Искусственные нейронные сети

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)»

17 апреля 2012

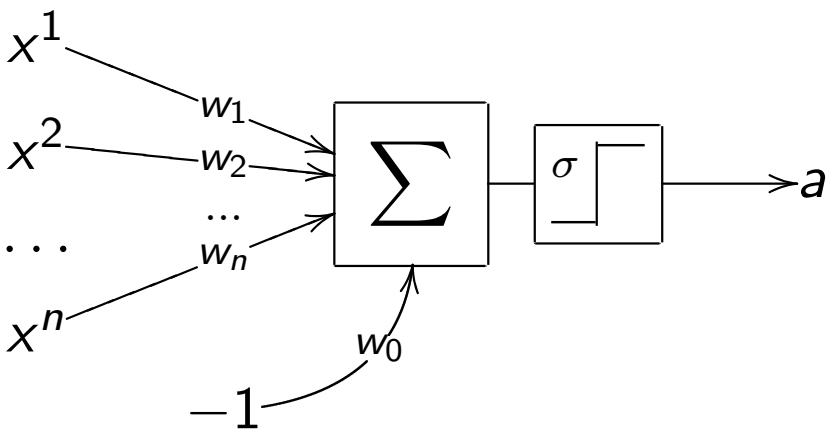
Содержание

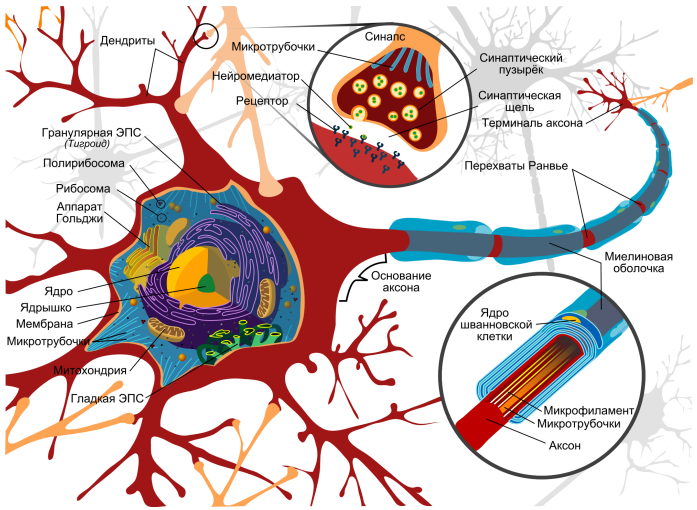
Напоминание: линейная модель нейрона МакКаллока-Питтса

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

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

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





Линейные алгоритмы классификации и регрессии

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

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

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

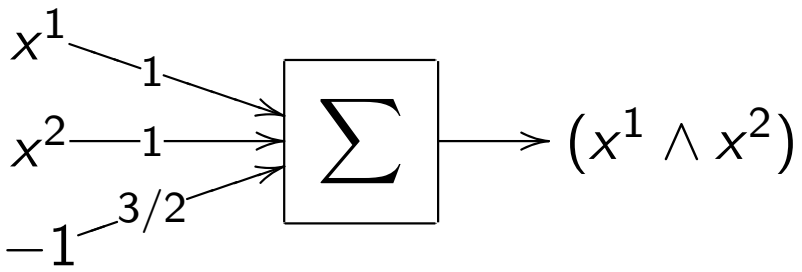
$$Q(w; X^\ell) = \sum_{i=1}^{\ell} \big(\sigma(\langle w, x_i \rangle) - y_i\big)^2 \to \min_w;$$

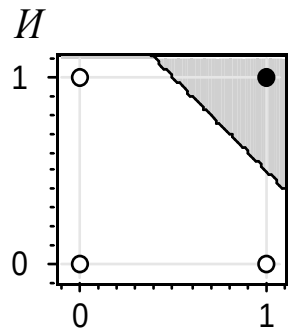
Насколько богатый класс функций реализуется нейроном?
А сетью (суперпозицией) нейронов?

