Лекции по искусственным нейронным сетям

(черновик)

К. В. Воронцов

9 апреля 2006 г.

Содержание

1 Искусственные нейронные сети **2**
1.1 Однослойные нейронные сети 2
1.1.1 Естественный нейрон и его формальная модель 2
1.1.2 Методы обучения отдельного нейрона 4
1.1.3 Проблема полноты 9
1.2 Многослойные нейронные сети 12
1.2.1 Метод обратного распространения ошибок 12
1.2.2 Улучшение сходимости и качества градиентного обучения 15
1.2.3 Неградиентные методы обучения 19
1.3 Сети Кохонена 20
1.3.1 Модели конкурентного обучения 20
1.3.2 Самоорганизующиеся карты Кохонена 22
1.3.3 Гибридные сети встречного распространения 24

1 Искусственные нейронные сети

Человеку и высшим животным буквально на каждом шагу приходится распознавать, принимать решения и обучаться. Нейросетевой подход возник благодаря стремлению 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 реально применяемые разновидности ANN представляют собой математические модели, имеющие лишь отдалённое сходство с нейрофизиологией.

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 другим нервным клеткам. 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, на подобии 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 несколько сотен миллисекунд. 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 существенной доли нейронов не нарушает обычного хода распространения нервного импульса.

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

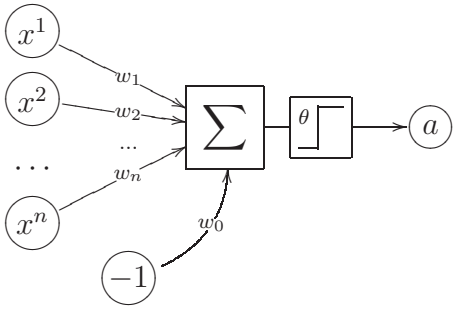


Рис. 1. Модель МакКаллока-Питтса.

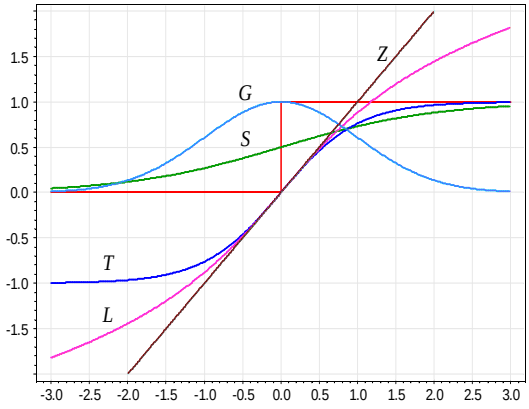


Рис. 2. Функции активации.

Модель МакКаллока-Питтса. В 1943 году МакКаллок и Питтс [8] предложили математическую модель нейрона, Рис. 1. Пусть имеется n входных величин $x^1, \ldots, x^n$ — бинарных признаков, описывающих объект x. Значения этих признаков будем трактовать как величины импульсов, поступающих на вход нейрона через n входных синапсов. Будем считать, что, попадая в нейрон, импульсы складываются с весами $w_1, \ldots, w_n$. 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

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

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*. Таким образом, модель МакКаллока-Питтса эквивалентна линейному пороговому классификатору.

Функции активации. Позже модель МакКаллока-Питтса была обобщена на случай произвольных вещественных входов и выходов, а также произвольных функций активации. На практике чаще всего используются функции активации, показанные на Рис. 2:

$\theta(z) = [z \geqslant 0]$	— ступенчатая функция Хэвисайда;
$\sigma(z) = (1 + e^{-z})^{-1}$	— S, сигмоидная (логистическая) функция;
$\mathrm{th}(z) = 2\sigma(2z) - 1$	— T, гиперболический тангенс;
$\ln(z + \sqrt{z^2 + 1})$	— L, логарифмическая функция;
$\exp(-z^2/2)$	— G, гауссовская функция;
z	— Z, тождественная (линейная) функция;

Векторная запись. Введём дополнительный константный признак $x^0 \equiv -1$ и векторные обозначения $w = (w_0, w_1, \ldots, w_n)$, $x = (x^0, x^1, \ldots, x^n)$. Пусть φ — выбранная функция активации. Тогда линейную модель нейрона можно записать более компактно через скалярное произведение:

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

1.1.2 Методы обучения отдельного 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.

