Приложение № 2
к распоряжению РТУ МИРЭА
от__ _____№_____

Титульный лист материалов по дисциплине
(заполняется по каждому виду учебного материала)

ДИСЦИЛИНА Проектирование и обучение нейронных сетей
(полное наименование дисциплины без сокращений)

ИНСТИТУТ Информационые технологии

КАФЕДРА Вычислительная техника
полное наименование кафедры

ВИД УЧЕБНОГО МАТЕРИАЛА Лекция
(в соответствии с пп.1-11)

ПРЕПОДАВАТЕЛЬ Сорокин Алексей Борисович
(фамилия, имя, отчество)

СЕМЕСТР 7 семестр, 2023/2024
(указать семестр обучения, учебный год)

3. ЛЕКЦИЯ. Обучение по дельта-правилу

Дельта-правило — метод обучения перцептрона на основе градиентного спуска. Дельта-правило развилось из первого и второго правил Хебба.

Отличия алгоритма обучения Розенблатта от алгоритма обучения по правилу Хебба сводятся к следующему:

1. вводится эмпирический коэффициент η с целью улучшения процесса сходимости: η – скорость обучения $0 < \eta < 1$;

2. исключены некоторые действия, которые замедляют процесс обучения, например, обновление весовых коэффициентов.

3. входные образы на вход нейронной сети подаются до тех пор, пока не будет достигнута допустимая величина ошибки обучения.

4. если решение существует, алгоритм обучения Розенблатта сходится за конечное число шагов.

Алгоритм обучения персептрона Розенблатта заключается в настройке коэффициентов межнейронных связей и порогов в ***слое А***. Пусть $X = x_1, x_2, \dots x_r, \dots x_m$, — вектор входных сигналов, а вектор $D = d_1, d_2, \dots d_k, \dots d_n$ — вектор сигналов, которые должны быть получены от перцептрона под воздействием входного вектора. Здесь n — число нейронов, составляющих перцептрон. Входные сигналы, поступив на входы перцептрона, были взвешены и просуммированы, в результате чего получен вектор $Y = y_1, y_2, \dots y_k, \dots y_n$ выходных значений перцептрона. Тогда можно определить вектор ошибки $E = e_1, e_2, \dots e_k, \dots e_n$, размерность которого совпадает размерностью вектором выходных сигналов.

Компоненты вектора ошибок определяются как разность между ожидаемыми и фактическими значениями на выходных нейронахперсептрона:

$$E = D - Y$$

На каждом шаге обучения вычисляется ошибка обучения e_j:

$$\mathrm{e}_j = d_j - y_j$$

,где d_j – эталонное значение входного сигнала; y_j – реальное значение входного сигнала.

При таких обозначениях формулу для корректировки j-го веса i-го нейрона можно записать следующим образом:

$$w_j(t+1) = w_j(t) + e_i x_j,$$

где номер сигнала j изменяется в пределах от единицы до размерности входного вектора m. Номер нейрона i изменяется в пределах от единицы до количества нейронов n. Величина t — номер текущей итерации обучения.

Таким образом, вес входного сигнала нейрона изменяется в сторону уменьшения ошибки пропорционально величине суммарной ошибки нейрона. Часто вводят коэффициент пропорциональности η, на который умножается величина ошибки. Этот коэффициент называют скоростью обучения.

Таким образом, итоговая формула для корректировки весов:

$$w_j(t+1) = w_j(t) + \eta e_i x_j$$

Алгоритм обучения Розенблатта (дельта-правило) сводится к следующей последовательности действий:

1. Инициализация весовых коэффициентов и порогов значениями, близкими к нулю.

2. Подача на вход нейронной сети очередного входного образа (входного вектора ***X***), взятого из обучающей выборки, и вычисление суммарного сигнала по всем входам для каждого нейрона ***j***:

$$\boldsymbol{S_j = \sum_{i=1}^{n} x_i w_{ij}}$$

, где ***n*** – размерность входного вектора, xi – ***i-я*** компонента входного вектора, $\boldsymbol{w_{ij}}$ – весовой коэффициент связи нейрона ***j*** и входа ***i***.

