Лекции по искусственным нейронным сетям

К. В. Воронцов

19 января 2009 г.

Материал находится в стадии разработки, может содержать ошибки и неточности. Автор будет благодарен за любые замечания и предложения, направленные по адресу `vokov@forecsys.ru`, либо высказанные в обсуждении страницы «Машинное обучение (курс лекций, К.В.Воронцов)» вики-ресурса `www.MachineLearning.ru`.

Содержание

1 Искусственные нейронные сети **2**
1.1 Проблема полноты . . . 2
1.1.1 Задача «исключающего ИЛИ» . . . 2
1.1.2 Вычислительные возможности нейронных сетей. . . . 4
1.2 Многослойные нейронные сети . . . 5
1.2.1 Метод обратного распространения ошибок . . . 6
1.2.2 Оптимизация структуры сети . . . 9
1.3 Сети Кохонена . . . 12
1.3.1 Модели конкурентного обучения . . . 12
1.3.2 Самоорганизующиеся карты Кохонена . . . 14
1.3.3 Гибридные сети встречного распространения . . . 16

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 Проблема полноты

Итак, отдельно взятый нейрон вида (??) 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.

1.1.1 Задача «исключающего ИЛИ»

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

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

Однако функцию $x^1 \oplus x^2 = [x^1 \neq x^2]$ — *исключающее ИЛИ* (exclusive or, XOR) 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 3.

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:

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

Расширенные пространства признаков, в которых линейный классификатор безошибочно разделяет обучающую выборку, называют *спрямляющими*. Проблема

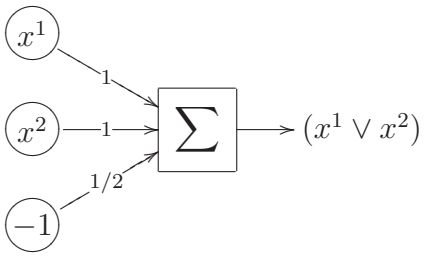


Рис. 1. Однослойный персептрон, реализующий операцию ИЛИ.

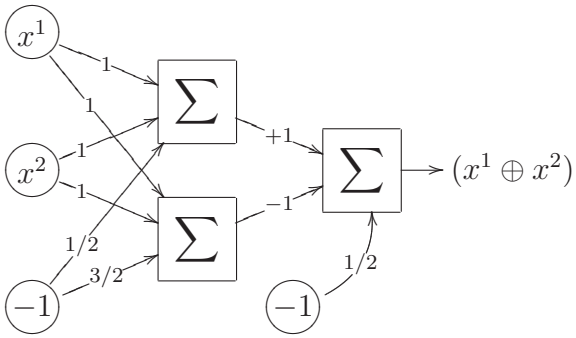


Рис. 2. Двухслойная сеть, реализующая операцию исключающего ИЛИ.

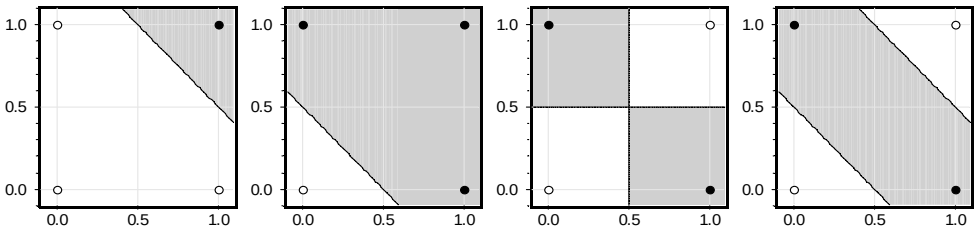


Рис. 3. Персептронная реализация элементарных логических функций. Слева направо: И, ИЛИ, XOR с помощью добавления признака x^1x^2, XOR с помощью двухслойной сети.

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 2:

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

Дальнейшее обобщение этой идеи приводит к построению многослойных нейронных сетей, состоящих из огромного количества связанных нейронов и напоминающих естественные нейронные сети. Пример такой композиции показан на Рис. 4. 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.