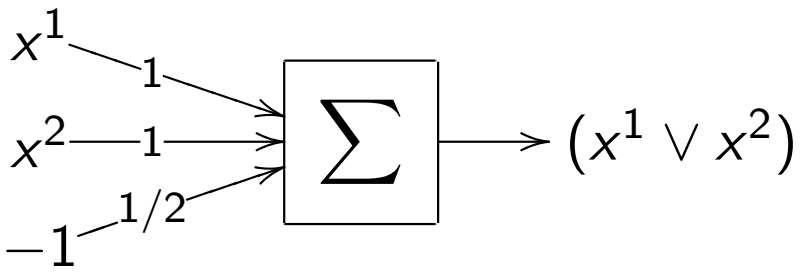
Нейронная реализация логических функций

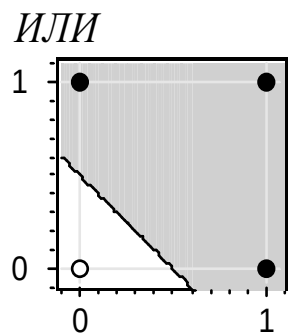
Функции И, ИЛИ, НЕ от бинарных переменных x^1 и x^2:

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







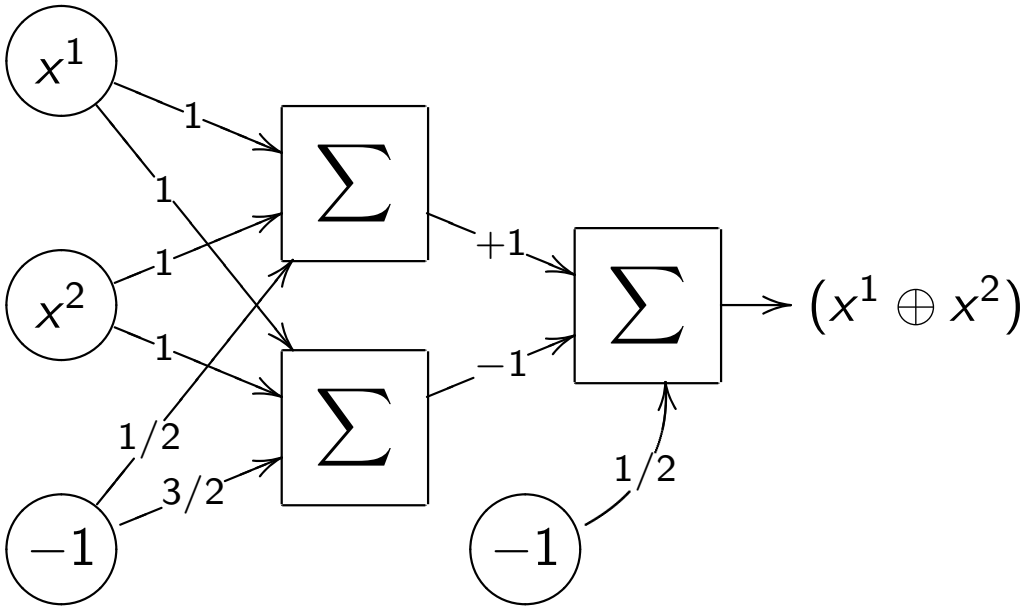


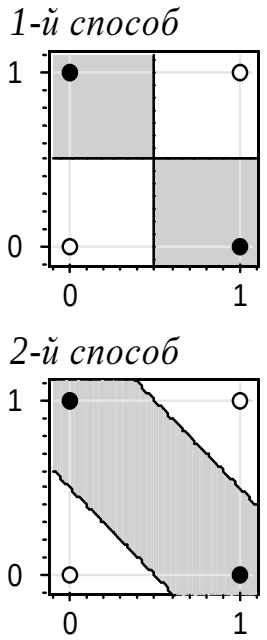
Логическая функция XOR (исключающее ИЛИ)

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?

Утверждение

Любая булева функция представима в виде ДНФ, следовательно, и в виде двухслойной сети.

Решение тринадцатой проблемы Гильберта:

Теорема (Колмогоров, 1957)

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

$$f(x^1, x^2, \ldots, x^n) = \sum_{k=1}^{2n+1} h_k \left(\sum_{i=1}^{n} \varphi_{ik}(x^i) \right),$$

где h_k, φ_{ik} — непрерывные функции, и φ_{ik} не зависят от f.

Любую ли функцию можно представить нейросетью?

