Мат.модели машинного обучения: 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
«Машинное обучение (курс лекций, К.В.Воронцов)»

МФТИ • 19 сентября 2025

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 $a(x, w) = \text{sign}\ g(x, w)$

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

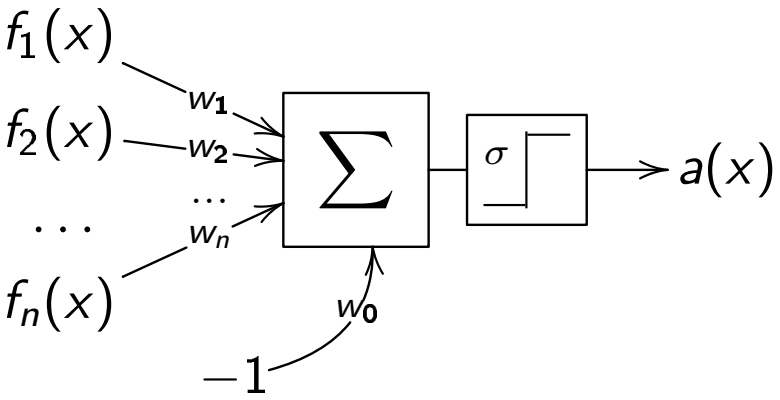
где L невозрастающая: $L(M) = \ln(1 + e^{-M})$, $(1 - M)_+$, и др.

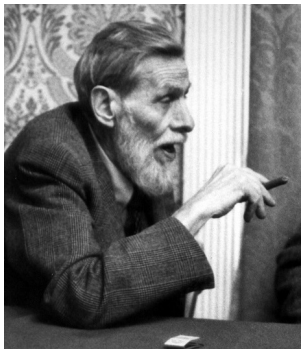
Искусственный нейрон — линейная модель классификации

Линейная модель нейрона (1943):

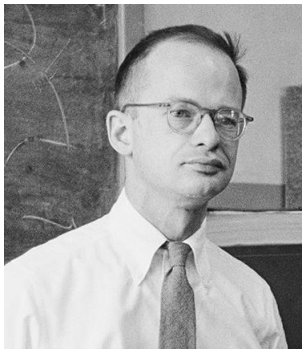
$$a(x,w) = \sigma\left(\sum_{j=1}^{n} w_j f_j(x) - w_0\right)$$

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

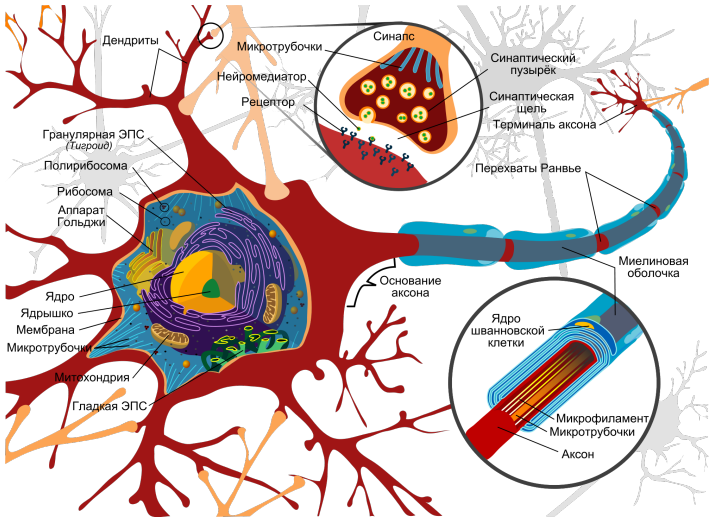




Уоррен МакКаллок (1898–1969)

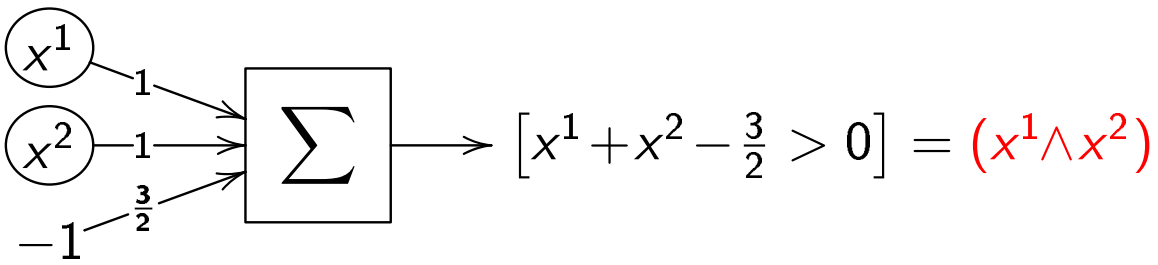


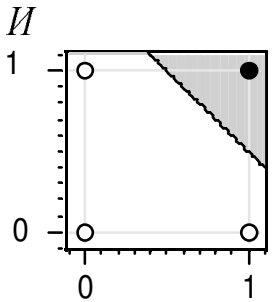
Вальтер Питтс (1923–1969)

