Лекции по искусственным нейронным сетям

К. В. Воронцов

21 декабря 2007 г.

Материал находится в стадии разработки, может содержать ошибки и неточности. Автор будет благодарен за любые замечания и предложения, направленные по адресу `voron@ccas.ru`. Перепечатка любых фрагментов данного материала без согласия автора является плагиатом.

Содержание

1 Искусственные нейронные сети

Человеку и высшим животным буквально на каждом шагу приходится распознавать, принимать решения и обучаться. Нейросетевой подход возник из стремления понять, каким образом мозг решает столь сложные задачи, и реализовать эти принципы в автоматических устройствах.

Пока *искусственные нейронные сети* (artificial neural networks, ANN) являются лишь предельно упрощёнными аналогами естественных нейронных сетей. Нервные системы животных и человека гораздо сложнее тех устройств, которые можно создать с помощью современных технологий. Однако для успешного решения многих практических задач оказалось вполне достаточно «подсмотреть» лишь общие принципы функционирования нервной системы. Некоторые разновидности ANN представляют собой математические модели, имеющие лишь отдалённое сходство с нейрофизиологией, что отнюдь не препятствует их практическому применению.

§1.1 Однослойные нейронные сети

1.1.1 Естественный нейрон и его формальная модель

Рассмотрим в общих чертах устройство и принципы работы нервной клетки — естественного *нейрона*. Клетка имеет множество разветвлённых отростков — *дендритов*, и одно длинное тонкое волокно — *аксон*, на конце которого находятся *синапсы*, примыкающие к дендритам других нервных клеток. Каждая нервная клетка может находиться в двух состояниях: обычном и возбуждённом. В возбуждённом состоянии клетка генерирует электрический импульс величиной около 100 мВ и длительностью 1 мс, который проходит по аксону до синапсов. Синапс при приходе импульса выделяет вещество, способствующее проникновению положительных зарядов внутрь соседней клетки. Синапсы имеют разную способность концентрировать это вещество, причём некоторые даже препятствуют его выделению — они называются *тормозящими*. Если суммарный заряд, попавший в клетку, превосходит некоторый порог, клетка возбуждается и генерирует импульс, который распространяется по аксону и доходит до синапсов, что способствует возбуждению следующих клеток. После возбуждения клетки наступает *период релаксации* — некоторое время она не способна генерировать новые импульсы. Благодаря этому клетки работают по тактам, наподобие дискретных автоматов, а сеть в целом передаёт направленную волну импульсов.

Скорость распространения нервного импульса составляет приблизительно 100 м/с, что в миллион раз меньше скорости распространения электрического сигнала в медной проволоке. Тем не менее, сложные задачи распознавания человек решает за десятые доли секунды. Это означает, что нейровычисления требуют порядка 10^2 последовательных тактов и выполняются с большой степенью параллелизма.

Кора головного мозга человека содержит порядка 10^{11} нейронов, и каждый нейрон связан с 10^3–10^4 других нейронов. Это обеспечивает высокую взаимозаменяемость нервных клеток и надежность нервной системы в целом. Отказ даже существенной доли нейронов не нарушает нормального хода распространения нервного импульса.

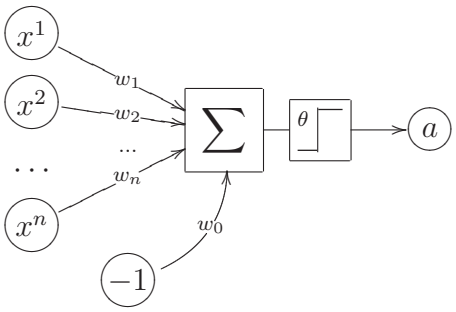


Рис. 1. Модель МакКаллока–Питтса.

Описанная выше схема сильно упрощена. На сегодняшний день нейрофизиологами выделено около 50 различных типов нейронов, которые функционируют поразному и не являются взаимозаменяемыми.

Модель МакКаллока–Питтса. Пусть X — пространство объектов; Y — множество допустимых ответов; $y^*\colon X \to Y$ — целевая зависимость, известная только на объектах обучающей выборки $X^\ell = (x_i, y_i)_{i=1}^\ell$, $y_i = y^*(x_i)$. Требуется построить алгоритм $a\colon X \to Y$, аппроксимирующий целевую зависимость y^* на всём множестве X.

Будем предполагать, что объекты описываются n числовыми признаками $f_j\colon X \to \mathbb{R}$, $j = 1, \dots, n$. Вектор $\big(f_1(x), \dots, f_n(x)\big) \in \mathbb{R}^n$ называется *признаковым описанием* объекта x.

В 1943 году МакКаллок и Питтс [12] предложили математическую модель нейрона, Рис. 1. Это алгоритм, принимающий на входе n-мерный вектор признакового описания $x = (x^1, \dots, x^n)$. Для простоты будем пока полагать, что все признаки бинарные. Значения этих признаков будем трактовать как величины импульсов, поступающих на вход нейрона через n входных синапсов. Поступающие в нейрон импульсы складываются с весами $w_1, \dots, w_n$. Если вес положительный, то соответствующий синапс возбуждающий, если отрицательный, то тормозящий. Если суммарный импульс превышает заданный *порог активации* w_0, то нейрон возбуждается и выдаёт на выходе 1, иначе выдаётся 0. Таким образом, нейрон вычисляет n-арную булеву функцию вида

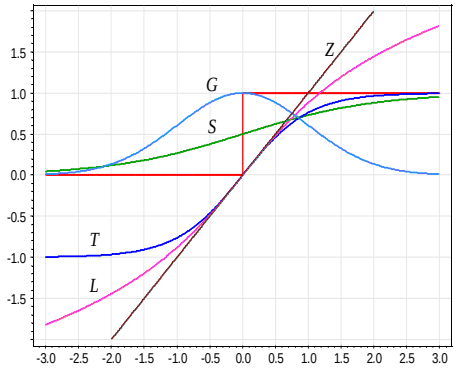
$$a(x) = \varphi\left(\sum_{j=1}^{n} w_j x^j - w_0\right),$$

где $\varphi(z) = [z \geqslant 0]$ — ступенчатая функция Хэвисайда. В теории нейронных сетей функцию φ, преобразующую значение суммарного импульса в выходное значение нейрона, принято называть *функцией активации*.

Введём дополнительный константный признак $x^0 \equiv -1$ и векторные обозначения $w = (w_0, w_1, \dots, w_n)^\intercal$, $x = (x^0, x^1, \dots, x^n)^\intercal$. Тогда линейную модель нейрона можно записать более компактно через скалярное произведение:

$$a(x) = \varphi\left(\sum_{j=0}^{n} w_j x^j\right) = \varphi(\langle w, x\rangle). \tag{1.1}$$

Итак, модель МакКаллока-Питтса эквивалентна линейному пороговому классификатору.



$\theta(z) = [z \geqslant 0]$	ступенчатая функция Хэвисайда;
$\sigma(z) = (1+e^{-z})^{-1}$	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);

Рис. 2. Функции активации.

Функции активации. Позже модель МакКаллока–Питтса была обобщена на случай произвольных вещественных входов и выходов, и произвольных функций активации. Чаще всего используются функции активации, показанные на Рис. 2.

1.1.2 Методы обучения синаптических весов нейрона

Персептрон Розенблатта. В 1957 году Розенблатт предложил эвристический алгоритм обучения нейрона, основанный на принципах, «подсмотренных» в нейрофизиологии. Экспериментально было обнаружено, что при синхронном возбуждении двух связанных нервных клеток синаптическая связь между ними усиливается. Чем чаще синапс угадывает правильный ответ, тем сильнее становится связь. Своеобразная тренировка связи приводит к постепенному запоминанию информации. Если же синапс начинает часто ошибаться или вообще перестаёт использоваться, связь ослабевает, информация начинается забываться. Таким образом, память реализуется в синапсах. В математической модели нейрона роль памяти играет вектор синаптических весов w.

Данное правило обучения нетрудно формализовать. Как признаки, так и ответы будем пока полагать бинарными. Перед началом обучения вектор весов некоторым способом инициализируется, например, заполняется нулевыми или случайными значениями. Затем обучающие объекты x_i по очереди подаются на вход модели МакКаллока–Питтса (1.1), и выданные ответы сравниваются с правильными.

Если ответ $a(x_i)$ совпадает с y_i, то вектор весов не изменяется.

Если $a(x_i) = 0$ и $y_i = 1$, то вектор весов w увеличивается. Увеличивать имеет смысл только те веса w_j, которые соответствуют ненулевым компонентам x_i^j, так как изменение других компонент не повлияет на результат. Поэтому можно положить $w := w + \eta x_i$, где η — некоторая положительная константа, называемая *темпом обучения* (learning rate).

Если $a(x_i) = 1$ и $y_i = 0$, то вектор весов уменьшается: $w := w - \eta x_i$.

Поскольку все величины бинарные, эти три случая легко объединить в одну формулу:

$$w := w - \eta\big(a(x_i) - y_i\big)x_i. \tag{1.2}$$

Обучающие объекты проходят через это правило многократно, до тех пор, пока веса изменяются, см. Алгоритм 1.1.

Алгоритм 1.1. Обучение персептрона Розенблатта

Вход:
X^ℓ — обучающая выборка;
η — темп обучения;
Выход:
синаптические веса $w_0, w_1, \ldots, w_n$;

1: инициализировать веса w_j;
2: **повторять**
3: **для всех** $i = 1, \ldots, \ell$
4: $w := w - \eta\big(a(x_i) - y_i\big)x_i$;
5: **пока** веса w изменяются;

Правило Хэбба. Иногда удобнее полагать, что классы помечены числами -1 и 1, а нейрон выдаёт знак скалярного произведения:

$$a(x) = \text{sign}(\langle w, x\rangle).$$

Тогда несовпадение знаков $\langle w, x_i\rangle$ и y_i означает, что нейрон ошибается на объекте x_i. При этом выражение (1.2) преобразуется в следующее правило модификации весов:

$$\textbf{если } \langle w, x_i\rangle\, y_i < 0 \textbf{ то } w := w + \eta x_i y_i, \tag{1.3}$$

называемое *правилом Хэбба*. Соответственно, в Алгоритме 1.1 заменяется шаг 4.

Для правила Хэбба докажем теорему сходимости. Она справедлива не только для бинарных, но и для произвольных действительных признаков.

