSProp, AdaDelta, Adam, Nadam и др.)
3. преодоление локальных экстремумов, мультистарт

Специфические эвристики для нейросетевых архитектур:

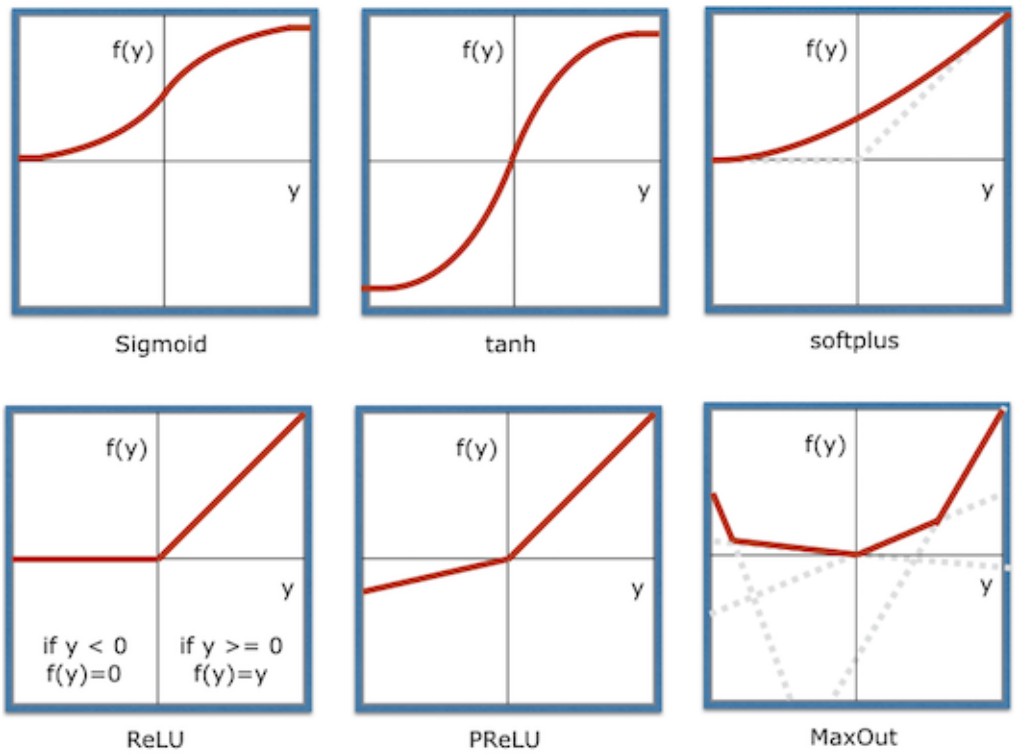
1. инициализация путём быстрого послойного обучения
2. кусочно-линейные функции активации ReLU и др.
3. регуляризации: L_2, L_1, Dropout, Skip Connection
4. различные нормировки для вычислительной устойчивости
5. разреживание сети (L_1, Optimal Brain Damage, OBS и др.)

Функции активации ReLU и PReLU (LeakyReLU)

Функции $\sigma(y) = \frac{1}{1+e^{-y}}$ и $\text{th}(y) = \frac{e^{y}-e^{-y}}{e^{y}+e^{-y}}$ могут приводить к затуханию градиентов или «параличу сети»

