Тема 6

Стандартные архитектуры нейронных сетей

Сеть из одного нейрона

Рассмотрим возможности сети состоящей из одного нейрона. Пусть, для начала, этот формальный нейрон не имеет нелинейного преобразователя, т. е. функционирует по формуле $y = (\bar{w}, \bar{x}) + w_0$. Какие задачи можно решать с его помощью?

Линейная регрессия

Первая, и весьма важная задача, решаемая нейросетью из одного нейрона – это задача линейной регрессии, которая формулируется так: найти наилучшее линейное приближение функции r, заданной обучающей выборкой $(\bar{x}^i, y^i), r(\bar{x}^i) = y^i, i = 1,2,...,m$. Для этого требуется найти линейную функцию $\varphi(\bar{x}) = (\bar{w}, \bar{x}) + w_0$, ближайшую к r.

Функцию ошибки можно определить как сумму квадратов разностей:

$$D_r(\varphi) = \sum_{i=1}^{m} (r(\bar{x}^i) - \varphi(\bar{x}^i))^2 = \sum_{i=1}^{m} (r(\bar{x}^i) - (\bar{w}, \bar{x}^i) - w_0)^2 = \sum_{i=1}^{m} \Delta_i^2 . \quad (6.1)$$

Решение. Эту функцию необходимо минимизировать, для чего найдём производные по изменяемым параметрам:

$$\frac{\partial D}{\partial w_j} = -2\sum_{i=1}^{m} \Delta_i x_j^i, (j = 1,...,n); \frac{\partial D}{\partial w_0} = -2\sum_{i=1}^{m} \Delta_i . \quad (6.2)$$

Приравнивая эти производные нулю и вводя в рассмотрение $(n+1)$-мерные вектора x^i, в которых все элементы те же, а $x_0^i \equiv 1$, мы можем записать

$$\begin{aligned} &\Delta_i = y^i - (\bar{w}, x^i), \\ &w_0 = \frac{1}{m}\left(\sum_{i=1}^{m} (y^i - (\bar{w}, x^i) \right). \end{aligned} \quad (6.3)$$

Введем обозначения $\tilde{y}_m = \frac{1}{m}\sum_{i=1}^{m} y^i; \quad \tilde{x}_m = \frac{1}{m}\sum_{i=1}^{m} \bar{x}^i$. В этих обозначениях (6.3) примет вид

$$w_0 = \tilde{y}^m - (\bar{w}, \tilde{x}^m) . \quad (6.4)$$

Подставив это значение в (6.2) и приравнивая к нулю, мы получим систему:

$$\sum_{k=1}^{n} w_k \cdot \left(\sum_{i=1}^{m} (x_j^i - \tilde{x}_j^m)(x_k^i - \tilde{x}_k^m) \right) = \sum_{i=1}^{m} (x_j^i - \tilde{x}_j^m)(y^i - \tilde{y}^m), \quad (6.5)$$

или, в более короткой форме

$$\sum_{k=1}^{n} w_k r_{kj} = h_j. \qquad (6.6)$$

В векторно-матричной форме полученное равенство имеет вид

$$\bar{w}R = \bar{h}\,. \qquad (6.6')$$

Решение этой системы в случае невырожденности матрицы R может быть найдено как

$$\bar{w} = \bar{h} \cdot R^{-1}, \qquad (6.7)$$

в противном случае решения либо не существует, либо оно не единственно. При этом обычно ищется решение минимальной длины:

$$\bar{w} = \bar{h} \cdot R^{+}, \qquad (6.8)$$

где R^{+} - псевдообратная матрица. Способы решения систем линейных уравнений хорошо известны, и могут быть применены и в данном случае.

Пусть теперь нейрон кроме возможности скалярного умножения входного вектора на вектор весов (которое осуществляется адаптивным сумматором) имеет также возможность порогового преобразования, т. е. его функционирование описывается формулой

$$y = F((\bar{w}, \bar{x}) + w_0) = \begin{cases} 1, (\bar{w}, \bar{x}) + w_0 \geq 0, \\ 0, (\bar{w}, \bar{x}) + w_0 < 0. \end{cases} \qquad (6.9)$$

Рассмотрим его возможности.

