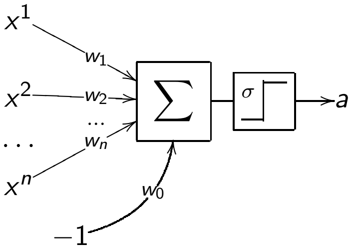
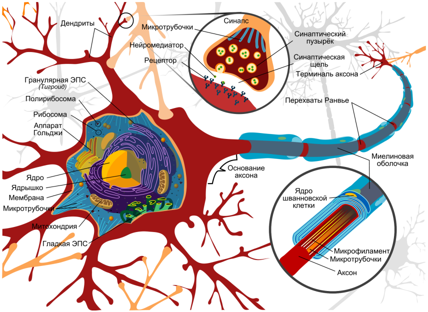
Линейная модель нейрона МакКаллока-Питтса

$f_j\colon X \to \mathbb{R}$, $j = 1, \ldots, n$ — числовые признаки, $x^j = f_j(x)$;

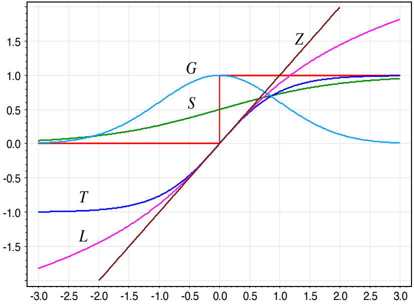
$$a(x, w) = \sigma\big(\langle w, x\rangle\big) = \sigma\left(\sum_{j=1}^{n} w_j f_j(x) - w_0\right),$$

где $w_0, w_1, \ldots, w_n \in \mathbb{R}$ — веса признаков;
$\sigma(s)$ — функция активации (в частности, sign).





Часто используемые функции активации $\sigma(z)$



$\theta(z) = [z \geqslant 0]$	— пороговая функция Хевисайда;
$\sigma(z) = (1 + e^{-z})^{-1}$	— сигмоидная функция (S);
$\mathrm{th}(z) = 2\sigma(2z) - 1$	— гиперболический тангенс (T);
$\ln(z + \sqrt{z^2 + 1})$	— логарифмическая функция (L);
$\exp(-z^2/2)$	— гауссовская функция (G);
z	— линейная функция (Z);

Задача классификации: $Y = \{\pm 1\}$, $a(x, w) = \operatorname{sign}\langle w, x_i \rangle$;

$$Q(w; X^\ell) = \sum_{i=1}^{\ell} \mathscr{L}\bigl(\underbrace{\langle w, x_i \rangle y_i}_{M_i(w)}\bigr) \to \min_w;$$

Задача регрессии: $Y = \mathbb{R}$, $a(x, w) = \sigma(\langle w, x_i \rangle)$;

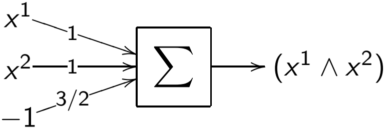
$$Q(w; X^\ell) = \sum_{i=1}^{\ell} \bigl(\sigma(\langle w, x_i \rangle) - y_i\bigr)^2 \to \min_w;$$

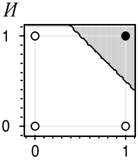
Насколько богатый класс функций реализуется нейроном?
А сетью (суперпозицией) нейронов?

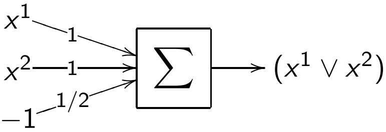
Нейронная реализация логических функций

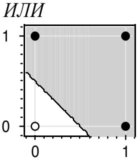
Функции И, ИЛИ, НЕ от бинарных переменных x^1 и x^2:

$$x^1 \wedge x^2 = \left[x^1 + x^2 - \tfrac{3}{2} > 0\right];$$
$$x^1 \vee x^2 = \left[x^1 + x^2 - \tfrac{1}{2} > 0\right];$$
$$\neg x^1 = \left[-x^1 + \tfrac{1}{2} > 0\right];$$





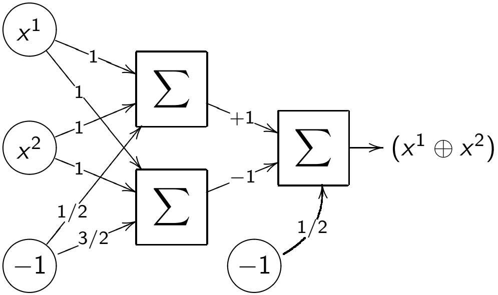


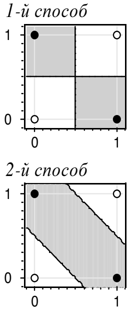


Логическая функция XOR (исключающее ИЛИ)

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

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





Любую ли функцию можно представить нейросетью?

