Искусственные нейронные сети: градиентные методы оптимизации

К. В. Воронцов
vokov@forecsys.ru

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

МФТИ • 16 сентября 2022

Содержание

Напоминание: линейные модели классификации и регрессии

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

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

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

Задача классификации с двумя классами: $Y = \{\pm 1\}$
$a(x, w) = \operatorname{sign}\langle w, x_i \rangle$ — линейная модель классификации
$\mathscr{L}(M)$ — невозрастающая функция отступа, например,
$\mathscr{L}(M) = \ln(1 + e^{-M})$, $(1 - M)_+$, e^{-M}, $\frac{1}{1+e^M}$, и др.

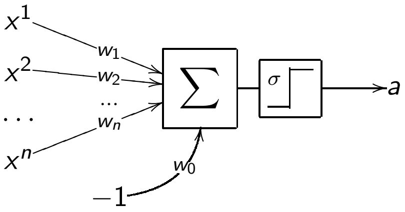
$$Q(w; X^\ell) = \sum_{i=1}^{\ell} \mathscr{L}\bigl(\underbrace{\langle w, x_i \rangle y_i}_{M_i(w)}\bigr) \to \min_w;$$

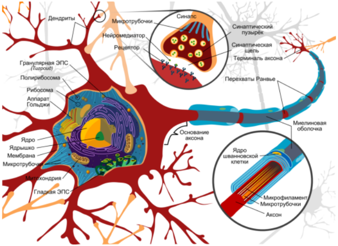
Напоминание: линейная модель нейрона МакКаллока-Питтса

$f_j\colon X \to \mathbb{R},\ \ j = 1, \dots, n$ — числовые признаки;

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

w_j — веса признаков, $\sigma(z)$ — функция активации, $x^j \equiv f_j(x)$



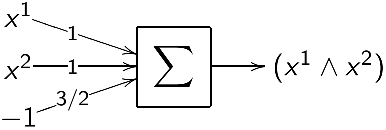


Насколько богатый класс функций реализуется нейроном?
А сетью (суперпозицией) нейронов?

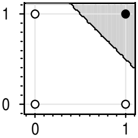
Нейронная реализация логических функций

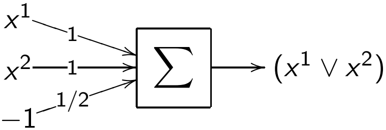
Функции И, ИЛИ, НЕ бинарных переменных (признаков) x^1, x^2:

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

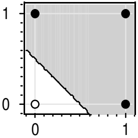


И





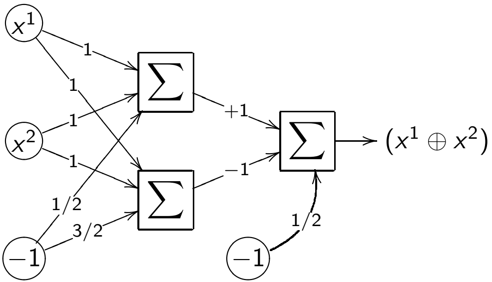
ИЛИ

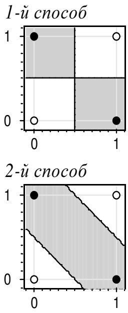


Логическая функция XOR (исключающее ИЛИ)

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* σ можно приблизить любую непрерывную функцию с любой желаемой точностью.

Практические рекомендации:

- Двух-трёх слоёв теоретически достаточно.
- Глубокие сети — это встроенное обучение признаков.

Любую ли функцию можно представить нейросетью?

