

Искусственные нейронные сети

К. В. Воронцов, А. В. Зухба
vokov@forecsys.ru
a__l@mail.ru

май 2016

Содержание

- 1 **Многослойные нейронные сети**
 - Проблема полноты
 - Вычислительные возможности нейронных сетей
 - Многослойная нейронная сеть
- 2 **Метод обратного распространения ошибок**
 - Алгоритм BackProp
 - Эвристики для улучшения сходимости
 - Эвристики для оптимизации структуры сети
- 3 **Сети Кохонена**
 - Модели конкурентного обучения
 - Карты Кохонена
 - Гибридные сети: кластеризация + регрессия

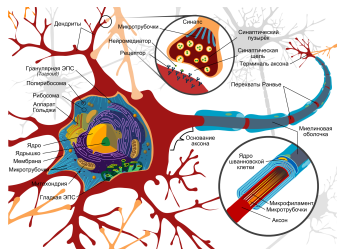
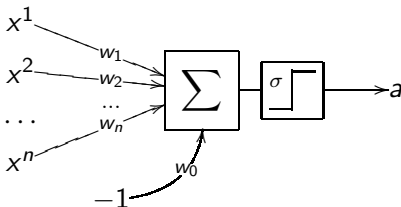
Напоминание: линейная модель нейрона МакКаллока-Питтса

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

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

где $w_0, w_1, \dots, w_n \in \mathbb{R}$ — веса признаков;

$\sigma(s)$ — функция активации (в частности, sign).



Линейные алгоритмы классификации и регрессии

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

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

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

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

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

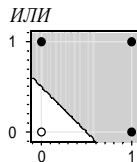
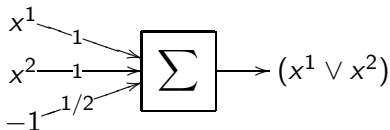
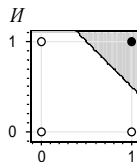
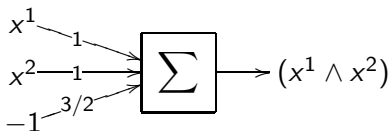
Нейронная реализация логических функций

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

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

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

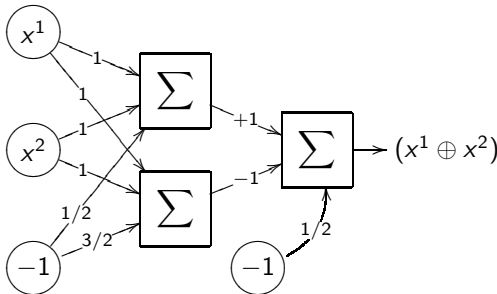
$$\neg x^1 = [-x^1 + \frac{1}{2} > 0] ;$$



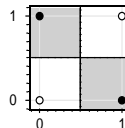
Логическая функция XOR (исключающее ИЛИ)

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

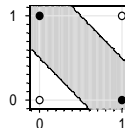
- Добавлением нелинейного признака:
$$x^1 \oplus x^2 = [x^1 + x^2 - 2x^1x^2 - \frac{1}{2} > 0];$$
- Сетью (двухслойной суперпозицией) функций И, ИЛИ, НЕ:
$$x^1 \oplus x^2 = [(x^1 \vee x^2) - (x^1 \wedge x^2) - \frac{1}{2} > 0].$$



1-й способ



2-й способ



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:

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

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* $\varphi : \mathbb{R} \rightarrow \mathbb{R}$, 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:

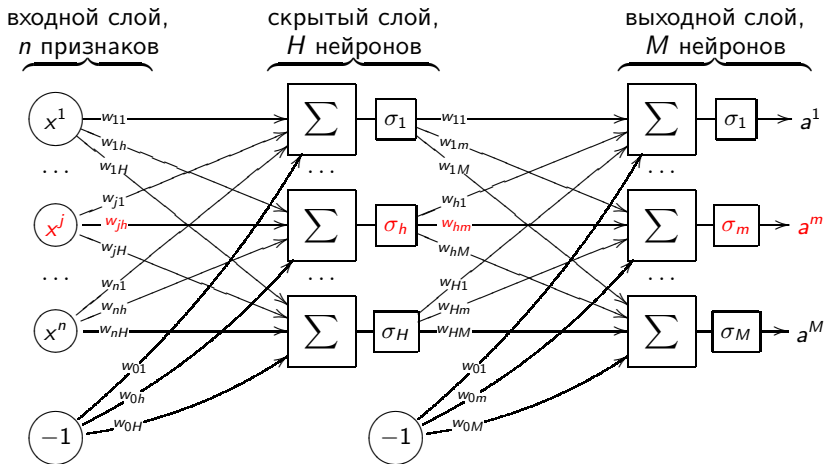
- С помощью линейных операций и одной нелинейной функции активации φ можно вычислять любую непрерывную функцию с любой желаемой точностью.
- Однако эти теоремы ничего не говорят о числе слоёв нейронной сети и о числе нейронов в каждом слое.