Задача линейного разделения двух классов

Эта задача ставится как задача построения решающего правила, в соответствии с которым, часть векторов из выборки $\{\bar{x}^{\alpha}\}_{\alpha=1,\ldots,p}$ будет отнесена к одному классу, а остальные вектора выборки – к другому. В терминах нейросети из одного нейрона, это означает необходимость подобрать такие весовые коэффициенты w_j для нашего нейрона, что выход нейрона (6.9) равен единице, если входной вектор принадлежит первому классу и нулю, если входной вектор принадлежит другому классу.

Решение. Поиск такого решающего правила можно рассматривать как разделение точек посредством проекции их прямую. Вектор $\bar{w}$ задает прямую, на которую ортогонально проектируются все точки, а число w_0 - точку на этой прямой, отделяющую первый класс от второго.

Простейший и подчас очень удобный выбор состоит в проектировании на прямую, соединяющую центры масс выборок. Центр масс вычисляется в предположении, что массы всех точек одинаковы и равны 1. Это соответствует заданию $\bar{w}$ в виде

$$\overline{w} = \frac{1}{m}\sum_{i=1}^{m}\overline{x}^{\alpha_i} + \frac{1}{n}\sum_{i=m+1}^{p}\overline{x}^{\alpha_j}\ , \qquad (6.10)$$

где $\alpha_1, \alpha_2, ..., \alpha_m$ - номера векторов принадлежащих первому классу, $\alpha_{m+1}, \alpha_{m+2}, ..., \alpha_p$ - номера векторов принадлежащих второму классу. Простейший вариант выбора значения w_0 - посередине между центрами масс групп векторов:

$$w_0 = \frac{1}{2}\left(\left(\frac{1}{m}\sum_{i=1}^{m}\overline{x}^{\alpha_i}, \overline{w}\right) + \left(\frac{1}{n}\sum_{i=m+1}^{p}\overline{x}^{\alpha_j}, \overline{w}\right)\right). \qquad (6.11)$$

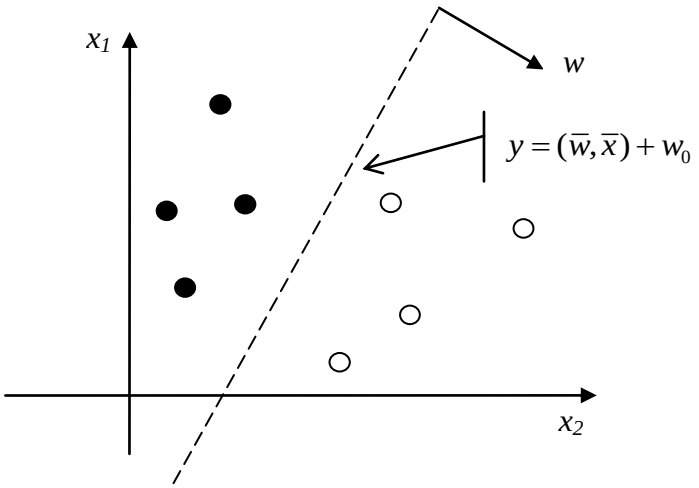

Рис. 6.1. Разделение векторов на классы при помощи прямой

Более чувствительные методы выбора границы раздела классов w_0 учитывают различные вероятности появления объектов разных классов, и оценки плотности распределения точек классов на прямой.

Рассмотрим теперь более сложные структуры - сети, состоящие из нескольких нейронов.

Однослойный персептрон Розенблатта

Самым простым и исторически первым вариантом многослойного персептрона является персептрон Розенблатта – нейросеть с единственным слоем нейронов [1]. Персептрон рассматривался его автором не как конкретное техническое вычислительное устройство, а как модель работы мозга.

