2. ЛЕКЦИЯ. Функция «исключающего ИЛИ»

Ограниченность однослойного персептрона

Итак, ученым удалось обучить персептрон распознавать буквы алфавита. Это был колоссальный успех: Электронное устройство, созданное по образу и подобию человеческого мозга, обученное подобно человеку, успешно моделировало интеллектуальные функции человека. Это был успех в познании самой природы человеческого мышления. Мозг начал раскрывать свои тайны. Появилась возможность исследовать мозг методами моделирования, не прибегая к сложнейшим антигуманным и мало что дающим натурным экспериментам. Это была сенсация, приковавшая к себе внимание мыслящих людей всего мира. Казалось, что ключ к интеллекту был найден и полное воспроизведение человеческого мозга и всех его функций—всего лишь вопрос времени. Писателям-фантастам, ученым, инженерам, бизнесменам, политикам виделись самые радужные перспективы практического применения идей искусственного интеллекта. Правительство Соединенных Штатов Америки выделило крупные субсидии на развитие нового перспективного научного направления.

Благодаря изобретению сигмоидных активационных функций и алгоритма градиентного спуска класс решаемых нейросетями задач расширялся. Делались попытки применения персептронов для решения задач прогнозирования, таких как предсказание погоды, курсов валют и акций. Персептроны пытались применять для анализа электрокардиограмм, для решения задач медицинской диагностики.

Но по мере расширения фронта научных исследований появлялись трудности. Неожиданно оказалось, что многие новые задачи персептрон решить не мог, потому что с ростом числа эпох ошибка обучения ε не стремилась к нулю. Если для одних задач кривая, изображающая зависимость ε от t быстро приближалась к оси абсцисс (рис. 1, а), то для других задач (и их было подавляющее большинство!) погрешность ε обучения не удавалось снизить даже при большом количестве эпох t (рис. 1, б).

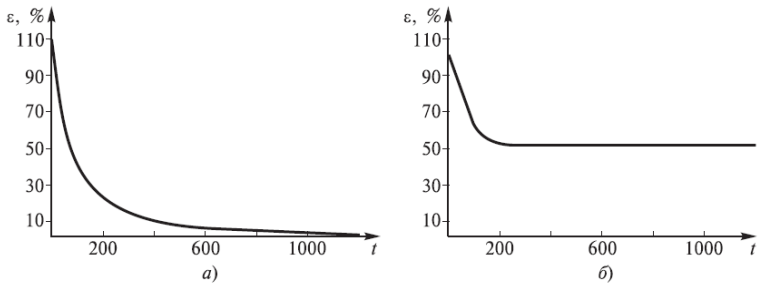

Рис. 1. Случай, когда ошибка обучения ε с ростом числа эпох t стремится к нулю (а), и случай, когда такого стремления нет (б)

В качестве примера можно привести провал американского проекта создания системы противовоздушной обороны с использования персептрона. Ни огромные гранты научных исследований, ни успехи в создании быстродействующих компьютеров не помогли обучить персептрон решать задачи распознавания движущихся военных объектов на «свой» и «чужой». Причем эти новые важные задачи в плане математической постановки практически ничем не отличались от тех, с которыми персептрон успешно справлялся ранее. Время шло, итерации продолжались, а погрешность обучения не падала. Возникла необходимость объяснения парадоксов, глубокого анализа и создания теоретической базы нейроинформатики.

Следующий период истории искусственного интеллекта начался с появления в 1969 г. книги двух известных американских математиков *М. Минского* и *С. Пайперта «Персептроны»*. Авторы этой книги математически строго доказали, что использовавшиеся в то время однослойные персептроны в принципе не способны решать многие простые задачи. Одну из таких задач, вошедшую в историю нейроинформатики под названием проблемы «Исключающего ИЛИ», мы рассмотрим подробно.

Напомню, «Исключающее ИЛИ» - это логическая функция двух аргументов, каждый из которых может иметь значение «истинно» либо «ложно». Сама она принимает значение «истинно», когда только один из аргументов имеет значение «истинно». Во всех остальных случаях эта функция принимает значение «ложно» (табл.1).