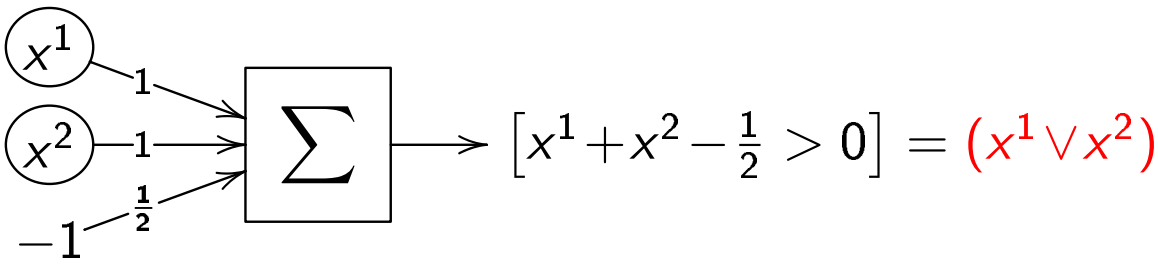


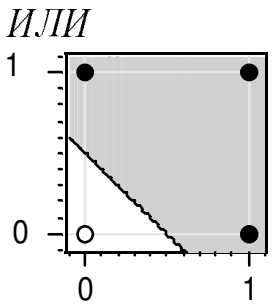
Нейронная реализация булевых функций

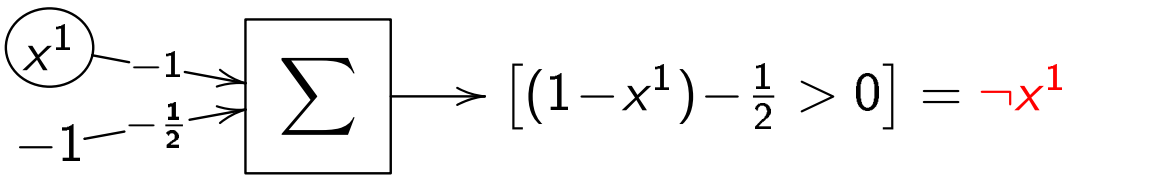
Функции И, ИЛИ, НЕ бинарных признаков x^1, x^2:

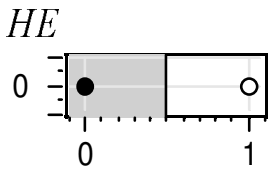












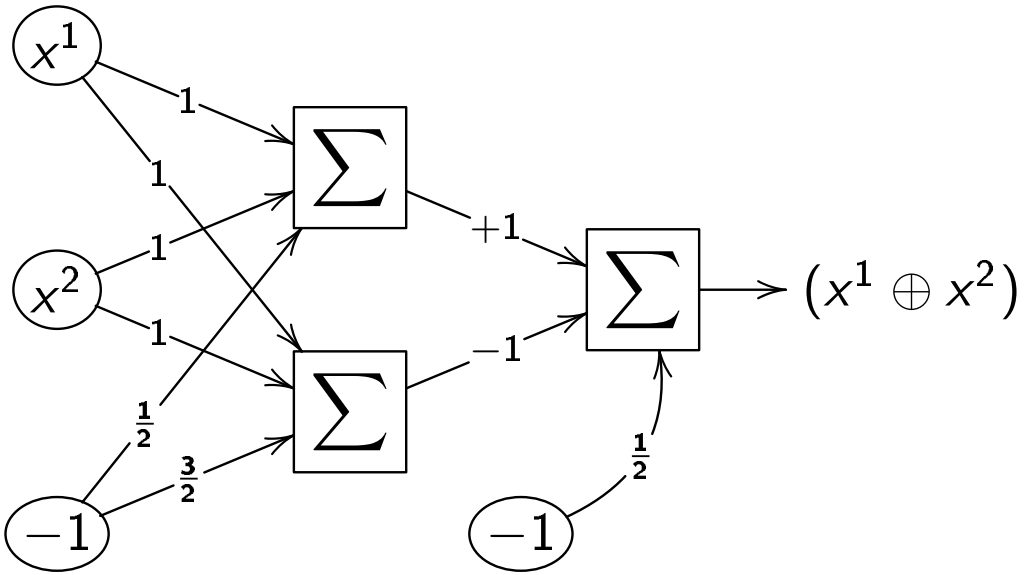
Дизъюнктивная нормальная форма $\Longleftrightarrow$ двухслойная нейросеть

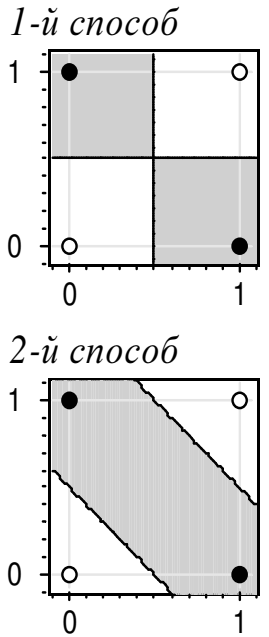
Ограниченность линейных моделей: функция XOR

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)
Обучение — метод коррекции ошибки
Архитектура — двухслойная сеть