3. Вычисление значения выхода каждого нейрона:

$$OUT = \begin{cases} y = -1, \text{если } S_j > b_j \\ y = 1, \text{если } S_j \le b_j \end{cases}$$

, где $\boldsymbol{b_j}$ – порог, соответствующий нейрону ***j***

4. Вычисление значения ошибки обучения для каждого нейрона $\boldsymbol{e_j = d_j - y_j}$

5. Проводится модификация весового коэффициента связи по формуле

$\boldsymbol{w_{ij}(t+1) = w_{ij}(t) + \eta x_i e_j}$.

6. Повторение пунктов 2 – 5 до тех пор, пока ошибка сети не станет меньше заданной $\boldsymbol{e_j < e_{зад}}$.

Особенности алгоритма обучения Розенблатта: должна быть задан коэффициент обучения ***α*** *и ошибка обучения* ***e.***

Пример. Воспроизведение логической функции «И» с помощью нейронной сети.

X_1	X_2	У
1	**1**	**1**
1	**-1**	**-1**
-1	**1**	**-1**
-1	**-1**	**-1**

Обучение нейронной сети производится в соответствии с алгоритмом

обучения Розенблатта (рис.1).

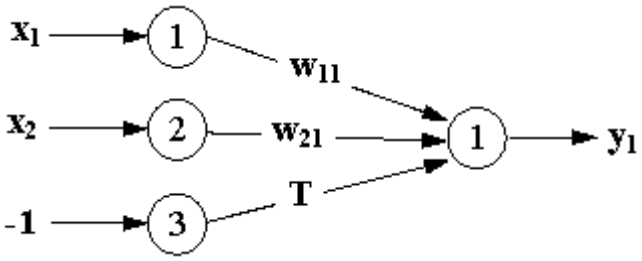


Рис.1. Архитектура сети

Значения сигналов – биполярные: **1** и **-1**. $OUT = \begin{cases} 1, \text{если } NET > 0 \\ -1, \text{если } NET \leq 0 \end{cases}$

Задаём коэффициент обучения $\eta = 1$, ошибка $e = 0$

Инициализируем работу НС: $w_{11} = 0,\ w_{21} = 0,\ T = 0$

Рассматриваем ***первую строку*** таблицы $x_1 = 1$, $x_2 = 1$, $y = 1$.

Определяем ошибку

$NET_0 = 1*0+1*0 - 1*0 = 0$, тогда $y_0 = -1$, а должно быть **1**

Вычисляем значение коэффициентов на каждом шаге:

$$w_{11}(t+1) = w_{11}(t) + \alpha(d - y_0)x_1 = 0 + 1*(1 - (-1))*1 = 2$$

$$w_{21}(t+1) = w_{21}(t) + \alpha(d - y_0)x_2 = 0 + 1*(1 - (-1))*1 = 2$$

$$T(t+1) = T(t) + \alpha(d - y_0)x_3 = 0 + 1(1 - (-1))-1 = -2$$

Рассматриваем ***вторую строку*** таблицы $x_1 = 1$, $x_2 = -1$, $y = -1$

Определяем ошибку

$NET_1 = 1*2 - 1*2 + 1*2 = 2$, тогда $y_1 = 1$, а должно быть **-1**

Вычисляем значение коэффициентов на каждом шаге:

$$w_{11}(t+2) = w_{11}(t) + \alpha(d - y_1)x_1 = 2 + 1*(-1 -1))*1 = 0$$

$$w_{21}(t+2) = w_{21}(t) + \alpha(d - y_1)x_2 = 2 + 1*(-1 -1)*-1 = 4$$

$$T(t+2) = T(t) + \alpha(d - y_0)x_3 = -2 + 1(-1 -1)*-1 = 0$$

Рассматриваем ***третью строку*** таблицы $x_1 = -1$, $x_2 = 1$, $y = -1$

Определяем ошибку

$NET_2 = -1*0 + 1*4 - 1*0 = 4$, тогда $y_2 = 1$, а должно быть **-1**

Вычисляем значение коэффициентов на каждом шаге:

$$w_{11}(t+3) = w_{11}(t) + \alpha(d - y_1)x_1 = 0 + 1*(-1 -1))*(-1) = 2$$

$$w_{21}(t+3) = w_{21}(t) + \alpha(d - y_1)x_2 = 4 + 1*(-1 -1)*1 = 2$$

$$T(t+3) = T(t) + \alpha(d - y_0)x_3 = 0 + 1(-1 -1)*-1 = 2$$

Рассматриваем ***четвертую строку*** таблицы $x_1 = -1$, $x_2 = -1$, $y = -1$

$NET_3 = -1*2 - 1*2 - 1*2 = -6$, тогда $y_2 = -1$, а должно быть **-1**

Ошибка e = 0

Соответственно, разделительная линия будет равна:

$2x_1 + 2x_2 - 2 = 0 \rightarrow x_1 + x_2 - 1 = 0$

Представленный алгоритм относится к широкому классу алгоритмов обучения с учителем, поскольку известны как входные вектора, так и требуемые значения выходных векторов (имеется учитель, способный оценить правильность ответа ученика).

Естественно, возникает вопрос, всегда ли алгоритм обучения персептрона приводит к желаемому результату. Ответ на этот вопрос дает теорема сходимости персептрона:

Если существует множество значений весов, которые обеспечивают требуемое распознавание образов, то в конечном итоге алгоритм обучения персептрона приводит либо к этому множеству, либо к другому множеству, такому, что требуемое распознавание образов будет достигнуто.

Как следует из этой теоремы, задача нахождения матрицы весовых коэффициентов w_j, обеспечивающих распознавание образов, может иметь множество решений—таких матриц может быть много. С другой стороны, в формулировке теоремы не говорится, что такие матрицы всегда существуют, и значит, не всегда существует решение задачи.

Распознавание букв

Дальнейшее развитие идеи персептрона и алгоритмов обучения связано с усложнением его структуры и функциональных свойств. На рис. 2 представлена схема персептрона, предназначенного для распознавания букв русского алфавита.

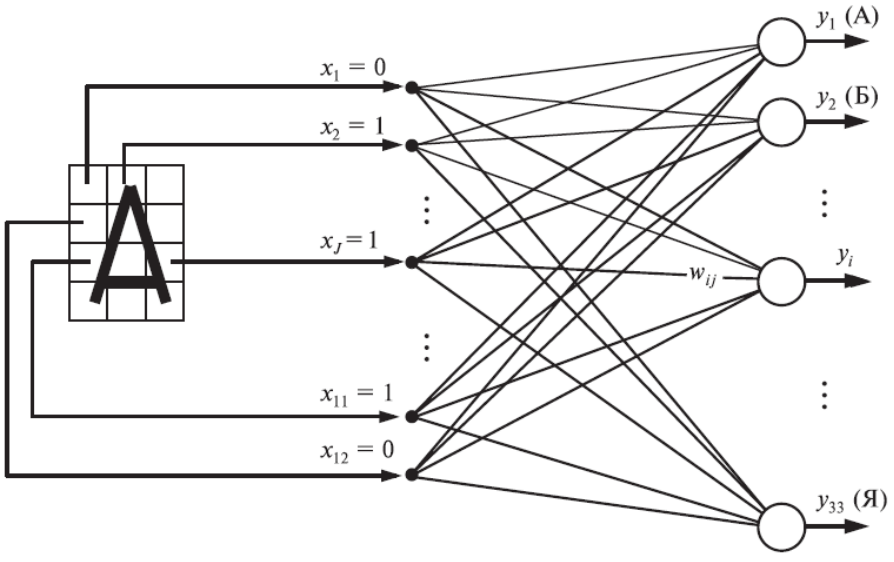


Рис.2. Персептрон, предназначенный для распознавания букв русского алфавита

В отличие от предыдущей схемы, такой персептрон имеет 33 выходных нейрона: каждой букве алфавита соответствует свой выходной нейрон. Полагается, что сигнал первого выходного нейрона y_1 должен быть равен единице, если персептрону предъявлена буква «А», и равен нулю, если предъявляется любая другая буква. Выход второго нейрона y_2 должен быть равен единице, если персептрону предъявлена буква «Б», и равен нулю во всех остальных случаях. И так далее до буквы «Я».