Теорема 1.1 (Новиков, 1962 [14]). *Пусть $X = \mathbb{R}^{n+1}$, $Y = \{-1, 1\}$, и выборка X^ℓ линейно разделима — существует вектор $\tilde{w}$ и положительное число δ такие, что $\langle \tilde{w}, x_i\rangle\, y_i > \delta$ для всех $i = 1, \ldots, \ell$. Тогда Алгоритм 1.1 сходится за конечное число шагов к вектору весов, разделяющему обучающие объекты без ошибок, из любого начального положения w^0, при любом положительном η, независимо от порядка предъявления объектов. Если $w^0 = 0$, то достаточное число исправлений вектора весов не превосходит*

$$t_{\max} = \left(\frac{D}{\delta}\right)^2, \quad \text{где } D = \max_{x \in X^\ell} \|x\|.$$

Доказательство.

Запишем выражение для косинуса угла между вектором $\tilde{w}$ и вектором весов после t-го исправления w^t, полагая без ограничения общности $\|\tilde{w}\| = 1$:

$$\cos(\widehat{\tilde{w}, w^t}) = \frac{\langle \tilde{w}, w^t\rangle}{\|w^t\|}.$$

При t-м исправлении нейрону с вектором весов w^{t-1} предъявляется обучающий объект x, правильный ответ y, и при этом нейрон совершает ошибку: $\langle x, w^{t-1}\rangle\, y < 0$. Согласно правилу Хэбба (1.3) в этом случае происходит модификация весов. В силу условия линейной разделимости, справедлива оценка снизу:

$$\big\langle \tilde{w}, w^t\big\rangle = \big\langle \tilde{w}, w^{t-1}\big\rangle + \eta\,\langle \tilde{w}, x\rangle\, y > \big\langle \tilde{w}, w^{t-1}\big\rangle + \eta\delta > \big\langle \tilde{w}, w^0\big\rangle + t\eta\delta.$$

В силу ограниченности выборки, $\|x\| < D$, справедлива оценка сверху:

$$\|w^t\|^2 = \|w^{t-1}\|^2 + \eta^2\|x\|^2 + 2\eta\left\langle x, w^{t-1}\right\rangle y < \|w^{t-1}\|^2 + \eta^2 D^2 < \|w^0\|^2 + t\eta^2 D^2.$$

Подставим полученные соотношения в выражение для косинуса:

$$\cos(\widehat{\tilde{w}, w^t}) > \frac{\langle \tilde{w}, w^0\rangle + t\eta\delta}{\sqrt{\|w^0\|^2 + t\eta^2 D^2}} \to \infty \text{ при } t \to \infty.$$

Косинус не может превышать единицы. Следовательно, при некотором достаточно большом t не найдётся ни одного $x \in X^\ell$ такого, что $\langle x, w^t\rangle\, y < 0$, то есть выборка окажется поделенной безошибочно.

Если $w^0 = 0$, то нетрудно оценить сверху достаточное число исправлений. Из условия $\cos \leqslant 1$ следует $\sqrt{t}\delta/D \leqslant 1$, откуда $t_{\max} = (D/\delta)^2$. ■

На практике линейная разделимость выборки является скорее исключением, чем правилом. Если условия теоремы Новикова не выполнены, то процесс обучения вполне может оказаться расходящимся.

Метод стохастического градиента. Исходя из принципа минимизации эмпирического риска задача настройки синаптических весов может быть сведена к поиску вектора w, доставляющего минимум функционалу качества:

$$Q(w) = \sum_{i=1}^{\ell} \mathscr{L}\big(a(x_i), y_i\big) \to \min_w, \tag{1.4}$$

где $\mathscr{L}(a, y)$ — заданная функция потерь, характеризующая величину ошибки ответа a при правильном ответе y. Применим для минимизации $Q(w)$ метод градиентного спуска. Запишем правило изменения вектора весов на каждой итерации:

$$w := w - \eta\frac{\partial Q}{\partial w},$$

где $\eta > 0$ — величина шага в направлении антиградиента. Предполагая, что функция потерь $\mathscr{L}$ и функция активации φ дифференцируемы, распишем градиент:

$$w := w - \eta\sum_{i=1}^{\ell} \mathscr{L}'_a\big(a(x_i), y_i\big)\varphi'(\langle w, x_i\rangle)x_i.$$

Здесь каждый обучающий объект вносит свой аддитивный вклад в изменение вектора w, но вектор w изменяется только после предъявления всех ℓ объектов. Можно делать и по-другому — как и в персептроне Розенблатта, брать объекты по одному, и для каждого обновлять вектор весов. Этот метод называется *стохастическим градиентом* (stochastic gradient, SG). Объекты перебираются в случайном порядке, для каждого объекта x_i делается градиентный шаг и сразу обновляется вектор весов:

$$w := w - \eta\mathscr{L}'_a\big(a(x_i), y_i\big)\varphi'(\langle w, x_i\rangle)x_i.$$

Практика показывает, что такой процесс обучения сходится быстрее обычного градиентного метода. Когда объекты предъявляются в некотором фиксированном порядке, процесс чаще оказывается расходящимся или зацикливается.

Алгоритм 1.2. Обучение персептрона методом стохастического градиента.

Вход:
X^ℓ — обучающая выборка;
η — темп обучения;
Выход:
Синаптические веса $w_0, w_1, \ldots, w_n$;

1: инициализировать веса:
$w_j := \text{random}\left(-\frac{1}{2n}, \frac{1}{2n}\right)$;
2: инициализировать текущую оценку функционала:
$Q := \sum_{i=1}^{\ell} \mathscr{L}\big(a(x_i), y_i\big)$;
3: **повторять**
4: выбрать объект x_i из X^ℓ случайным образом;
5: вычислить выходное значение алгоритма $a(x_i)$ и ошибку:
$\varepsilon_i := \mathscr{L}\big(a(x_i), y_i\big)$;
6: сделать шаг градиентного спуска:
$w := w - \eta \mathscr{L}'_a\big(a(x_i), y_i\big)\varphi'(\langle w, x_i\rangle)x_i$;
7: оценить значение функционала:
$Q := \frac{\ell-1}{\ell}Q + \frac{1}{\ell}\varepsilon_i^2$;
8: **пока** значение Q не стабилизируется;

Пример 1.1 (Адаптивный линейный элемент). Пусть все признаки вещественны: $X = \mathbb{R}^n$, функция потерь квадратична: $\mathscr{L}(a, y) = (a - y)^2$, функция активации линейна: $\varphi(z) = z$. Тогда правило обновления весов в методе стохастического градиента в точности совпадает с правилом Розенблатта (1.2):

$$w := w - \eta\big(a(x_i) - y_i\big)x_i.$$

Это правило обучения было предложено Видроу и Хоффом в 1960 году и называется *дельта-правилом* (delta-rule), а сам линейный нейрон — *адаптивным линейным элементом* или ADALINE [18].

В Алгоритме 1.2 показана реализация метода SG при произвольных функциях $\mathscr{L}(a, y)$ и $\varphi(z)$. Обратим внимание на инициализацию весов и критерий останова. И то, и другое — не более чем эвристики.

Веса инициализируются небольшими по абсолютной величине случайными значениями, но это лишь один из возможных вариантов, неплохо зарекомендовавший себя на практике. Можно было бы инициализировать веса нулём.

Критерий останова основан на приблизительной оценке значения функционала Q. Вычислять его точно довольно накладно, так как это потребовало бы на каждой итерации пропускать все объекты выборки через алгоритм $a(x)$. Поэтому Q оценивается методом экспоненциальной скользящей средней. Когда градиентный метод подходит к окрестности минимума, значение Q стабилизируется, и оценка скользящей средней приближается к точному значению функционала.

Преимущества метода SG в том, что он легко реализуется, легко обобщается на нейронные сети более сложных архитектур, позволяет настраивать сети на выборках сколь угодно больших объёмов (за счёт того, что случайной подвыборки мо-

жет оказаться достаточно для обучения). Метод стохастического градиента считается классическим инструментом настройки нейронных сетей.

Недостаток SG в том, что сходимость в общем случае не гарантируется. На практике сходимость определяют методом проб и ошибок. Существует довольно обширный набор эвристик и рекомендаций, способствующих улучшению сходимости. Некоторые из них будут рассмотрены ниже, в разделе 1.2.2.

Сигмоидная функция активации $\sigma(z) = \frac{1}{1+e^{-z}}$ играет особую роль в задачах классификации. Рассмотрим для простоты случай двух классов, $Y = \{-1, +1\}$. Если справедливы условия Теоремы **??**, стр. **??**, то значение на выходе сигмоидной функции совпадает с апостериорной вероятностью класса y для объекта x:

$$\sigma\big(\langle w, x\rangle\, y\big) = \mathsf{P}\{y|x\}. \tag{1.5}$$

Таким образом, нейрон с сигмоидной функцией активации $a(x) = \sigma\big(\langle w, x\rangle\big)$ вычисляет вероятность принадлежности объекта x классу $+1$. Во многих приложениях оценка этой вероятности необходима для решения «сопутствующих» задач, в частности, для оценки рисков, связанных с возможными ошибками классификации.

В то же время, не стоит забывать, что равенство (1.5) получено при довольно сильных теоретико-вероятностных предположениях: функции правдоподобия классов должны принадлежать экспонентному семейству распределений, и классы должны иметь схожую форму, точнее, одинаковый параметр разброса. На практике гарантировать выполнение этих условий вряд ли возможно. Поэтому трактовать выходы сигмоидных функций активации как вероятности следует с большой осторожностью. На самом деле они дают лишь оценку удалённости объекта от границы классов, нормированную так, чтобы она принимала значения из отрезка $[0, 1]$.

Связь персептрона и логистической регрессии. Градиентная техника легко распространяется на другие функции активации и другие функционалы качества.