Пусть имеется обучающая выборка прецедентов $X^\ell = \{x_1, \ldots, x_\ell\}$, и для каждого объекта $x_i = (x_i^1, \ldots, x_i^n)$ известен правильный ответ y_i. 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, 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. Заметим, что увеличивать имеет смысл только те веса w_j, которые соответствуют ненулевым компонентам x_i^j, например, 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.

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

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

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;

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*. Для этого правила сформулируем и докажем теорему сходимости, справедливую не только для бинарных, но и для произвольных действительных признаков.

Теорема 1.1. (Новиков, [10]) *Пусть $X = \mathbb{R}^{n+1}$, $Y = \{-1, 1\}$, и выборка X^ℓ линейно разделима — существует вектор w^* и положительное число δ такие, что $\langle w^*, x_i\rangle\, y_i > \delta$ для всех $i = 1, \ldots, \ell$. Тогда алгоритм 1.1 сходится к вектору весов, разделяющему обучающие объекты без ошибок, из любого начального положения w^0, независимо от порядка предъявления объектов, за конечное число шагов. 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 w^* и вектором весов после t-го исправления w^t, полагая без ограничения общности $\|w^*\| = 1$:

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

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

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

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

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

Подставляем полученные соотношения в выражение для косинуса:

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

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

Если $w^0 = 0$, то нетрудно оценить сверху достаточное число исправлений. Из условия $\cos = 1$ следует $\sqrt{t}\delta/D = 1$, 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.

TODO: *Модельный пример расходимости.* ToDo[1]

Адаптивный линейный элемент. В 1960 году Видроу и Хофф предложили правило настройки синаптических весов, основанное на минимизации среднеквадратичной ошибки [13]:

$$Q(w) = \sum_{i=1}^{\ell} \big(a(x_i) - y_i\big)^2 \to \min_w.$$

В случае вещественных признаков, $X = \mathbb{R}^n$, и линейной функции активации, $a(x) = \langle w, x\rangle$, данная задача ничем не отличается от классической задачи линейной регрессии. Применим для её решения метод градиентного спуска. Запишем правило изменения вектора весов на каждой итерации:

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

Расписывая градиент, и учитывая, что $a(x_i) = \langle w, x_i\rangle$, получим правило:

$$w := w - \eta \sum_{i=1}^{\ell} \big(a(x_i) - y_i\big) x_i,$$

где $\eta > 0$ — величина шага в направлении антиградиента. Формально это выражение совпадает с правилом Розенблатта (1.2), если каждый градиентный шаг делать только для одного объекта. Данное правило обучения называют *дельта-правилом* (delta-rule), а сам линейный нейрон — *адаптивным линейным элементом* ADALINE.

Метод стохастического градиента. Благодаря аддитивности функционала $Q(w)$ и градиента $\partial Q/\partial w$ нейрон можно обучать, предъявляя объекты по одному. Но тогда возникают вопросы относительно организации процесса обучения. В каком порядке предъявлять объекты? Достаточно ли сделать для данного объекта только один градиентный шаг или лучше несколько? Нужно ли обновлять вектор весов w после каждого шага?

Практика показывает, что процесс обучения сходится быстрее, если предъявлять объекты по одному в случайном порядке, делать для каждого объекта только

один градиентный шаг, и сразу обновлять вектор весов. Этот метод получил название *стохастического градиента* (stochastic gradient, SG). Удивительно, что именно так и устроен персептрон Розенблатта, за исключением разве что последовательности предъявления объектов. Когда объекты предъявляются в некотором фиксированном порядке, процесс чаще оказывается расходящимся или зацикливается [???].

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 может 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.