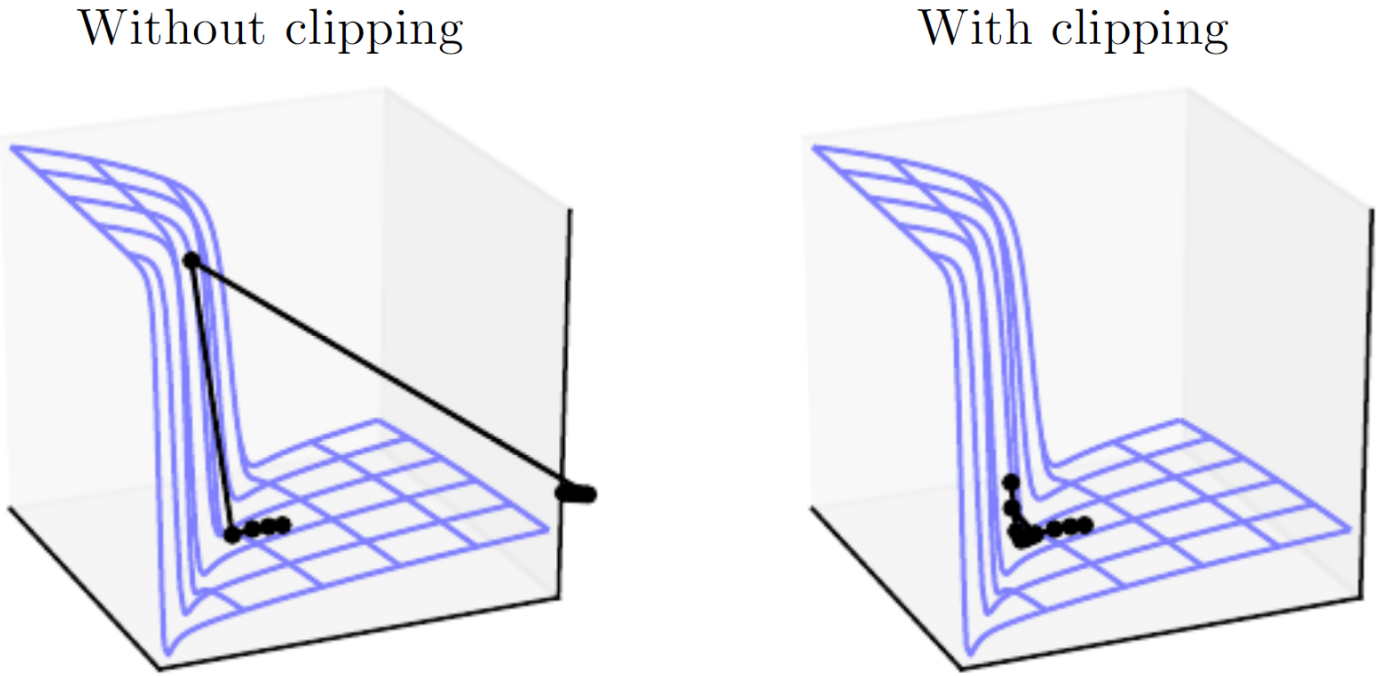
Функция положительной срезки (rectified linear unit)

$$\text{ReLU}(y) = \max\{0, y\}; \qquad \text{PReLU}(y) = \max\{0, y\} + \alpha \min\{0, y\}$$



Проблема взрыва градиента и эвристика gradient clipping

Проблема взрыва градиента (gradient exploding)
Причина — овражный ландшафт оптимизируемого критерия



Эвристика Gradient Clipping:
если $\|g\| > \theta$ **то** $g := g\theta/\|g\|$

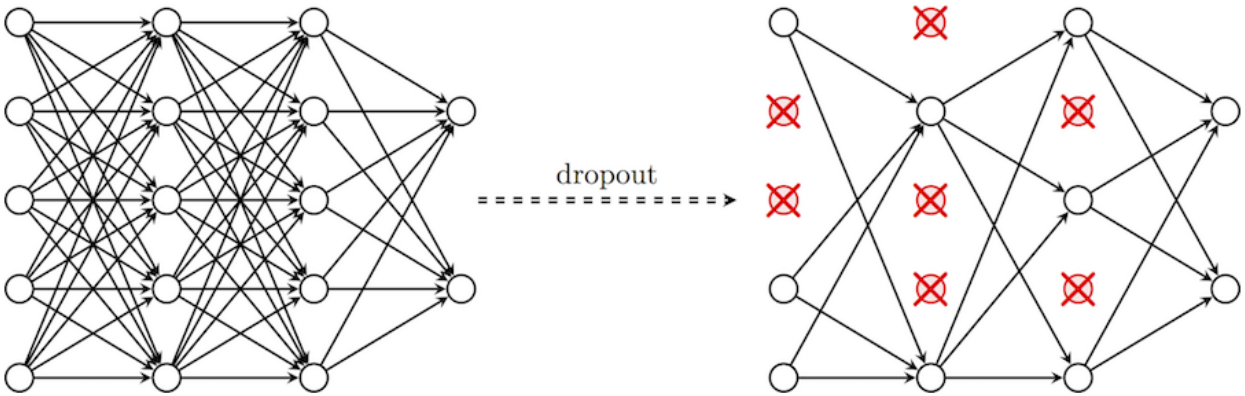
Метод случайных отключений нейронов (Dropout)