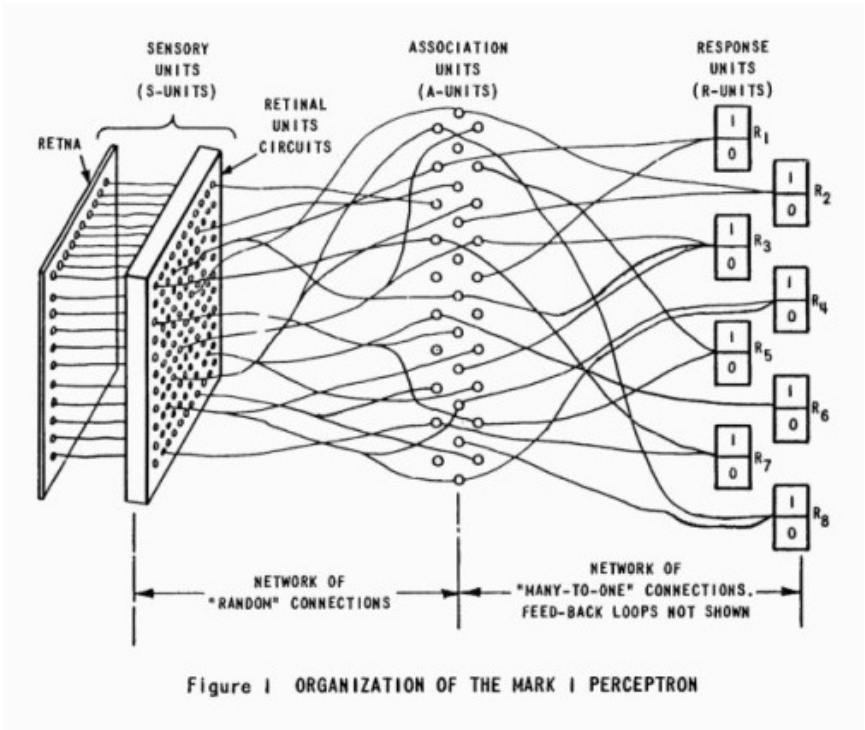
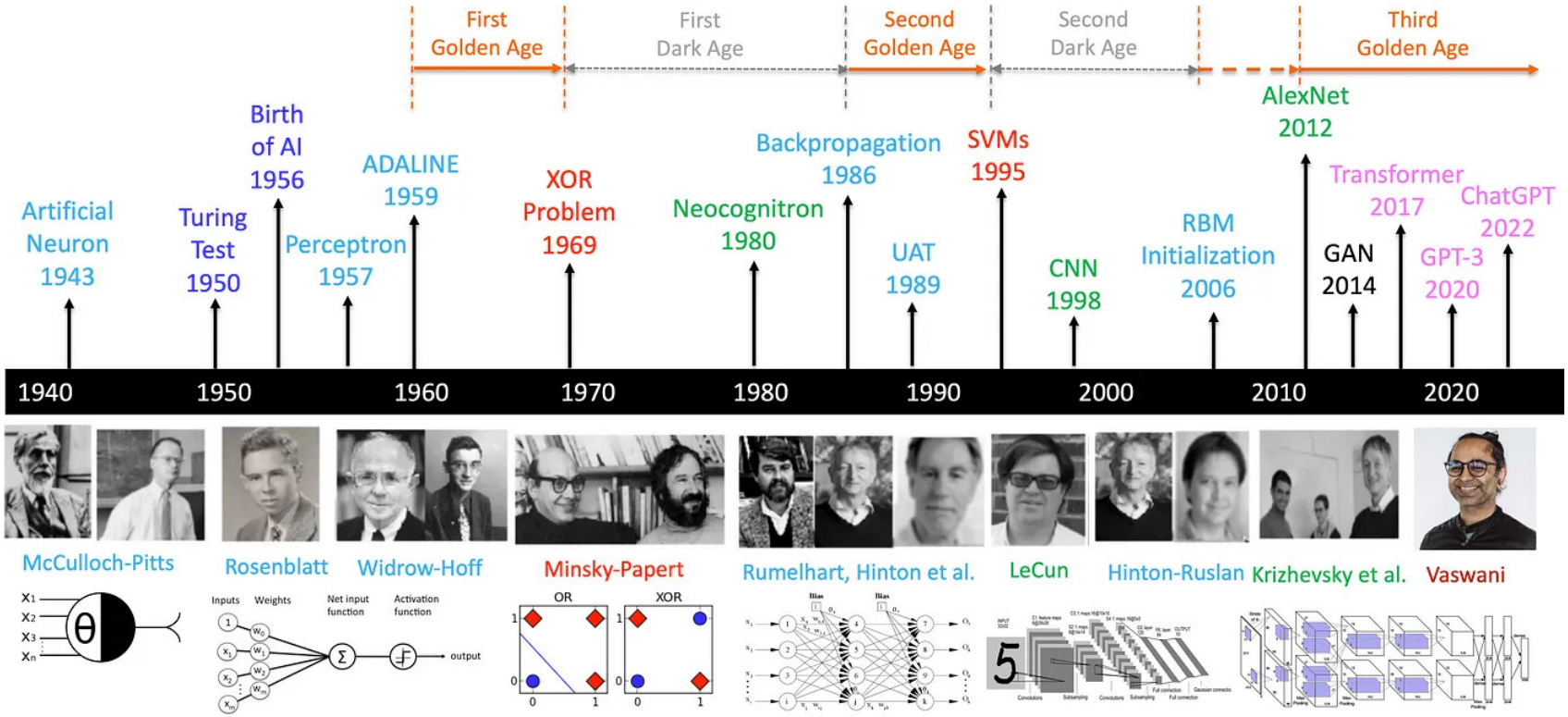


Figure 1 ORGANIZATION OF THE MARK I PERCEPTRON

Фрэнк Розенблатт (1928–1971)

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

Основные вехи развития нейронных сетей (AI winters)



Минский М., Пайперт С. Персептроны. 1971 (1969)
Галушкин А. И. Синтез многослойных систем распознавания образов. 1974
Ивáхненко А. Г., Лапа В. Г. Кибернетические предсказывающие устройства. 1965
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

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.

Имеет ли теорема Колмогорова отношение к нейросетям?

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.