Проблема выбора шага η подробно исследуется в теории численной оптимизации. В частности, можно воспользоваться методом скорейшего спуска. Для ADALINE одномерную задачу минимизации функционала Q по параметру η удаётся решить аналитически. Это приводит к следующему выражению для вычисления *адаптивного шага* [1] при предъявлении обучающего объекта x_i:

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

Доказано, что при таком выборе шага процесс обучения ADALINE сходится [???].

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 классификации 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:

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

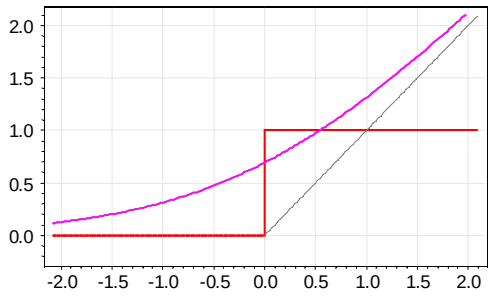


Рис. 3. Логарифмическая аппроксимация пороговой функции потерь.

где σ — сигмоидная функция.

При определённых вероятностных предположениях справедливо соотношение

$$\sigma\big(\langle w, x_i \rangle \, y_i\big) = \mathsf{P}\{y^*(x_i) = y_i\}. \tag{1.4}$$

В этом случае минимизация функционала $\widetilde{Q}(w)$ эквивалентна применению принципа максимума правдоподобия:

$$\widetilde{Q}(w) = -\ln \prod_{i=1}^{\ell} \sigma\big(\langle w, x_i\rangle\, y_i\big) = -\ln \prod_{i=1}^{\ell} \mathsf{P}\{y^*(x_i) = y_i\} = -L(X^\ell; w).$$

Из соотношения (1.4) также вытекает, что нейрон с сигмоидной функции активации $a(x) = \sigma\big(\langle w, x\rangle\big)$ вычисляет вероятность принадлежности объекта x классу $+1$. Это замечательное свойство служит дополнительным обоснованием для применения сигмоидной функции в классифицирующих нейронных сетях.

В то же время, не стоит забывать, что равенство (1.4) получено при довольно сильных предположениях. Во-первых, функции правдоподобия классов должны принадлежать экспоненциальному семейству распределений. Во-вторых, классы должны иметь равные параметры разброса, т. е. в некотором смысле иметь «схожую форму». Кроме того, среди признаков должна присутствовать константа, но это условие удовлетворяется автоматически при добавлении фиктивного признака $x^0 = -1$. В практических задачах гарантировать выполнение первых двух 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.

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

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

Тогда в методе стохастического градиента правило обновления весов при предъявления 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:

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

TODO: *Рисунок* ToDo[2]

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.

В остальном алгоритм обучения остаётся тем же, что и для персептрона Розенблатта.

Заметим также, что описанный метод настройки весов существенно проще метода логистической регрессии IRLS, рассмотренного в **??**. Различие в том, что здесь

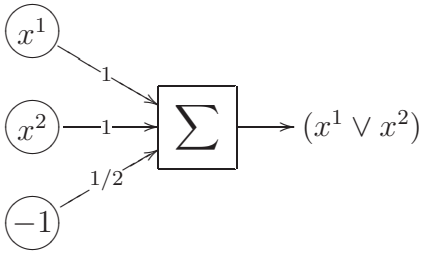


Рис. 4. Однослойный персептрон, реализующий операцию ИЛИ.

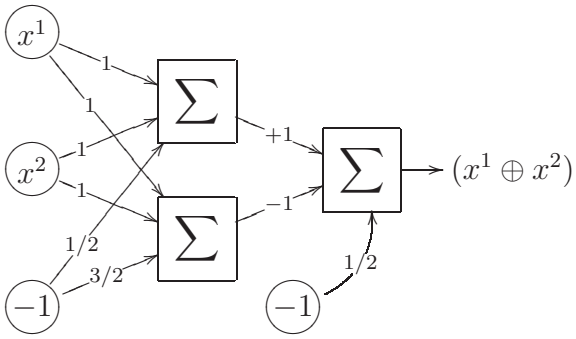


Рис. 5. Двухслойная сеть, реализующая операцию исключающего ИЛИ.

