Искусственные нейронные сети

Жорникова Полина, 622 гр.

2017г.

Задача аппроксимации с особым классом функций

X — множество объектов, Y — множество ответов;
$(f_1(x), \ldots, f_p(x))$ — признаки объекта $x \in X$, $f : X \to \mathbb{R}$; $x^j := f_j(x)$;
$X^n = (x_i, y_i)_{j=1}^n$ — обучающая выборка.

Рассмотрим стандартную **задачу построения предсказывающей модели**:

$$Q(a, X^n) = \frac{1}{n} \sum_{j=1}^{n} \mathscr{L}(a, x_j, y_j) \to \min_w,$$

где алгоритм a задаётся следующим образом:

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

где $w_j \in \mathbb{R}$, $j = 0, \ldots, p$; $\sigma : \mathbb{R} \to \mathbb{R}$ — функция активации (например, sign).

Преимущества особого класса функций

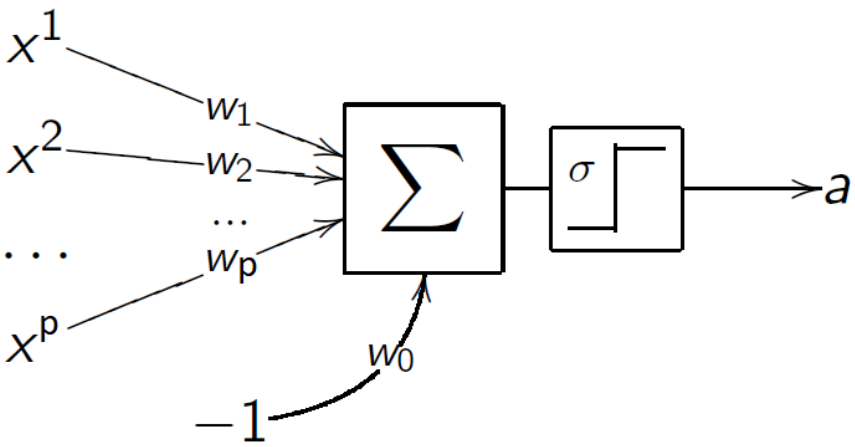
$$a(x, w) = \sigma(\langle w, x \rangle) = \sigma\left(\sum_{j=1}^{p} w_j f_j(x) - w_0\right).$$

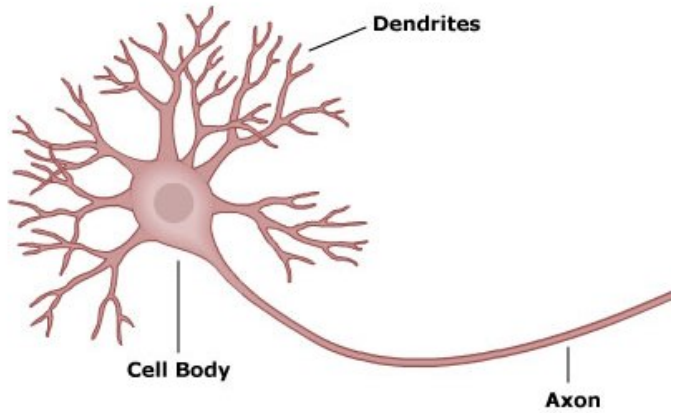
1. Биологическая интерпретация.
2. Способность аппроксимировать широкий класс предсказательных функций.
3. Расширяемость класса.
4. Возможность построения эффективных алгоритмов оптимизации (BackProp).

Интерпретация. Модель нейрона МакКаллока-Питса

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

- $f_j(x)$, $x^j := f_j(x)$, $j = 1, \ldots, p$, — числовые признаки, входы;
- $w_j \in \mathbb{R}$, $j = 1, \ldots, p$ — весовые коэффициенты;
- $\sigma(z)$ — функция активации (например, sign);
- w_0 — порог активации.





Способность аппроксимировать. Модель нейрона для задач машинного обучения

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

$$Q(w; X^n) = \sum_{j=1}^{n} \mathscr{L}\left(\langle w, x_j \rangle, y_j\right) = \sum_{j=1}^{n} \left(\sigma(\langle w, x_j \rangle) - y_j\right)^2 \to \min_w .$$

При $\sigma(z) = z$ получаем многомерную линейную регрессию.