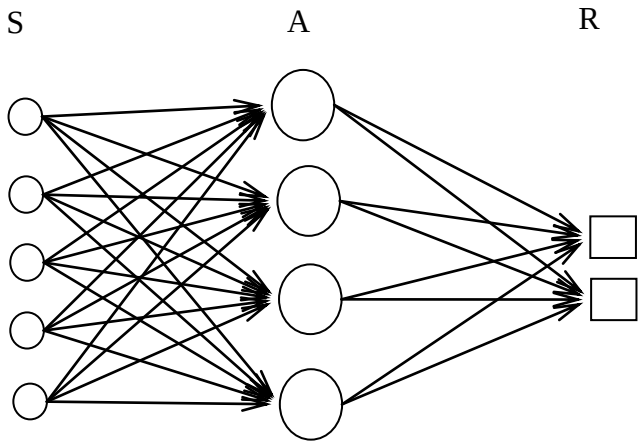

Рис. 6.2. Вид персептрона Розенблатта

Простейший классический персептрон содержит нейроподобные элементы трех типов. S-элементы (сенсорные) формируют входной вектор двоичных сигналов, поступающих из внешнего мира. Далее сигналы поступают в слой ассоциативных или A-элементов. Только ассоциативные элементы, представляющие собой формальные нейроны, выполняют нелинейную обработку информации и имеют изменяемые веса связей. Нелинейная часть A-элемента реализуется посредством пороговой функции:

$$f(x)=\begin{cases}1, & \text{åñëè } x\ge 0,\\ -1, & \text{åñëè } x<0.\end{cases} \qquad (6.12)$$

R-элементы (рефлекторы) с фиксированными весами формируют сигнал реакции персептрона на входной стимул.

Однослойный персептрон характеризуется матрицей синаптических связей W от S- к A-элементам. Элемент матрицы W_{ij} отвечает связи, ведущей от *i*-го S-элемента к *j*-му A-элементу.

Алгоритм обучения персептрона

Обучение сети состоит в подстройке весовых коэффициентов каждого нейрона. Пусть имеется набор пар векторов $(\bar{x}^{\alpha},\bar{y}^{\alpha}),\alpha=1,...,p,\quad \bar{x}^{\alpha},\bar{y}^{\alpha}\in R^{N}$, называемый обучающей выборкой. Будем называть нейронную сеть обученной на данной обучающей выборке, если при подаче на входы сети каждого вектора $\bar{x}^{\alpha}$ на выходах всякий раз получается соответствующий вектор $\bar{y}^{\alpha}$.

1. Функционирование. Допустим, что вектор $\bar{x}^{\alpha}$ является образом распознаваемой демонстрационной карты, а значит, подаётся на каждый входной A-элемент. Например, для *j*-го элемента - компоненты $\bar{x}_i^{\alpha}$ – умножаются на

соответствующие компоненты вектора весов w_{ij}. Эти произведения суммируются. Если сумма превышает нулевой порог, то выход j-го нейрона равен единице, в противном случае он равен -1. Эта операция компактно записывается в векторной форме как

$$\bar{y}^* = F(\bar{x}^\alpha \cdot W), \qquad (6.13)$$

$$F(\bar{x}) = (\mathrm{sgn}(x_1), \mathrm{sgn}(x_2), ..., \mathrm{sgn}(x_p)). \qquad (6.14)$$

2. Обучение. Для обучения сети образ $\bar{x}^\alpha$ подается на вход и вычисляется выход $\bar{y}^*$ (в соответствии с описанием в п. 1). Если $\bar{y}^*$ правильный, то ничего не меняется. Однако, если выход неправилен, то веса, присоединенные к входам, усиливающим ошибочный результат, модифицируются, чтобы уменьшить ошибку: если выход неправильный и равен нулю, то добавить все входы к соответствующим им весам; или если выход неправильный и равен единице, то вычесть каждый вход из соответствующего ему веса:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \cdot (y_j^*(t) - y_j^{\alpha(t)}) \cdot x_i^{\alpha(t)}. \qquad (6.15)$$

где η - константа, определяющая размер шага обучения.

3. Если ошибка существенна, перейти на шаг 1.

В работах Розенблатта был сделано заключение о том, что нейронная сеть рассмотренной архитектуры будет способна к воспроизведению любой логической функции, однако, как было показано позднее М.Минским и С.Пейпертом (M.Minsky, S.Papert, 1969), этот вывод оказался неточным. Были выявлены принципиальные неустранимые ограничения однослойных персептронов, и в последствии стал в основном рассматриваться многослойный вариант персептрона, в котором имеются несколько слоев процессорных элементов.