применялся метод первого порядка (градиентный спуск), в то время как IRLS основан на методе второго порядка (Ньютона-Рафсона), имеющем лучшие характеристики сходимости.

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» [9]. Следующий классический контрпример иллюстрирует невозможность нейронной реализации даже очень простых функций.

Проблема «исключающего ИЛИ». Легко построить нейроны, реализующие логические функции И, ИЛИ, НЕ (Рис. 4):

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

Однако функцию исключающего ИЛИ

$$x^1 \oplus x^2 = \begin{cases} 1, & x^1 = x^2; \\ 0, & x^1 \neq x^2; \end{cases}$$

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.

Первый путь — пополнить состав признаков, подавая на вход нейрона нелинейные преобразования исходных признаков. В частности, если разрешить образовывать произвольные 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:

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

Второй путь технически более прост. Кроме того, он приводит к построению нейронных сетей из огромного количества связанных нейронов, напоминающих естественные нейронные сети.

Многослойные нейронные сети. Рассмотрим композицию нейронов, представленную на Рис. 6. 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 [4].

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. Известна теорема Колмогорова о том, что любая непрерывная функция n аргументов представима в виде суперпозиции непрерывных функций одного аргумента и суммирования [2].

[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.

Теорема 1.2. (Колмогоров, 1957) *Любая непрерывная функция n аргументов на единичном кубе $[0,1]^n$ представима в виде*

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

где h_k, φ_{ik} — непрерывные функции одной переменной, 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 h_k различны, 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 [12].

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 и др., 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 непрерывную функцию с любой желаемой точностью.

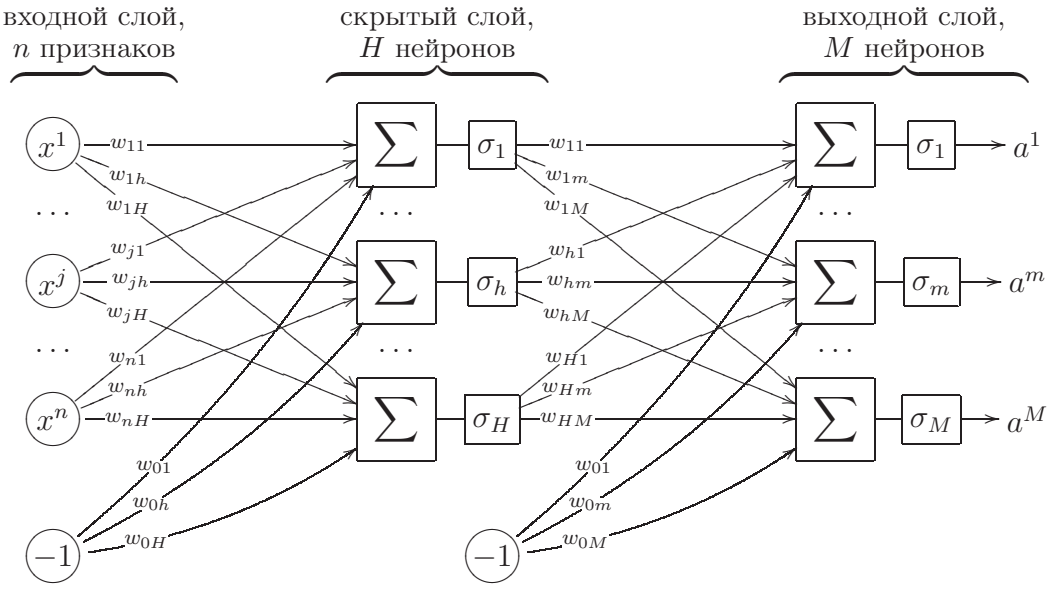


Рис. 6. Многослойная сеть с одним скрытым слоем.

Данная теорема ничего не говорит о количестве слоёв нейронной сети (уровней вложенности суперпозиции) и о количестве нейронов, необходимых для аппроксимации произвольной функции. Следующий результат имеет более конкретное отношение к нейронным сетям.