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

$$Q(w; X^n) = \sum_{j=1}^{n} \mathscr{L}\left(\langle w, x_j \rangle, y_j\right) = \sum_{j=1}^{n} [y_j \langle w, x_j \rangle < 0] \to \min_w .$$

Нейронная сеть — суперпозиция нейронов.
Насколько богатый класс функций реализуется нейроном? А сетью?

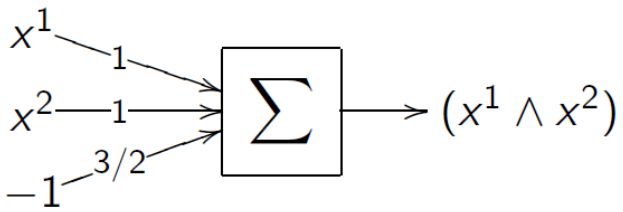
Способность аппроксимировать. Нейронная реализация логических функций

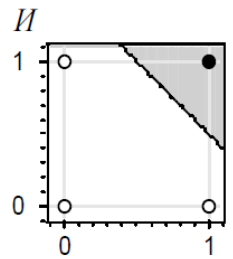
Функции И, ИЛИ, НЕ от бинарных переменных x^1 и x^2:

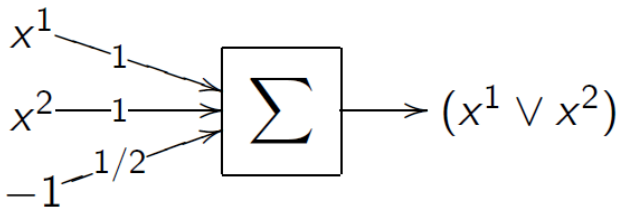
$$x^1 \wedge x^2 = \left[x^1 + x^2 - \frac{3}{2} > 0\right];$$

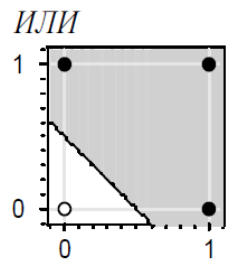
$$x^1 \vee x^2 = \left[x^1 + x^2 - \frac{1}{2} > 0\right];$$

$$\neg x^1 = \left[-x^1 - \frac{1}{2} > 0\right].$$









Способность аппроксимировать. Функция XOR

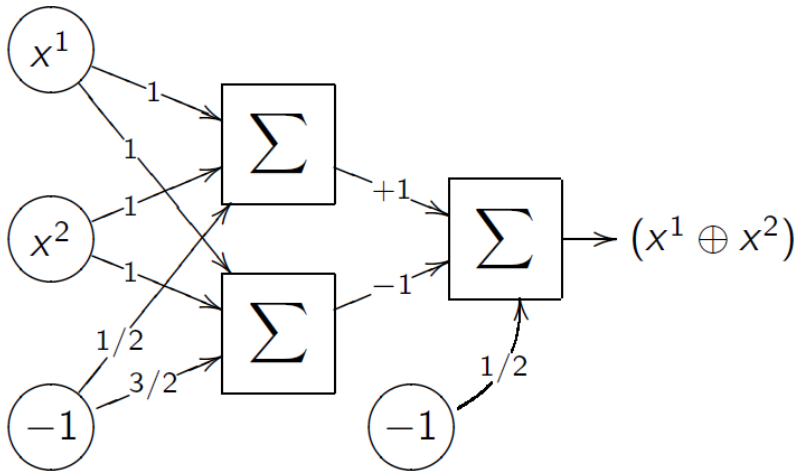
$x^1 \oplus x^2 = [x^1 \neq x^2]$ не реализуема одним нейроном. 2 варианта реализации:

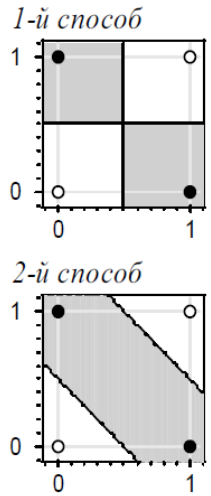
1. добавление нелинейного признака:

$$x^1 \oplus x^2 = \left[x^1 + x^2 - 2x^1x^2 - \frac{1}{2} > 0\right];$$

2. двухслойной нейронной сетью (суперпозицией) функций И, ИЛИ, НЕ:

$$x^1 \oplus x^2 = \left[\neg\left(x^1 \wedge x^2 - x^1 \vee x^2\right) > 0\right].$$





Способность аппроксимировать. Можно ли любую функцию представить нейросетью?