Как уже отмечалось ранее, первый действующий персептрон был создан в 1958–1961 гг. Он был предназначен для распознавания букв латинского алфавита. Буквы, отпечатанные на карточках, поочередно накладывали на табло фотоэлементов и осуществляли процесс обучения персептрона согласно приведенному здесь алгоритму. После выполнения достаточно большого количества эпох персептрон научился безошибочно распознавать все буквы, участвовавшие в обучении. Таким образом, была подтверждена гипотеза о том, что компьютер, построенный по образу и подобию человеческого мозга, может решать интеллектуальные задач и, в частности, решать задачу распознавания образов—букв латинского алфавита.

Но это было не все. Помимо того, что персептрон научился распознавать знакомые образы, т. е. те образы, которые демонстрировались ему в процессе обучения, он успешно справлялся с распознаванием образов, которые «видел» впервые. Выяснилось, что персептрон оказался способным распознавать буквы, отпечатанные с небольшими искажениями и даже другим шрифтом, если шрифт не слишком сильно отличался от используемого при обучении персептрона.

Свойство мозга узнавать образы, которые ему встретились впервые, называется свойством обобщения. Это свойство было унаследовано персептроном непосредственно от его прототипа – мозга. Оно было унаследовано благодаря тому, что персептрон является адекватной моделью мозга, удачно отражающей как его структурные, так и функциональные качества. Именно свойство обобщения впоследствии позволило применять нейронные сети для решения широчайшего круга практических задач, недоступных для традиционных методов информатики. Именно благодаря этому свойству нейронные сети стали эффективнейшим инструментом научных исследований и практических приложений. Именно благодаря этому свойству нейросетевые и нейрокомпьютерные технологии заняли то лидирующее положение, которое они занимают в настоящее время.

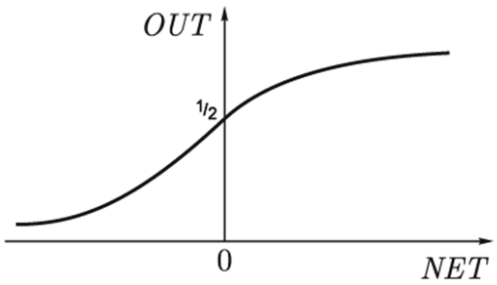
Обобщенное дельта-правило

Персептрон, схема которого приведена на рис. 2, предназначен для распознавания букв алфавита. Можно попытаться использовать его для решения других практических задач. Например, ставить диагнозы болезней или определять: свой или чужой самолет подлетает к границам страны. Все зависит от того, какой смысл придавать входному вектору x_j и выходному вектору y_i.

Так, например, если в качестве x_j на вход персептрона подавать сигналы, кодирующие симптомы заболевания человека, а в качестве y_i на выходе персептрона снимать сигналы, кодирующие диагнозы его заболеваний, то на основе такого персептрона можно построить систему медицинской диагностики.

Однако следует заметить, что для других классов задач, например прогнозирование погоды, температуры воздуха, прогнозирование котировок акций и курсов валют, такой персептрон не годится, так как он может выдавать только бинарные результаты типа «нуль» и «единица».

Круг решаемых задач значительно расширится, если научить персептрон выдавать не только бинарные выходные сигналы, но и аналоговые, т. е. имеющие непрерывные значения. Такое развитие персептрона предусматривает вместо ступенчатой активационной функции использовать непрерывную функцию. Например, сигмоид (рис. 3)



$$OUT = \frac{1}{1 + e^{-NET}}$$

Рис. 3. Сигмоидная активационная функция

Подобно обычной пороговой функции активации, сигмоида отображает точки области определения (−∞, +∞) в значения из интервала (0, +1). Практически сигмоида обеспечивает непрерывную аппроксимацию классической пороговой функции.

Появление персептронов с непрерывными активационными функциями обусловило появление новых подходов к их обучению. Б. Уидроу и М. Е. Хофф предложили минимизировать квадратичную ошибку, определяемую формулой:

$$\varepsilon = \frac{1}{2}\sum_{i=1}^{I}(d_i - y_i)^2 \qquad (1)$$

в которой, как и раньше, d_i —требуемый (желаемый) выход i-го нейрона, а y_i —выход, который получился в результате вычислений персептрона.