5.

TODO: *Теорема из книги Бишопа [5].* ToDo[3]

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. Известен весьма эффективный способ вычисления градиента, при котором каждый градиентный шаг выполняется за число операций, лишь немногим большее, чем при обычном вычислении сети на одном объекте.

На первый взгляд, количество операций, необходимых для вычисления градиента, должно было бы возрасти пропорционально числу весовых коэффициентов. Однако этого удаётся избежать путём аналитического дифференцирования суперпозиции с сохранением всех необходимых промежуточных величин. Данный метод получил название *обратного распространения ошибок* (error back-propagation) [11].

1.2.1 Метод обратного распространения ошибок

Рассмотрим многослойную сеть, в который каждый нейрон предыдущего слоя связан со всеми нейронами последующего слоя, Рис. 6. 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 $J = n$ и $v^j \equiv x^j$. Обозначим через w вектор всех синаптических весов сети.

Выходные значения сети на объекте x_i вычисляются как суперпозиция:

$$a^m(x_i) = \sigma_m\left(\sum_{h=0}^{H} w_{hm} u^h(x_i)\right); \qquad u^h(x_i) = \sigma_h\left(\sum_{j=0}^{J} w_{jh} v^j(x_i)\right). \tag{1.5}$$

Запишем функционал среднеквадратичной ошибки для отдельного объекта x_i:

$$Q(w) = \frac{1}{2}\sum_{m=1}^{M}\big(a^m(x_i) - y_i^m\big)^2. \tag{1.6}$$

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

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

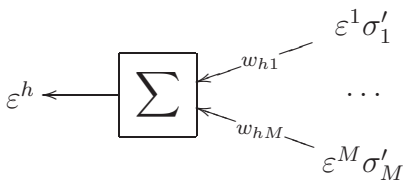
Оказывается, эта частная производная равна величине ошибки ε^m на объекте x_i. Теперь выпишем частные производные по выходам скрытого слоя:

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

Эту величину, по аналогии с ε^m, будем называть ошибкой сети на скрытом слое и обозначать через ε^h.

Через σ'_m обозначена производная функции активации, вычисленная при том же значении аргумента, что и в (1.5). Если используется сигмоидная функция активации, то для эффективного вычисления производной можно воспользоваться формулой $\sigma'_m = \sigma_m(1 - \sigma_m)$.

Заметим, что ε^h вычисляется по ε^m, если запустить сеть «задом наперёд», подавая на выходы нейронов скрытого слоя значения $\varepsilon^m\sigma'_m$ с теми же весами w_{hm}, а результат ε^h считывать на входе:



Если слоёв больше, чем два, то остальные частные производные вычисляются аналогично:

$$\frac{\partial Q(w)}{\partial v^j} = \sum_{h=1}^{H}\varepsilon^h\sigma'_h w_{jh} = \varepsilon^j.$$

Алгоритм 1.2. Обучение двухслойной сети методом обратного распространения ошибки

Вход:

$(x_i, y_i)_{i=1}^{\ell}$ — 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 — вычислить:
$u^h(x_i)$, σ'_h для всех $h = 1, \ldots, H$;
$a^m(x_i)$, σ'_m, ε^m для всех $m = 1, \ldots, M$;
$Q_i := \sqrt{\sum_{m=1}^{M} (\varepsilon^m)^2}$;
5: Обратный ход — вычислить:
$\varepsilon^h := \sum_{m=1}^{M} \varepsilon^m \sigma'_m w_{hm}$ для всех $h = 1, \ldots, H$;
6: Градиентный шаг:
$w_{hm} := w_{hm} - \eta \varepsilon^m \sigma'_m u^h$, для всех $h = 0, \ldots, H$, $m = 1, \ldots, M$;
$w_{jh} := w_{jh} - \eta \varepsilon^h \sigma'_h x^j$, 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;