Рассмотрим задачу классификации с двумя классами, $Y = \{-1, +1\}$. Введём функционал качества, равный числу ошибок на обучающей выборке X^ℓ:

$$Q(w) = \sum_{i=1}^{\ell} \big[\langle w, x_i\rangle\, y_i < 0\big] \to \min_w.$$

Рассматривая логистическую регрессию в разделе **??**, мы заменили этот функционал его гладкой аппроксимацией, Рис. 3:

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

где σ — сигмоидная функция. Оказывается, выражение, стоящее в этой формуле под знаком логарифма, тесно связано с функциями правдоподобия классов $p_y(x)$, $y \in Y$. Это легко показать, воспользовавшись определением условной вероятности, соотношением (1.5), а также тем фактом, что априорные вероятности классов P_y и вероятности появления объектов $p(x)$ не зависят от весов w:

$$p_y(x) = \mathsf{P}\{x|y\} = \mathsf{P}\{y|x\} \cdot p(x)/P_y = \sigma\big(\langle w, x\rangle\, y\big) \cdot \mathrm{const}(w).$$

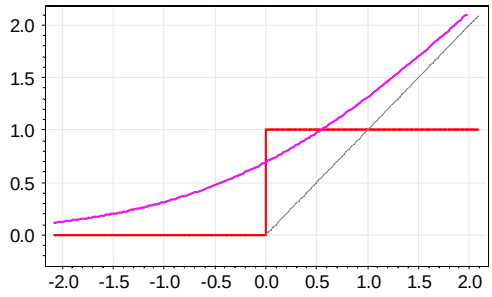


Рис. 3. Логарифмическая аппроксимация пороговой функции потерь.

Отсюда следует, что, если верно (1.5), то минимизация функционала $\widetilde{Q}(w)$ эквивалентна принципу максимума правдоподобия:

$$L(X^\ell; w) = \ln \prod_{i=1}^{\ell} p_{y_i}(x_i) = -\widetilde{Q}(w) + \text{const}(w) \to \max_w .$$

Данное обстоятельство служит дополнительным обоснованием для использования функционала $\widetilde{Q}(w)$ в качестве критерия настройки линейного классификатора.

Рассмотрим теперь градиентный метод обучения нейрона с сигмоидной функцией активации. Распишем градиент функционала $\widetilde{Q}(w)$, воспользовавшись выражением для производной сигмоидной функции $\sigma' = \sigma(1-\sigma)$:

$$\frac{\partial \widetilde{Q}}{\partial w} = -\sum_{i=1}^{\ell} \big(1 - \sigma\big(\langle w, x_i\rangle\, y_i\big)\big) y_i x_i.$$

Тогда в методе стохастического градиента правило обновления весов при предъявлении прецедента x_i, y_i будет иметь вид:

$$w := w + \eta\big(1 - \sigma\big(\langle w, x_i\rangle\, y_i\big)\big) y_i x_i.$$

Сопоставим его с правилом Хэбба:

$$w := w + \eta\big[\langle w, x_i\rangle\, y_i < 0\big] y_i x_i.$$

Легко видеть, что логистичекое правило есть ни что иное, как сглаженный вариант правила Хэбба. В правиле Хэбба смещение весов происходит только когда на объекте x_i допускается ошибка. В логистическом правиле смещение тем больше, чем меньше значение $\langle w, x_i\rangle\, y_i$, т. е. чем серьёзнее ошибка. Даже если ошибки нет, но объект близок к границе классов, веса модифицируются так, чтобы граница прошла как можно дальше от объекта. Стратегия увеличения *зазора* (margin) между объектами обучающей выборки и разделяющей поверхностью способствует повышению качества классификации и увеличению скорости сходимости [6, 16].

В остальном алгоритм обучения повторяет персептрон Розенблатта.

Заметим также, что описанный метод настройки весов существенно проще метода логистической регрессии IRLS, рассмотренного в **??**. Различие в том, что здесь применялся метод первого порядка (градиентный спуск), в то время как IRLS основан на методе второго порядка (Ньютона-Рафсона), имеющем лучшие характеристики сходимости, но требующим обращения матрицы на каждой итерации.

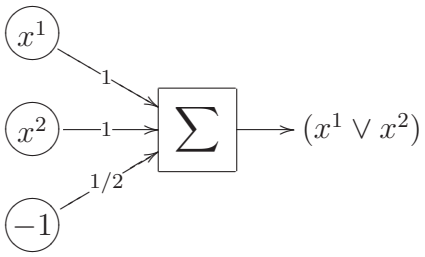


Рис. 4. Однослойный персептрон, реализующий операцию ИЛИ.

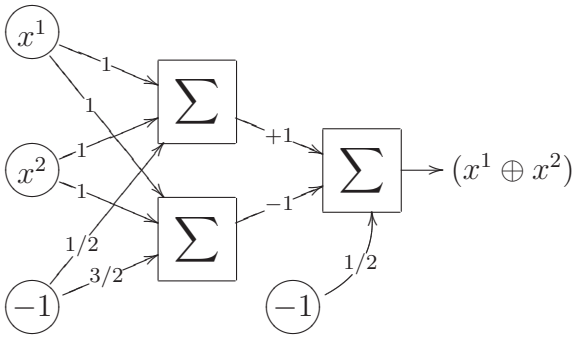


Рис. 5. Двухслойная сеть, реализующая операцию исключающего ИЛИ.

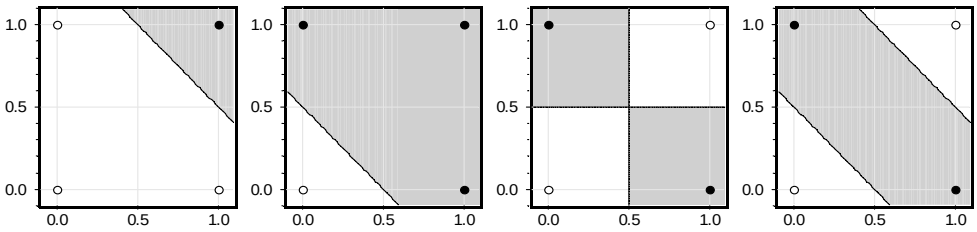


Рис. 6. Персептронная реализация элементарных логических функций: И, ИЛИ, исключающее ИЛИ двумя способами — добавлением признака x^1x^2 и двухслойной сетью.

1.1.3 Проблема полноты

Итак, отдельно взятый нейрон вида (1.1) позволяет реализовать линейный классификатор или линейную регрессию. При решении практических задач линейность оказывается чрезмерно сильным ограничением. На ограниченность персептрона указывали Минский и Пайперт в своей знаменитой книге «Персептроны» [13]. Следующий классический контрпример иллюстрирует невозможность нейронной реализации даже очень простых функций.

Проблема «исключающего ИЛИ». Легко построить нейроны, реализующие логические функции И, ИЛИ, НЕ от бинарных переменных x^1 и x^2, см. Рис. 4:

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

Однако функцию $x^1 \oplus x^2 = [x^1 \neq x^2]$, называемую *исключающим ИЛИ*, принципиально невозможно реализовать одним нейроном с двумя входами x^1 и x^2, поскольку множества нулей и единиц этой функции линейно неразделимы.

Возможны два пути решения этой проблемы, см. Рис 6.

Первый путь — пополнить состав признаков, подавая на вход нейрона нелинейные преобразования исходных признаков. В частности, если разрешить образовывать всевозможные произведения исходных признаков, то нейрон будет строить уже не линейную, а полиномиальную разделяющую поверхность. В случае исключающего ИЛИ достаточно добавить только один вход x^1x^2, чтобы в расширенном

пространстве множества нулей и единиц оказались линейно разделимыми:

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

Такие расширенные пространства признаков называют *спрямляющими*. Проблема заключается в том, что подбор нужных нелинейных преобразований является нетривиальной задачей, которая для общего случая до сих пор остаётся нерешённой.

Второй путь — построить композицию из нескольких нейронов. Например, исключающее ИЛИ можно реализовать, подав выходы И-нейрона и ИЛИ-нейрона на вход ещё одному ИЛИ-нейрону, Рис 5:

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

Дальнейшее обобщение этой идеи приводит к построению многослойных нейронных сетей, состоящих из огромного количества связанных нейронов и напоминающих естественные нейронные сети.

Многослойные нейронные сети. Рассмотрим композицию нейронов, представленную на Рис. 7. Значения всех n признаков одновременно подаются на вход всех H нейронов первого слоя. Затем их выходные значения подаются на вход всех M нейронов следующего слоя. В данном случае этот слой является выходным — такая сеть называется *двухслойной*.[1] В общем случае сеть может содержать произвольное число слоёв. Все слои, за исключением последнего, называются *скрытыми* (hidden layers).

Вычисление выходных значений сети может осуществляться с высокой степенью параллелизма, за число тактов, равное числу слоёв. Существуют эффективные аппаратные реализации нейронных сетей, в которых вычисления действительно происходят параллельно. Но пока на практике чаще используются программные реализации, выполненные на обычных последовательных компьютерах.

Вычислительные возможности нейронных сетей. Возникает вопрос: любую ли функцию можно представить (хотя бы приближённо) с помощью нейронной сети? Следующие факты позволяют ответить на этот вопрос утвердительно.

1. Любая булева функция представима в виде двухслойной сети. Это тривиальное следствие нейронной представимости функций И, ИЛИ, НЕ и представимости произвольной булевой функции в виде дизъюнктивной нормальной формы [5].

2. Из простых геометрических соображений вытекает, что двухслойная сеть с пороговыми функциями активации позволяет выделить произвольный выпуклый многогранник в n-мерном пространстве признаков. Трёхслойная сеть позволяет вычислить любую конечную линейную комбинацию характеристических функций выпуклых многогранников, следовательно, аппроксимировать любые области с непрерывной границей, включая неодносвязные, а также аппроксимировать любые непрерывные функции.

3. В 1900 году Гильберт предложил список из 23 нерешённых задач, которые, по его мнению, должны были стать вызовом для математиков XX века. Тринадцатая проблема заключалась в следующем: возможно ли произвольную непрерывную

[1] Существует терминологическая путаница с подсчётом числа слоёв. Иногда такую сеть (видимо, глядя на картинку) называют трёхслойной, считая входы $x^0, x^1, \ldots, x^n$ особым, «распределительным» слоем. По делу, в счёт должны идти только слои, состоящие из суммирующих нейронов.

функцию n аргументов представить в виде суперпозиции функций меньшего числа аргументов. Ответ был дан А. Н. Колмогоровым в [2].

Теорема 1.2 (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, причём φ_{ik} не зависят от выбора f.

Нетрудно видеть, что записанное здесь выражение имеет структуру нейронной сети с одним скрытым слоем. Таким образом, двух слоёв уже достаточно, чтобы вычислять произвольные непрерывные функции, и не приближённо, а точно. К сожалению, представление Колмогорова не является персептроном: функции φ_{ik} не линейны, а функции h_k зависят от f, и в общем случае не являются дифференцируемыми.