Теорема (Вейерштрасс)

Любую непрерывную функцию n переменных можно равномерно приблизить полиномом с любой точностью.

Определение

Набор функций F называется *разделяющим точки* множества X, если для любых различных $x, x' \in X$ существует функция $f \in F$ такая, что $f(x) \neq f(x')$.

Теорема (Стоун, 1948)

Пусть X — компактное пространство, $C(X)$ — алгебра непрерывных на X вещественных функций, F — кольцо в $C(X)$, содержащее константу ($1 \in F$) и разделяющее точки множества X. Тогда F плотно в $C(X)$.

Любую ли функцию можно представить нейросетью?

Определение

Набор функций $F \subseteq C(X)$ называется *замкнутым относительно функции* $\varphi : \mathbb{R} \to \mathbb{R}$, если для любого $f \in F$ выполнено $\varphi(f) \in F$.

Теорема (Горбань, 1998)

Пусть X — компактное пространство, $C(X)$ — алгебра непрерывных на X вещественных функций, F — линейное подпространство в $C(X)$, замкнутое относительно нелинейной непрерывной функции φ, содержащее константу ($1 \in F$) и разделяющее точки множества X. Тогда F плотно в $C(X)$.

Любую ли функцию можно представить нейросетью?

Выводы:

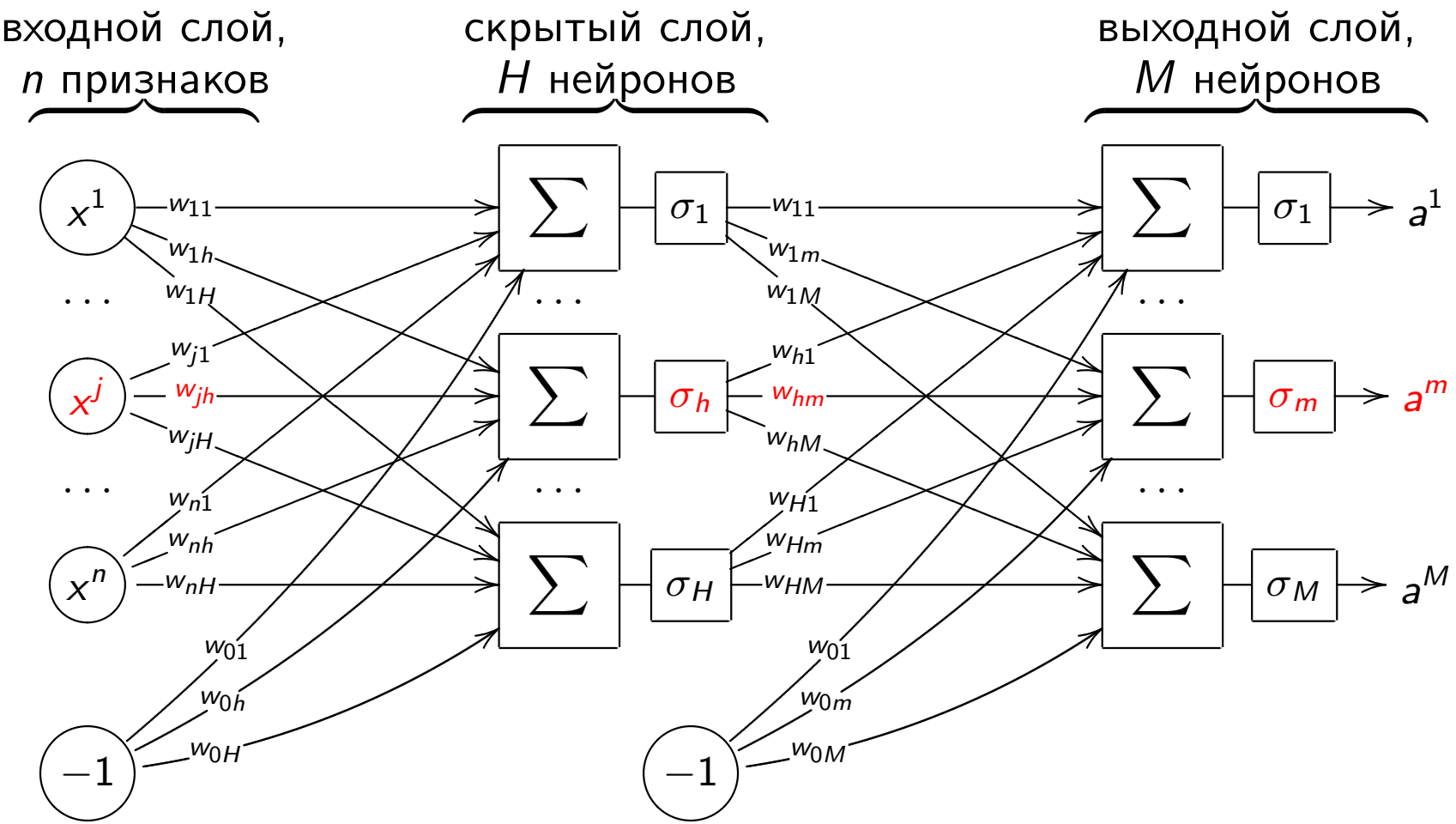
- С помощью линейных операций и одной нелинейной *функции активации* φ можно вычислять любую непрерывную функцию с любой желаемой точностью.
- Однако эти теоремы ничего не говорят о числе слоёв нейронной сети и о числе нейронов в каждом слое.