- Двухслойная сеть в $\{0,1\}^n$ позволяет реализовать произвольную булеву функцию (ДНФ).
- Двухслойная сеть в $\mathbb{R}^n$ позволяет отделить произвольный выпуклый многогранник.
- Трёхслойная сеть $\mathbb{R}^n$ позволяет отделить произвольную многогранную область, не обязательно выпуклую, и даже не обязательно связную.
- С помощью линейных операций и одной нелинейной *функции активации* φ можно приблизить любую непрерывную функцию с любой желаемой точностью.

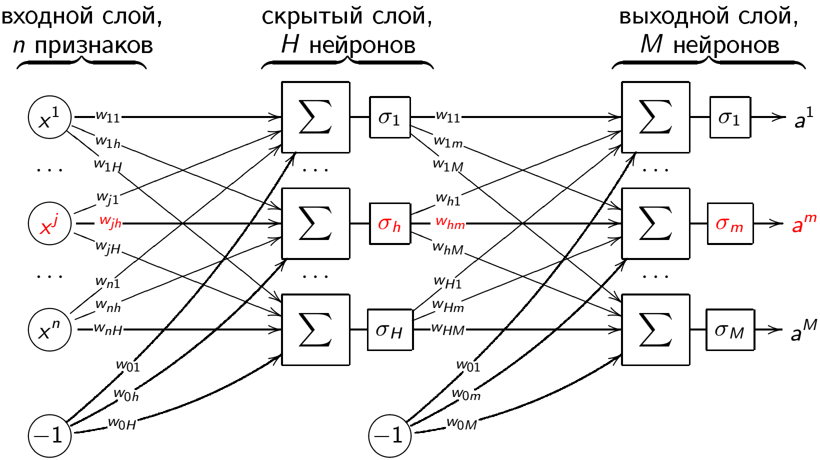
Практические рекомендации:

- Двух-трёх слоёв обычно достаточно.
- Можно достраивать нейроны в произвольных местах сети по необходимости, вообще не заботясь о числе слоёв.

Многослойная нейронная сеть

Пусть для общности $Y = \mathbb{R}^M$, для простоты слоёв только два:

$$a^m(x, w) = \sigma_m\left(\sum_{h=0}^{H} w_{hm}\sigma_h\left(\sum_{j=0}^{n} w_{jh} f_j(x_i)\right)\right).$$



Напоминание: Алгоритм SG (Stochastic Gradient)

Задача минимизации суммарных потерь:

$$Q(w) := \sum_{i=1}^{\ell} \mathscr{L}_i(w) \to \min_{w}.$$

Вход: выборка X^ℓ; темп обучения η; параметр λ;

Выход: веса $w \equiv \bigl(w_{jh}, w_{hm}\bigr) \in \mathbb{R}^{H(n+M+1)+M}$;

1: инициализировать веса w и текущую оценку $Q(w)$;

2: **повторять**

3: выбрать объект x_i из X^ℓ (например, случайно);

4: вычислить потерю $\mathscr{L}_i := \mathscr{L}_i(w)$;

5: градиентный шаг: $w := w - \eta \nabla \mathscr{L}_i(w)$;

6: оценить значение функционала: $Q := (1 - \lambda)Q + \lambda \mathscr{L}_i$;

7: **пока** значение Q и/или веса w не стабилизируются;

Задача дифференцирования суперпозиции функций

Выходные значения сети $a^m(x_i)$, $m = 1, \dots, M$ на объекте x_i:

$$a^m(x_i) = \sigma_m\left(\sum_{h=0}^{H} w_{hm} u^h(x_i)\right); \qquad u^h(x_i) = \sigma_h\left(\sum_{j=0}^{J} w_{jh} f_j(x_i)\right).$$