4. Известна классическая теорема Вейерштрасса о том, что любую непрерывную функцию n переменных можно равномерно приблизить полиномом с любой степенью точности. Более общая теорема Стоуна утверждает, что любую непрерывную функцию на произвольном компакте X можно приблизить не только многочленом от исходных переменных, но и многочленом от любого конечного набора функций F, разделяющих точки [17].

Опр. 1.1. *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 1.3 (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 $C(X)$.*

На самом деле справедливо ещё более общее утверждение. Оказывается, вместо многочленов (суперпозиции операций сложения и умножения) можно пользоваться суперпозициями сложения и какой-нибудь (практически произвольной) непрерывной нелинейной функции одного аргумента [3]. Этот результат имеет прямое отношение к нейронным сетям, поскольку они строятся из операций сложения, умножения и нелинейных функций активации.

Опр. 1.2. *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 1.4 (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 $C(X)$.*

Это интерпретируется как утверждение об универсальных аппроксимационных возможностях произвольной нелинейности: с помощью линейных операций и единственного нелинейного элемента φ можно получить устройство, вычисляющее любую

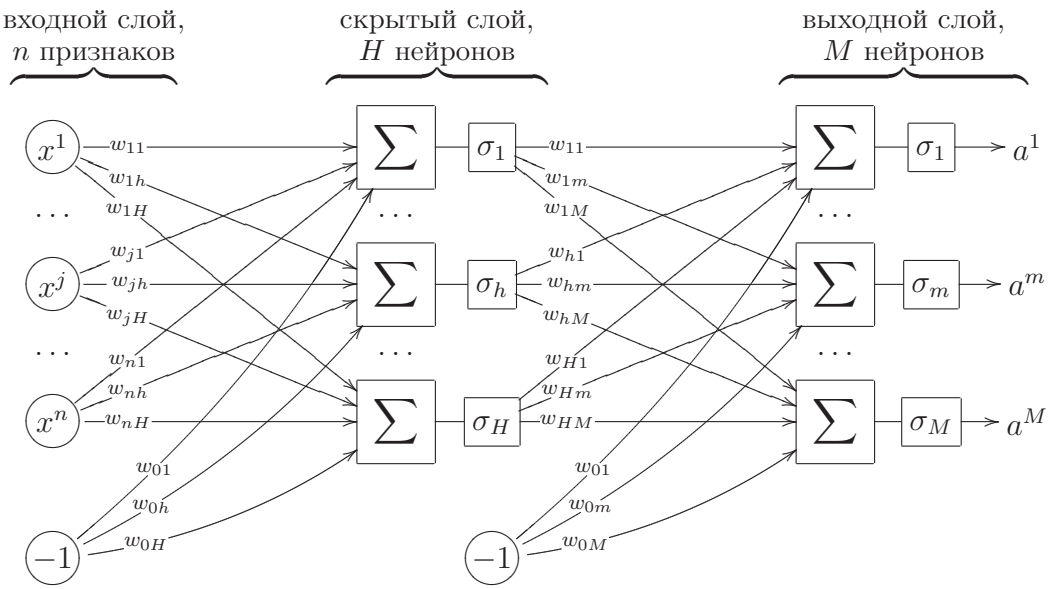


Рис. 7. Многослойная сеть с одним скрытым слоем.

непрерывную функцию с любой желаемой точностью. Однако данная теорема ничего не говорит о количестве слоёв нейронной сети (уровней вложенности суперпозиции) и о количестве нейронов, необходимых для аппроксимации произвольной функции.

Таким образом, нейронные сети являются универсальными аппроксиматорами функций. Возможности сети возрастают с увеличением числа слоёв и числа нейронов в них. Двух-трёх слоёв, как правило, достаточно для решения подавляющего большинства практических задач классификации, регрессии и прогнозирования.

§1.2 Многослойные нейронные сети

Многослойные сети также можно настраивать градиентными методами, несмотря на огромное количество весовых коэффициентов. В середине 80-х одновременно несколькими исследователями был предложен эффективный способ вычисления градиента, при котором каждый градиентный шаг выполняется за число операций, лишь немногим большее, чем при обычном вычислении сети на одном объекте. Это кажется удивительным — ведь количество операций, необходимых для вычисления градиента, обычно возрастает пропорционально числу весовых коэффициентов. Здесь этого удаётся избежать благодаря аналитическому дифференцированию суперпозиции с сохранением необходимых промежуточных величин. Метод получил название *обратного распространения ошибок* (error back-propagation) [15].

1.2.1 Метод обратного распространения ошибок

Рассмотрим многослойную сеть, в который каждый нейрон предыдущего слоя связан со всеми нейронами последующего слоя, Рис. 7. Для большей общности положим $X = \mathbb{R}^n$, $Y = \mathbb{R}^M$.

Введём следующие обозначения. Пусть выходной слой состоит из M нейронов с функциями активации σ_m и выходами a^m, $m = 1, \ldots, M$. Перед ним находится скрытый слой из H нейронов с функциями активации σ_h и выходами u^h, $h = 1, \ldots, H$.

Веса синаптических связей между h-м нейроном скрытого слоя и m-м нейроном выходного слоя будем обозначать через w_{hm}. Перед этим слоем может находиться либо распределительный слой, либо ещё один скрытый слой с выходами v^j, $j = 1, \ldots, J$ и синаптическими весами w_{jh}. В общем случае число слоёв может быть произвольным. Если сеть двухслойная, то v^j есть просто j-й признак: $v^j(x) \equiv f_j(x) \equiv x^j$, и $J = n$. Обозначим через w вектор всех синаптических весов сети.

Выходные значения сети на объекте x_i вычисляются как суперпозиция:

$$a^m(x_i) = \sigma_m\left(\sum_{h=0}^{H} w_{hm} u^h(x_i)\right); \qquad u^h(x_i) = \sigma_h\left(\sum_{j=0}^{J} w_{jh} v^j(x_i)\right). \tag{1.6}$$

Запишем функционал среднеквадратичной ошибки для отдельного объекта x_i:

$$Q(w) = \frac{1}{2}\sum_{m=1}^{M}\left(a^m(x_i) - y_i^m\right)^2. \tag{1.7}$$

В дальнейшем нам понадобятся частные производные Q по выходам нейронов. Выпишем их сначала для выходного слоя:

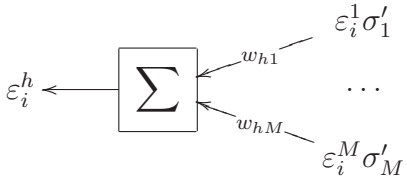
$$\frac{\partial Q(w)}{\partial a^m} = a^m(x_i) - y_i^m = \varepsilon_i^m.$$

Оказывается, частная производная Q по a^m равна величине ошибки ε_i^m на объекте x_i. Теперь выпишем частные производные по выходам скрытого слоя:

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

Эту величину, по аналогии с ε_i^m, будем называть ошибкой сети на скрытом слое и обозначать через ε_i^h. Через σ'_m обозначена производная функции активации, вычисленная при том же значении аргумента, что и в (1.6). Если используется сигмоидная функция активации, то для эффективного вычисления производной можно воспользоваться формулой $\sigma'_m = \sigma_m(1 - \sigma_m)$.

Заметим, что ε_i^h вычисляется по ε_i^m, если запустить сеть «задом наперёд», подав на выходы нейронов скрытого слоя значения $\varepsilon_i^m \sigma'_m$, а результат ε^h получив на входе. При этом входной вектор скалярно умножается на вектор весов w_{hm}, находящихся справа от нейрона, а не слева, как при прямом вычислении (отсюда и название алгоритма — *обратное распространение ошибок*):



Имея частные производные по a^m и u^h, легко выписать градиент Q по весам:

$$\frac{\partial Q(w)}{\partial w_{hm}} = \frac{\partial Q(w)}{\partial a^m}\frac{\partial a^m}{\partial w_{hm}} = \varepsilon_i^m \sigma'_m u^h, \qquad m = 1, \ldots, M, \quad h = 0, \ldots, H; \tag{1.8}$$

$$\frac{\partial Q(w)}{\partial w_{jh}} = \frac{\partial Q(w)}{\partial u^h}\frac{\partial u^h}{\partial w_{jh}} = \varepsilon_i^h \sigma'_h v^j, \qquad h = 1, \ldots, H, \quad j = 0, \ldots, J; \tag{1.9}$$

Алгоритм 1.3. Обучение двухслойной сети методом back-propagation — обратного распространения ошибки

Вход:
$X^\ell = (x_i, y_i)_{i=1}^\ell$ — обучающая выборка, $x_i \in \mathbb{R}^n$, $y_i \in \mathbb{R}^M$;
H — число нейронов в скрытом слое;
η — темп обучения;

Выход:
синаптические веса w_{jh}, w_{hm};

1: инициализировать веса небольшими случайными значениями:
$w_{jh} := \text{random}\left(-\frac{1}{2n}, \frac{1}{2n}\right)$;
$w_{hm} := \text{random}\left(-\frac{1}{2H}, \frac{1}{2H}\right)$;
2: **повторять**
3: выбрать объект x_i случайным образом;
4: прямой ход:
$u_i^h := \sigma_h\left(\sum_{j=0}^J w_{jh} v^j(x_i)\right)$, для всех $h = 1, \ldots, H$;
$a_i^m := \sigma_m\left(\sum_{h=0}^H w_{hm} u^h(x_i)\right)$, для всех $m = 1, \ldots, M$;
$\varepsilon_i^m := a_i^m - y_i^m$, для всех $m = 1, \ldots, M$;
$Q_i := \sum_{m=1}^M (\varepsilon_i^m)^2$;
5: обратный ход:
$\varepsilon_i^h := \sum_{m=1}^M \varepsilon_i^m \sigma'_m w_{hm}$, для всех $h = 1, \ldots, H$;
6: градиентный шаг:
$w_{hm} := w_{hm} - \eta \varepsilon_i^m \sigma'_m u^h$, для всех $h = 0, \ldots, H$, $m = 1, \ldots, M$;
$w_{jh} := w_{jh} - \eta \varepsilon_i^h \sigma'_h x^j$, для всех $j = 0, \ldots, n$, $h = 1, \ldots, H$;
7: $Q := \frac{\ell-1}{\ell} Q + \frac{1}{\ell} Q_i$;
8: **пока** Q не стабилизируется;

и так далее для каждого слоя. Если слоёв больше двух, то остальные частные производные вычисляются аналогично — обратным ходом по слоям сети справа налево.