Универсальная теорема аппроксимации (Цыбенко, 1989)

Пусть $\sigma(x)$ *— непостоянная, ограниченная и монотонно возрастающая непрерывная функция;*
I_{p_0} *—* p_0*-мерный единичный гиперкуб;*
$C(I_{p_0})$ *— множество непрерывных функций на* I_{p_0}*.*
Тогда для любой $f \in C(I_{p_0})$ *и* $\varepsilon > 0$ *существуют* $p_1 \in \mathbb{Z}$ *и* α_i, b_i, $w_{ij} \in \mathbb{R}$, $i = 1, \ldots, p_1$, $j = 1, \ldots, p_0$, *такие что функция*

$$F(x^1, \ldots, x^{p_0}) = \sum_{i=1}^{p_1} \alpha_i \, \sigma \left(\sum_{j=1}^{p_0} w_{ij} x^j + b_i \right),$$

аппроксимирует функцию f *с точностью* ε*:*

$$|F(x^1, \ldots, x^{p_0}) - f(x^1, \ldots, x^{p_0})| < \varepsilon$$

для любого $x = (x^1, \ldots, x^{p_0}) \in I_{p_0}$*.*

Способность аппроксимировать. Можно ли любую функцию представить нейросетью?

$$F(x^1, \ldots, x^{p_0}) = \sum_{i=1}^{p_1} \alpha_i \, \sigma \left(\sum_{j=1}^{p_0} w_{ij} x^j + b_i \right),$$

Нейросеть, выход которой, соответствует F:
сеть с p_0 входными узлами, одним скрытым слоем с p_1 узлами и функцией активации $\sigma(z)$.

Выводы:

- Любую непрерывную функцию можно приблизить нейросетью с любой заранее заданной точностью.
- Для этой нейросети нужна одна нелинейная функция активации и один скрытый слой.

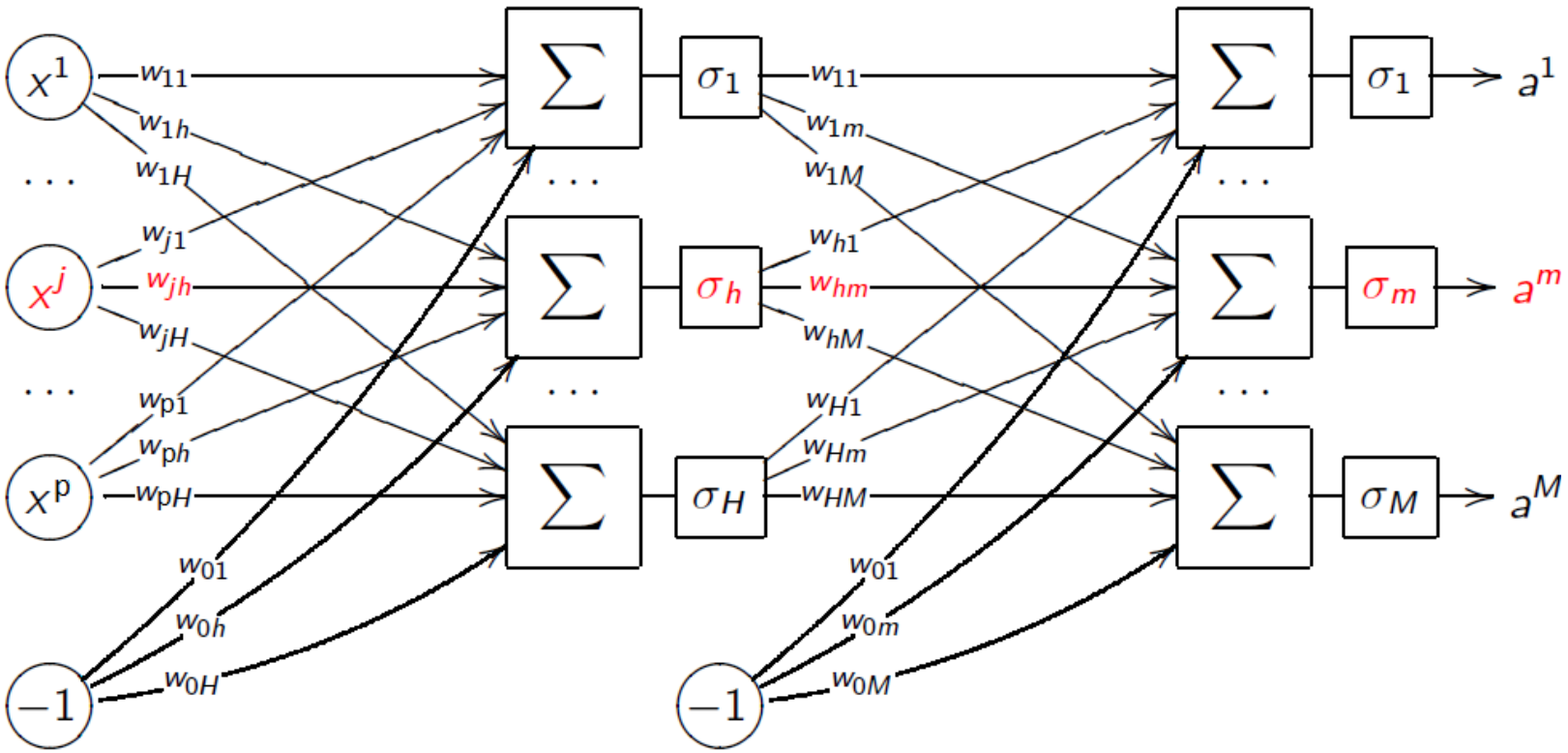
Расширяемость. Многослойная нейронная сеть