Рассмотрим алгоритм коррекции весовых коэффициентов персептрона, имеющего J входов и I выходов (рис. 4).

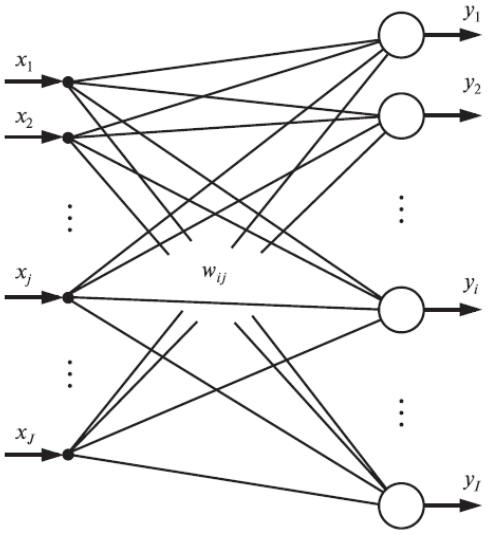


Рис. 4. Персептрон с J входами и I выходами

Квадратичная ошибка обучения персептрона ε зависит от того, какими являются весовые коэффициенты w_{ij} . Другими словами, ε является функцией от множества весовых коэффициентов: $\varepsilon = \varepsilon(w_{ij})$.

Для ее графического представления требуется многомерная система координат, которую мы в нашем трехмерном мире представить себе не можем. В этой многомерной системе координат функция $\varepsilon = \varepsilon(w_{ij})$ изображается в виде многомерной поверхности, называемой гиперповерхностью.

Чтобы хоть как-то представить себе гиперповерхность, предположим, что все аргументы w_{ij} «заморожены», т. е. не меняются, за исключением двух, например w_{ij} и w_{ij+1}, которые являются переменными. Тогда в трехмерной системе координат $(w_{ij}, w_{ij+1}, \varepsilon)$ гиперповерхность будет иметь вид фигуры, напоминающей параболоид, которую назовем псевдопарабалоидом (рис. 5).

Процесс обучения персептрона теперь можно представить, как отыскание такого сочетания весовых коэффициентов w_{ij}, которому соответствует самая нижняя точка гиперпсевдопараболоида. Задачи подобного рода называются оптимизационными. Говорят, что оптимизационная задача состоит в минимизации функции $\varepsilon = \varepsilon(w_{ij})$ в многомерном пространстве параметров w_{ij}.

Таким образом, если раньше мы говорили, что персептрон обучают методом «поощрения—наказания», то теперь мы будем говорить, что задача обучения персептрона—это задача оптимизации (минимизации) функции-ошибки персептрона $\varepsilon = \varepsilon(w_{ij})$. Иногда ее называют погрешностью персептрона.

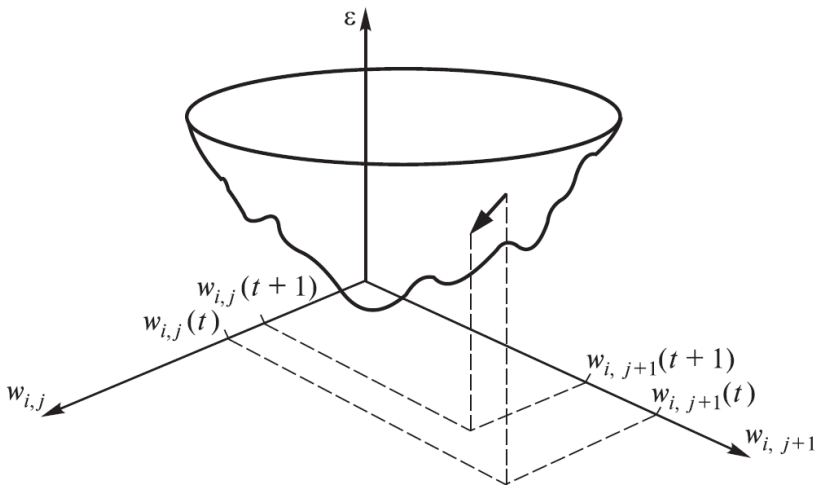


Рис. 5. Графическое изображение функции-ошибки персептрона $\varepsilon = \varepsilon(w_{ij})$ в трехмерной системе координат $w_{ij}, w_{ij+1}, \varepsilon$