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

$$\frac{\partial Q(w)}{\partial w_{hm}} = \frac{\partial Q(w)}{\partial a^m} \frac{\partial a^m}{\partial w_{hm}} = \varepsilon^m \sigma'_m u^h, \qquad m = 1, \ldots, M, \quad h = 0, \ldots, H; \tag{1.7}$$

$$\frac{\partial Q(w)}{\partial w_{jh}} = \frac{\partial Q(w)}{\partial u^h} \frac{\partial u^h}{\partial w_{jh}} = \varepsilon^h \sigma'_h v^j, \qquad h = 1, \ldots, H, \quad j = 0, \ldots, J; \tag{1.8}$$

и так далее для каждого слоя.

Теперь мы обладаем всем необходимым, чтобы полностью записать алгоритм обратного распространения, см. Алгоритм 1.2.

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 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 любыми градиентными методами: скорейшего спуска, сопряженных градиентов, и другими.

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 сложности 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 (1.7)–(1.8). 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 большим или меньшим успехом преодолеть недостатки метода обратного распространения. Различных тонкостей настолько много, что умение хорошо настроить нейронную сеть по праву считается искусством, см. обзор [7].

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 (1.4), эффективность вычисления производной, ограниченность выходного значения. Применение нечётных

функций, таких как $\mathrm{th}(z) = 2\sigma(2z) - 1$, увеличивает скорость сходимости примерно в полтора раза.

Для предотвращения эффекта паралича имеет смысл добавлять небольшой линейный член: $\mathrm{th}\, z + \gamma z$. Можно использовать и другие медленно растущие функции, например $\ln(z + \sqrt{z^2+1})$.

Ещё одна скрытая опасность связана с применением функционала среднеквадратичной невязки (1.6) 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 [7] 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 1.2 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.

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 вероятностью 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. Этот метод 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.

Регуляризация или сокращение весов. Этот метод является частным случаем регуляризации некорректно поставленных задач по А. Н. Тихонову. Он эффективно противодействует возникновению паралича сети и предотвращает нежелательные эффекты мультиколлинеарности и численной неустойчивости весов, свойственные всем линейным методам. Аналогичный приём применяется в линейном дискриминантном анализе и в линейной регрессии.

Допустим, что при обучении сети минимизируется функционал $Q(w)$. Идея заключается в том, чтобы ограничить возможный рост абсолютных значений весов, добавив к имеющемуся функционалу штрафное слагаемое:

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

Изменение функционала приводит к появлению аддитивной поправки у градиента:

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

При этом правило обновления весов принимает вид

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

Единственная модификация заключается в появлении неотрицательного множителя $(1-\eta\lambda)$, что приводит к постоянному уменьшению весов. Отсюда название метода — *сокращение весов* (weights decay).

Преимущества данного метода в том, что он очень просто реализуется и сочетается с любыми функционалами и любыми градиентными методами оптимизации. Сокращение весов необходимо в тех случаях, когда среди исходных признаков имеются линейно зависимые или сильно коррелированные.

Недостаток метода сокращения весов — появление дополнительного управляющего параметра λ, который приходится подбирать экспериментально.

Сокращение числа связей.

TODO: *optimal brain damage* ToDo[4]

Выбор числа слоёв. Если в конкретной задаче гипотеза о линейной разделимости классов выглядит правдоподобно, то можно ограничиться однослойной сетью. Двухслойные 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 хуже сходятся градиентные методы, и тем труднее обучить сеть.

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. В частности, могут использоваться критерии 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.

3. 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 новый нейрон. Веса новых связей инициализируются небольшими случайными числами, и обучение повторяется снова. После каждого добавления ошибка сначала резко возрастает, затем быстро сходится к меньшему значению. Интересно, что общее время обучения обычно оказывается лишь в 1.5 раза больше, чем если бы в сети сразу было нужное количество нейронов. Это означает, что информация, накопленная в сети, не теряется полностью при добавлении нейрона со случайными параметрами.

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 переусложнении сети и появлении эффекта переобученности.