Обобщение: полносвязная нейронная сеть с L слоями

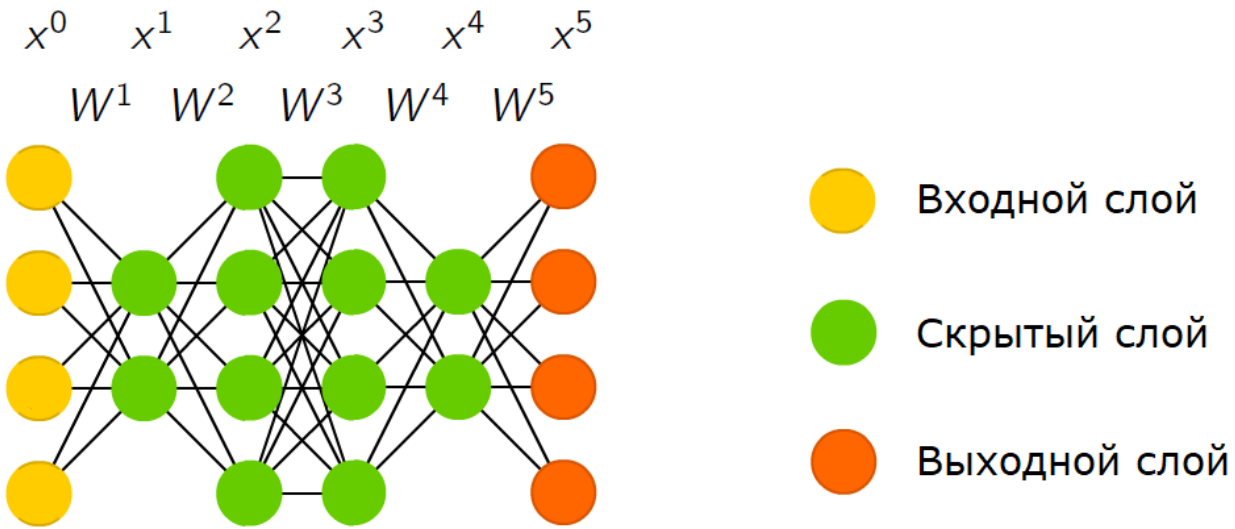
Архитектура сети: H_l — число нейронов в l-м слое, $l = 1, \ldots, L$

$x^0 = x = \big(f_j(x)\big)_{j=0}^{n}$ — 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, $x_0^l = -1$

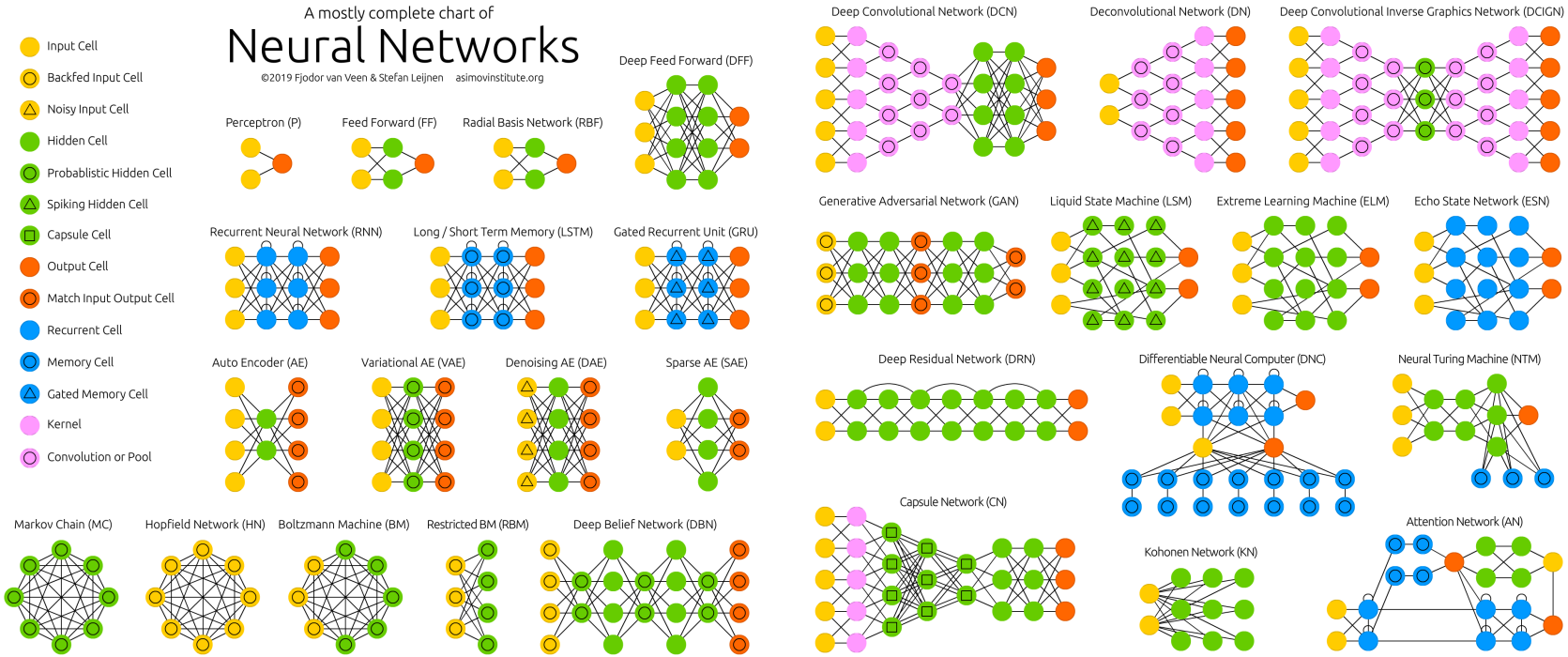
$x^L = a(x) = \big(a_m(x)\big)_{m=1}^{M}$ — выходной вектор сети, $H_L = M$

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



Много (M) выходов нужно для многоклассовой классификации

Зоопарк архитектур нейронных сетей



- *Архитектура сети* — структура слоёв и связей между ними, позволяющая наделять сеть нужными свойствами
- Все архитектуры обучаются методом BackProp (ну, почти), в неполносвязной сети отсутствующие связи игнорируются

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:

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

Если сеть не полносвязная, то $w_{kh}^l = 0$, $\frac{\partial}{\partial w_{kh}^l} = 0$ для части k, h

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, $l = L, \ldots, 2$:

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

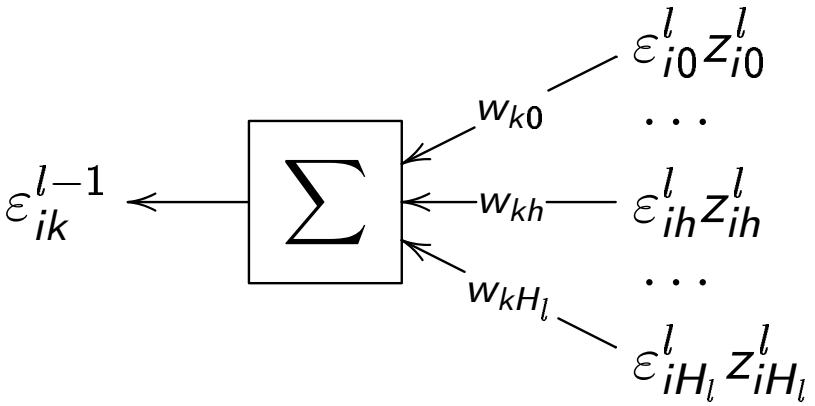
— формально назовём это *ошибкой скрытого слоя*.