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

входной слой, p признаков

скрытый слой, H нейронов

выходной слой, M нейронов



Вектор параметров $w \equiv (w_{jh}, w_{hm}) = \left(\{w_{jh}\}_{j,h=0}^{p,H}, \{w_{hm}\}_{h,m=0}^{H,M}\right) \in \mathbb{R}^{H(p+M+1)+M}$.

Расширяемость. Многослойная нейронная сеть. Функция активации

- логистическая функция: $\sigma(z) = \frac{1}{1+e^{-az}}$, $a \in \mathbb{R}$;
- гиперболический тангенс: $\sigma(z) = \frac{e^{az}-e^{-az}}{e^{az}+e^{-az}}$, $a \in \mathbb{R}$;
- rectifier: $f(z) = \max(0, x) \approx \ln(1 + e^z)$.

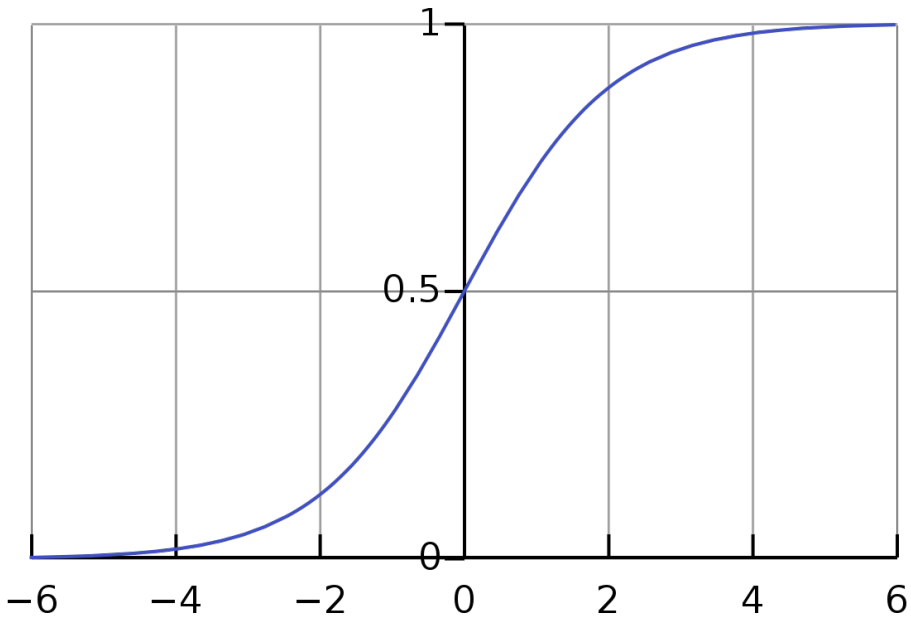


Рис.: График функции $\frac{1}{1+e^{-z}}$.