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:

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

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

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

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

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

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

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

Промежуточная задача: найти частные производные

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

Быстрое вычисление градиента

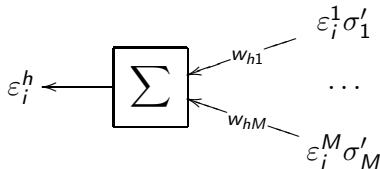
Промежуточная задача: частные производные

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

— это ошибка на выходном слое;

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

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



Быстрое вычисление градиента

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

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

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

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:
$$u_i^h := \sigma_h\left(\sum_{j=0}^J w_{jh}x_i^j\right), \quad h = 1..H;$$
$$a_i^m := \sigma_m\left(\sum_{h=0}^H w_{hm}u_i^h\right), \quad \varepsilon_i^m := a_i^m - y_i^m, \quad m = 1..M;$$
$$\mathcal{L}_i := \sum_{m=1}^M (\varepsilon_i^m)^2;$$
- 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: $Q := (1 - \lambda)Q + \lambda \mathcal{L}_i;$
- 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 , \mathcal{L} ;
- возможно динамическое (потокковое) 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 $Q(w) = \sum_{i=1}^{\ell} \mathcal{L}_i(w)$ на блоки (chunks).

Диагональный метод Левенберга-Марквардта

Метод Ньютона-Рафсона (второго порядка):

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

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

Эвристика. Считаем, что гессиан диагонален:

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

η — темп обучения,

μ — параметр, предотвращающий обнуление знаменателя.

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

Динамическое наращивание сети

- ❶ обучение при заведомо недостаточном числе нейронов N ;
- ❷ 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;
- ❸ 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:

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

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

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

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

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.

OBS — Optimal Brain Surgeon

N — количество прецедентов;

$r(n)$ — реальный отклик на n -й прецедент;

$d(n)$ — желаемый отклик на n -й прецедент.

$r(n)$ представим в виде $r(n) = F(w, x)$.

Вычисление матрицы обратной Гессияну:

$$Q(w) = \frac{1}{2N} \sum_{n=1}^N (d(n) - r(n))^2;$$

$$\frac{\partial Q}{\partial w} = -\frac{1}{N} \sum_{n=1}^N \frac{\partial F(w, x(n))}{\partial w} (d(n) - r(n));$$

$$Q'' = \frac{1}{N} \sum_{n=1}^N \left\{ \left(\frac{\partial F(w, x(n))}{\partial w} \right) \left(\frac{\partial F(w, x(n))}{\partial w} \right)^T - \left(\frac{\partial^2 F(w, x(n))}{\partial w^2} \right) (d(n) - r(n)) \right\}.$$

OBS — Optimal Brain Surgeon

Если сеть полностью обучена, $r(n)$ достаточно близко к $d(n)$.

$$Q'' \simeq \frac{1}{N} \sum_{n=1}^N \left(\frac{\partial F(w, x(n))}{\partial w} \right) \left(\frac{\partial F(w, x(n))}{\partial w} \right)^T.$$

Обозначим $\xi(n) = N^{-0,5} \frac{\partial F(w, x(n))}{\partial w}$.

Тогда Гессиан можно вычислить по рекуррентной формуле:

$$Q''(n) = \sum_{k=1}^n \xi(k) \xi^T(k);$$

А матрицу, обратную Гессиану:

$$Q''^{-1}(n) = Q''^{-1}(n-1) - \frac{Q''^{-1}(n-1) \xi(n) \xi^T(n) Q''^{-1}(n-1)}{1 + \xi^T(n) Q''^{-1}(n-1) \xi(n)}.$$

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;

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

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

$\rho: X \times X \rightarrow [0, \infty)$ — функция расстояния между объектами.