Пусть для конкретности $\mathscr{L}_i(w)$ — средний квадрат ошибки:

$$\mathscr{L}_i(w) = \frac{1}{2}\sum_{m=1}^{M}\left(a^m(x_i) - y_i^m\right)^2.$$

Промежуточная задача: найти частные производные

$$\frac{\partial \mathscr{L}_i(w)}{\partial a^m}; \qquad \frac{\partial \mathscr{L}_i(w)}{\partial u^h}.$$

Быстрое вычисление градиента

Промежуточная задача: частные производные

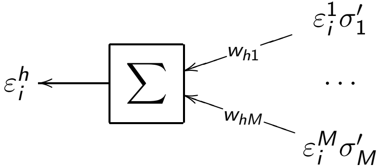
$$\frac{\partial \mathscr{L}_i(w)}{\partial a^m} = a^m(x_i) - y_i^m = \varepsilon_i^m$$

— это ошибка на выходном слое;

$$\frac{\partial \mathscr{L}_i(w)}{\partial u^h} = \sum_{m=1}^{M} \bigl(a^m(x_i) - y_i^m\bigr)\sigma'_m w_{hm} = \sum_{m=1}^{M} \varepsilon_i^m \sigma'_m w_{hm} = \varepsilon_i^h$$

— назовём это *ошибкой на скрытом слое*.

Похоже, что ε_i^h вычисляется по ε_i^m путём его пропускания через сеть в обратном направлении:



Теперь, имея частные производные $\mathscr{L}_i(w)$ по a^m и u^h, легко выписать градиент $\mathscr{L}_i(w)$ по весам w:

$$\frac{\partial \mathscr{L}_i(w)}{\partial w_{hm}} = \frac{\partial \mathscr{L}_i(w)}{\partial a^m} \frac{\partial a^m}{\partial w_{hm}} = \varepsilon_i^m \sigma'_m u^h(x_i), \quad m = 1..M, \quad h = 0..H;$$

$$\frac{\partial \mathscr{L}_i(w)}{\partial w_{jh}} = \frac{\partial \mathscr{L}_i(w)}{\partial u^h} \frac{\partial u^h}{\partial w_{jh}} = \varepsilon_i^h \sigma'_h f_j(x_i), \quad h = 1..H, \quad j = 0..n;$$

Алгоритм обратного распространения ошибки BackProp:

Вход: $X^\ell = (x_i, y_i)_{i=1}^\ell \subset \mathbb{R}^n \times \mathbb{R}^M$; параметры H, λ, η;
Выход: синаптические веса w_{jh}, w_{hm};

1: ...

Алгоритм BackProp

инициализировать веса w_{jh}, w_{hm};
повторять
  выбрать объект x_i из X^ℓ (например, случайно);
  прямой ход:
  $u_i^h := \sigma_h\bigl(\sum_{j=0}^J w_{jh} x_i^j\bigr),\ \ h = 1..H;$
  $a_i^m := \sigma_m\bigl(\sum_{h=0}^H w_{hm} u_i^h\bigr),\ \ \varepsilon_i^m := a_i^m - y_i^m,\ \ m = 1..M;$
  $\mathscr{L}_i := \sum\limits_{m=1}^{M} (\varepsilon_i^m)^2;$
  обратный ход: $\varepsilon_i^h := \sum\limits_{m=1}^{M} \varepsilon_i^m \sigma'_m w_{hm},\ \ h = 1..H;$
  градиентный шаг:
  $w_{hm} := w_{hm} - \eta \varepsilon_i^m \sigma'_m u_i^h,\ \ h = 0..H,\ \ m = 1..M;$
  $w_{jh} := w_{jh} - \eta \varepsilon_i^h \sigma'_h x_i^j,\ \ j = 0..n,\ \ h = 1..H;$
  $Q := (1-\lambda) Q + \lambda \mathscr{L}_i;$
пока Q не стабилизируется;

Алгоритм BackProp: преимущества и недостатки

Преимущества:

- быстрое вычисление градиента;
- метод легко обобщается на любые σ, $\mathscr{L}$;
- возможно динамическое (потоковое) обучение;
- на сверхбольших выборках не обязательно брать все x_i;
- возможность распараллеливания;

Недостатки — все те же, свойственные SG:

- возможна медленная сходимость;
- застревание в локальных минимумах;
- проблема «паралича сети» (горизонтальные асимптоты σ);
- проблема переобучения;
- подбор комплекса эвристик является искусством;

Стандартные эвристики для метода стохастического градиента

Применимы все те же эвристики, что и в обычном SG:

- инициализация весов;
- порядок предъявления объектов;
- оптимизация величины градиентного шага;
- регуляризация (сокращение весов);

Кроме того, появляются новые проблемы:

- выбор функций активации в каждом нейроне;
- выбор числа слоёв и числа нейронов;
- выбор значимых связей;

Ускорение сходимости

1. Начальное приближение — послойное обучение сети. Нейроны настраиваются как отдельные линейные алгоритмы

- либо по случайной подвыборке $X' \subseteq X^\ell$;
- либо по случайному подмножеству входов;
- либо из различных случайных начальных приближений;

тем самым обеспечивается *различность* нейронов.