Таблица 1. Истинность логической функции «Исключающее ИЛИ»

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Логическая функция «Исключающее ИЛИ» может быть выражена через функции логического умножения «И» («AND»), логического сложения «ИЛИ» («OR») и логического отрицания «НЕТ» («NOT») с помощью логической формулы

$$y = (x_1 AND\ NOT\ x_2)\ OR\ (x_2 AND\ NOT\ x_1) \quad (1)$$

Задача состоит в том, чтобы научиться моделировать функцию «Исключающее ИЛИ» с помощью однонейронного персептрона с двумя входами x_1 и x_2 и одним выходом y (рис. 2).

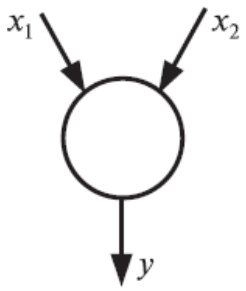

Рис. 2. Однонейронный персептрон с двумя входами и одним выходом

М. Минский и С. Пайперт предложили геометрическую интерпретацию к проблеме «Исключающего ИЛИ», состоящую в следующем. Они предложили изобразить на координатной плоскости x_1, x_2 все возможные комбинации входных сигналов в виде четырех точек: A, B, C, D, как показано на рис. 3.

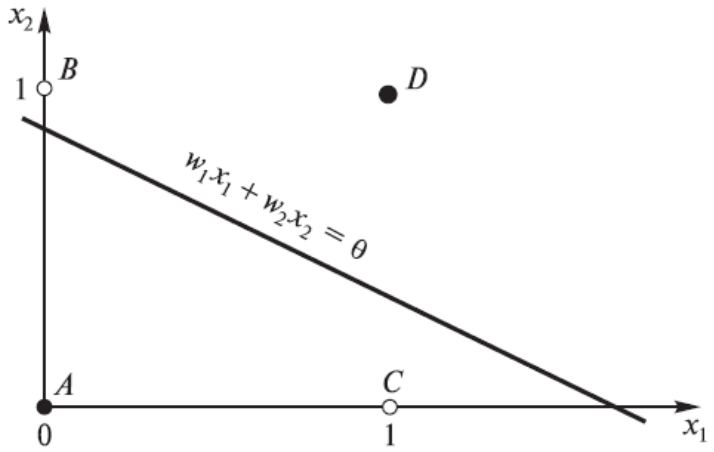

Рис. 3. Геометрическая интерпретация к объяснению проблемы «Исключающего ИЛИ»

Точка A имеет координаты $x_1 = 0, x_2 = 0$; точка B имеет координаты $x_1 = 0, x_2 = 1$ и т. д. согласно табл. 2.

Таблица 2. Истинность логической функции «Исключающее ИЛИ», дополненная точками A, B, C, D

Точки	x_1	x_2	y
A	0	0	0
B	0	1	1
C	1	0	1
D	1	1	0

В соответствии с формулами однонейронный персептрон осуществляет преобразование

$$S = w_1x_1 + w_2x_2; \tag{2}$$

$$y = \begin{cases} 1, \text{если } S \geq \theta \\ 0, \text{если } S < \theta \end{cases} \tag{3}$$

Рассмотрим случай, когда $S = \theta$. Это значит, что, согласно (1), выполняется равенство

$$w_1 x_1 + w_2 x_2 = \theta$$

Если в этом уравнении величины x_1 и x_2 считать переменными, а θ, w_1 и w_2 – константами, то на координатной плоскости x_1, x_2 рассматриваемое уравнение изобразится в виде прямой линии, положение и наклон которой определяются значениями коэффициентов w_1, w_2 и порога θ. Для всех точек плоскости x_1, x_2, лежащих на этой линии, выполняется равенство $S = \theta$, и поэтому, согласно формуле (2), выходной сигнал персептрона равен единице. Для точек, лежащих выше указанной линии, сумма $w_1 x_1 + w_2 x_2$ больше θ, и поэтому, согласно формулам (1) и (2), выходной сигнал персептрона так-же равен единице. Для точек же, лежащих ниже этой линии, сумма $w_1 x_1 + w_2 x_2$ меньше θ, и выходной сигнал персептрона равен нулю. Поэтому линию, изображающую уравнение (3), называют *пороговой прямой*.