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.



Напоминание: Алгоритм SG (Stochastic Gradient)

Задача минимизации суммарных потерь:

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

Вход: выборка X^ℓ; темп обучения η; параметр λ;
Выход: веса $w \equiv \left(w_{jh}, w_{hm}\right) \in \mathbb{R}^{H(n+M+1)+M}$;

1: инициализировать веса w и текущую оценку $Q(w)$;
2: **повторять**
3: выбрать объект x_i из X^ℓ (например, случайно);
4: вычислить потерю $\mathscr{L}_i := \mathscr{L}(w, x_i, y_i)$;
5: градиентный шаг: $w := w - \eta \nabla \mathscr{L}(w, x_i, y_i)$;
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 $a^m(x_i)$, $m = 1..M$ 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

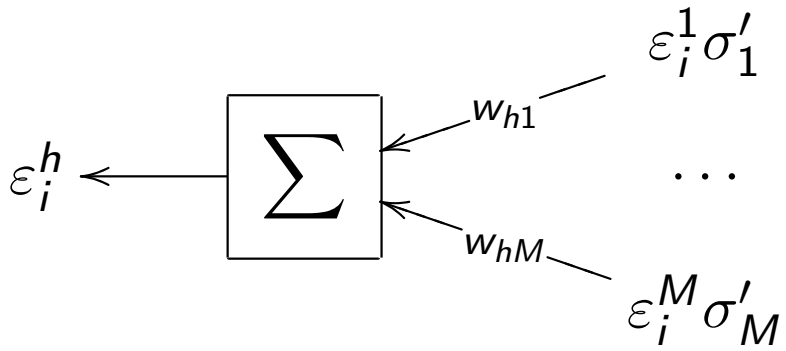
Промежуточная задача: частные производные

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

— это ошибка на выходном слое;

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

— назовём это *ошибкой на выходном слое*. Похоже, что ε_i^h вычисляется по ε_i^m, если запустить сеть «задом наперёд»:



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:

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

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

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

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

Алгоритм BackProp: преимущества и недостатки

Преимущества:

- эффективность: градиент вычисляется за время, сравнимое со временем вычисления самой сети;
- 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 переобучения;
- подбор комплекса эвристик является искусством;

Стандартные эвристики для метода SG

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. Более тщательный подбор начального приближения.
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 $Q(w) = \sum_{i=1}^{\ell} \mathscr{L}_i(w)$ на блоки (chunks).

Диагональный метод Левенберга-Марквардта

Метод Ньютона-Рафсона (второго порядка):

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

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

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 $\big(H(n+M+1)+M\big)^2$.

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 по многослойным сетям

- Нейрон = линейная классификация или регрессия.
- Нейронная сеть = суперпозиция нейронов с нелинейной функцией активации. Двух-трёх слоёв достаточно для решения очень широкого класса задач.
- 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)

Постановка задачи кластеризации (обучения без учителя)

Дано:

$X = \mathbb{R}^n$ — пространство объектов;

$Y = \{1, \ldots, M\}$ — множество кластеров, M фиксировано;

$X^\ell = \{x_i\}_{i=1}^\ell$ — обучающая выборка объектов;

$\rho\colon X \times X \to \mathbb{R}_+$ — метрика.