Теперь мы обладаем всем необходимым, чтобы полностью выписать алгоритм обратного распространения, см. Алгоритм 1.3.

Достоинства метода обратного распространения.

- Достаточно высокая эффективность. Прямой ход, обратный ход и вычисления градиента требуют порядка $O(Hn + HM)$ операций.
- Через каждый нейрон проходит информация только о связных с ним нейронах. Поэтому back-propagation легко реализуется на вычислительных устройствах с параллельной архитектурой.
- Высокая степень общности. Алгоритм легко записать для произвольного числа слоёв, произвольной размерности выходов и выходов, произвольной функции потерь и произвольных функций активации, возможно, различных у разных нейронов. Кроме того, back-propagation не накладывает никаких ограничений на используемый метод оптимизации. Его можно применять вместе с методом скорейшего спуска, сопряженных градиентов, Ньютона-Рафсона и др.

Недостатки метода обратного распространения.

- Метод не всегда сходится. Для улучшения сходимости приходится применять большое количество различных эвристических ухищрений.
- Процесс градиентного спуска склонен застревать в многочисленных локальных минимумах функционала Q.
- Приходится заранее фиксировать число нейронов скрытого слоя H. В то же время, это критичный параметр сложности сети, от которого может существенно зависеть качество обучения и скорость сходимости.
- При чрезмерном увеличении числа весов сеть склонна к переобучению.
- Если применяются функции активации с горизонтальными асимптотами, типа сигмоидной или th, то сеть может попадать в состояние «паралича». Чем больше значения синаптических весов на входе нейрона, тем ближе значение производной σ' к нулю, тем меньше изменение синаптических весов в соответствии с формулами (1.8)–(1.9). Если нейрон один раз попадает в такую «мёртвую зону», то у него практически не остаётся шансов из неё выбраться. Парализоваться могут отдельные связи, нейроны, или вся сеть в целом.

1.2.2 Улучшение сходимости и качества градиентного обучения

В этом разделе рассматриваются эвристические приёмы, позволяющие с бóльшим или меньшим успехом преодолеть недостатки метода обратного распространения. Различных тонкостей настолько много, что умение хорошо настроить нейронную сеть по праву считается искусством, см. обзор [10].

Нормализация данных. Процесс настройки сети чувствителен к различиям в масштабах измерения признаков. В случае больших различий сеть может оказаться парализованной. Поэтому общей практикой при градиентном обучении является предварительная *нормализация* признаков:

$$x^j := \frac{x^j - x^j_{\min}}{x^j_{\max} - x^j_{\min}}, \quad \text{либо} \quad x^j := \frac{x^j - x^j_{\text{ср}}}{x^j_{\text{ско}}}, \quad j = 1, \ldots, n,$$

где $x^j_{\min}$, $x^j_{\max}$, $x^j_{\text{ср}}$, $x^j_{\text{ско}}$ — соответственно минимальное, максимальное, среднее значения и среднеквадратичное отклонение признака x^j.

Выбор функций активации. Сигмоидная функция $\sigma(z) = (1+e^{-z})^{-1}$ часто применяется при решении задач классификации. Её преимущества — возможность оценить вероятность принадлежности объекта классу по формуле (1.5), эффективность вычисления производной, ограниченность выходного значения. Применение нечётных функций, таких как $\text{th}(z) = 2\sigma(2z) - 1$, увеличивает скорость сходимости примерно в полтора раза.

Для предотвращения эффекта паралича имеет смысл добавлять небольшой линейный член: $\text{th}\, z + \gamma z$. Можно использовать и другие медленно растущие функции.

Применение функции активации $\ln(z+\sqrt{z^2+1})$ для решения задач прогнозирования рассмотрено в [1].

Ещё одна скрытая опасность связана с применением функционала среднеквадратичной невязки (1.7) в задачах классификации. Если значения меток классов $\{-1,+1\}$ совпадают со значениями горизонтальных асимптот функции активации выходного слоя, то оптимизационный процесс будет стремиться к неограниченному увеличению аргумента функции активации, следовательно, к параличу выходного нейрона. Таким образом, значения меток классов должны находиться внутри области значений функции активации. В работе [10] рекомендуется использовать функцию активации $1.7159\,\mathrm{th}(\frac{2}{3}z)$, у которой в точках $\{-1,+1\}$ достигается максимум второй производной.

Выбор начального приближения. Из тех же соображений предотвращения паралича синаптические веса должны инициализироваться небольшими по модулю значениями. В Алгоритме 1.3 на шаге 1 веса инициализируются случайными значениями из отрезка $\left[-\frac{1}{2k}, \frac{1}{2k}\right]$, где k — число нейронов в том слое, из которого выходит данный синапс. В этом случае (и при условии, что все признаки нормализованы) значения скалярных произведений гарантированно попадают в «рабочую зону» функций активации, представленных на Рис. 2.

Существует и более тонкий способ формирования начального приближения. Идея заключается в том, чтобы сначала настроить нейроны первого слоя по-отдельности, как H однослойных персептронов. Затем по-отдельности настраиваются нейроны второго слоя, которым на вход подаётся вектор выходных значений первого слоя. Чтобы сеть не получилась вырожденной, нейроны первого слоя должны быть существенно различными. Ещё лучше, если они будут хоть как-то приближать целевую зависимость, тогда второму слою останется только усреднить результаты первого слоя, сгладив ошибки некоторых нейронов[2]. Добиться этого совсем несложно, если обучать нейроны первого слоя на различных случайных подвыборках, либо подавать им на вход различные случайные подмножества признаков. Отметим, что при формировании начального приближения не требуется особая точность настройки, поэтому отдельные нейроны можно обучать простейшими градиентными методами.

Порядок предъявления объектов. Кроме стандартной рекомендации использовать метод стохастического градиента (т. е. предъявлять объекты в случайном порядке), имеются ещё следующие соображения.

1. Наибольшее смещение весов ожидается для того объекта, который наименее похож на объекты, предъявленные до него. В общем случае довольно трудно указать, какой из объектов максимально информативен на данном шаге обучения. Простая для реализации эвристика заключается в том, чтобы попеременно предъявлять объекты из разных классов, поскольку объекты одного класса с большей ве-

[2]Фактически, такая двуслойная сеть является простейшей алгоритмической композицией. Нейроны первого скрытого слоя играют роль *базовых алгоритмов*, нейроны второго слоя реализуют *корректирующую операцию*. Обучение первого слоя по случайным подвыборкам с последующим взвешенным усреднением во втором слое в точности соответствует методу *бэггинга* (bagging), см. раздел **??**. Композиции общего вида рассматриваются в главе **??**.

роятностью содержат схожую информацию. Эта техника называется *перетасовкой объектов* (shuffling).

2. Ещё одна эвристика состоит в том, чтобы чаще предъявлять те объекты, на которых была допущена ошибка. Для этого вероятность появления каждого объекта вычисляется в соответствии с величиной ошибки сети на данном объекте. Эту эвристику рекомендуется применять только в тех случаях, когда исходные данные не содержат выбросов, иначе процесс обучения может сосредоточиться на шумовых объектах, которые вообще следовало бы исключить из обучающей выборки.

3. Другой вариант предыдущей эвристики отличается простотой реализации и заключается в том, чтобы после прямого хода (шага 4 в Алгоритме 1.3) сравнить величину ошибки на i-м объекте с некоторым порогом. Если ошибка окажется меньше порога, вектор весов не модифицируется. Иначе выполняется обратный ход, вычисляется градиент и изменяются веса (шаги 5 и 6). Логика этой эвристики в точности та же, что у персептрона Розенблатта: если объект уже неплохо классифицируется, то менять веса не нужно. При этом увеличивается и скорость настройки.

Сокращение весов является частным случаем регуляризации некорректно поставленных задач по А. Н. Тихонову [4]. Аналогичный приём применяется в линейном дискриминантном анализе и в линейной регрессии. Идея заключается в том, чтобы ограничить возможный рост абсолютных значений весов, добавив к минимизируемому функционалу $Q(w)$ штрафное слагаемое:

$$Q_\tau(w) = Q(w) + \frac{\tau}{2}\|w\|^2.$$

Изменение функционала приводит к появлению аддитивной поправки у градиента:

$$\frac{\partial Q_\tau(w)}{\partial w} = \frac{\partial Q(w)}{\partial w} + \tau w.$$

При этом правило обновления весов принимает вид

$$w := w(1 - \eta\tau) - \eta\frac{\partial Q(w)}{\partial w}.$$

Таким образом, вся модификация алгоритма сводится к появлению неотрицательного множителя $(1-\eta\tau)$, приводящего к постоянному уменьшению весов. Отсюда название метода — *сокращение весов* (weights decay).

Преимущества метода в том, что он очень просто реализуется, сочетается со многими функционалами и многими градиентными методами оптимизации. Он предотвращает паралич сети и повышает устойчивость весов в случае мультиколлинеарности — когда имеются линейно зависимые или сильно коррелированные признаки. В конечном итоге сокращение весов способствует повышению обобщающей способности алгоритма и снижению риска переобучения.

Дополнительный управляющий параметр τ позволяет найти компромисс между точностью настройки на конкретную выборку и устойчивостью весов.

Недостаток метода в том, что параметр τ приходится подбирать в режиме скользящего контроля, что связано с большими затратами времени.

Сокращение весов уменьшает эффективную сложность сети, но количество параметров остаётся тем же. Это означает, что ресурс на хранение и вычисление сети

расходуется не продуктивно. В некоторых случаях более предпочтительным является альтернативный метод, в котором избыточные синаптические связи не уменьшают свой вес, а полностью удаляются. Он будет рассмотрен чуть позже.

Выбор величины шага.

1. Известно, что градиентные методы сходятся при определённых условиях, если величину шага η уменьшать с числом итераций t. Точнее, сходимость гарантируется при $\eta_t \to 0$, $\sum_{t=1}^{\infty} \eta_t = \infty$, $\sum_{t=1}^{\infty} \eta_t^2 < \infty$, в частности можно положить $\eta_t = 1/t$. При настройке нейронных сетей дополнительные условия сходимости могут не выполняться, поэтому стратегию постепенного уменьшения шага следует воспринимать скорее как эвристическую рекомендацию.