Этап обучения: делая градиентный шаг $\mathscr{L}_i(w) \to \min_w$,
отключаем h-ый нейрон l-го слоя с вероятностью p_l:

$$x_{hi}^l = \xi_h^l \, \sigma_h^l\Big(\sum_k w_{kh}^l x_{ki}^{l-1}\Big), \qquad \mathsf{P}(\xi_h^l = 0) = p_l$$

Этап применения: включаем все нейроны, но с поправкой:

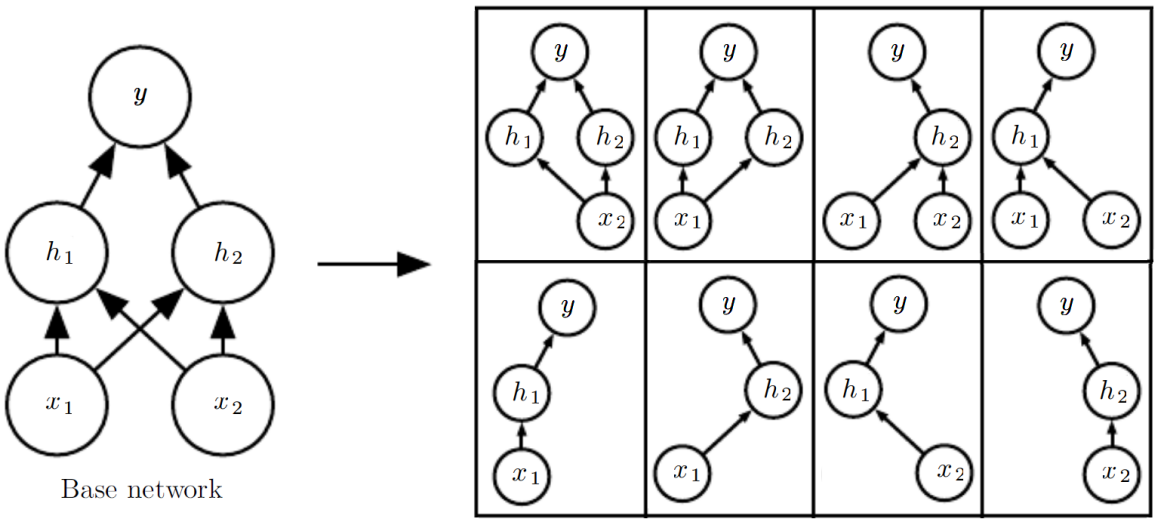
$$x_{hi}^l = (1 - p_l)\sigma_h^l\Big(\sum_k w_{kh}^l x_{ki}^{l-1}\Big)$$



N.Srivastava, G.Hinton, A.Krizhevsky, I.Sutskever, R.Salakhutdinov.
Dropout: a simple way to prevent neural networks from overfitting. 2014.

Интерпретации Dropout

1. регуляризация: из всех сетей выбираем более устойчивую к утрате pN нейронов, моделируя надёжность мозга
2. сокращаем переобучение, заставляя разные части сети решать одну и ту же исходную задачу вместо того, чтобы подстраивать их под компенсацию ошибок друг друга
3. аппроксимируем простое голосование по 2^N сетям с общим набором из N весов, но с различной архитектурой связей



Base network

Обратный Dropout и L_2-регуляризация

На практике чаще используют не Dropout, а *Inverted Dropout*.

Этап обучения:

$$x_{hi}^l = \frac{1}{1-p_l}\xi_h^l\,\sigma_h^l\Big(\sum_k w_{kh}^l x_{ki}^{l-1}\Big), \qquad \mathsf{P}(\xi_h^l = 0) = p_\ell$$

Этап применения не требует ни модификаций, ни знания p_ℓ:

$$x_{hi}^l = \sigma_h^l\Big(\sum_k w_{kh}^l x_{ki}^{l-1}\Big)$$

L_2-регуляризация предотвращает рост параметров на обучении:

$$\mathscr{L}(w, x_i) + \tfrac{\lambda}{2}\|w\|^2 \to \min_w$$

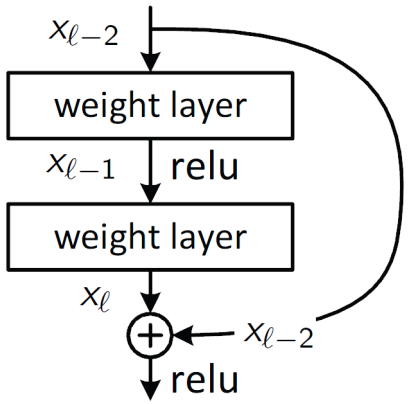
Градиентный шаг с Dropout и L_2-регуляризацией:

$$w := w(1-\eta\lambda) - \eta\,\frac{1}{1-p_\ell}\xi_h^\ell\,\nabla\mathscr{L}(w, x_i)$$