Требуется:

построить алгоритм кластеризации $a\colon X \to Y$.

Предлагается:

- ввести центры кластеров $w_m \in \mathbb{R}^n$, $m = 1, \ldots, M$;
- относить объект $x \in X$ к ближайшему кластеру (правило жёсткой конкуренции WTA — Winner Takes All):

$$a(x) = \arg\min_{m \in Y} \rho(x, w_m).$$

Метод стохастического градиента

Минимизация среднего внутрикластерного расстояния:

$$Q(w; X^\ell) = \frac{1}{2}\sum_{i=1}^{\ell} \rho^2(x_i, w_{a(x_i)}) \to \min_w, \quad w = (w_1, \ldots, w_M);$$

Пусть метрика евклидова, $\rho^2(x_i, w_m) = \|w_m - x_i\|^2$.

$$\frac{\partial Q(w; X^\ell)}{\partial w_m} = \sum_{i=1}^{\ell} (w_m - x_i)\big[a(x_i) = m\big].$$

Градиентный шаг в методе SG: для случайного $x_i \in X^\ell$

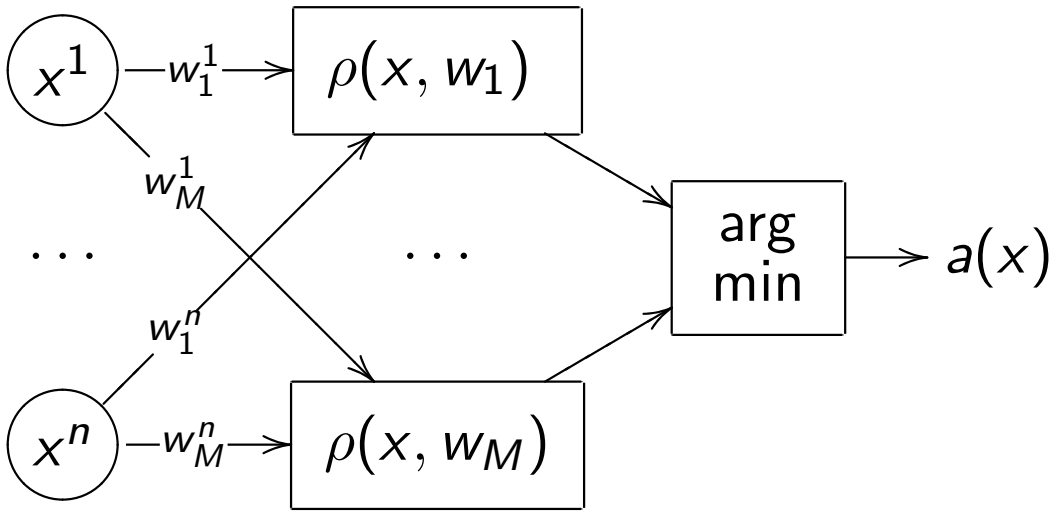
$$w_m := w_m + \eta(x_i - w_m)\big[a(x_i) = m\big]$$

(если x_i относится к кластеру m, то w_m сдвигается в сторону x_i).

Сеть Кохонена (сеть с конкурентным обучением)

Структура алгоритма — двухслойная нейронная сеть:

$$a(x) = \arg\min_{m \in Y} \rho(x, w_m) :$$



Градиентное правило обучения напоминает перцептрон:

$$\text{если} \quad a(x_i) = m, \quad \text{то} \quad w_m := w_m + \eta(x_i - w_m).$$

Алгоритм SG (Stochastic Gradient)

Вход: выборка X^ℓ; темп обучения η; параметр λ;
Выход: центры кластеров $w_1, \ldots, w_M \in \mathbb{R}^n$;

1: инициализировать центры w_m, $m = 1, \ldots, M$;
2: инициализировать текущую оценку функционала:
$Q := \sum_{i=1}^{\ell} \rho^2(x_i, w_{a(x_i)})$;
3: **повторять**
4: выбрать объект x_i из X^ℓ (например, случайно);
5: вычислить кластеризацию: $m := \arg\min_{m \in Y} \rho(x_i, w_m)$;
6: градиентный шаг: $w_m := w_m + \eta(x_i - w_m)$;
7: оценить значение функционала:
$Q := (1 - \lambda)Q + \lambda\rho^2(x_i, w_m)$;
8: **пока** значение Q и/или веса w не стабилизируются;

Жёсткая и мягкая конкуренция

Правило жёсткой конкуренции WTA (winner takes all):

$$w_m := w_m + \eta(x_i - w_m)\big[a(x_i) = m\big], \quad m = 1, \ldots, M,$$

Недостатки правила WTM:

- медленная скорость сходимости;
- некоторые w_m могут никогда не выбираться.