Существует множество методов решения оптимизационных задач. Наиболее простым методом является перебор весовых коэффициентов w_{ij} с последующими вычислениями и сравнениями между собой значений функции ε, соответствующих этим коэффициентам. Однако более эффективны так называемые *градиентные методы (которые мы с Вами проходили)*.

Градиент функции является очень важным математическим понятием, с которым обычно знакомятся на первых курсах вузов. Напомним, что градиент функции $\varepsilon = \varepsilon(w_{ij})$ представляет собой вектор, проекциями которого на оси координат являются частные производные от функции ε по этим координатам $\partial\varepsilon/\partial w_{ij}$, и что градиент функции всегда направлен в сторону ее наибольшего возрастания. Поскольку задача состоит в отыскании минимума функции $\varepsilon = \varepsilon(w_{ij})$, то нам надо опускаться по поверхности ошибок, что обеспечивается движением в сторону, противоположную градиенту этой функции. Отсюда название—метод градиентного спуска.

Движение в сторону, противоположную градиенту, будет осуществляться, если на каждой эпохе к координатам текущей точки w_{ij} мы, используя знакомую нам итерационную формулу $w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}$, будем добавлять величину, прямо пропорциональную частной производной по координате w_{ij}, взятую с противоположным знаком:

$$\Delta w_{ij} = -\eta \frac{\partial \varepsilon}{\partial w_{ij}} \qquad (2)$$

Здесь η - некоторый коэффициент, обычно задаваемый в пределах от 0 до 1 и называемый, как и раньше, коэффициентом скорости обучения.

Квадратичная ошибка ε является сложной функцией, зависящей от

выходных сигналов персептрона $\boldsymbol{y_i}$, которые, в свою очередь, зависят от w_{ij}, т. е. $\varepsilon = \varepsilon(y_i(w_{ij}))$. По правилу дифференцирования сложной функции

$$\frac{\partial \varepsilon}{\partial w_{ij}} = \frac{\partial \varepsilon}{\partial y_i}\frac{\partial y_i}{\partial w_{ij}} \tag{3}$$

Выходные сигналы нейронов y_i вычисляются с помощью сигмоидных активационных функций $y_i = f_\sigma(S_j)$, аргументом которых являются суммы $S_j = \sum_{i=1}^{n} x_i\, w_{ij}$. Следовательно

$$\frac{\partial y_i}{\partial w_{ij}} = \frac{\partial f_\sigma(S_i)}{\partial S_i}\frac{\partial S_i}{\partial w_{ij}} = f'_\sigma(S_i)x_j \tag{4}$$

Аналогичным образом, вспоминая формулу (1) и выполняя дифференцирование ε по y_i, получаем

$$\frac{\partial \varepsilon}{\partial y_i} = -(d_i - y_i) \tag{5}$$

Подставив (4) и (5) в (3) и затем полученное выражение в (2), окончательно будем иметь

$$\Delta w_{ij} = -\eta\big(-(d_i - y_i)f'_\sigma(S_i)x_j\big) = \eta(d_i - y_i)f'_\sigma(S_i)x_j \tag{6}$$

Это выражение получено для нейронов с активационными функциями любого вида. Из этой формулы видно, что в качестве активационной функции при использовании обобщенного дельта-правила функция активации нейронов должна быть непрерывно дифференцируемой на всей оси абсцисс. Преимущество имеют функции активации с простой производной (например — логистическая кривая (сигмоид) или гиперболический тангенс).