[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.

1.1.2 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 [3].

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 невыпуклые и даже 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 функцию n аргументов представить в виде суперпозиции функций меньшего числа аргументов. Ответ был дан А. Н. Колмогоровым в [1].

Теорема 1.1 (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 из $2n+1$ нейронов. 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 [11].

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 1.2 (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 $C(X)$.*

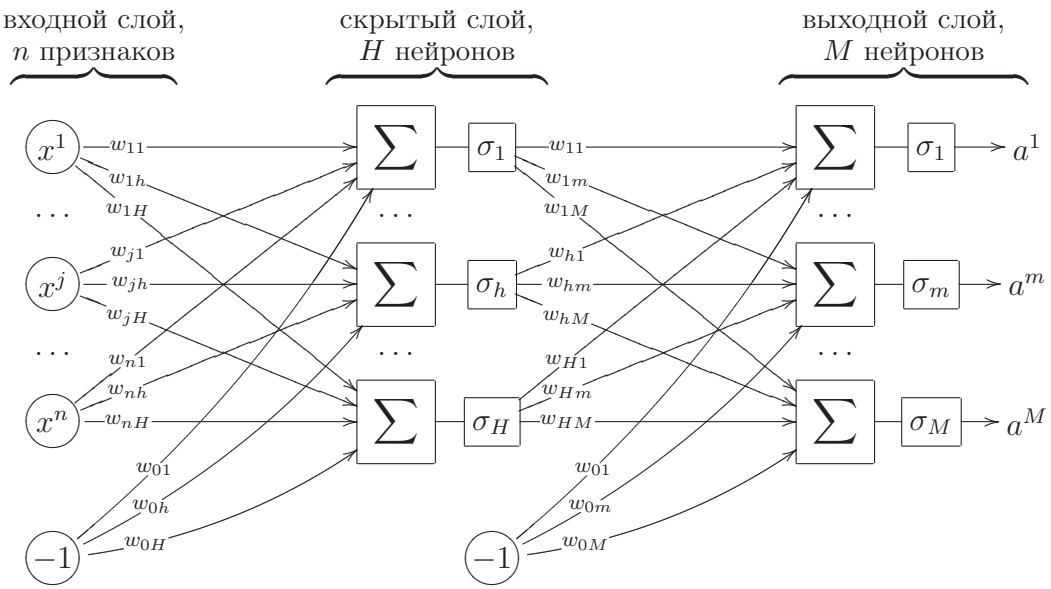


Рис. 4. Многослойная сеть с одним скрытым слоем.

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 [2]. 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 1.3 (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 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, так же, как и однослойный персептрон, можно настраивать градиентными методами, несмотря на огромное количество весовых коэффици-

ентов. 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 backpropagation) [10].

1.2.1 Метод обратного распространения ошибок

Рассмотрим многослойную сеть, в который каждый нейрон предыдущего слоя связан со всеми нейронами последующего слоя, Рис. 4. Такая сеть называется *полносвязной*. 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:

$$a^m(x_i) = \sigma_m\left(\sum_{h=0}^{H} w_{hm} u^h(x_i)\right); \qquad u^h(x_i) = \sigma_h\left(\sum_{j=0}^{J} w_{jh} v^j(x_i)\right). \tag{1.1}$$

Запишем функционал среднеквадратичной ошибки для отдельного объекта x_i:

$$Q(w) = \frac{1}{2}\sum_{m=1}^{M}\big(a^m(x_i) - y_i^m\big)^2. \tag{1.2}$$

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

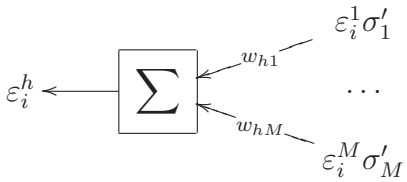
$$\frac{\partial Q(w)}{\partial a^m} = a^m(x_i) - y_i^m = \varepsilon_i^m.$$

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:

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

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 (1.1). Если используется сигмоидная функция активации, то для эффективного вычисления производной можно воспользоваться формулой $\sigma'_m = \sigma_m(1 - \sigma_m) = a^m(x_i)\big(1 - a^m(x_i)\big)$.