Сегодня однослойный персептрон представляет только исторический интерес, однако на его примере могут быть изучены основные понятия и простые алгоритмы обучения нейронных сетей.

Слоистые архитектуры

Особый интерес в нейроинформатике представляют слоистые однородные нейросети, состоящие из однотипных формальных нейронов, каждый из которых может быть соединён с любыми нейронами следующего слоя не более чем одной связью. Нейросеть с такими свойствами называется *многослойным персептроном*.

Многослойная нейронная сеть и алгоритм обратного распространения ошибки

В середине 1980-х несколькими исследователями независимо друг от друга был предложен эффективный алгоритм обучения многослойных персептронов [2,3,4], основанный на вычислении градиента функции ошибки. Алгоритм был назван "обратным распространением ошибки".

Алгоритм обратного распространения - это итеративный градиентный алгоритм, который используется с целью минимизации среднеквадратичного отклонения текущего выхода многослойного персептрона и желаемого выхода.

В сетях, обучаемых по методу обратного распространения ошибки могут быть целые или действительные входные сигналы. Выходные сигналы сети - это действительные числа из интервала, заданного передаточной функцией нейронов.

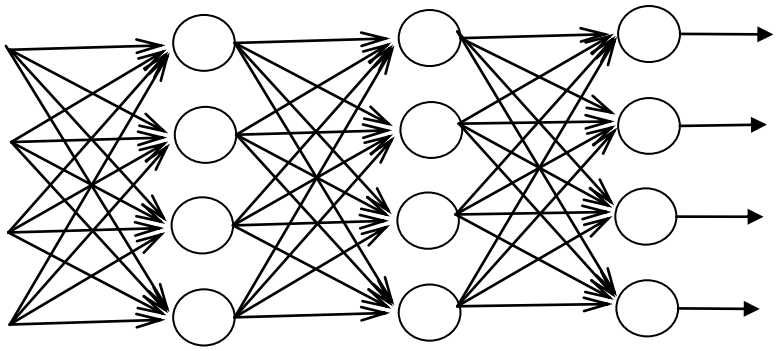

Рис. 6.3. Многослойная сеть с последовательными связями

Рассмотрим многослойный персептрон со стандартными элементами.

Функционирование многослойного персептрона

Функционирование многослойного персептрона нейросети описывается следующими формулами:

$$y_i^0 = x_i^0,$$
$$s_i^k = \sum_{j=1}^{P_{k-1}} y_j^{k-1} w_{ij}^k, \qquad (6.16)$$
$$y_i^k = f(s_i^k),$$

где

x_i^0 - значение подаваемое в момент $t=0$ на i-й синапс,

k - номер слоя ($k = 1,...,q$);

P_k - число нейронов в *k*-м слое;

y_i^k - выход *i*-го нейрона в *k*-м слое;

w_{ij}^k - вес синапса связи *i*-го нейрона в *(k*-1*)*-м слое с *j*-м нейроном в *k*-м слое;

s_i^k - выход сумматора *i*-го нейрона в *k*-м слое;

$f(s)$ - функция активации нейрона, одинаковая для всех нейронов сети.

Предполагается, что выход *i*-го нейрона в *(k*-1*)*-м подается на *i*-й вход *j*-го нейрона в *k*-м слое.

Очевидно, что формулы (6.16) можно записать и в матрично-векторном виде:

$$\begin{aligned} \bar{y}^0 &= \bar{x}^0, \\ \bar{s}^k &= W^k \bar{y}^{k-1}, \\ \bar{y}^k &= F(\bar{s}^k), \end{aligned} \qquad (6.16')$$

где

$\bar{x}^0$ - вектор, подаваемый на вход нейросети,

$W^k = (w_{ij}^k)$ - матрица размерности $(P_k \times P_{k-1})$, элементами которой являются веса связей нейронов $(k-1)$-го слоя с нейронами k-го слоя,

$\bar{y}^k$ - вектор полученный на выходе нейронов k-го слоя,

$F(\bar{s}^k) := (f(s_1^k), f(s_2^k), \ldots, f(s_{P_k}^k))$ - функция нелинейного преобразования.