Если $f_\sigma(S_i)$ – сигмоида, заданная формулой $F(x) = {1}/{(1 + e^{-x})}$ то

$$f'_\sigma(S_i) = ((1 + e^{-S_t})^{-1})' = f_\sigma(S_i)(1 - f_\sigma(S_i)) \tag{7}$$

Подставив это выражение в (6), получим:

$$\Delta w_{ij} = \eta(d_i - y_i)f_\sigma(S_i)\big(1 - f_\sigma(S_i)\big)x_j = \eta(d_i - y_i)y_i(1 - y_i)x_j \tag{8}$$

Таким образом, мы получили итерационную формулу для обучения персептрона

$$w_{ij}(t + 1) = w_{ij}(t) + \Delta w_{ij} \tag{9}$$

в которой

$$\Delta w_{ij} = \eta\delta_i x_j \tag{10}$$

$$\delta_i = y_i(1 - y_i)(d_i - y_i) \tag{11}$$

Величину δ_i, введенную здесь с помощью формулы (11), в дальнейшем будем называть нейронной ошибкой. Алгоритм (9)–(11) называют обобщенным дельта-правилом. Его преимущество по сравнению с обычным дельта-правилом состоит в более быстрой сходимости и в возможности более точной обработки

входных и выходных непрерывных сигналов, т. е. в расширении круга решаемых персептронами задач.

Итак, введение сигмоидной функции активации вместо функции ступеньки и появление нового алгоритма обучения – обобщенного дельта-правила – расширило область применения персептрона. Теперь он может оперировать не только с бинарными (типа «нуль» и «единица»), но и с непрерывными (аналоговыми) выходными сигналами.

На основе дельта-правила разработан ADALINE, который сразу начал использоваться для задач предсказания и адаптивного управления. Сейчас Адалин (адаптивный сумматор) является стандартным элементом многих систем обработки сигналов (рис.6).

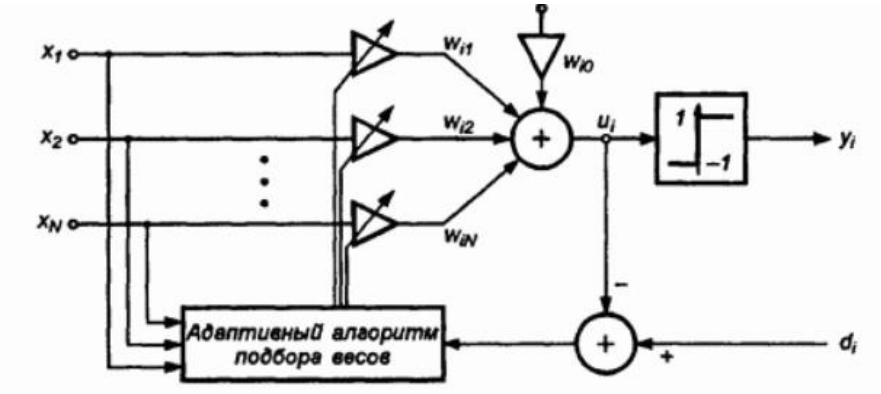


Рис. 6. Структурная схема типа ADALINE,

Продолжением дельта-правила является алгоритм обратного распространения ошибки.

Представление входных данных

Основная особенность нейросетей в том, что в них все входные и выходные параметры представлены в виде чисел с плавающей точкой обычно в диапазоне [0..1]. В тоже время данные предметной области часто имеют другое кодирование. Так это могут быть числа в произвольном диапазоне, даты, символьные строки. Таким образом, данные о проблеме могут быть как количественными так и качественными.

Рассмотрим сначала преобразование качественных данных в числовые, а затем рассмотрим способ преобразования входных данных в требуемый диапазон.

Качественные данные мы можем разделить на две группы: упорядоченные (ординальные) и неупорядоченные. Для рассмотрения способов кодирования этих данных мы рассмотрим задачу о прогнозировании успешности лечения какого-либо заболевания. Примером упорядоченных данных могут например

являться данные, например, о дополнительных факторах риска при данном заболевании.

Нет	Ожирение	Алкоголь	Курение	Гипертония

А также возможным примером может быть например возраст больного

До 25 лет	25-39 лет	40-49 лет	50-59 лет	60 и старше

Опасность каждого фактора возрастает в таблицах при движении слева направо.