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

Теорема Цыбенко (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

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

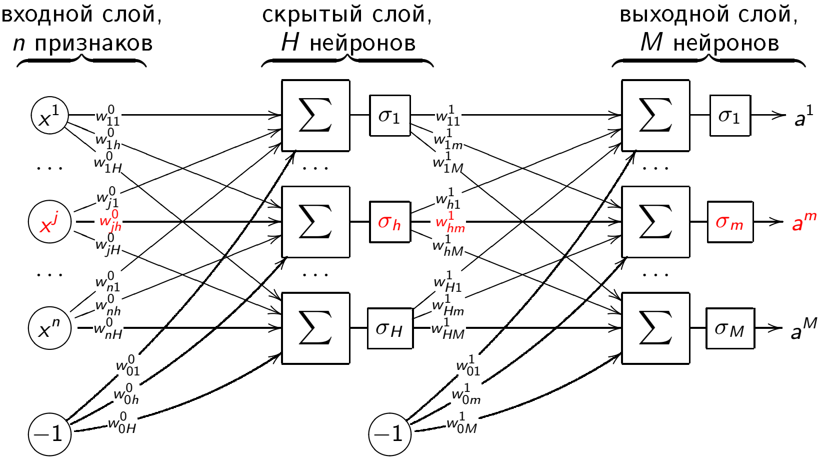
равномерно приближает $f(x)$ с любой точностью ε:

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

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

Двухслойная нейронная сеть с M-мерным выходом

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



Параметры модели $w \equiv \left(w_{jh}^0, w_{hm}^1\right) \in \mathbb{R}^{H(n+1)+M(H+1)}$.

Обобщение: нейронная сеть с заданным числом слоёв

L — число слоёв

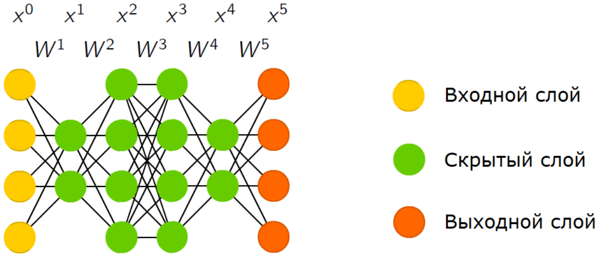
H_l — число нейронов в l-м слое, $l = 1, \dots, L$

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

$x^l = \bigl(x_h^l\bigr)_{h=0}^{H_l}$ — вектор признаков на выходе l-го слоя

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

$W^l = (w_{jh}^l)$ — матрица весов l-го слоя



Напоминание: алгоритм SG (Stochastic Gradient)

Минимизация средних потерь на обучающей выборке:

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

Вход: выборка X^ℓ; темп обучения η; параметр λ;
Выход: вектор весов всех L слоёв $w \equiv \bigl(w^l_{kh} : l = 1, \ldots, L\bigr)$;

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

пока значение Q и/или веса w не стабилизируются;

Задача дифференцирования суперпозиции функций

$x_i^0 = x_i$ — вектор объекта обучающей выборки на входе сети
$x_i^l = \bigl(x_{hi}^l\bigr)_{h=0}^{H_l}$ — вектор на выходе l-го слоя, $l = 1, \dots, L$:

$$x_{hi}^l = \sigma_h^l \left(\sum_{k=0}^{H_{l-1}} w_{kh}^l x_{ki}^{l-1} \right), \quad \text{в матричной записи} \quad x_i^l = \sigma^l \bigl(W^l x_i^{l-1} \bigr)$$

Функция потерь (для примера — среднеквадратичная):

$$\mathscr{L}_i(w) = \sum_{m=1}^{M} \mathscr{L}\bigl(a_m(x_i, w), y_{mi}\bigr) = \sum_{m=1}^{M} \frac{1}{2} \bigl(a_m(x_i, w) - y_{mi}\bigr)^2.$$

Промежуточная задача: найти частные производные

$$\frac{\partial \mathscr{L}_i(w)}{\partial x_h^l}$$

Рекуррентное вычисление частных производных

Найдём сначала частные производные по $x_h^L \equiv a_h(x_i, w)$:

$$\frac{\partial \mathscr{L}_i(w)}{\partial x_h^L} = \frac{\partial \mathscr{L}(x_h^L, y_{hi})}{\partial x_h^L} = a_h(x_i, w) - y_{hi} \equiv \varepsilon_{hi}^L;$$

для квадратичной функции потерь это *ошибка выходного слоя*.