Функционирование сети происходит следующим образом:

1-й шаг: на вход сети подается вектор сигналов $X = (x_1, x_2, \ldots, x_{P_1})$. Происходит срабатывание нейронов первого слоя.

k-й шаг: выход каждого нейрона k-го слоя определяемый формулами (6.16) передается по синаптическим связям нейронам (k+1)-го слоя.

q-й шаг: на выходе нейронной сети снимается вектор сигналов $Y = (y_1, y_2, \ldots, y_{P_q})$, который и является результатом.

Алгоритм обучения многослойного персептрона

В многослойных сетях оптимальные выходные значения нейронов всех слоев, кроме последнего, как правило, не известны, и многослойный перцептрон уже невозможно обучить, руководствуясь только величинами ошибок на выходах НС. Одним из вариантов решения этой проблемы является так называемое «распространение сигналов ошибки» от выходов НС к ее входам, в направлении, обратном прямому распространению сигналов в обычном режиме работы. Этот алгоритм обучения НС

получил название *процедуры обратного распространения*. Именно он будет рассмотрен далее.

Пусть имеется обучающая выборка $(\bar{x}(i), \bar{d}(i))$, $i = 1,..., N$. Согласно методу наименьших квадратов, минимизируемой целевой функцией ошибки НС является величина:

$$D(\bar{W}) = \frac{1}{2}\sum_{j,p}(y_j^q(p) - d_j(p))^2 \qquad (6.17)$$

где $y_j^q(p)$ – реальное выходное состояние нейрона *j* выходного слоя *q* нейронной сети при подаче на ее входы *p*-го образа; $d_j(p)$ – идеальное (желаемое) выходное состояние этого нейрона.

Суммирование ведется по всем нейронам выходного слоя и по всем обрабатываемым сетью образам. Минимизация ведется методом градиентного спуска, что означает подстройку весовых коэффициентов следующим образом:

$$w_{ij}^k(t+1) = w_{ij}^k(t) + \Delta w_{ij}^k,$$

$$\Delta w_{ij}^k = -\eta \cdot \frac{\partial D}{\partial w_{ij}^k}. \qquad (6.18)$$

Здесь w_{ij}^k – весовой коэффициент синаптической связи, соединяющей *i*-ый нейрон слоя *k-1* с *j*-ым нейроном слоя *k*, η – коэффициент скорости обучения, $0<\eta<1$, *t* – номер итерации алгоритма

Из (6.17) и (6.18) следует равенство

$$\frac{\partial D}{\partial w_{ij}^k} = \frac{\partial D}{\partial y_j^k} \cdot \frac{dy_j^k}{ds_j^k} \cdot \frac{\partial s_j^k}{\partial w_{ij}^k}. \qquad (6.19)$$

Так как множитель $\frac{dy_j^k}{ds_j^k}$ является производной этой функции по ее аргументу, из этого следует, что производная активационной функция должна быть определена на всей оси абсцисс. В связи с этим функция единичного скачка и прочие активационные функции с неоднородностями не подходят для рассматриваемых НС. В них применяются такие гладкие функции, как гиперболический тангенс или классический сигмоид с экспонентой.

Третий множитель $\frac{\partial s_j^k}{\partial w_{ij}^k}$, очевидно, равен выходу нейрона предыдущего слоя $y_i^{(k-1)}$.

Что касается первого множителя в (6.19), он легко раскладывается следующим образом:

$$\frac{\partial D}{\partial y_j^k} = \sum_i \frac{\partial D}{\partial y_i^{k+1}} \cdot \frac{dy_i^{k+1}}{ds_i^{k+1}} \cdot \frac{\partial s_i^{k+1}}{\partial y_j^{k+1}} = \sum_i \frac{\partial D}{\partial y_i^{k+1}} \cdot \frac{dy_i^{k+1}}{ds_i^{k+1}} \cdot w_{ji}^{k+1}. \qquad (6.20)$$

Здесь суммирование по i выполняется среди нейронов слоя k+1. Введя новую переменную

$$\delta_j^k = \frac{\partial D}{\partial y_j^k} \cdot \frac{dy_j^k}{ds_j^k}. \qquad (6.21)$$

мы получим рекурсивную формулу для расчетов величин δ_j^k слоя n из величин δ_i^{k+1} более старшего слоя k+1.