В первом случае мы видим, что у больного может быть несколько факторов риска одновременно. В таком случае нам необходимо использовать такое кодирование, при котором отсутствует ситуация, когда разным комбинациям факторов соответствует одно и то же значение. Наиболее распространен способ кодирования, когда каждому фактору ставится в соответствие разряд двоичного числа.

Единица в этом разряде говорит о наличии фактора, а 0 о его отсутствии. Параметру нет можно поставить в соответствии число 0. Таким образом, для представления всех факторов достаточно 4-х разрядного двоичного числа. Таким образом число $1010_2 = 10_{10}$ означает наличие у больного гипертонии и употребления алкоголя, а числу 0000_2 соответствует отсутствие у больного факторов риска. Таким образом, факторы риска будут представлены числами в диапазоне [0..15].

Во втором случае мы также можем кодировать все значения двоичными весами, но это будет нецелесообразно, т.к. набор возможных значений будет слишком неравномерным. В этом случае более правильным будет установка в соответствие каждому значению своего веса, отличающегося на 1 от веса соседнего значения. Так число 3 будет соответствовать возрасту 50-59лет. Таким образом, возраст будет закодирован числами в диапазоне [0..4].

В принципе аналогично можно поступать и для неупорядоченных данных, поставив в соответствие каждому значению какое-либо число. Однако это вводит нежелательную упорядоченность, которая может исказить данные, и сильно затруднить процесс обучения. В качестве одного из способов решения этой проблемы можно предложить поставить в соответствие каждому значению одного из входов нейросети. В этом случае при наличии этого значения соответствующий ему вход устанавливается в 1 или в 0 при противном случае. К сожалению данный способ не является панацеей, ибо при большом количестве вариантов входного значения число входов нейросети разрастается до огромного количества. Это резко увеличит затраты времени на обучение. В качестве варианта обхода этой проблемы можно использовать несколько другое решение.

В соответствие каждому значению входного параметра ставится бинарный вектор, каждый разряд которого соответствует отдельному входу неросети. Например, если число возможных значений параметра 128, то можно использовать 7 разрядный вектор. Тогда 1 значению будет соответствовать вектор 0000000 а 128 - вектор 1111111, а ,например, 26 значению – 0011011. Тогда число требуемых для кодирования параметров входов можно определить как

$$N = log_2 n, \text{ где}$$

n - количество значений параметра

N - количество входов

Преобразование числовых входных данных

Для нейросети необходимо чтобы входные данные лежали в диапазоне [0..1], в то время как данные проблемной области могут лежать в любом диапазоне. Предположим, что данные по одному из параметров лежат в диапазоне [Min…Max]. Тогда наиболее простым способом нормирования будет

$$\tilde{x} = \frac{x - min}{max - min}$$

где x - исходное значение параметра;

$\tilde{x}$ - значение, подаваемое на вход нейросети.

К сожалению, этот способ кодирования не лишен недостатков. Так в случае если $max \gg \tilde{x}$ то распределение данных на входе может принять вид (рис. 7)

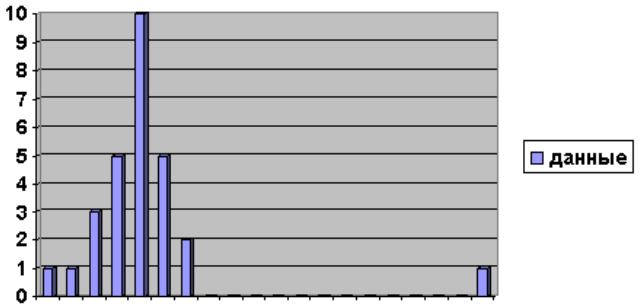


Рис. 7. Недостаток кодирования по max и min

Т.е. распределение входных параметров будет крайне неравномерным, что приведет к ухудшению качества обучения. Поэтому в подобных ситуациях , а также в случае, когда значение входа лежит в диапазоне $[0, \infty)$ можно использовать нормировку с помощью функции вида

$$\tilde{x} = \frac{1}{1 + e^{-x}}$$

Посмотрите примеры на сайте научного совета РАН по методологии искусственного интеллекта http://www.permai.ru/# (в разделе «ПРОЕКТЫ»)