ResNet: остаточная нейронная сеть (Residual NN)

Сквозная связь (skip connection) слоя l
с предшествующим слоем $l - d$:

$$x_l = \sigma(Wx_{l-1}) + x_{l-d}$$



Слои выучивают малую поправку $x_l - x_{l-d}$
а не заново всё преобразование $x_{l-d} \to x_l$

- Приращения более устойчивы $\Rightarrow$ улучшается сходимость
- Появляется возможность увеличивать число слоёв
- Обобщение — Highway Networks:

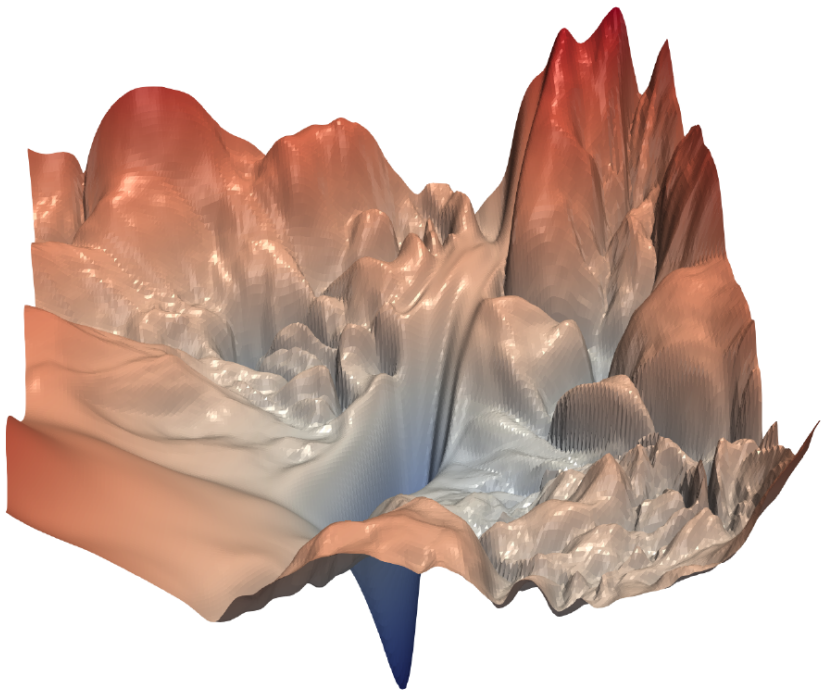
$$x_l = \sigma(Wx_{l-1})\underbrace{\tau(W'x_{l-1})}_{\text{transform gate}} + x_{l-d}\underbrace{\big(1 - \tau(W'x_{l-1})\big)}_{\text{carry gate}}$$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. 2015

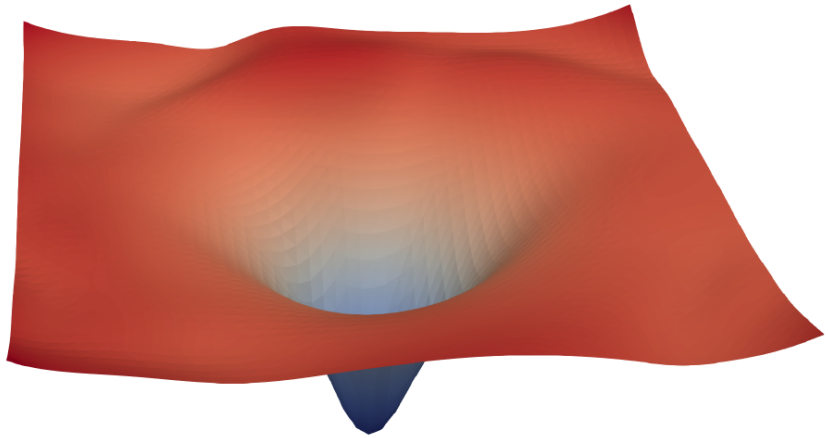
R.K.Srivastava, K.Greff, J.Schmidhuber. Highway Networks. 2015