2. Метод скорейшего градиентного спуска приводит к выбору *адаптивного шага* η исходя из решения одномерной задачи минимизации $Q\left(w - \eta \frac{\partial Q}{\partial w}\right) \to \min\limits_{\eta}$. Во многих случаях эту задачу удаётся решить аналитически. В частности, для алгоритма ADALINE с линейной функцией активации

$$\eta = \frac{1}{1 + \sum_{j=0}^{n} \left(x_i^j\right)^2} = \frac{1}{1 + \|x_i\|^2}. \tag{1.10}$$

Формулы вычисления адаптивного шага получены и для более сложных случаев, в том числе для метода обратного распространения ошибок [1].

Выбивание сети из локальных минимумов обязательно должно быть предусмотрено в любой хоть сколько-нибудь серьёзной реализации back-propagation. Один из простейших способов заключается в том, чтобы при каждой стабилизации функционала ошибки производить случайные модификации вектора весов в довольно большой окрестности текущего значения и запускать процесс градиентного спуска из новых точек. Этот способ называют *потряхиванием коэффициентов* (jog of weights). По сути дела, он является симбиозом градиентного метода и *случайного локального поиска* (stochastic local search).

Выбор критерия останова. В Алгоритме 1.3 в качестве критерия останова используется условие стабилизации среднеквадратичной ошибки Q. Точное вычисление этой величины потребовало бы пропускать через сеть все обучающие объекты после каждой итерации. Разумеется, это недопустимо из соображений эффективности. Вместо этого функционал Q оценивается приближённо, как экспоненциальное скользящее среднее ошибок Q_i, допущенных на объектах x_i, при этом больший вес получают объекты, предъявленные последними (шаг 7).

Ранний останов. Слишком глубокая оптимизация также может привести к переобучению. Узкий глобальный минимум функционала Q менее предпочтителен, чем более пологий и устойчивый локальный минимум. Для предотвращения попадания в такие «расщелины» применяется техника *раннего останова* (early stopping). По ходу итераций вычисляется какой-нибудь *внешний критерий* (стр. **??**), и если он начинает возрастать, процесс настройки прекращается.

Выбор градиентного метода оптимизации. К сожалению, градиентные методы первого порядка сходятся довольно медленно, и потому редко применяются на практике. Ньютоновские методы второго порядка также непрактичны, но по другой причине — они требуют вычисления матрицы вторых производных функционала $Q(w)$, имеющей слишком большой размер. Метод сопряжённых градиентов этого не требует, однако применять его непосредственно нельзя, так как он существенно опирается на предположение неизменности функционал $Q(w)$, а в методе стохастического градиента функционал меняется при предъявлении каждого нового объекта. Необходимы специальные ухищрения, чтобы приспособить стандартные методы оптимизации для настройки нейронных сетей. В обзоре [10] даются следующие рекомендации.

1. Если обучающая выборка имеет большой объём (порядка нескольких сотен объектов и более), или если решается задача классификации, то можно использовать метод стохастического градиента с адаптивным шагом.

2. Диагональный метод Левенберга–Марквардта сходится в несколько раз быстрее. В этом методе величина шага вычисляется индивидуально для каждого весового коэффициента, при этом используется только один диагональный элемент матрицы вторых производных:

$$\eta_{jh} = \frac{\eta}{\frac{\partial^2 Q}{\partial w_{jh}^2} + \mu},$$

где η остаётся глобальным параметром темпа обучения, μ — новый параметр, предотвращающий обнуление знаменателя, и, соответственно, неограниченное увеличение шага. Отношение η/μ есть темп обучения на ровных участках функционала $Q(w)$, где вторая производная обращается в нуль. Диагональный элемент матрицы вторых производных вычисляется с помощью специального варианта back-propagation.

3. Если обучающая выборка имеет небольшой объём, или если решается задача регрессии, то лучше использовать адаптированные варианты метода сопряжённых градиентов. Адаптация заключается в том, что объекты предъявляются не по одному, а *пакетами* (batch learning). Состав пакета может формироваться случайным образом. Для каждого пакета минимизируемый функционал остаётся фиксированным, что позволяет применить метод сопряжённых градиентов.

1.2.3 Оптимизация структуры сети

Выбор структуры сети, то есть числа слоёв, числа нейронов и числа связей для каждого нейрона, является, пожалуй, наиболее сложной проблемой. Существуют различные стратегии поиска оптимальной структуры сети: постепенное наращивание, построение заведомо слишком сложной сети с последующим упрощением, поочерёдное наращивание и упрощение.

Проблема выбора структуры тесно связана с проблемами недообучения и переобучения. Слишком простые сети не способны адекватно моделировать целевые зависимости в реальных задачах. Слишком сложные сети имеют избыточное число свободных параметров, которые в процессе обучения настраиваются не только на восстановление целевой зависимости, но и на воспроизведение шума.

Выбор числа слоёв. Если в конкретной задаче гипотеза о линейной разделимости классов выглядит правдоподобно, то можно ограничиться однослойной сетью. Двух-

слойные сети позволяют представлять извилистые нелинейные границы, и в большинстве случаев этого хватает. Трёхслойными сетями имеет смысл пользоваться для представления сложных многосвязных областей. Чем больше слоёв, тем более богатый класс функций реализует сеть, но тем хуже сходятся градиентные методы, и тем труднее её обучить.

Выбор числа нейронов в скрытом слое H производят различными способами, но ни один из них не является лучшим.

1. Визуальный способ. Если граница классов (или кривая регрессии) слишком сглажена, значит, сеть переупрощена, и необходимо увеличивать число нейронов в скрытом слое. Если граница классов (или кривая регрессии) испытывает слишком резкие колебания, на тестовых данных наблюдаются большие выбросы, веса сети принимают большие по модулю значения, то сеть переусложнена, и скрытый слой следует сократить. Недостаток этого способа в том, что он подходит только для задач с низкой размерностью пространства (небольшим числом признаков).

2. Оптимизация H по *внешнему критерию*, например, по критерию скользящего контроля или средней ошибки на независимой контрольной выборке $Q(X^k)$. Зависимость внешних критериев от параметра сложности, каким является H, обычно имеет характерный оптимум. Недостаток этого способа в том, что приходится много раз заново строить сеть при различных значениях параметра H, а в случае скользящего контроля — ещё и при различных разбиениях выборки на обучающую и контрольную части.

Динамическое добавление нейронов. Сначала сеть обучается при заведомо недостаточной мощности среднего слоя $H \ll \ell$. Обучение происходит до тех пор, пока ошибка не перестанет убывать. Тогда добавляется один или несколько новых нейронов. Веса новых связей инициализируются небольшими случайными числами, либо добавленные нейроны обучаются по-отдельности как однослойные персептроны. Во втором случае можно рекомендовать обучать новый персептрон на случайной подвыборке, возможно, добавив в неё те объекты, на которых текущая сеть допустила наибольшие ошибки. Веса старых связей не меняются. Затем проводится настройка сети методом обратного распространения.

После добавления новых нейронов ошибка, как правило, сначала резко возрастает, затем быстро сходится к меньшему значению. Интересно, что общее время обучения обычно оказывается лишь в 1.5–2 раза больше, чем если бы в сети сразу было нужное количество нейронов. Это означает, что информация, накопленная в сети, является полезной и не теряется при добавлении новых нейронов.

При постепенном наращивании сети целесообразно наблюдать за динамикой какого-нибудь внешнего критерия. Прохождение значения $Q(X^k)$ через минимум является надёжным критерием останова, так как свидетельствует о переобученности, вызванной чрезмерным усложнением сети.

Удаление избыточных связей. Метод *оптимального усечения сети* (optimal brain damage, OBD) [11, 8] удаляет те связи, к изменению которых функционал Q наименее чувствителен. Уменьшение числа весов снижает склонность сети к переобучению.

Метод OBD основан на предположении, что после стабилизации функционала ошибки Q вектор весов w находится в локальном минимуме, где функционал может

быть аппроксимирован квадратичной формой:

$$Q(w+\delta) = Q(w) + \tfrac{1}{2}\delta^{\intercal} H(w)\delta + o(\|\delta\|^2),$$

где $H(w) = \left(\frac{\partial^2 Q(w)}{\partial w_{jh}\partial w_{j'h'}}\right)$ — гессиан, матрица вторых производных. Как и в диагональном методе Левенберга–Марквардта, предполагается, что диагональные элементы доминируют в гессиане, а остальными частными производными можно пренебречь, положив их равными нулю. Это предположение носит эвристический характер и вводится для того, чтобы избежать трудоёмкого вычисления всего гессиана.

Если гессиан $H(w)$ диагонален, то

$$\delta^{\intercal} H(w)\delta = \sum_{j=0}^{J}\sum_{h=1}^{H} \delta_{jh}^2 \frac{\partial^2 Q(w)}{\partial w_{jh}^2}.$$

Обнуление веса w_{jh} эквивалентно выполнению условия $w_{jh} + \delta_{jh} = 0$. Введём величину *значимости* (salience) синаптической связи, равную изменению функционала $Q(w)$ при обнулении веса: $S_{jh} = w_{jh}^2 \frac{\partial^2 Q(w)}{\partial w_{jh}^2}$.

Эвристика OBD заключается в том, чтобы удалить из сети d синапсов, соответствующих наименьшим значениям S_{jh}. Здесь d — это ещё один параметр метода настройки. После удаления производится цикл итераций до следующей стабилизации функционала Q. При относительно небольших значениях d градиентный алгоритм довольно быстро находит новый локальный минимум Q. Процесс упрощения сети останавливается, когда *внутренний критерий* стабилизируется, либо когда заданный *внешний критерий* начинает возрастать.

Обнуление веса w_{jh} между входным и скрытым слоями означает, что h-й нейрон скрытого слоя не будет учитывать j-й признак. Это можно интерпретировать как отбор информативных признаков для h-го нейрона скрытого слоя.

Обнуление веса w_{hm} между скрытым и выходным слоями означает удаление h-го нейрона скрытого слоя.

§1.3 Сети Кохонена

До сих пор мы рассматривали нейронные сети, предназначенные для обучения с учителем, когда для каждого объекта x_i задан соответствующий ему ответ y_i. Сети Кохонена решают задачи обучения без учителя, когда задаются только сами объекты x_i, и требуется выделить обособленные «плотные сгустки» объектов — кластеры, и научиться относить новые объекты к этим кластерам.