BackProp. Напоминание: алгоритм Stochastic Gradient

Идея: на каждом шаге учитывать только одно наблюдение.

Алгоритм:

- Инициализация вектора параметров $w^{(0)}$, выбор скорости обучения η, темп забывания λ, начальная оценка функционала

$$\overline{Q}(w_0) = \frac{1}{n} Q(w_0) = \frac{1}{n} \sum_{i=1}^{n} \mathscr{L}(w_0, x_i, y_i).$$

- Случайный выбор элемента из выборки x_i и вычисление функции потерь $\mathscr{L}_i := \mathscr{L}(w, x_i, y_i)$.
- Обновление вектора параметров $w := w - \eta \nabla \mathscr{L}(w, x_i, y_i)$.
- Оценка функционала $Q := (1 - \lambda)Q + \lambda \mathscr{L}_i$.
- Повторять до сходимости Q или w.

BackProp. Дифференцирование суперпозиции функций

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

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

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

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

Промежуточная задача: найти частые производные

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

BackProp. Быстрое вычисление градиента

Промежуточная задача: частая производная

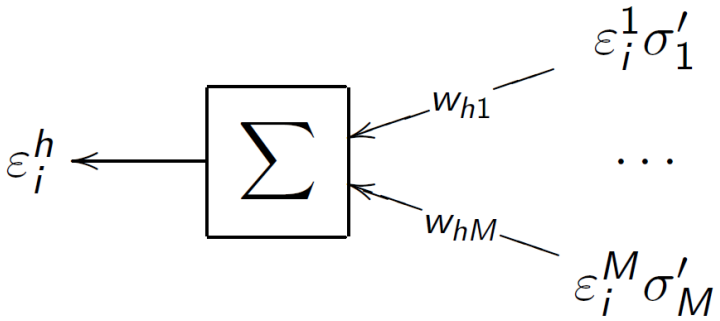
$$\frac{\partial \mathscr{L}_i(w)}{\partial a^m} = a^m(x_i) - y_i^m = \varepsilon_i^m$$

— это ошибка на выходном слое.

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

— назовём это *ошибкой на скрытом слое*.

Похоже, что ε_i^h вычисляется по ε_i^m, если запустить сеть «задом наперёд»:



BackProp. Быстрое вычисление градиента

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

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

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

Под $\sigma_m^{'}$ всегда будет подразумеваться производная в точке $\sum_{h=0}^{H} w_{hm} u^h(x_i)$. Аналогично, под $\sigma_h^{'}$ всегда будет подразумеваться производная в точке $\sum_{j=0}^{p} w_{jh} f_j(x_i)$.

BackProp. Алгоритм

Вход: $X^n = (x_i, y_i)_{i=1}^n \subset \mathbb{R}^p \times \mathbb{R}^M$; параметры H, λ, η.

Выход: веса w_{jh}, w_{hm}.

1: инициализировать веса w_{jh}, w_{hm};

2: **повторять**

3: случайно выбрать элемент x_i из выборки X^n;

4: прямой ход:

$u_i^h := \sigma_h\left(\sum_{j=0}^p w_{jh} x_i^j\right),\ h = 1, \ldots, H;$

$a_i^m := \sigma_m\left(\sum_{h=0}^H w_{hm} u_i^h\right),\ \varepsilon_i^m := a_i^m - y_{im},\ m = 1, \ldots, M;$

$\mathscr{L}_i := \sum_{m=1}^M (\varepsilon_i^m)^2$, вычисление производных $\sigma_m^{'}, \sigma_h^{'}$;

5: обратный ход:

$\varepsilon_i^h := \sum_{m=1}^M \varepsilon_i^m \sigma^{'} w_{hm},\ h = 1, \ldots, H;$

6: градиентный шаг:

$w_{hm} := w_{hm} - \eta \varepsilon_i^m \sigma_m^{'} u_i^h,\ h = 0, \ldots, H,\ m = 1, \ldots, M;$

$w_{jh} := w_{jh} - \eta \varepsilon_i^h \sigma_h^{'} x_i^j,\ j = 0, \ldots, p,\ h = 1, \ldots, H;$

7: $Q := (1 - \lambda) Q + \lambda \mathscr{L}_i;$

8: **пока:** Q не сойдется.