2. Выбивание из локальных минимумов (jogging of weights).

3. Адаптивный градиентный шаг (метод скорейшего спуска).

4. Метод сопряжённых градиентов и chunking — разбиение суммы $Q(w) = \sum\limits_{i=1}^{\ell} \mathscr{L}_i(w)$ по подмножествам объектов (chunks).

Диагональный метод Левенберга-Марквардта

Метод Ньютона-Рафсона (второго порядка):

$$w := w - \eta\bigl(\mathscr{L}_i''(w)\bigr)^{-1}\mathscr{L}_i'(w),$$

где $\bigl(\mathscr{L}_i''(w)\bigr) = \bigl(\frac{\partial^2\mathscr{L}_i(w)}{\partial w_{jh}\partial w_{j'h'}}\bigr)$ — гессиан, размера $\bigl(H(n{+}M{+}1){+}M\bigr)^2$.

Эвристика. Считаем, что гессиан диагонален:

$$w_{jh} := w_{jh} - \eta\left(\frac{\partial^2\mathscr{L}_i(w)}{\partial w_{jh}^2} + \mu\right)^{-1}\frac{\partial\mathscr{L}_i(w)}{\partial w_{jh}},$$

η — темп обучения,
μ — параметр, предотвращающий обнуление знаменателя.

Отношение η/μ есть темп обучения на ровных участках функционала $\mathscr{L}_i(w)$, где вторая производная обнуляется.

Динамическое наращивание сети

1. обучение при заведомо недостаточном числе нейронов H;
2. после стабилизации $Q(w)$ — добавление нового нейрона и его инициализация путём обучения
   - либо по случайной подвыборке $X' \subseteq X^\ell$;
   - либо по объектам с наибольшими значениями потерь;
   - либо по случайному подмножеству входов;
   - либо из различных случайных начальных приближений;
3. снова итерации BackProp;

Эмпирический опыт: Общее время обучения обычно лишь в 1.5–2 раза больше, чем если бы в сети сразу было нужное количество нейронов. Полезная информация, накопленная сетью, не теряется при добавлении новых нейронов.

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 $Q''(w) = \left(\frac{\partial^2 Q(w)}{\partial w_{jh}\partial w_{j'h'}}\right)$ — гессиан, размера $\bigl(H(n+M+1)+M\bigr)^2$.

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

$$\delta^{\mathsf{T}} Q''(w)\delta = \sum_{j=0}^{n}\sum_{h=1}^{H} \delta_{jh}^2 \frac{\partial^2 Q(w)}{\partial w_{jh}^2} + \sum_{h=0}^{H}\sum_{m=0}^{M} \delta_{hm}^2 \frac{\partial^2 Q(w)}{\partial w_{hm}^2}.$$

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

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

Прореживание сети (OBD — Optimal Brain Damage)

1. В BackProp вычислять вторые производные $\frac{\partial^2 Q}{\partial w_{jh}^2}$, $\frac{\partial^2 Q}{\partial w_{hm}^2}$.
2. Если процесс минимизации $Q(w)$ пришёл в минимум, то
   - упорядочить все веса по убыванию S_{jh};
   - 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: $S_j = \sum_{h=1}^{H} S_{jh}$.

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. Двух-трёх слоёв достаточно для решения очень широкого класса задач.
- Для решения очень сложных задач используются глубокие сети (Deep Learning).
- BackProp = быстрое дифференцирование суперпозиций. Позволяет обучать сети практически любой конфигурации.
- Некоторые меры по улучшению сходимости и качества:
  - регуляризация
  - перетасовка объектов
  - инициализация нейронов как отдельных алгоритмов
  - адаптивный градиентный шаг
  - метод сопряжённых градиентов и chunking
  - динамическое наращивание
  - прореживание (OBD)