Частные производные по x_h^l вычисляются рекуррентно, по уровням справа налево, $l = L, \dots, 2$:

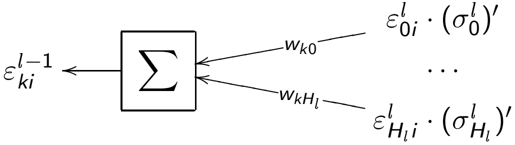
$$\frac{\partial \mathscr{L}_i(w)}{\partial x_k^{l-1}} = \sum_{h=0}^{H_l} \frac{\partial \mathscr{L}_i(w)}{\partial x_h^l} (\sigma_h^l)' w_{kh}^l = \sum_{h=0}^{H_l} \varepsilon_{hi}^l (\sigma_h^l)' w_{kh}^l = \varepsilon_{ki}^{l-1}$$

— формально назовём это *ошибкой скрытого слоя*. Здесь и далее опускаем аргумент у производной функции активации:

$$(\sigma_h^l)' = (\sigma_h^l)'\left(\sum_{k=0}^{H_l} w_{kh}^l x_{ki}^{l-1}\right)$$

Быстрое вычисление градиента

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



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

$$\frac{\partial \mathscr{L}_i(w)}{\partial w_{kh}} = \frac{\partial \mathscr{L}_i(w)}{\partial x_h^l} \frac{\partial x_h^l}{\partial w_{kh}} = \varepsilon_{hi}^l (\sigma_h^l)' x_{ki}^{l-1}, \quad k = 0..H_{l-1}, \quad h = 0..H_l.$$

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_l, λ, η;
Выход: синаптические веса w_{kh};

инициализировать веса w_{kh};
повторять

выбрать объект x_i из X^ℓ (например, случайно);
прямой ход:
$x^l_{hi} := \sigma^l_h\bigl(\sum_{k=0}^{H_l} w^l_{kh} x^{l-1}_{ki}\bigr)$, $l = 1..L$, $h = 1..H_l$;
$\varepsilon^L_{hi} := \dfrac{\partial \mathscr{L}_i(w)}{\partial x^L_{hi}}$, $h = 1..H_L$;
обратный ход:
$\varepsilon^{l-1}_{ki} = \sum_{h=0}^{H_l} \varepsilon^l_{hi} (\sigma^l_h)' w^l_{kh}$, $l = L..2$, $k = 0..H_{l-1}$;
градиентный шаг:
$w_{kh} := w_{kh} - \eta \varepsilon^l_{hi} (\sigma^l_h)' x^{l-1}_{ki}$, $l = 1..L$, $k = 0..H_{l-1}$, $h = 0..H_l$;

пока Q не стабилизируется;

Алгоритм BackProp: преимущества и недостатки

Преимущества:

- быстрое вычисление градиента
- обобщение на любые σ, $\mathscr{L}$ и любое число слоёв
- возможность динамического (потокового) обучения
- сублинейное обучение на сверхбольших выборках (когда части объектов x_i уже достаточно для обучения)
- возможно распараллеливание

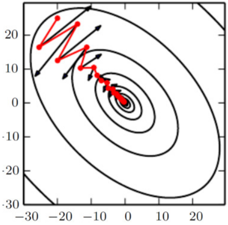
Недостатки — все те же, свойственные SG:

- медленная сходимость
- застревание в локальных экстремумах
- «паралич сети» из-за горизонтальных асимптот σ
- проблема переобучения
- подбор комплекса эвристик является искусством

Напоминание: метод накопления инерции (momentum)