Правило мягкой конкуренции WTM (winner takes most):

$$w_m := w_m + \eta(x_i - w_m)\, K\big(\rho(x_i, w_m)\big), \quad m = 1, \ldots, M,$$

где ядро $K(\rho)$ — неотрицательная невозрастающая функция.

Теперь центры *всех* кластеров смещаются в сторону x_i,
но чем дальше от x_i, тем меньше величина смещения.

Карта Кохонена (Self Organizing Map, SOM)

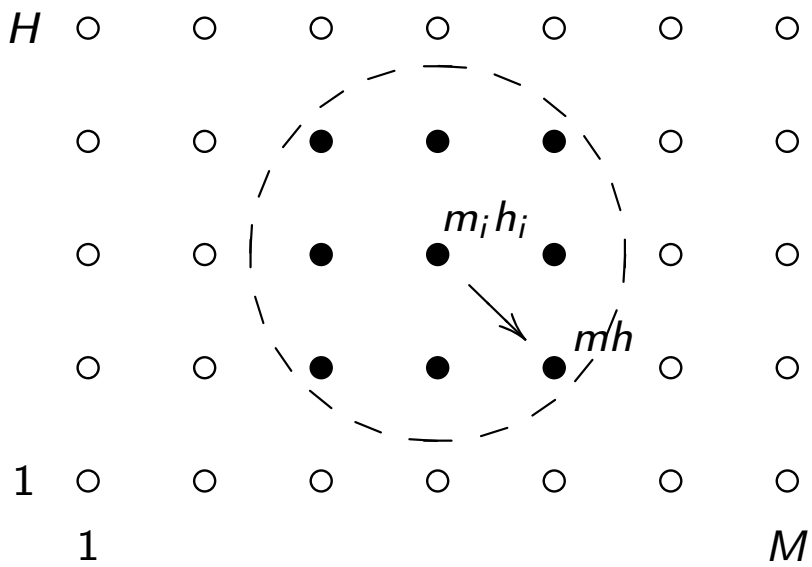
$Y = \{1, \ldots, M\} \times \{1, \ldots, H\}$ — прямоугольная сетка кластеров.

Каждому узлу (m, h) приписан нейрон Кохонена $w_{mh} \in \mathbb{R}^n$.

Наряду с метрикой $\rho(x_i, x)$ на X вводится метрика на сетке Y:

$$r\big((m_i, h_i), (m, h)\big) = \sqrt{(m - m_i)^2 + (h - h_i)^2}.$$

Окрестность(m_i, h_i):



Обучение карты Кохонена

Вход: X^ℓ — обучающая выборка; η — темп обучения;
Выход: $w_{mh} \in \mathbb{R}^n$ — векторы весов, $m = 1..M$, $h = 1..H$;

1: $w_{mh} := \text{random}\left(-\frac{1}{2MH}, \frac{1}{2MH}\right)$ — инициализация весов;
2: **повторять**
3: выбрать объект x_i из X^ℓ случайным образом;
4: WTA: вычислить координаты кластера:
$(m_i, h_i) := a(x_i) \equiv \arg\min_{(m,h)\in Y} \rho(x_i, w_{mh})$;
5: **для всех** $(m, h) \in$ Окрестность(m_i, h_i)
6: WTM: сделать шаг градиентного спуска:
$w_{mh} := w_{mh} + \eta(x_i - w_{mh})\, K\big(r((m_i, h_i), (m, h))\big)$;
7: **пока** кластеризация не стабилизируется;