ResNet: визуализация оптимизационного критерия

Сквозные связи (skip connection) упрощают оптимизируемый критерий, устраняя локальные экстремумы и седловые точки:



without skip connections



with skip connections

Hao Li et al. Visualizing the Loss Landscape of Neural Nets. 2018

Пакетная нормализация данных (Batch Normalization)

$B = \{x_i\}$ — пакеты (mini-batch) данных
$B^l = \{x_i^l\}$ — векторы объектов x_i на выходе l-го слоя

Нормировка каждой h-й компоненты вектора x_i^l по пакету:

$$\hat{x}_{hi}^l = \frac{x_{hi}^l - \mu_h}{\sqrt{\sigma_h^2 + \varepsilon}}; \quad \mu_h = \frac{1}{|B^l|}\sum_{x_i^l \in B^l} x_{hi}^l; \quad \sigma_h^2 = \frac{1}{|B^l|}\sum_{x_i^l \in B^l} (x_{hi}^l - \mu_h)^2$$

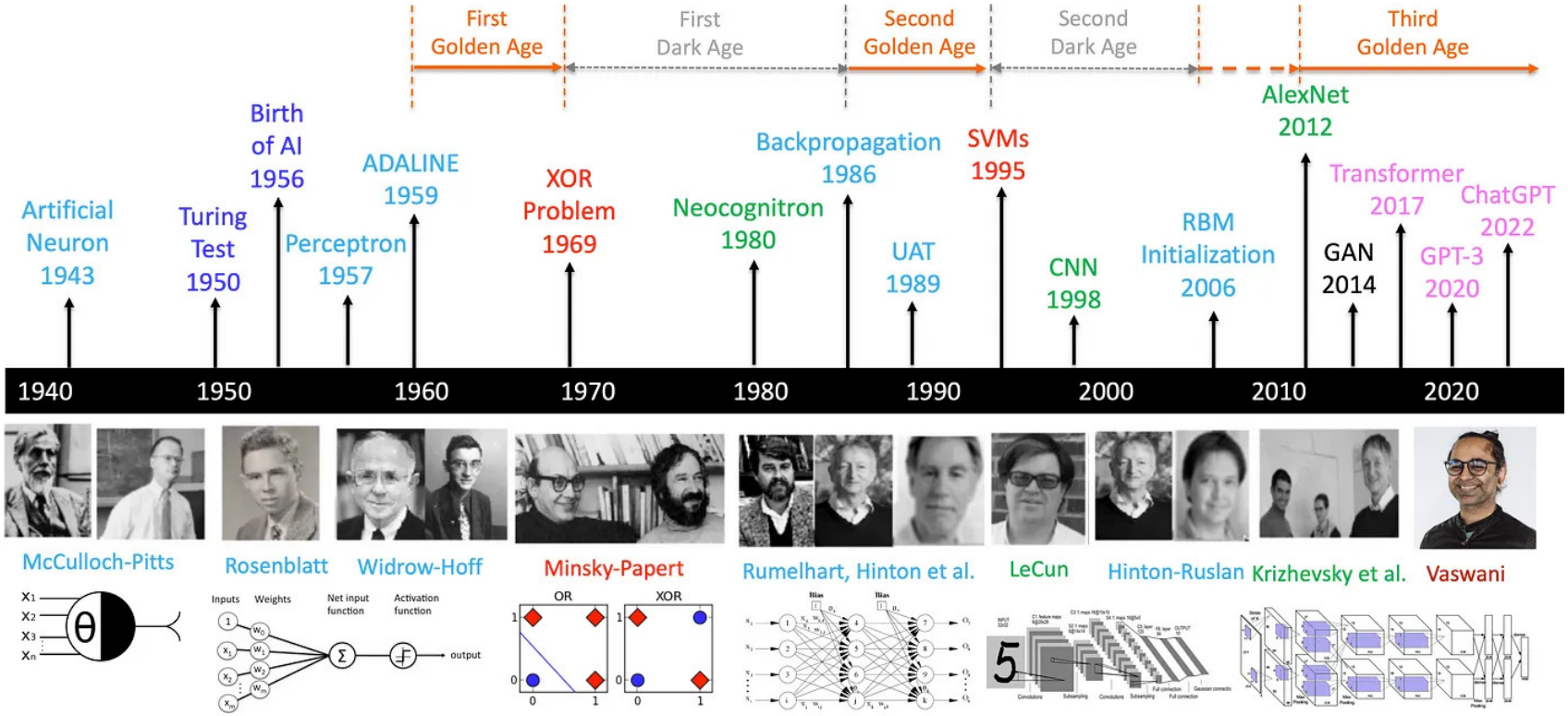
Компенсация: $\tilde{x}_{hi}^l = \gamma_h^l \hat{x}_{hi}^l + \beta_h^l$ — линейный слой,
параметры γ_h^l и β_h^l настраиваются BackProp