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

Изменения минимальны, 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:

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

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

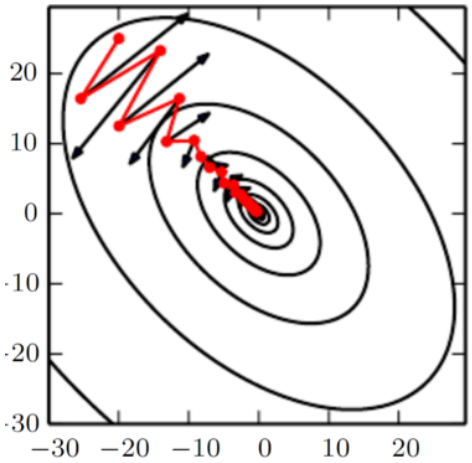
Недостатки — все те же, свойственные SG:

- 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

Напоминание: метод накопления инерции (momentum)

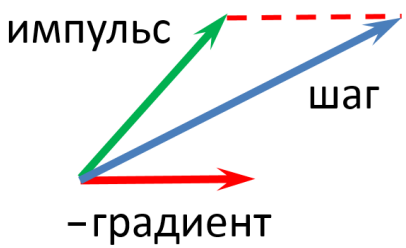
Momentum — экспоненциальное скользящее среднее градиента по $\approx \frac{1}{1-\gamma}$ последним итерациям [Б.Т.Поляк, 1964]:

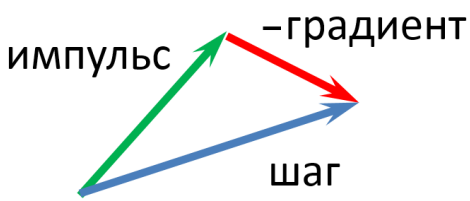
$$v := \gamma v + (1-\gamma)\nabla \mathscr{L}(w, x_i)$$
$$w := w - \eta v$$



NAG (Nesterov's accelerated gradient) — ускоренный стохастический градиент с инерцией [Ю.Е.Нестеров, 1983]:

$$v := \gamma v + (1-\gamma)\nabla \mathscr{L}(w - \eta\gamma v, x_i)$$
$$w := w - \eta v$$





Адаптивные градиенты

RMSProp (running mean square) — выравнивание скоростей изменения весов скользящим средним по $\approx \frac{1}{1-\alpha}$ итерациям, ускоряет обучение по весам, которые пока мало изменялись:

$$G := \alpha G + (1-\alpha)\,\nabla\mathscr{L}(w, x_i) \odot \nabla\mathscr{L}(w, x_i)$$

$$w := w - \eta \nabla\mathscr{L}(w, x_i) \oslash \left(\sqrt{G} + \varepsilon\right)$$

где $\odot$ и $\oslash$ — покоординатное умножение и деление векторов.

AdaDelta (adaptive learning rate) — двойная нормировка приращений весов, после которой можно брать $\eta = 1$:

$$G := \alpha G + (1-\alpha)\,\nabla\mathscr{L}(w, x_i) \odot \nabla\mathscr{L}(w, x_i)$$

$$\delta := \nabla\mathscr{L}(w, x_i) \odot \frac{\sqrt{\Delta}+\varepsilon}{\sqrt{G}+\varepsilon}$$

$$\Delta := \alpha\Delta + (1-\alpha)\,\delta \odot \delta$$

$$w := w - \eta\delta$$

Комбинированные градиентные методы

Adam (adaptive momentum) = инерция + RMSProp:

$$v := \gamma v + (1-\gamma)\nabla\mathscr{L}(w, x_i) \qquad \hat{v} := v(1-\gamma^k)^{-1}$$
$$G := \alpha G + (1-\alpha)\nabla\mathscr{L}(w, x_i) \odot \nabla\mathscr{L}(w, x_i) \qquad \hat{G} := G(1-\alpha^k)^{-1}$$
$$w := w - \eta\hat{v} \oslash \left(\sqrt{\hat{G}} + \varepsilon\right)$$