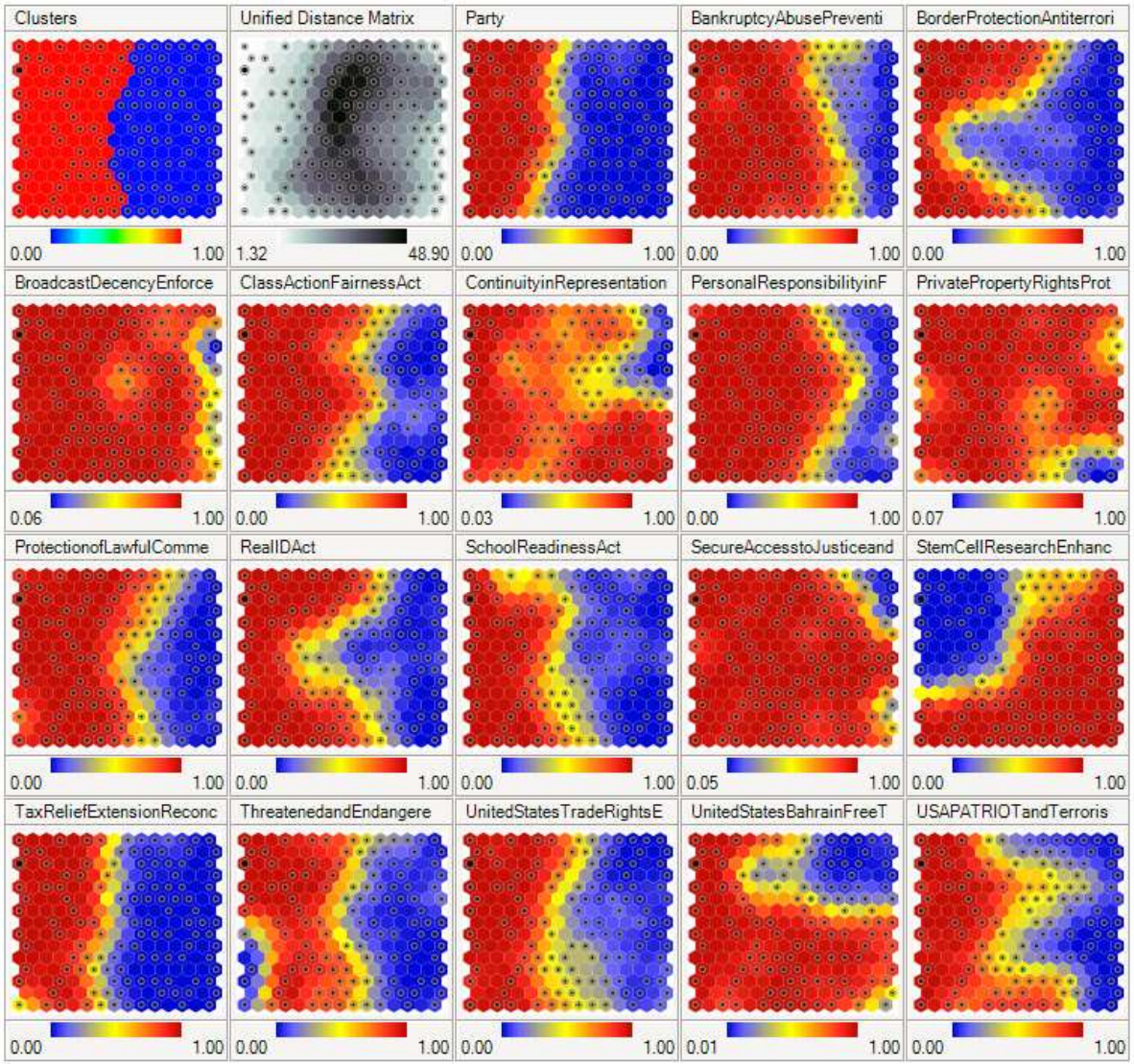
Интерпретация карт Кохонена

Два типа графиков — цветных карт $M \times H$:

- Цвет узла (m, h) — локальная плотность в точке (m, h) — среднее расстояние до k ближайших точек выборки;
- По одной карте на каждый признак:
  цвет узла (m, h) — значение j-й компоненты вектора $w_{m,h}$.

Пример:
Задача UCI house-votes (US Congress voting patterns)
Объекты — конгрессмены;
Признаки — вопросы, выносившиеся на голосование;
Есть целевой признак {демократ,республиканец}.