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:

$$\frac{\partial Q(w)}{\partial w_{hm}} = \frac{\partial Q(w)}{\partial a^m}\frac{\partial a^m}{\partial w_{hm}} = \varepsilon_i^m \sigma'_m u^h, \qquad m = 1,\dots,M, \quad h = 0,\dots,H; \tag{1.3}$$

$$\frac{\partial Q(w)}{\partial w_{jh}} = \frac{\partial Q(w)}{\partial u^h}\frac{\partial u^h}{\partial w_{jh}} = \varepsilon_i^h \sigma'_h v^j, \qquad h = 1,\dots,H, \quad j = 0,\dots,J; \tag{1.4}$$

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 1.1.

Достоинства метода обратного распространения.

- Достаточно высокая эффективность. В случае двухслойной сети 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 можно применять совместно с различными градиентными методами оптимизации: методом скорейшего спуска, сопряженных градиентов, Ньютона-Рафсона и др.

Недостатки метода обратного распространения.

- Метод наследует известные недостатки градиентной настройки весов в однослойном персептроне. Здесь также возникают проблемы медленной сходимости или расходимости, «застревания» в локальных минимумах функционала Q, переобучения и паралича. Причём парализоваться могут отдельные связи, нейроны, или вся сеть в целом.
- 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.

Алгоритм 1.1. 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;

Для улучшения сходимости и качества градиентного обучения применяются все те же приёмы, которые рекомендовались в **??** для обучения однослойного персептрона методом стохастического градиента. К ним добавляется ряд новых рекомендаций, связанных с многослойностью.

Выбор начального приближения. Для предотвращения паралича синаптические веса должны инициализироваться небольшими по модулю значениями. В Алгоритме 1.1 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. **??**.

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.

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 [7] 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.

1.2.2 Оптимизация структуры сети

Выбор структуры сети, то есть числа слоёв, числа нейронов и числа связей для каждого нейрона, является, пожалуй, наиболее сложной проблемой. Существу-

[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 **??**.

ют 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 однослойным персептроном. 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 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 сра-

зу 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 прореживания сети* (optimal brain damage, OBD) [8, 5] 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. Тем самым происходит отбор информативных признаков для h-го нейрона скрытого слоя.

Метод OBD легко приспособить и для настоящего *отбора признаков*. Вводится суммарная значимость признака $S_j = \sum_{h=1}^{H} S_{jh}$, и из сети удаляется один или несколько признаков с наименьшим значением S_j.

Обнуление веса w_{hm} между скрытым и выходным слоями означает, что m-е выходное значение не зависит от h-го нейрона скрытого слоя. Если выход одномерный ($M = 1$), то h-й нейрон можно удалить. В случае многомерного выхода для удаления нейронов скрытого слоя вычисляется суммарная значимость $S_h = \sum_{m=1}^{M} S_{hm}$.

§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 $w_m \in \mathbb{R}^n$, $m = 1, \ldots, M$, описывающие центры кластеров, и относить произвольный объект $x \in X$ к ближайшему кластеру:

$$a(x) = \arg\min_{m \in Y} \rho(x, w_m). \tag{1.5}$$

Образно говоря, кластеры соревнуются за право присоединить к себе объект x. Кластер, ближайший к x, называется кластером-победителем, а выражение (1.5) — *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 **??**, практически без модификаций. Единственное отличие заключается в том, что теперь правило обновления весов на шаге 6 будет иметь вид

$$w_m := w_m + \eta(x_i - w_m) \big[a(x_i) = m\big], \tag{1.6}$$

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.