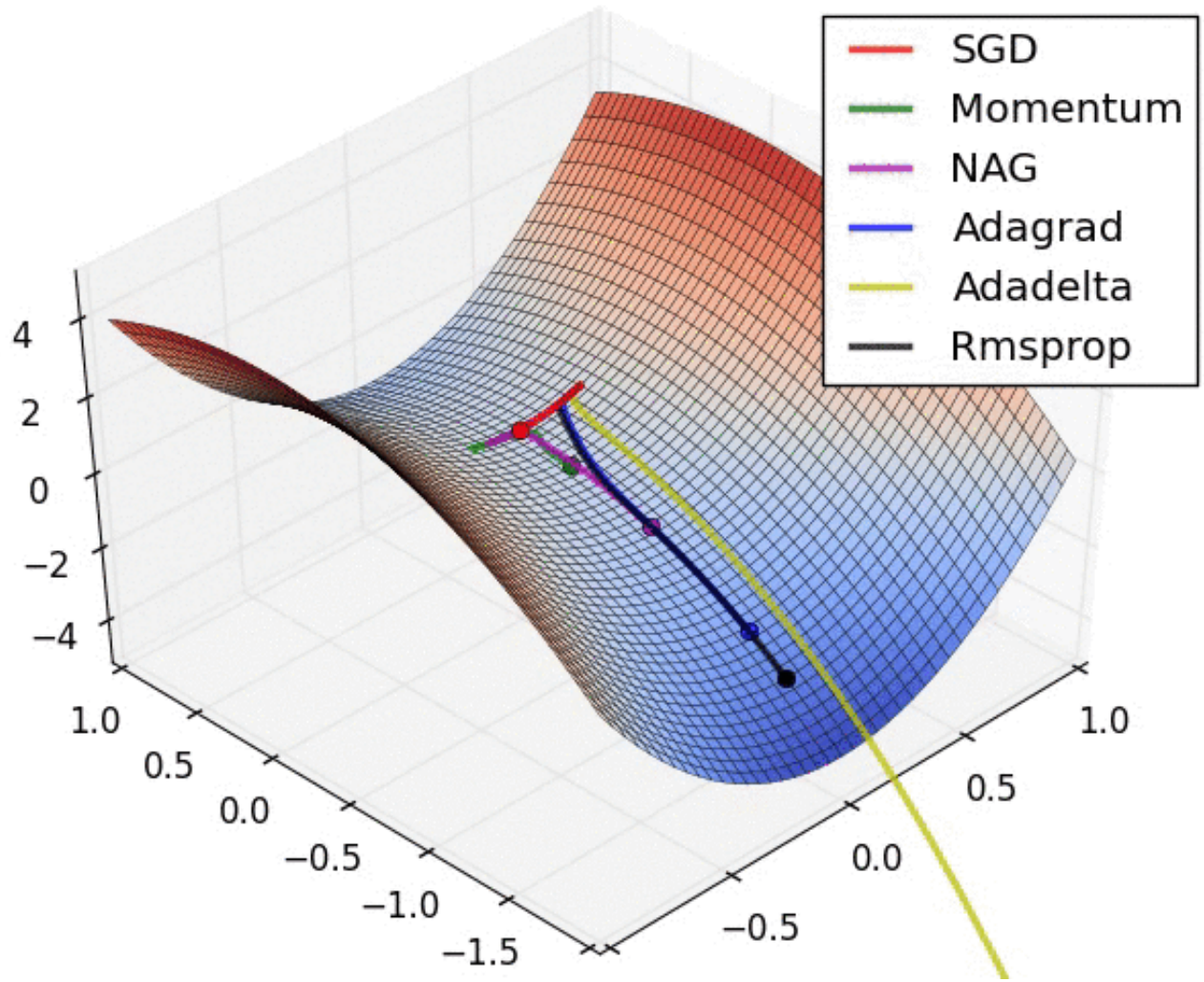
Калибровка $\hat{v}$, $\hat{G}$ увеличивает v, G на первых итерациях,
где k — номер итерации; $\gamma = 0.9$, $\alpha = 0.999$, $\varepsilon = 10^{-8}$

Nadam (Nesterov-accelerated adaptive momentum):
те же формулы для v, $\hat{v}$, G, $\hat{G}$,

$$w := w - \eta\left(\gamma\hat{v} + \tfrac{1-\gamma}{1-\gamma^k}\nabla\mathscr{L}(w - \eta\gamma\hat{v}, x_i)\right) \oslash \left(\sqrt{\hat{G}} + \varepsilon\right)$$

Timothy Dozat. Incorporating Nesterov Momentum into Adam. ICLR-2016.