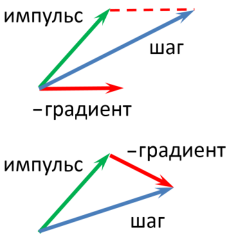
Momentum — экспоненциальное скользящее среднее градиента по $\approx \frac{1}{1-\gamma}$ последним итерациям [Б.Т.Поляк, 1964]:

$$v := \gamma v + (1-\gamma)\mathscr{L}'_i(w)$$
$$w := w - \eta v$$



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

$$v := \gamma v + (1-\gamma)\mathscr{L}'_i(w - \eta\gamma v)$$
$$w := w - \eta v$$



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:

$$G := \alpha G + (1-\alpha)\,\mathscr{L}_i'(w) \odot \mathscr{L}_i'(w)$$
$$w := w - \eta \mathscr{L}_i'(w) \oslash \left(\sqrt{G} + \varepsilon\right)$$

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

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

$$G := \alpha G + (1-\alpha)\,\mathscr{L}_i'(w) \odot \mathscr{L}_i'(w)$$
$$\delta := \mathscr{L}_i'(w) \odot \frac{\sqrt{\Delta}+\varepsilon}{\sqrt{G}+\varepsilon}$$
$$\Delta := \alpha \Delta + (1-\alpha)\,\delta \odot \delta$$
$$w := w - \eta \delta$$

Комбинированные градиентные методы

Adam (adaptive momentum) = инерция + RMSProp:

$$v := \gamma v + (1-\gamma)\mathscr{L}_i'(w) \qquad \hat{v} := v(1-\gamma^k)^{-1}$$

$$G := \alpha G + (1-\alpha)\mathscr{L}_i'(w) \odot \mathscr{L}_i'(w) \qquad \hat{G} := G(1-\alpha^k)^{-1}$$

$$w := w - \eta\hat{v} \oslash \left(\sqrt{\hat{G}} + \varepsilon\right)$$

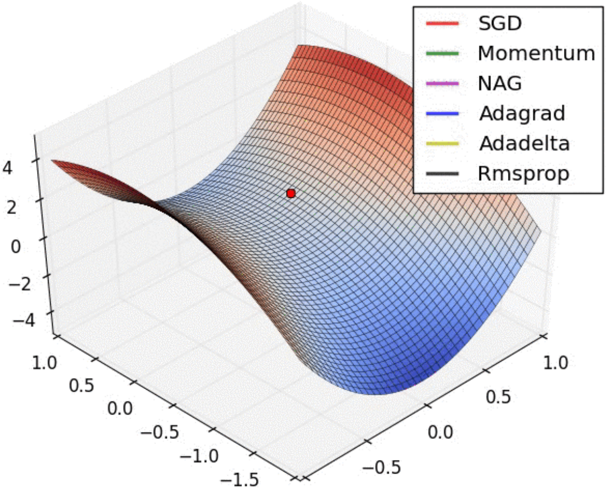
Калибровка $\hat{v}$, $\hat{G}$ увеличивает v, G на первых итерациях, где k — номер итерации; $\gamma = 0.9$, $\alpha = 0.999$, $\varepsilon = 10^{-8}$

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

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

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`

Напоминание: 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 $\dim^2(w)$.

Эвристика. Считаем, что гессиан диагонален:

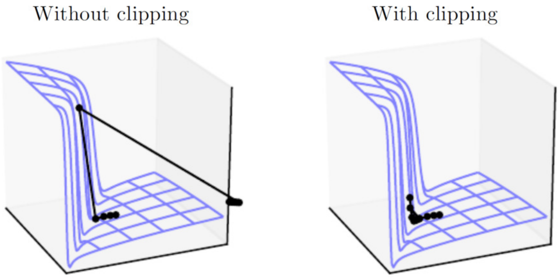
$$w_{jh} := w_{jh} - \eta\left(\frac{\partial^2 \mathscr{L}_i(w)}{\partial w_{jh}^2} + \mu\right)^{-1} \frac{\partial \mathscr{L}_i(w)}{\partial w_{jh}},$$

η — темп обучения (можно брать $\eta = 1$),
μ — параметр, предотвращающий обнуление знаменателя.