Обоснования (как это работает, до конца не ясно):

- градиент становится ограниченным и более устойчивым
- ускоряется сходимость, можно увеличить градиентные шаги
- регуляризация, повышение численной устойчивости

S.Ioffe, C.Szegedy (Google) Batch normalization: accelerating deep network training by reducing internal covariate shift. 2015.

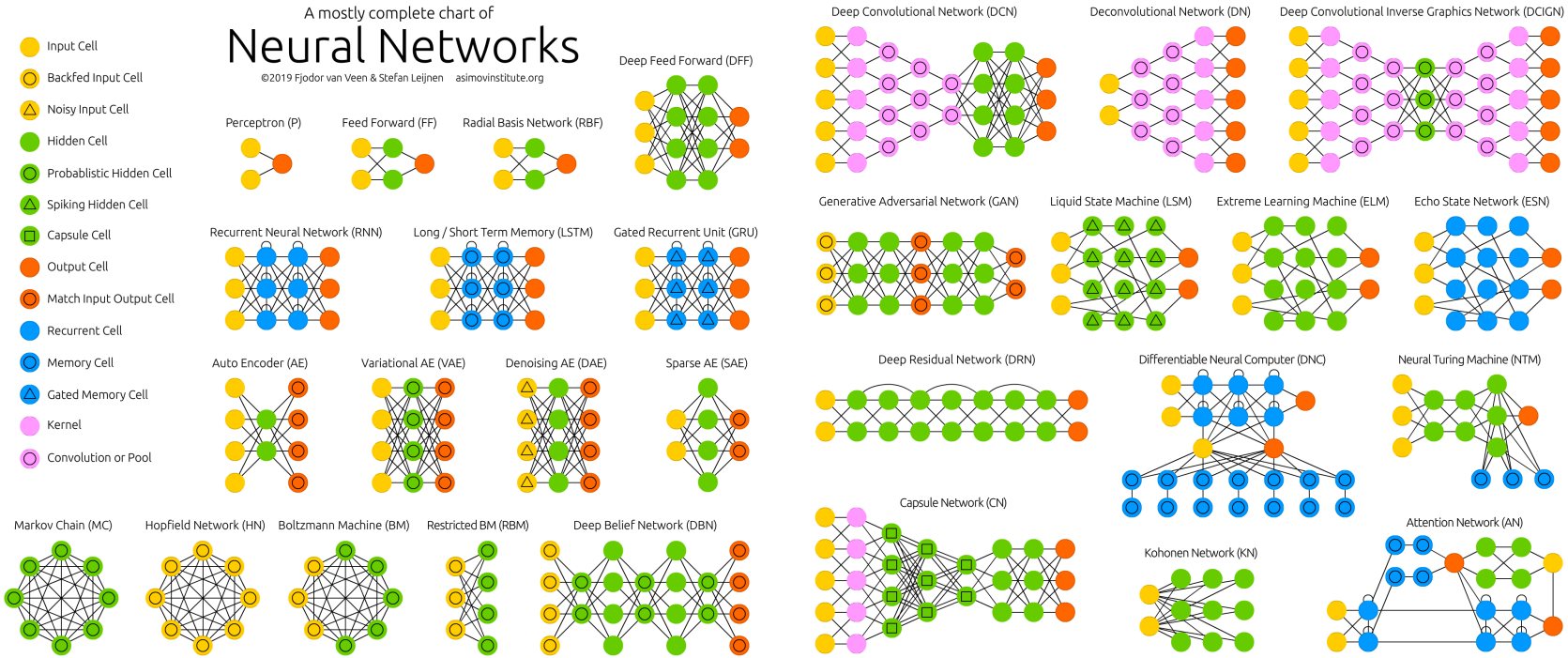
Основные вехи развития нейронных сетей (AI winters)