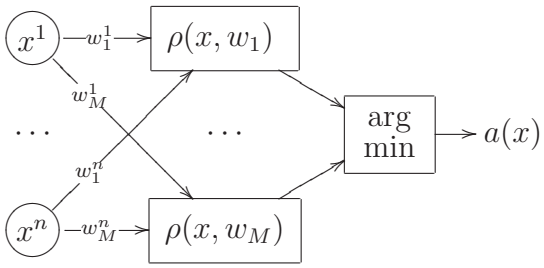


Рис. 5. Представление алгоритма кластеризации (1.5) в виде двухслойной нейронной сети.

Формальное сходство формулы (1.6) с персептронным правилом наводит на мысль, что кластеризация осуществляется алгоритмом, аналогичным нейронной сети. Выражение (1.5) 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. 5. 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 нейрофизиологам удалось найти некоторые её аналоги в биологических нейронных сетях [4].

Альтернативное название — *обучающееся векторное квантование* (learning vector quantization, LVQ) — связано с тем, что по исходной выборке из ℓ объектов x_i строятся 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.

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 (1.5) немного модифицируется. Вводится «механизм утомления» победителей — *правило CWTA* (conscience WTA):

$$a(x) = \arg\min_{m \in Y} C_m \rho(x, w_m), \tag{1.7}$$

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):

$$w_m := w_m + \eta(x_i - w_m)\, K\big(\rho(x_i, w_m)\big), \quad m = 1, \ldots, M. \tag{1.8}$$

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 (1.6) является частным случаем (1.8), 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 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 $M \times H$. На практике M и H имеют порядок десятков или сотен. Каждому узлу сетки приписывается нейрон Кохонена с вектором весов $w_{mh} \in \mathbb{R}^n$, $m = 1, \ldots, M$, $h = 1, \ldots, H$. Таким образом, множество Y совпадает с множеством узлов сетки, $Y = \{1, \ldots, M\} \times \{1, \ldots, H\}$.

Алгоритм кластеризации $a(x)$ выдаёт пару индексов $(m, h) \in Y$, показывающих, в какой узел сетки проецируется объект x. Чтобы карта отражала кластерную структуру выборки, близкие объекты должны попадать в близкие узлы сетки.

Обучение нейронов производится методом стохастического градиента, см. Алгоритм 1.2. 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, h_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 (1.8), только вместо метрики $\rho(x, x')$, определённой на пространстве объектов, используется евклидова метрика на множестве узлов сетки Y:

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

По прежнему, $K(\rho)$ — ядро сглаживания, 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

Алгоритм 1.2. Обучение карты Кохонена методом стохастического градиента

Вход:
X^ℓ — обучающая выборка;
η — темп обучения;
Выход:
Векторы синаптических весов w_{mh}, $m = 1, \ldots, M$, $h = 1, \ldots, H$;

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

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 $(m_i, h_i) = a(x_i)$ 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 средних расстояний отображают в виде двухцветной полутоновой карты. Чем меньше среднее расстояние, тем выше локальная плотность точек на карте. На той же карте отмечают и сами точки обучающей выборки. На карте хорошо видно, сколько кластеров выделено, в каких местах они расположились, и какие объекты выборки в них попадают.

Следующий шаг — поиск интерпретации кластеров. Для этого строятся ещё n карт, по одной карте на каждый признак. Теперь цвет узла (m, h) соответствует значению j-й компоненты вектора весов $w_{m,h}$. 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.

В результате содержательная интерпретация кластеров формулируется путём перечисления всех признаков, имеющих либо повышенные, либо пониженные значения в каждом из кластеров.

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 сглаживающего ядра. 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 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. Иногда рекомендуется построить несколько карт и выбрать из них наиболее «удачную».

Не секрет, что популярность карт Кохонена обусловлена в значительной степени их эстетической привлекательностью и впечатлением наукообразной загадочности, которое они производят на неискушённого пользователя. На практике они применяются в основном как вспомогательное средство для предварительного визуального анализа и выявления некоторых закономерностей в 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 жёсткой конкуренции WTA эти аппроксимации образуют кусочно-непрерывную функцию. Мягкая конкуренция WTM «склеивает» из локальных кусков гладкую аппроксимацию.