BackProp. Преимущества и недостатки

Преимущества:

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

Недостатки (есть все те же, что и у SG):

- возможна медленная сходимость;
- застревание в локальных минимумах;
- проблема «паралича» сети (горизонтальные асимптоты σ);
- проблема переобучения;
- сложно подбирать эвристики.

BackProp. Эвристики

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

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

И появляются новые эвристики для улучшения сходимости, специфичные для нейросети.

Эвристики для нейросети. Начальное приближение

Более тщательный подбор начального приближения.

Нейроны первого слоя настраиваются как H отдельных однослойных сетей:

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

тем самым обеспечивается *различность нейронов*.

Затем по отдельности настраиваются нейроны второго слоя, которым на вход подается вектор выходных значений первого.

Эвристики для нейросети. Выбор градиентного метода

1. **Адаптивный градиентный шаг** (метод скорейшего спуска). Ищем η_*:

$$\mathscr{L}_i(w - \nabla\mathscr{L}_i(w)) \to \min_{\eta}.$$

2. **Диагональный метод Левенберга-Марквардта.**

Метод Ньютона-Рафсона (второго порядка):

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

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

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 элемента.

Построение нейросети

Выбор числа скрытых слоёв. Двух-трёх слоёв достаточно для очень широкого класса задач. Если знаем, что классы линейно разделимы, то достаточно одного слоя.

Выбор числа нейронов в скрытом слое H.

- **Визуальный способ** (для задач с небольшим числом признаков). Если граница классов (или кривая регрессии) слишком сглажена — размер слоя нужно увеличить, а если есть резкие колебания, то, наоборот, уменьшить.
- **По внешнему критерию.**
  - Средняя ошибка на тестовой выборке.
  - CV (главный недостаток — высокая трудоёмкость).

Построение нейросети. Динамическое наращивание сети

1. Обучение при заведомо недостаточном числе нейронов $H \ll n$ до тех пор, пока ошибка не перестаёт убывать.
2. Добавление нового нейрона и инициализация его связей небольшими случайными весами или путем обучения
   - либо по случайной подвыборке $X' \subseteq X^n$;
   - либо по объектам с наибольшими значениями потерь;
   - либо по случайному подмножеству входов;
   - либо из различных случайных начальных приближений.

   Веса старых связей не меняются.
3. Снова итерации BrackPop.

Полезно наблюдать за внешним критерием. Например, прохождение $Q(X^k)$ через минимум — надежный критерий остановки.

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.

Построение нейросети. Удаление избыточных связей. Optimal Brain Damage (OBD)

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

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

где $Q''(w) = \frac{\partial^2 Q(w)}{\partial w_{jh}\partial w_{j'h'}}$ — гессиан размера $(H(p+M+1)+M)^2$.

Эвристика. Пусть гессиан диагонален, тогда

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

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

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

Построение нейросети. Удаление избыточных связей. Optimal Brain Damage (OBD)

1. В BackProp вычислять вторые производные $\frac{\partial^2 Q(w)}{\partial w_{jh}^2}$, $\frac{\partial^2 Q(w)}{\partial w_{hm}^2}$.
2. Если процесс минимизации $Q(w)$ пришел в минимум, то
   - упорядочить веса по убыванию S_{jh};
   - удалить d связей с наименьшей значимостью;
   - снова запустить BackProp.
3. Если $Q(w, X^n)$ или $Q(w, X^k)$ существенно ухудшился, то вернуть последние удаленные связи и выйти.

Аналогично, OBD можно использовать для **отбора признаков**.
Суммарная значимость признака: $S_j = \sum_{h=1}^{H} S_{jh}$.

Эмпирический опыт: сеть, построенная с помощью OBD, меньше переобучается, чем сеть сразу построенная по полученной структуре (со случайно инициализированными весами).