Ива́хненко А. Г., Лапа В. Г. Кибернетические предсказывающие устройства. 1965
Минский М., Пайперт С. Персептроны. 1971 (1969)
Галушкин А. И. Синтез многослойных систем распознавания образов. 1974
Rummelhart D. et al. Learning internal representations by error propagation. 1986
Krizhevsky A. et al. ImageNet classification with deep convolutional neural networks. 2012
Vaswani A. et al. Attention is all you need. 2017

Глубокие нейронные сети (Deep Neural Network, DNN)

Архитектура сети — структура слоёв и связей между ними, позволяющая наделять DNN нужными свойствами



Все архитектуры обучаются методом BackProp (ну, почти)

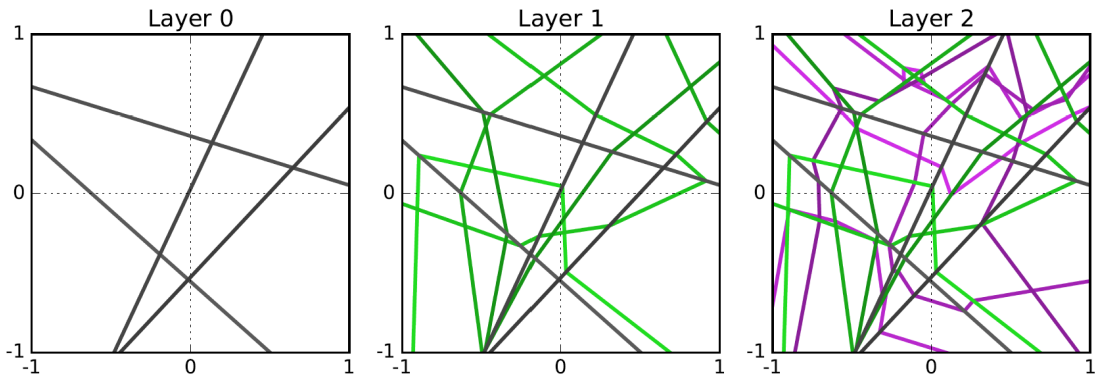
Sanjeev Arora. Toward theoretical understanding of deep learning. ICML-2018 Tutorial
https://unsupervised.cs.princeton.edu/deeplearningtutorial.html

Аппроксимационная способность: глубина важнее ширины

A^n_{LH} — семейство полносвязных многослойных сетей $a(x, w)$: n признаков, L слоёв, H нейронов в каждом слое, $x \in \mathbb{R}^n$, функции активации кусочно-линейные: ReLU, hard-tanh и т.п.

Мера разнообразия семейства A^n_{LH} — максимальное число участков линейности $a(x, w)$ — выпуклых многогранников в $\mathbb{R}^n$.

Пример. Участки линейности, $n = 2$, $L = 3$, $H = 4$:



Теорема. Разнообразие семейства A^n_{LH} растёт как $O(H^{nL})$.

M.Raghu et al. On the Expressive Power of Deep Neural Networks, 2016.

Избыточная параметризация может ускорять сходимость

Рассмотрим t-й шаг SGD: $\mathscr{L}(x_i w) \to \min\limits_w$, $x_i, w \in \mathbb{R}^n$, $i \equiv i(t)$:

$$w^{t+1} := w^t - \eta\, x_i \mathscr{L}'(x_i w^t)$$

Пример избыточной параметризации: $\mathscr{L}(x_i w_1 v) \to \min\limits_{w_1, v}$, $v \in \mathbb{R}$:

$$w_1^{t+1} := w_1^t - \eta\, x_i v^t \mathscr{L}'(x_i w_1^t v^t)$$
$$v^{t+1} := v^t - \eta\, (x_i w_1^t)\, \mathscr{L}'(x_i w_1^t v^t)$$

Рекуррентная формула для $w^t = w_1^t v^t$:

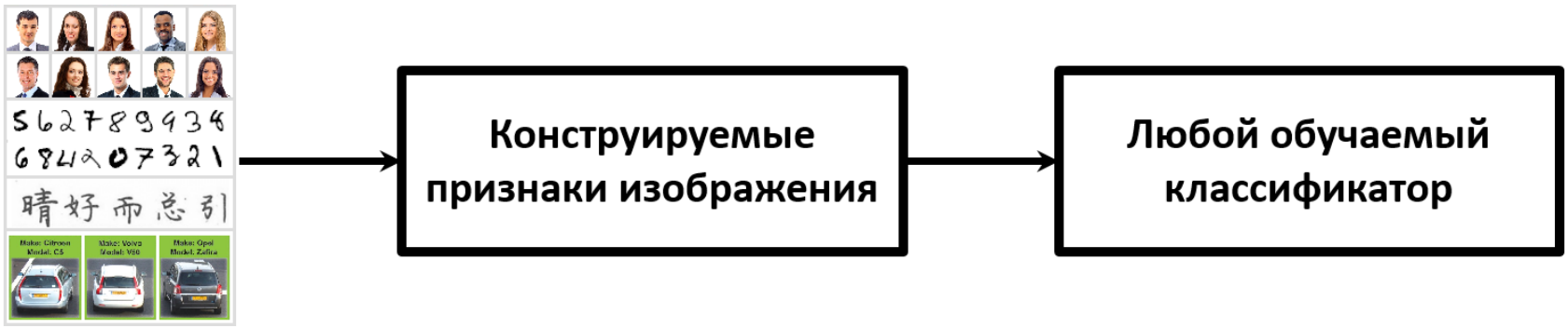
$$w^{t+1} := w_1^{t+1} v^{t+1} = w^t - \eta^t x_i \mathscr{L}'(x_i w^t) - \sum_{\tau=1}^{t-1} \eta^{t,\tau} x_{i(\tau)} \mathscr{L}'(x_{i(\tau)} w^\tau)$$

Это (неожиданно!) метод Momentum с адаптивным шагом η^t и адаптивными коэффициентами сглаживания $\eta^{t,\tau}$.

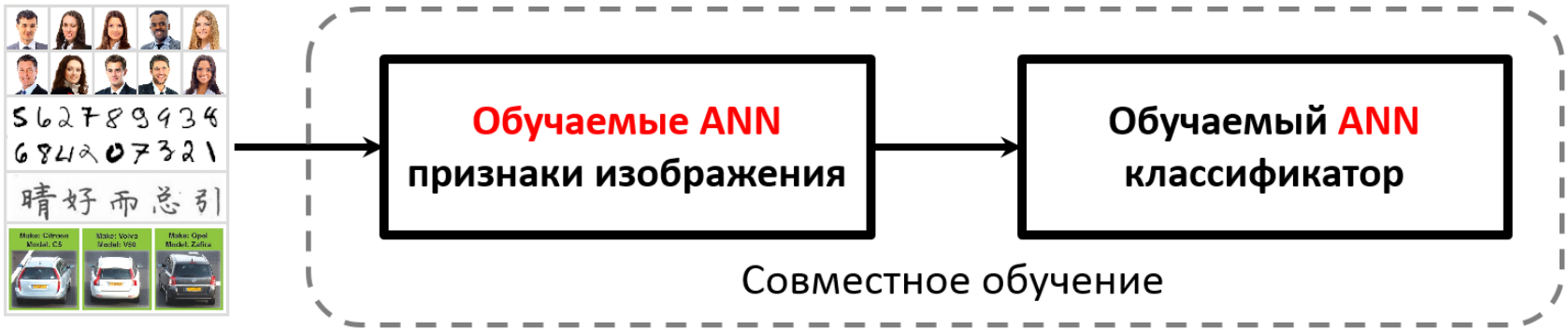
Sanjeev Arora, Nadav Cohen, Elad Hazan. On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization. 2018

Генерация признаков для распознавания изображений

Классический подход к распознаванию изображений:



Современный подход DNN — end-to-end deep learning:



Yann LeCun et al. Learning algorithms for classification: A comparison on handwritten digit recognition. 1995

Резюме

- Нейронная сеть = суперпозиция линейных нейронов с нелинейными функциями активации
- BackProp = быстрое дифференцирование суперпозиций. Позволяет обучать сети практически любой архитектуры.
- Некоторые меры по улучшению сходимости и качества:
  - адаптивный градиентный шаг: Momentum, NAG, Adam и др.
  - функции активации типа ReLU
  - регуляризации: L_2, L_1, Dropout, Skip Connection и др.
  - различные виды нормализации: batch normalization и др.
  - инициализация нейронов как отдельных алгоритмов
- Обоснования глубоких нейронных сетей
  - нейрофизиология: в естественной нейросети 10–20 слоёв
  - глубина важнее ширины для универсальной аппроксимации
  - автоматическое выделение признаков (feature extraction)
  - обучаемая векторизация сложноструктурированных данных