Требуется построить алгоритм кластеризации $a: X \rightarrow Y$:

- каждый кластер состоит из близких объектов;
- объекты разных кластеров существенно различны.

Предлагается:

- ввести центры кластеров $w_m \in \mathbb{R}^n$, $m = 1, \dots, M$;
- 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:

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

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

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

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

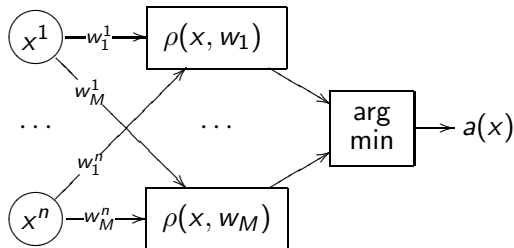
$$w_m := w_m + \eta(x_i - w_m) [a(x_i) = m]$$

(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:

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

Алгоритм SG (Stochastic Gradient)

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

Выход: центры кластеров $w_1, \dots, w_M \in \mathbb{R}^n$;

-
- 1: инициализировать центры w_m , $m = 1, \dots, M$;
 - 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):

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

Недостатки правила WTM:

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

Правило мягкой конкуренции WTM (winner takes most):

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

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)

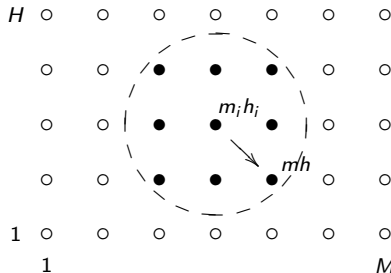
$Y = \{1, \dots, M\} \times \{1, \dots, H\}$ — прямоугольная сетка кластеров.

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 :

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

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** $(m, h) \in \text{Окрестность}(m_i, h_i)$
- 6: WTM: сделать шаг градиентного спуска:
 $w_{mh} := w_{mh} + \eta(x_i - w_{mh}) K(r((m_i, h_i), (m, h)));$
- 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:

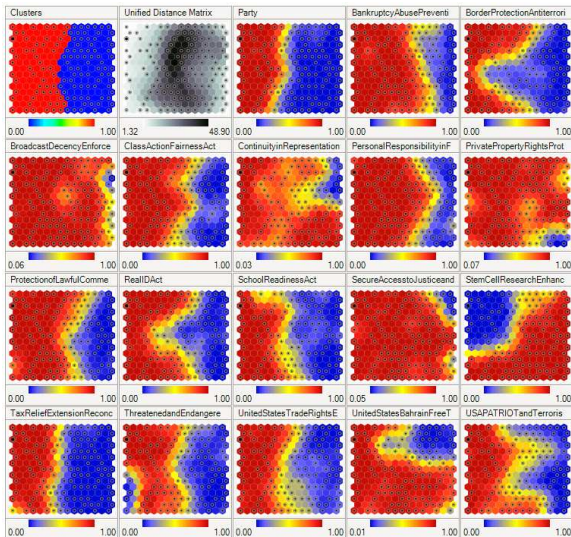
Задача UCI house-votes (US Congress voting patterns)

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 :

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

2) на каждом кластере построить регрессию-константу:

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

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:

$$Q(v) = \frac{1}{2} \sum_{i=1}^{\ell} (a(x_i) - y_i)^2 \rightarrow \min_v;$$
$$\frac{\partial Q}{\partial v_m} = \sum_{i=1}^{\ell} (a(x_i) - y_i) [c(x_i) = m] = 0.$$

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

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

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 :

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

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

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

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

Кусочно-гладкая непараметрическая регрессия

Настройка центров w_m сетью Кохонена — WTM:

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

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

Настройка уровней v_m линейным нейроном:

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

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

Кусочно-гладкая непараметрическая регрессия

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

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

Достоинства этого метода:

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

Резюме по сетям Кохонена

- Сеть Кохонена решает задачу кластеризации.
- Основные стратегии — мягкая WTM и жёсткая WTA.
- Мягкая конкуренция ускоряет сходимость.
- Карта Кохонена используется для визуализации многомерных данных, разведочного анализа данных, интерпретации кластеров по признакам.
- Карта Кохонена может быть субъективной и искажённой
- В сетях встречного распространения сначала решается задача кластеризации, затем — классификации или регрессии.