Кусочно-постоянная аппроксимация. Чтобы алгоритм кластеризации (1.5) превратить в алгоритм кусочной аппроксимации, к нему добавляется ещё один нейрон

с линейной функцией активации, образующий третий слой с весами $v_1, \ldots, v_M$:

$$a(x) = v_{m^*(x)} = \sum_{m=1}^{M} v_m \big[m^*(x) = m\big]; \tag{1.9}$$
$$m^*(x) = \arg\min_{m=1,\ldots,M} \rho(x, w_m);$$

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:

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

Подставляя сюда выражение для $a(x_i)$ из (1.9), 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.

Резюме

1. *Проблема полноты* связана с вопросом «насколько богатый класс функций способны реализовать нейронные сети той или иной структуры?» Однослойный персептрон способен разделять множества точек гиперплоскостью. Двухслойные сети способны аппроксимировать произвольные выпуклые области. Трёхслойные сети способны аппроксимировать области произвольной формы. Для этого функция активации обязана быть нелинейной.

2. *Метод обратного распространения ошибок* основан на технике быстрого дифференцирования суперпозиции функций. Он позволяет настраивать сети практически любой архитектуры, с любым числом слоёв. Однако он не всегда сходится, может «застревать» в локальных минимумах, попадать в состояние паралича (если функция активации имеет горизонтальную асимптоту) и переобучаться (если сеть имеет избыточное количество весовых коэффициентов). Для решения этих проблем выработано большое количество подходов и эвристик.

3. *Проблема паралича сети* решается с помощью нормировки исходных данных, аккуратного выбора начального приближения, использования функций активации без горизонтальных асимптот, сокращения весов (регуляризации).

4. *Проблема сходимости*, связанная с «застреванием» в локальных минимумах, решается путём смешения стратегий градиентного спуска и случайного локального поиска. Для увеличения скорости сходимости применяется оптимизация величины темпа обучения (метод скорейшего спуска), оптимизация порядка предъявления объектов (перетасовка объектов), диагональный метод Левенберга-Марквардта или другие методы второго порядка.

5. *Проблема переобучения* решается с помощью регуляризации (сокращения весов), раннего останова и оптимизации структуры сети.

6. *Оптимизация структуры сети* — числа слоёв, числа нейронов в каждом слое и числа связей для каждого нейрона — может производиться различными способами. Выбор числа нейронов в скрытом слое по скользящему контролю является слишком трудоёмкой процедурой и годится только для очень простых задач. Более эффективны процедуры динамического добавления и удаления нейронов и отдельных связей. Метод оптимального прореживания сети (OBD) позволяет выявлять и удалять наименее значимые связи, неинформативные признаки и избыточные нейроны в скрытых слоях.

7. *Сети Кохонена* решают задачу кластеризации, используя евклидову метрику между объектами. Правила мягкой и жёсткой конкуренции возникают как следствия градиентной минимизации средних внутрикластерных расстояний.

8. *Карты Кохонена* — это специальная разновидность сети Кохонена, в которой нейроны скрытого слоя образуют прямоугольную решётку, которая визуализируется в виде разноцветной карты и применяется для разведочного анализа данных. Обычно строится $n + 1$ карта, где n — число признаков. Одна общая карта показывает собственно кластеры — места сгущений объектов. Карты по

каждому признаку позволяют определить, каким кластерам свойственны пониженные или повышенные значения признака, и тем самым найти интерпретацию кластеров. Недостатками карт Кохонена являются субъективность, наличие искажений и чувствительность к начальному приближению.

9. *Гибридные сети встречного распространения* являются симбиозом кластеризующего слоя Кохонена и обычного линейного нейрона. Они используются для кусочно-постоянной (с правилом жёсткой конкуренции) или гладкой (с правилом мягкой конкуренции) аппроксимации регрессионных зависимостей.

Упражнения

Упр. 1.1. 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] *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.

[2] 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.

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

[4] *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.

[5] *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.

[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] *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.

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

[10] *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.

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