Кластеризация основывается на предположении, что в пространстве X введена метрика $\rho\colon X \times X \to \mathbb{R}$, адекватно оценивающая степень сходства любой пары объектов.

1.3.1 Модели конкурентного обучения

Пусть $X = \mathbb{R}^n$ — пространство объектов, $Y = \{1, \ldots, M\}$ — множество кластеров, число M фиксировано. Задана обучающая выборка объектов $X^\ell = \{x_i\}_{i=1}^{\ell}$. Требуется выработать правило отнесения объектов к кластерам $a\colon X \to Y$.

Правило жёсткой конкуренции WTA. Наиболее очевидный подход заключается в том, чтобы ввести векторы w_m, $m = 1, \ldots, M$, описывающие центры кластеров, и относить произвольный объект $x \in X$ к ближайшему кластеру:

$$a(x) = \arg\min_{m \in Y} \rho(x, w_m). \tag{1.11}$$

Образно говоря, кластеры соревнуются за право присоединить к себе объект x. Кластер, ближайший к x, называется кластером-победителем, а выражение (1.11) — *правилом WTA* (winner takes all).

Настройка алгоритма кластеризации $a(x)$ сводится к оптимизации расположения центров w_m. Для этого минимизируется функционал качества кластеризации, равный среднему квадрату расстояния между объектами и центрами кластеров:

$$Q(w_1, \ldots, w_M) = \frac{1}{2} \sum_{i=1}^{\ell} \rho^2(x_i, w_{a(x_i)}) \to \min.$$

Допустим, что метрика евклидова: $\rho(x, w) = \|x - w\|$. Тогда можно выписать градиент функционала Q по вектору w_m:

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

Для поиска центров кластеров w_m можно применить метод стохастического градиента — Алгоритм 1.2, практически без модификаций. Единственное отличие заключается в том, что теперь правило обновления весов на шаге 6 будет иметь вид

$$w_m := w_m + \eta(x_i - w_m) \big[a(x_i) = m\big], \tag{1.12}$$

где $x_i \in X^\ell$ — случайным образом выбранный обучающий объект, η — градиентный шаг, он же *темп обучения*. Смысл данного правила очевиден: если объект x_i относится к кластеру m, то центр этого кластера w_m немного сдвигается в направлении объекта x_i, остальные центры не изменяются.

Формальное сходство формулы (1.12) с персептронным правилом наводит на мысль, что кластеризация осуществляется алгоритмом, аналогичным нейронной сети. Выражение (1.11) и в самом деле представимо в виде двухслойной суперпозиции, только вместо привычных скалярных произведений $\langle x, w_m \rangle$ вычисляются функции расстояния $\rho(x, w_m)$, а на выходе вместо суммирования применяется операция минимума, см. Рис. 8. При этом центры кластеров w_m взаимно однозначно соответствуют нейронам скрытого стоя, которые называются *нейронами Кохонена*. Нейрон, выход которого минимален, называется *нейроном-победителем*.

Рассмотренная разновидность нейронных сетей называется *сетью с конкурентным обучением*. Исходно она была предложена как чисто математическая модель. Позже удалось найти некоторые её аналоги и в нейрофизиологии [7].

Альтернативное название — *обучающееся векторное квантование* (learning vector quantization, LVQ) — связано с тем, что исходная выборка из ℓ объектов x_i порождает M новых объектов w_m, похожих на исходные. Это центры ячеек, по которым распределяются («квантуются») исходные объекты. Как правило, $M \ll \ell$, поэтому замена объектов на ближайшие к ним центры позволяет эффективно сжимать данные при незначительной потере информации. Объём сохраняемой информации регулируется единственным параметром M, что достаточно удобно в приложениях.

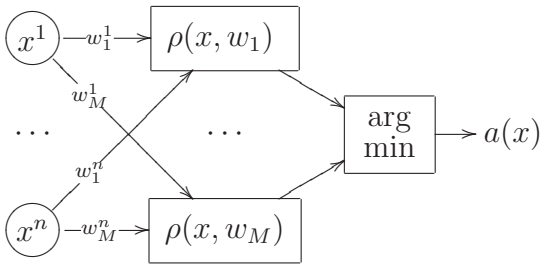


Рис. 8. Представление алгоритма кластеризации (1.11) в виде двухслойной нейронной сети.

Правило справедливой конкуренции CWTA. Недостаток конкурентного обучения по правилу WTA заключается в том, что при случайной инициализации весов нейрон Кохонена может попасть в такую область, где он никогда не станет победителем. В результате появится неинформативный пустой кластер.

Для преодоления этого недостатка алгоритм (1.11) немного модифицируется. Вводится «механизм утомления» победителей — *правило CWTA* (conscience WTA):

$$a(x) = \arg\min_{m \in Y} C_m \rho(x, w_m), \tag{1.13}$$

где C_m — количество побед m-го нейрона в ходе обучения. Таким образом, кластеры штрафуются за слишком частое присоединение объектов.

Правило мягкой конкуренции WTM. Другим недостатком правила WTA является медленная скорость сходимости, связанная с тем, что на каждой итерации модифицируется только один нейрон-победитель w_m. Для ускорения сходимости, особенно на начальных итерациях, можно подстраивать сразу несколько нейронов, близких к объекту x_i. Для этого вводится ядро — неотрицательная монотонно убывающая на $[0, +\infty)$ функция расстояния $K(\rho)$. Иногда накладывается дополнительное требование нормировки $K(0) = 1$. Часто берут гауссовское ядро $K(\rho) = \exp(-\beta\rho^2)$ при некотором $\beta > 0$. Вместо правила жёсткой конкуренции WTA вводится мягкая конкуренция — *правило WTM* (winner takes most):

$$w_m := w_m + \eta(x_i - w_m)\, K\big(\rho(x_i, w_m)\big), \quad m = 1, \ldots, M. \tag{1.14}$$

Теперь на каждой итерации центры *всех* кластеров смещаются в сторону x_i, но чем дальше центр находится от x_i, тем меньше величина смещения. Заметим, что (1.12) является частным случаем (1.14), если положить $K\big(\rho(x_i, w_m)\big) = [a(x_i) = m]$.

На начальных итерациях имеет смысл выбрать небольшое значение коэффициента β, чтобы все весовые векторы успели переместиться ближе к области входных векторов. Затем β можно увеличивать, делая конкуренцию всё более жёсткой, и постепенно переходя к коррекции только одного нейрона-победителя.

Благодаря способности к сглаживанию, правило WTM имеет многочисленные применения. Одно из важнейших — самоорганизующиеся карты Кохонена.

1.3.2 Самоорганизующиеся карты Кохонена

Самоорганизующиеся карты Кохонена (self-organizing maps, SOM) применяются для визуализации многомерных данных. Они дают лишь общую картину, довольно размытую и подверженную искажениям, поскольку спроецировать многомерную выборку на плоскость без искажений в общем случае невозможно. Тем не менее, карты

Алгоритм 1.4. Обучение карты Кохонена методом стохастического градиента

Вход:
X^ℓ — обучающая выборка;
η — темп обучения;
Выход:
Синаптические веса w_{mn}, $m = 1, \ldots, M$, $n = 1, \ldots, N$;

1: инициализировать веса:
$w_{mn} := \text{random}\left(-\frac{1}{2MN}, \frac{1}{2MN}\right)$;
2: **повторять**
3: выбрать объект x_i из X^ℓ случайным образом;
4: WTA: вычислить координаты узла, в который проецируется объект x_i:
$(m_i, n_i) := a(x_i) \equiv \arg\min_{(m,n)\in Y} \rho(x_i, w_{mn})$;
5: **для всех** $(m, n) \in Y$
6: WTM: сделать шаг градиентного спуска:
$w_{mn} := w_{mn} + \eta(x_i - w_{mn})\, K\big(r((m_i, n_i), (m, n))\big)$;
7: **пока** размещение всех объектов в узлах сетки не стабилизируется;

Кохонена позволяют увидеть ключевые особенности кластерной структуры выборки. Они используются на стадии *разведочного анализа данных*, скорее для общего понимания задачи, чем для получения каких-либо точных результатов.

Идея заключается в том, чтобы спроецировать все объекты выборки на плоскую карту, точнее, на множество узлов прямоугольной сетки заранее заданного размера $M \times N$. На практике M и N имеют порядок десятков или сотен. Чтобы карта отражала кластерную структуру выборки, близкие объекты должны попадать в близкие узлы сетки.

Каждому узлу сетки приписывается нейрон Кохонена с вектором весов w_{mn}, $m = 1, \ldots, M$, $n = 1, \ldots, N$. Таким образом, множество Y совпадает с множеством узлов сетки, $Y = \{1, \ldots, M\} \times \{1, \ldots, N\}$. Алгоритм классификации $a(x)$ выдаёт пару индексов $(m, n) \in Y$, показывающих, в какой узел сетки проецируется объект x.

Обучение нейронов производится методом стохастического градиента, см. Алгоритм 1.4. После случайного выбора объекта x_i на шаге 3 определяется нейрон-победитель согласно правилу WTA. Соответствующий ему узел сетки обозначается в алгоритме через (m_i, n_i). Затем, согласно правилу WTM, этот нейрон и нейроны, расположенные в ближайших узлах сетки, сдвигаются в сторону вектора x_i. Правило обучения, применяемое на шаге 6, схоже с (1.14), только вместо метрики $\rho(x, x')$, определённой на пространстве объектов, используется евклидова метрика на множестве узлов сетки Y:

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

По прежнему, $K(\rho)$ — заданная сглаживающая функция, например, $K(\rho) = \exp(-\beta\rho^2)$. Параметр β задаёт степень сглаженности карты: чем меньше β, тем мягче конкуренция нейронов, и тем более сглаженными будут выглядеть границы кластеров на карте. Имеет смысл увеличивать значение параметра β от итерации к итерации, чтобы сеть Кохонена сначала обучилась кластерной структуре «в общих чертах», а затем сосредоточилась на деталях.

Алгоритм останавливается, когда проекции всех, или хотя бы большинства, объектов выборки $(m_i, n_i) = a(x_i)$ перестанут меняться от итерации к итерации.

Искусство интерпретации карт Кохонена. Если через настроенную карту Кохонена (алгоритм $a(x)$) пропустить все объекты обучающей выборки X^ℓ, то кластеры исходного пространства отобразятся в сгустки точек на карте. При этом векторы весов в узлах-сгустках должны быть одновременно похожи на все объекты соответствующих кластеров.