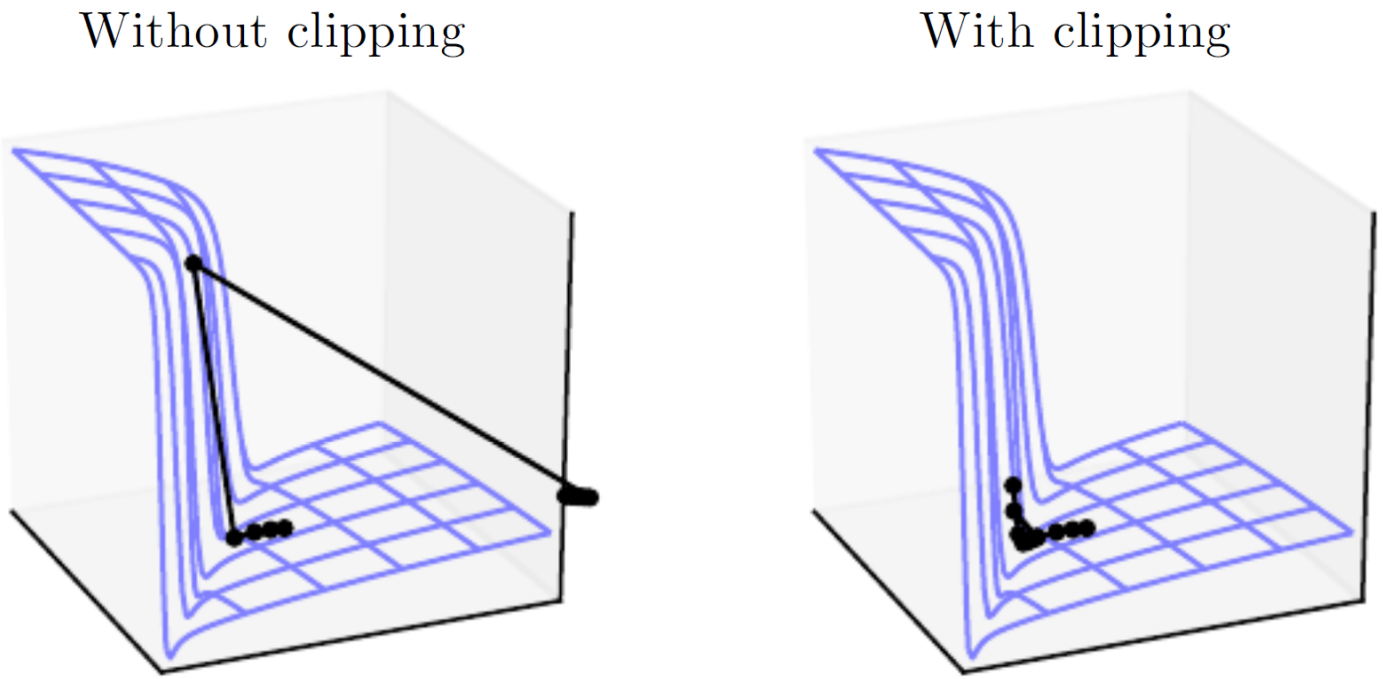
Сравнение сходимости методов



Alec Radford's animation:
`https://www.denizyuret.com/2015/03/alec-radfords-animations-for.html`

Проблема взрыва градиента и эвристика gradient clipping

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



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

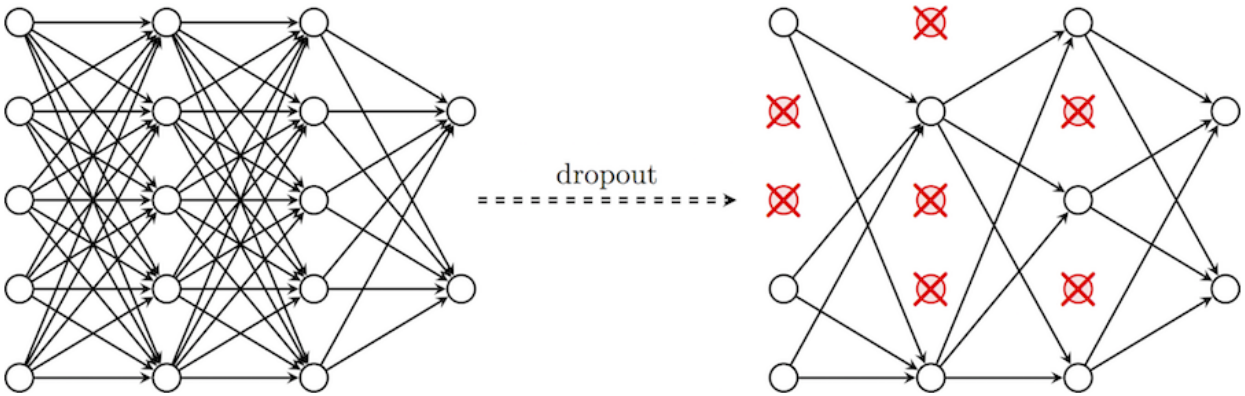
Метод случайных отключений нейронов (Dropout)

Этап обучения: делая градиентный шаг $\mathscr{L}_i(w) \to \min\limits_w$,
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:

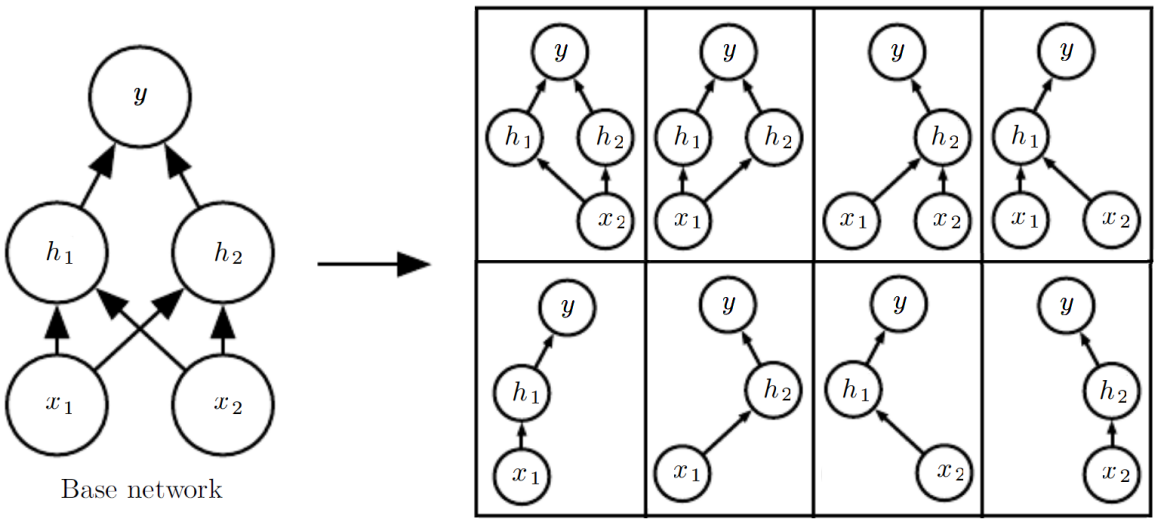
$$x_{hi}^l = (1 - p_l)\sigma_h^l\Big(\sum_k w_{kh}^l x_{ki}^{l-1}\Big)$$



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:

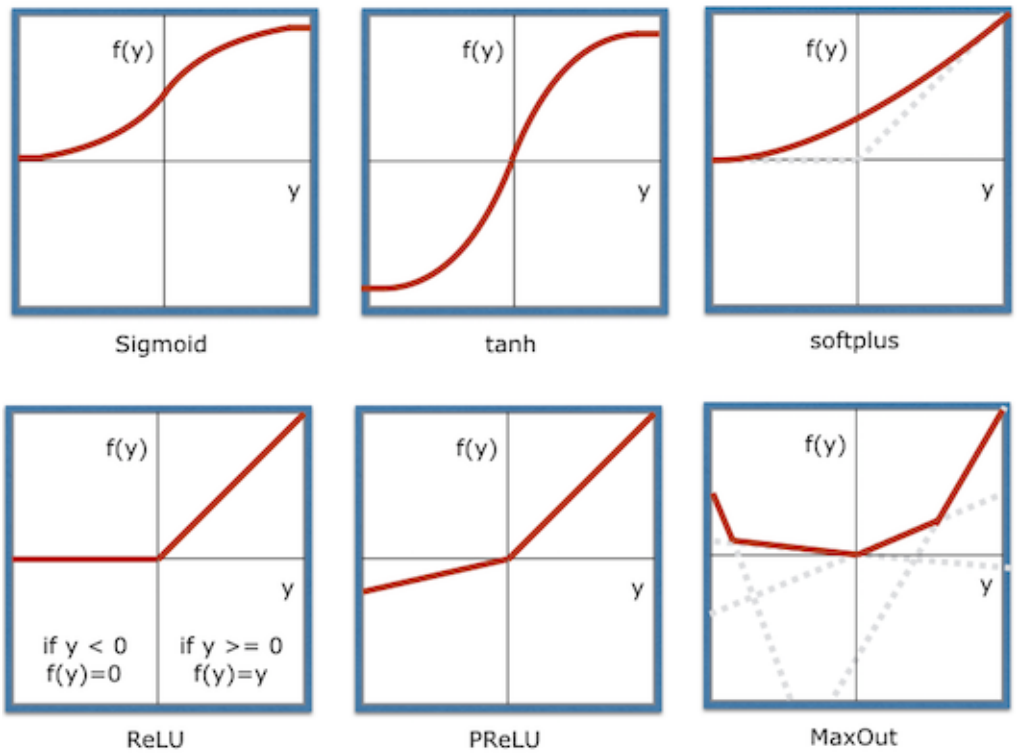
$$w := w(1-\eta\lambda) - \eta\,\frac{1}{1-p_\ell}\xi_h^\ell\,\nabla\mathscr{L}(w, x_i)$$

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)

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



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.

Эвристики для начального приближения

1. Выравнивание дисперсий выходов в разных слоях:
$$w_{kh} := \text{uniform}\left(-\frac{1}{\sqrt{H_l}}, \frac{1}{\sqrt{H_l}}\right)$$
2. Выравнивание дисперсий градиентов в разных слоях:
$$w_{kh} := \text{uniform}\left(-\frac{6}{\sqrt{H_{l-1}+H_l}}, \frac{6}{\sqrt{H_{l-1}+H_l}}\right),$$
где H_{l-1}, H_l — число нейронов в предыдущем и текущем слое
3. Послойное обучение нейронов как линейных моделей:
   - либо по случайной подвыборке $X' \subseteq X^\ell$;
   - либо по случайному подмножеству входов;
   - либо из различных случайных начальных приближений;

   тем самым обеспечивается *различность* нейронов.
4. Инициализация весами предобученной модели
5. Инициализация случайным ортогональным базисом

Прореживание сети (OBD — Optimal Brain Damage)

Пусть w — локальный минимум $Q(w)$, тогда $Q(w)$ можно аппроксимировать квадратичной формой:

$$Q(w+\delta) = Q(w) + \tfrac{1}{2}\delta^{\mathsf{T}} Q''(w)\delta + o(\|\delta\|^2),$$

где $Q''(w) = \left(\frac{\partial^2 Q(w)}{\partial w_{kh}\partial w_{k'h'}}\right)$ — гессиан, размера $\dim^2(w)$.

Эвристика. Пусть гессиан $Q''(w)$ диагонален, тогда

$$\delta^{\mathsf{T}} Q''(w)\delta = \sum_{l=1}^{L}\sum_{k=0}^{H_{l-1}}\sum_{h=1}^{H_l} \delta_{kh}^2 \frac{\partial^2 Q(w)}{\partial w_{kh}^2}$$

Хотим обнулить вес: $w_{kh} + \delta_{kh} = 0$. Как изменится $Q(w)$?

Определение. *Значимость* (salience) веса w_{kh} — это изменение функционала $Q(w)$ при его обнулении: $S_{kh} = w_{kh}^2 \frac{\partial^2 Q(w)}{\partial w_{kh}^2}$.

Yann LeCun, John Denker, Sara Solla. Optimal Brain Damage. 1989

Прореживание сети (OBD — Optimal Brain Damage)

1. В BackProp вычислять вторые производные $\frac{\partial^2 Q}{\partial w_{kh}^2}$.
2. Если процесс минимизации $Q(w)$ пришёл в минимум,то
   - упорядочить на каждом уровне веса по убыванию S_{kh};
   - удалить N связей с наименьшей значимостью;
   - снова запустить BackProp.
3. Если $Q(w, X^\ell)$ или $Q(w, X^k)$ существенно ухудшился, то вернуть последние удалённые связи и выйти.

Отбор признаков с помощью OBD — аналогично.

Суммарная значимость признака: $S_j = \sum_{h=1}^{H_1} S_{jh}$.

Эмпирический опыт: результат постепенного прореживания обычно лучше, чем BackProp изначально прореженной сети.

Резюме

- Нейрон = линейная классификация или регрессия.
- Нейронная сеть = суперпозиция нейронов с нелинейной функцией активации. Теоретически двух-трёх слоёв достаточно для решения очень широкого класса задач.
- Глубокие нейросети автоматизируют выделение признаков из сложно структурированных данных (feature extraction)
- BackProp = быстрое дифференцирование суперпозиций. Позволяет обучать сети практически любой архитектуры.
- Некоторые меры по улучшению сходимости и качества:
  - адаптивный градиентный шаг
  - функции активации типа ReLU
  - регуляризация и DropOut
  - пакетная нормализация (batch normalization)
  - инициализация нейронов как отдельных линейных моделей