Интерпретация карт Кохонена (пример)



Достоинства и недостатки карт Кохонена

Достоинства:

- Возможность визуального анализа многомерных данных.

Недостатки:

- **Субъективность.** Карта зависит не только от кластерной структуры данных, но и...
  — от свойств сглаживающего ядра;
  — от (случайной) инициализации;
  — от (случайного) выбора x_i в ходе итераций.
- **Искажения.** Близкие объекты исходного пространства могут переходить в далёкие точки на карте, и наоборот.

Рекомендуется только для разведочного анализа данных.

Кусочно-постоянная непараметрическая регрессия

Задача восстановления регрессии:

$X^\ell = \{x_i, y_i\}_{i=1}^\ell, \quad y_i \in \mathbb{R}.$

Основная идея — применить WTA-кластеризацию:

1) разбить выборку на M кластеров с центрами w_m:

$$c(x) = \arg\min_{m=1,\ldots,M} \rho(x, w_m);$$

2) на каждом кластере построить регрессию-константу:

$$a(x) = v_{c(x)} = \sum_{m=1}^{M} v_m \big[c(x) = m\big].$$

Требуется: по выборке X^ℓ найти центры w_m и уровни v_m.

Кусочно-постоянная непараметрическая регрессия

Настройка центров w_m сетью Кохонена — WTA.

Задача настройки уровней v_m решается аналитически:

$$Q(v) = \frac{1}{2}\sum_{i=1}^{\ell}\big(a(x_i) - y_i\big)^2 \to \min_v;$$

$$\frac{\partial Q}{\partial v_m} = \sum_{i=1}^{\ell}\big(a(x_i) - y_i\big)\big[c(x_i) = m\big] = 0.$$

Подставляя сюда $a(x_i) = v_m$, получаем среднее y_i по кластеру:

$$v_m = \frac{\sum_{i=1}^{\ell} y_i\big[c(x_i) = m\big]}{\sum_{i=1}^{\ell}\big[c(x_i) = m\big]}.$$

Кусочно-гладкая непараметрическая регрессия

Усложнение: теперь функция $a(x)$ должна быть гладкой.

Основная идея — применить WTM-кластеризацию:
1) «мягкая» кластеризация на M кластеров с центрами w_m:

$$c_m(x) = K\big(\rho(x, w_m)\big), \quad m = 1, \ldots, M;$$

2) формула Надарая-Ватсона сглаживает ответы v_m,
но не по ℓ объектам, а по M центрам кластеров w_m:

$$a(x) = \frac{\sum\limits_{m=1}^{M} v_m c_m(x)}{\sum\limits_{m=1}^{M} c_m(x)}.$$

Требуется: по выборке X^ℓ найти центры w_m и уровни v_m.

Кусочно-гладкая непараметрическая регрессия

Настройка центров w_m сетью Кохонена — WTM:

$$Q(w) = \frac{1}{2}\sum_{i=1}^{\ell}\sum_{m=1}^{M} c_m(x_i)\|w_m - x_i\|^2 \to \min_w;$$

$$\frac{\partial Q(w)}{\partial w_m} = \sum_{i=1}^{\ell} c_m(x_i)(w_m - x_i);$$

Настройка уровней v_m линейным нейроном:

$$Q(v) = \frac{1}{2}\sum_{i=1}^{\ell}\big(a(x_i) - y_i\big)^2 \to \min_v;$$

$$\frac{\partial Q(v)}{\partial v_m} = \sum_{i=1}^{\ell} \frac{c_m(x_i)}{\sum_{s=1}^{M} c_s(x_i)}\big(a(x_i) - y_i\big);$$

Кусочно-гладкая непараметрическая регрессия

Веса слоя Кохонена w_m и веса линейного слоя v_m на каждой итерации SG обновляются независимо:

$$\begin{cases} w_m := w_m + \eta_1(x_i - w_m)K\big(\rho(x, w_m)\big); \\ v_m := v_m - \eta_2(a(x_i) - y_i)K\big(\rho(x, w_m)\big); \end{cases}$$

Достоинства этого метода:

- регрессия учитывает кластерную структуру выборки;
- повышается эффективность вычисления $a(x)$.