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 сводится к выбору адаптивного шага η исходя из решения одномерной задачи минимизации $Q\left(w - \eta \frac{\partial Q}{\partial w}\right) \to \min_{\eta}$. Решение данной задачи для метода обратного распространения ошибок подробно рассматривается в [1].

3. В некоторых градиентных методах оптимальная величина шага вычисляется аналитически, возможно, индивидуально для каждого весового коэффициента (см. ниже).

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 1.2 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. Как правило, слишком 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, 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 в нём недопустимо менять функционал качества, что препятствует применению стохастического градиента. Таким образом, применение стандартных оптимизационных техник для настройки нейронных сетей требует специфических усовершенствований. В обзоре [7] приводятся 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). Для каждого пакета минимизируемый функционал остаётся фиксированным, что позволяет применить метод сопряжённых градиентов.

1.2.3 Неградиентные методы обучения

TODO: *Обучение по частям (bagging). Термодинамический метод или имитация отжига (simulated annealing). Стохастические методы.* ToDo[5]

Наиболее эффективным подходом представляется одновременное применение нескольких методов. Например, для поиска неплохого начального приближения можно использовать обучение по частям, затем запускать back-propagation для более

тонкой подстройки весов, а для выбивания из локальных минимумов применять стохастические методы.

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 позволяют решать 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 $\rho\colon X \times X \to \mathbb{R}$, позволяющая адекватно оценить 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 новый объект x к ближайшему кластеру:

$$a(x) = \arg\min_{m \in Y} \rho(x, w_m). \tag{1.9}$$

Образно говоря, кластеры соревнуются за право присоединить к себе очередной объект. Ближайший кластер объявляется победителем, и «получает всё», поэтому данное правило называют *WTA* (winner takes all).

Для оптимального выбора центров будем минимизировать 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:

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

Если для поиска центров кластеров применить метод стохастического градиента, то правило обновления весов при предъявлении обучающего объекта x_i будет выглядеть следующим образом:

$$w_m := \begin{cases} w_m + \eta(x_i - w_m), & \text{если } a(x_i) = m; \\ w_m, & \text{если } a(x_i) \neq m; \end{cases} \tag{1.10}$$

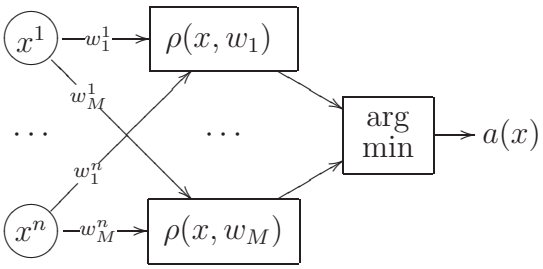


Рис. 7. Представление алгоритма кластеризации (1.9) в виде двухслойной нейронной сети.

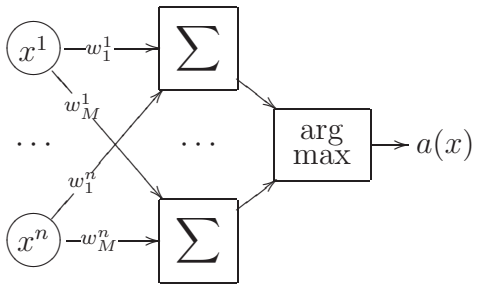


Рис. 8. Двухслойная сеть с конкурентным обучением.

где η — градиентный шаг, он же *темп обучения*. Смысл данного правила в том, что если объект x_i относится к кластеру m, то центр данного кластера немного сдвигается в направлении объекта x_i, остальные центры не изменяются.

Формальное сходство формулы (1.10) с персептронным правилом наводит на мысль, что кластеризация осуществляется алгоритмом, аналогичным нейронной сети. Выражение (1.9) 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. 7. При этом центры кластеров w_m взаимно однозначно соответствуют нейронам скрытого стоя (которые называются *нейронами Кохонена*) с весами w_m. Нейрон, выход которого минимален, называется *нейроном-победителем*.