$$\delta_j^k = \left[\sum_i \delta_i^{k+1} \cdot w_{ji}^{k+1} \right] \cdot \frac{dy_j^k}{ds_j^k} \qquad (6.22)$$

Для выходного же слоя

$$\delta_l^q = (y_l^q - d_l) \cdot \frac{dy_l^q}{ds_l^q}. \qquad (6.23)$$

Теперь мы можем записать (6.18) в раскрытом виде:

$$\Delta w_{ij}^k = -\eta \cdot \delta_j^k \cdot y_i^{k-1}. \qquad (6.24)$$

Иногда для придания процессу коррекции весов некоторой инерционности, сглаживающей резкие скачки при перемещении по поверхности целевой функции, (6.25) дополняется значением изменения веса на предыдущей итерации

$$\Delta w_{ij}^k(t) = -\eta \cdot (\mu \cdot \Delta w_{ij}^k(t-1) + (1-\mu) \cdot \delta_j^k \cdot y_i^{k-1}) \qquad (6.25)$$

где μ – коэффициент инерционности, t – номер текущей итерации. Такой вариант алгоритма получил название «метод тяжелого шарика».

Таким образом, полный алгоритм обучения НС с помощью процедуры обратного распространения строится так:

1. Задать константу η>0 и значение $E_{\max}$>0. Инициализировать веса $w^k{}_{ij}$ случайными значениями близкими к 0.
2. Подать на входы сети один из возможных образов $\bar{x}^0 = \bar{x}(p)$ и в режиме обычного функционирования НС, когда сигналы распространяются от входов к выходам, рассчитать значения последних по формулам (6.16).
3. Рассчитать δ_l^q для выходного слоя по формуле (6.24):

$$\delta_l^q = (y_l^q - d_l) \cdot \frac{dy_l^q}{ds_l^q}.$$

Рассчитать по формуле (6.25) изменения весов Δw^q слоя q.

4. Рассчитать по формулам (6.23) и (6.25) (или (6.23) и (6.26)) соответственно δ^k и Δw^k для всех остальных слоев, *k=q-1,...1*:

$$\delta_j^k = \left[\sum_i \delta_i^{k+1} \cdot w_{ji}^{k+1}\right] \cdot \frac{dy_j^k}{ds_j^k},$$

$$\Delta w_{ij}^k(t) = -\eta \cdot (\mu \cdot \Delta w_{ij}^k(t-1) + (1-\mu) \cdot \delta_j^k \cdot y_i^{k-1}).$$

5. Скорректировать все веса в НС

$$w_{ij}^k(t) = w_{ij}^k(t-1) + \Delta w_{ij}^k(t). \qquad (6.26)$$

5. Если ошибка сети $E>E_{\max}$ существенна, перейти на шаг 2. В противном случае – конец.

Нейронной сети на шаге 2 попеременно в случайном порядке предъявляются все тренировочные образы, чтобы сеть, образно говоря, не забывала одни по мере запоминания других.

Из выражения (6.25) следует, что когда выходное значение y_i^{k-1} стремится к нулю, эффективность обучения заметно снижается. При двоичных входных векторах в среднем половина весовых коэффициентов не будет корректироваться, поэтому область возможных значений выходов нейронов [0,1] желательно сдвинуть в пределы [-0.5,+0.5], что достигается простыми модификациями активационных функций. Например, сигмоид с экспонентой преобразуется к виду

$$f(x) = -0.5 + \frac{1}{1+e^{-\alpha \cdot x}}. \qquad (6.27)$$

Литература

1. Розенблат Ф. Принципы нейродинамики. М.: Мир, 1965. 480 с.
2. Охонин В.А. Вариационный принцип в теории адаптивных сетей. Препринт ИФ СО АН СССР, Красноярск, 1987, №61Б, 18 с.
3. Rumelhart D. E., Hinton G. E., Williams R. J. Learning internal reprentations by error propagation. In Parallel distributed processing, vol. 1, pp. 318-62. 1986. Cambridge, MA: MIT Press.
4. Parker D. B. Learning logic. Invention Report S81-64, File 1, Office of Technology Licensing, Stanford University, Stanford, CA. 1982.