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

Проблема взрыва градиента и эвристика gradient clipping

Проблема взрыва градиента (gradient exploding)



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

При грамотном подборе γ проблема взрыва градиента не возникает, и эвристика Gradient Clipping не нужна.

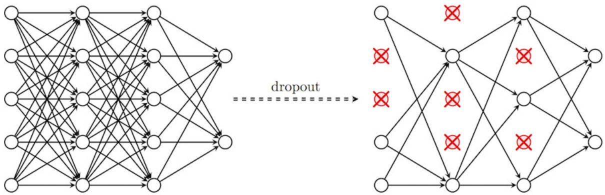
Метод случайных отключений нейронов (Dropout)

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

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

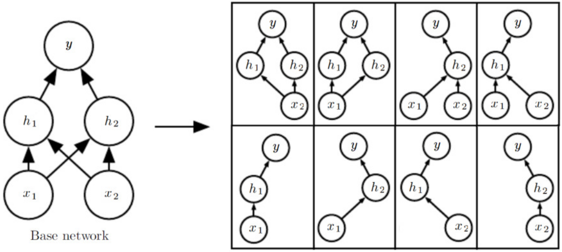
Этап применения: включаем все нейроны, но с поправкой:

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



Интерпретации Dropout

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



Обратный Dropout и L_2-регуляризация

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

Этап обучения:

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

Этап применения не требует ни модификаций, ни знания p_ℓ:

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

L_2-регуляризация предотвращает рост параметров на обучении:

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

Градиентный шаг с Dropout и L_2-регуляризацией:

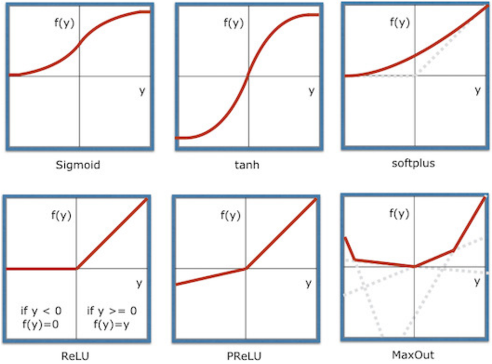
$$w := w(1 - \eta\lambda) - \eta\, \frac{1}{1-p_\ell}\xi_h^\ell\, \mathscr{L}_i'(w)$$

Функции активации ReLU и PReLU (LeakyReLU)

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

Функция положительной срезки (rectified linear unit)

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



Пакетная нормализация данных (Batch Normalization)

$B = \{x_i\}$ — пакеты (mini-batch) данных.
Усреднение градиентов $\mathscr{L}_i(w)$ по пакету ускоряет сходимость.

$B^l = \{x_i^l\}$ — векторы объектов x_i на выходе l-го слоя.

Batch Normalization:

1. Нормировать каждую h-ю компоненту вектора x_i^l по пакету:

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

2. Добавить линейный слой с настраиваемыми весами:

$$\tilde{x}_{hi}^l = \gamma_h^l \hat{x}_{hi}^l + \beta_h^l$$

3. Параметры γ_h^l и β_h^l настраиваются BackProp.

Эвристики для начального приближения

1. Выравнивание дисперсий выходов в разных слоях:

$$w_{kh} := \mathrm{uniform}\left(-\frac{1}{\sqrt{H_l}}, \frac{1}{\sqrt{H_l}}\right)$$

2. Выравнивание дисперсий градиентов в разных слоях:

$$w_{kh} := \mathrm{uniform}\left(-\frac{6}{\sqrt{H_{l-1}+H_l}}, \frac{6}{\sqrt{H_{l-1}+H_l}}\right),$$

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 $Q''(w) = \Bigl(\frac{\partial^2 Q(w)}{\partial w_{kh}\partial w_{k'h'}}\Bigr)$ — 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: $S_{kh} = w_{kh}^2 \frac{\partial^2 Q(w)}{\partial w_{kh}^2}$.

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: $S_j = \sum\limits_{h=1}^{H_1} S_{jh}$.

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 алгоритмов