Когда метрика евклидова, а векторы x_i и w_m нормированы, минимизация расстояния эквивалентна максимизации скалярного произведения:

$$\|x - w_m\|^2 = \|x\|^2 + \|w_m\|^2 - 2\langle x, w_m \rangle = 2 - 2\langle x, w_m \rangle .$$

В этом случае нейронная сеть приобретает более привычный вид, и отличается от стандартной сети только применением максимума вместо суммирования в выходном нейроне, см. Рис. 8.

Рассмотренная разновидность нейронных сетей не имеет прямых аналогов в нейрофизиологии и называется *сетью с конкурентным обучением*.

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

Для преодоления этого недостатка алгоритм (1.9) немного модифицируется. Вводится «механизм утомления» победителей — правило *CWTA* (conscience WTA):

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

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, особенно при использовании случайной инициализации весовых векторов w_m. Неплохая эвристика заключается в том, чтобы на начальных итерациях подстраивать все нейроны, а не только один нейрон-победитель.

Для этого вводится *потенциальная функция* — монотонно убывающая функция расстояния $\Pi(\rho)$. Например, можно взять $\Pi(\rho) = \exp(-\beta\rho^2)$, $\beta > 0$. Вместо жёсткого правила «победитель получает всё» вводится более мягкое правило «победитель получает больше» — *WTM* (winner takes most):

$$w_m := w_m + \eta(x_i - w_m)\,\Pi\big(\rho(x_i, w_m)\big).$$

Таким образом, центры всех кластеров смещаются в сторону объекта x_i, но центры, находящиеся дальше от x_i, смещаются меньше.

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 не точные количественные результаты, а лишь общую картину данных, причём довольно размытую и подверженную искажениям. Тем не менее, карты Кохонена весьма полезны. На стадии *разведочного анализа* они позволяют выявить ключевые особенности кластерной структуры данных и способствуют лучшему пониманию задачи.

Идея заключается в том, чтобы спроецировать все объекты выборки на плоскую карту, и чтобы близким объектам соответствовали близкие точки на карте. Для этого скрытый слой нейронов Кохонена наделяется топографической структурой. Теперь это прямоугольная сетка размера $M \times N$, в каждом узле которой находится нейрон с вектором весов w_{mn}. Множество кластеров 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 попадает объект x.

Обучение нейронов производится методом стохастического градиента. Сначала для объекта x_i определяется нейрон-победитель согласно правилу WTA. Затем этот нейрон и его ближайшие соседи сдвигаются в сторону обучающего вектора x_i. В этом правило обучения схоже с WTM, но только вместо метрики $\rho(x, x')$ на пространстве объектов используется евклидова метрика на множестве узлов сетки Y:

$$\rho\big((m, n), (m', n')\big) = \sqrt{(m - m')^2 + (n - n')^2}.$$

Таким образом, правило обучения весов в SOM имеет вид

$$w_{mn} := w_{mn} + \eta(x_i - w_{mn})\,\Pi\big(\rho(a(x_i), (m, n))\big), \tag{1.12}$$

где $\Pi(\rho)$ — выбранная потенциальная функция, например, $\Pi(\rho) = \exp(-\beta\rho^2)$. 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.

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 будут 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.

TODO: *Картинка небывалой красоты...* ToDo[6]

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. Пробуя различные подмножества объектов, можно найти 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 [6]. И наоборот, далёкие объекты могут случайно оказаться на карте рядом, особенно, если они были одинаково далеки от всех кластеров. Искажения неизбежны при проецировании многомерного пространства 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 научной загадочности, которое они производят на неискушённых пользователей. На практике они применяются лишь как вспомогательное средство для предварительного визуального анализа многомерных данных.

1.3.3 Гибридные сети встречного распространения

Кусочно-постоянная аппроксимация.

Гладкая аппроксимация. Возможность вычисления обратной функции.

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] *Яблонский С. В.* Введение в дискретную математику. — Москва: Наука, 1986.

[5] *Bishop C. M.* Neural Networks for Pattern Recognition. — Oxford University Press, 1995.

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

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

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

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

[10] *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.

[11] *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.

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

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