Обычно для каждой точки карты вычисляют среднее расстояние до k ближайших точек выборки, и полученное распределение расстояний отображают в виде двухцветной полутоновой карты. На ней же отмечают и сами точки обучающей выборки. На такой карте хорошо видно, сколько кластеров выделено, и в каких местах карты они расположились.

Следующий шаг — поиск интерпретации кластеров. Для этого строятся ещё n карт, по одной карте на каждый признак. Теперь цвет узла (m, n) соответствует значению j-й компоненты вектора весов $w_{m,n}$. Обычно эти карты раскрашивают в геодезические цвета от синего до коричневого. Синие участки карты соответствуют наименьшим значениям j-го признака, красные — наибольшим. Сравнивая карты признаков с общей картой кластеров, можно найти кластеры, которым свойственны повышенные или пониженные значения отдельных признаков.

Ещё один способ интерпретации — выделение на карте всех узлов, в которые попадает выбранное подмножество объектов. Если объекты сгруппировались в одном месте, значит мы имеем дело с кластером или его частью. Выделяя различные подмножества объектов, имеющие заведомо известную содержательную интерпретацию, можно находить интерпретации различных зон на карте.

Недостатки карт Кохонена.

- Субъективность. Не всегда ясно, какие особенности карты обусловлены кластерной структурой данных, а какие — свойствами потенциальной функции. От выбора параметра β существенно зависит сглаженность границ кластеров и степень детализации карты.

- Наличие искажений. Близкие объекты исходного пространства могут переходить в далёкие точки на карте. В частности, объективно существующие кластеры могут разрываться на фрагменты [9]. И наоборот, далёкие объекты могут случайно оказаться на карте рядом, особенно, если они были одинаково далеки от всех кластеров. Искажения неизбежны при проецировании многомерной выборки на плоскость. Распределение точек на карте позволяет судить лишь о локальной структуре многомерных данных, и то не всегда.

- Зависимость от инициализации. Начальное распределение весов существенно влияет на процесс обучения и может сказываться не только на расположении кластеров, но даже на их количестве.

Не секрет, что популярность карт Кохонена обусловлена в значительной степени их эстетической привлекательностью и общим впечатлением наукообразия и загадочности, которое они производят на неискушённых пользователей. На практике

они применяются в основном как вспомогательное средство для предварительного визуального анализа и понимания общей структуры многомерных данных.

1.3.3 Гибридные сети встречного распространения

Ещё одно важное применение нейронов Кохонена с их способностью кластеризовать исходные векторы — кусочная аппроксимация функций.

Рассмотрим задачу восстановления регрессии, когда на элементах обучающей выборки $X^\ell = \{x_i\}_{i=1}^\ell$ заданы вещественные ответы $y_i = y^*(x_i)$. Идея заключается в том, чтобы сначала разбить обучающие объекты (и всё пространство X) на кластеры, не пользуясь ответами y_i, затем для каждого кластера построить свою локальную аппроксимацию целевой зависимости y^*. При использовании правила жёсткой конкуренции WTA эти аппроксимации никак не «склеиваются», и зависимость $a(x)$ получается кусочно-непрерывной. Правило мягкой конкуренции WTM позволяет «склеить» локальные куски и получить сколь угодно гладкую аппроксимацию.

Кусочно-постоянная аппроксимация. Чтобы алгоритм кластеризации (1.11) превратить в алгоритм кусочной аппроксимации, к нему добавляется ещё один нейрон с линейной функцией активации, образующий третий слой с весами $v_1, \ldots, v_M$:

$$\begin{aligned} a(x) &= v_{m^*(x)} = \sum_{m=1}^{M} v_m \big[m^*(x) = m\big]; \\ m^*(x) &= \arg\min_{m=1,\ldots,M} \rho(x, w_m); \end{aligned} \tag{1.15}$$

где $m^*(x)$ — номер нейрона-победителя на объекте x, вычисляемый по правилу WTA.

При квадратичной функции потерь задача настройки весов v_m легко решается аналитически:

$$\begin{aligned} Q(v) &= \sum_{i=1}^{\ell} \big(a(x_i) - y_i\big)^2 \to \min_v; \\ \frac{1}{2}\frac{\partial Q}{\partial v_m} &= \sum_{i=1}^{\ell} \big(a(x_i) - y_i\big)\big[m^*(x_i) = m\big] = 0. \end{aligned}$$

Подставляя сюда выражение для $a(x_i)$ из (1.15), получаем

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

Проще говоря, оптимальное значение весового коэффициента v_m есть среднее y_i по всем объектам x_i, попавшим в m-й кластер. Ответ алгоритма $a(x)$ для всех объектов, попадающих в m-й кластер, будет одинаков и равен v_m. Это означает, что $a(x)$ реализует кусочно-постоянную функцию.

Гладкая аппроксимация строится с помощью правила мягкой конкуренции WTM при выбранном гладком ядре $K(\rho)$. Алгоритм $a(x)$ представляет собой формулу Надарая-Ватсона для сглаживания ответов v_m по M точкам — центрам кластеров:

$$a(x) = \sum_{m=1}^{M} v_m \frac{K\big(\rho(x, w_m)\big)}{\sum_{s=1}^{M} K\big(\rho(x, w_s)\big)}.$$

В этом случае задача минимизации $Q(v)$ сводится к решению системы линейных уравнений размера $M \times M$. Вместо явного решения системы можно снова воспользоваться методом стохастического градиента. На каждой итерации обновляются и веса слоя Кохонена w_m, и веса третьего (выходного) слоя v_m:

$$\begin{cases} w_m := w_m - \eta(w_m - x_i)K\big(\rho(x_i, w_m)\big); \\ v_m := v_m - \eta(a(x_i) - y_i)\dfrac{K\big(\rho(x_i, w_m)\big)}{\sum_{s=1}^{M} K\big(\rho(x_i, w_s)\big)}; \end{cases}$$

Заметим, что обновление весов w_m влияет на веса v_m, а обратного влияния нет.

Данный метод получил название *встречного распространения ошибки*, поскольку второй слой настраивается по правилу Кохонена, а третий слой — так же, как в методе back-propagation.

Упражнения

Упр. 1.1. Доказать: в задачах классификации с двумя классами $Y = \{0, 1\}$ персептронное правило (1.2) переходит в правило Хэбба (1.3), если изменить метку класса 0 на -1.

Упр. 1.2. Доказать, что в однослойном персептроне с линейной функцией активации при квадратичном функционале ошибки оптимальная величина темпа обучения η даётся формулой (1.10).

Упр. 1.3. Вывести формулы вычисления вторых производных $\frac{\partial^2 Q}{\partial w_{jh}^2}$, необходимые для реализации диагонального метода Левенберга-Марквардта, в алгоритме обратного распространения ошибок.

Список литературы

[1] *Головко В. А.* Нейронные сети: обучение, организация и применение. — М.: ИПРЖР, 2001.

[2] *Колмогоров А. Н.* О представлении непрерывных функций нескольких переменных в виде суперпозиции непрерывных функций одного переменного // *Докл. АН СССР.* — 1958. — Т. 114, № 5. — С. 953–956.

[3] Нейроинформатика / А. Н. Горбань, В. Л. Дунин-Барковский, А. Н. Кирдин, Е. М. Миркес, А. Ю. Новоходько, Д. А. Россиев, С. А. Терехов и др. — Новосибирск: Наука, 1998. — С. 296.

[4] *Тихонов А. Н., Арсенин В. Я.* Методы решения некорректных задач. — М.: Наука, 1986.

[5] *Яблонский С. В.* Введение в дискретную математику. — М.: Наука, 1986.

[6] *Bartlett P.* The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network // *IEEE Transactions on Information Theory.* — 1998. — Vol. 44, no. 2. — Pp. 525–536.
http://discus.anu.edu.au/~bartlett.

[7] *Durbin R., Rummelhart D. E.* Product units: A computationally powerful and biologically plausible extension to backpropagation networks // *Neural Computation.* — 1989. — Vol. 1, no. 4. — Pp. 133–142.

[8] *Hassibi B., Stork D. G.* Second order derivatives for network pruning: Optimal brain surgeon // Advances in Neural Information Processing Systems / Ed. by S. J. Hanson, J. D. Cowan, C. L. Giles. — Vol. 5. — Morgan Kaufmann, San Mateo, CA, 1993. — Pp. 164–171.
http://citeseer.ist.psu.edu/hassibi93second.html.

[9] *Hastie T., Tibshirani R., Friedman J.* The Elements of Statistical Learning. — Springer, 2001.

[10] *LeCun Y., Bottou L., Orr G. B., Muller K.-R.* Efficient BackProp // Neural Networks: tricks of the trade. — Springer, 1998.

[11] *LeCun Y., Denker J., Solla S., Howard R. E., Jackel L. D.* Optimal brain damage // Advances in Neural Information Processing Systems II / Ed. by D. S. Touretzky. — San Mateo, CA: Morgan Kauffman, 1990.
http://citeseer.ist.psu.edu/lecun90optimal.html.

[12] *McCulloch W. S., Pitts W.* A logical calculus of ideas immanent in nervous activity // *Bulletin of Mathematical Biophysics.* — 1943. — no. 5. — Pp. 115–133.

[13] *Minsky M., Papert S.* Perceptrons: an Introduction to Computational Geometry. — MIT Press, 1968.

[14] *Novikoff A. B. J.* On convergence proofs on perceptrons // Proceedings of the Symposium on the Mathematical Theory of Automata. — Vol. 12. — Polytechnic Institute of Brooklyn, 1962. — Pp. 615–622.

[15] *Rummelhart D. E., Hinton G. E., Williams R. J.* Learning internal representations by error propagation // Vol. 1 of Computational models of cognition and perception, chap. 8. — Cambridge, MA: MIT Press, 1986. — Pp. 319–362.

[16] *Smola A., Bartlett P., Scholkopf B., Schuurmans D.* Advances in large margin classifiers. — MIT Press, Cambridge, MA. — 2000.
http://citeseer.ist.psu.edu/article/smola00advances.html.

[17] *Stone M. N.* The generalized Weierstrass approximation theorem // *Math. Mag.* — 1948. — Vol. 21. — Pp. 167–183, 237–254.

[18] *Widrow B., Hoff M. E.* Adaptive switching circuits // IRE WESCON Convention Record. — DUNNO, 1960. — Pp. 96–104.