А теперь посмотрим на таблицу 2 истинности функции «Исключающее ИЛИ». Согласно этой таблице, в точках A и D выход персептрона должен быть нулевым, а в точках B и C —единичным. Но для этого надо расположить пороговую прямую так, чтобы точки A и D лежали ниже этой линии, а точки B и C —выше, что невозможно. Это значит, что, сколько бы персептрон ни обучали, какие бы значения ни придавали его синаптическим весам и порогу, персептрон в принципе не способен воспроизвести соотношение между входами и выходом, требуемое таблицей истинности функции «Исключающее ИЛИ».

Помимо проблемы «Исключающего ИЛИ» в упомянутой выше книге М. Минский и С. Пайперт привели ряд других задач, в которых точки, изображающие входные сигналы, не могут быть разделены пороговой прямой (в многомерных случаях—плоскостью, гиперплоскостью). Такие задачи получили название линейно неразделимых.

После выхода в свет книги М. Минского и С. Пайперта «Персептроны» всем стало ясно, что предпринимавшиеся в то время попытки обучать персептроны решению многих задач, которые, как оказалось, относятся к классу линейно неразделимых, с самого начала были обречены на провал. Это была пустая трата времени, сил и финансовых ресурсов. Успешность же обучения персептрона распознаванию букв латинского алфавита – это счастливая случайность – задача оказалась линейно разделимой, что в жизни встречается крайне редко.

Линейная разделимость

Линейная разделимость ограничивает однослойные сети задачами классификации, в которых множества точек (соответствующих входным значениям) могут быть разделены геометрически. Для нашего случая с двумя входами разделитель является прямой линией. В случае трех входов разделение осуществляется плоскостью, рассекающей трехмерное пространство. Для четырех или более входов визуализация невозможна, и необходимо мысленно представить ***n-мерное*** пространство, рассекаемое «гиперплоскостью» — геометрическим объектом, который делит пространство четырех или большего числа измерений.

Так как линейная разделимость ограничивает возможности персептронного представления, то важно знать, является ли данная функция разделимой. К сожалению, не существует простого способа определить это, если число переменных велико.

Нейрон с n двоичными входами может иметь 2^n различных входных образов, состоящих из нулей и единиц. Так как каждый входной образ может соответствовать двум различным бинарным выходам (единица и ноль), то всего имеется 2^{2^n} функций от n переменных. При возрастании числа аргументов ситуация еще более катастрофична: относительное число функций, которые обладают свойством линейной разделимости резко уменьшается.

Значит и резко сужается класс функций, который может быть реализован персептроном (так называемый класс функций, обладающий свойством персептронной представляемости). Соответствующие данные приведены в следующей таблице:

Число переменных N	Полное число возможных логических функций 2^{2^n}	Из них линейно разделимых функций
1	4	4
2	16	14
3	256	104
4	65536	1882
5	> 1000000000	94572

Видно, что однослойный персептрон крайне ограничен в своих возможностях точно представить наперед заданную логическую функцию.

Нужно отметить, что позднее, в начале 70-х годов, это ограничение было преодолено путем введения нескольких слоев нейронов, однако критическое отношение к классическому персептрону сильно заморозило общий круг интереса и научных исследований в области искусственных нейронных сетей.

Преодоление ограничения линейной разделимости

К концу 1960-х годов проблема линейной разделимости была хорошо понята. К тому же, было известно, что это серьезное ограничение представляемости однослойными сетями можно преодолеть, добавив дополнительные слои. Например, двухслойные сети можно получить каскадным соединением двух *однослойных сетей*. Они способны выполнять более общие классификации, отделяя те точки, которые содержатся в выпуклых ограниченных или неограниченных областях. Область называется выпуклой, если для любых двух ее точек соединяющий их *отрезок* целиком лежит в области. Область называется ограниченной, если ее можно заключить в некоторый круг. Неограниченную область невозможно заключить внутрь круга (например, область между двумя параллельными линиями). Примеры выпуклых ограниченных и неограниченных областей представлены на рис. 4.

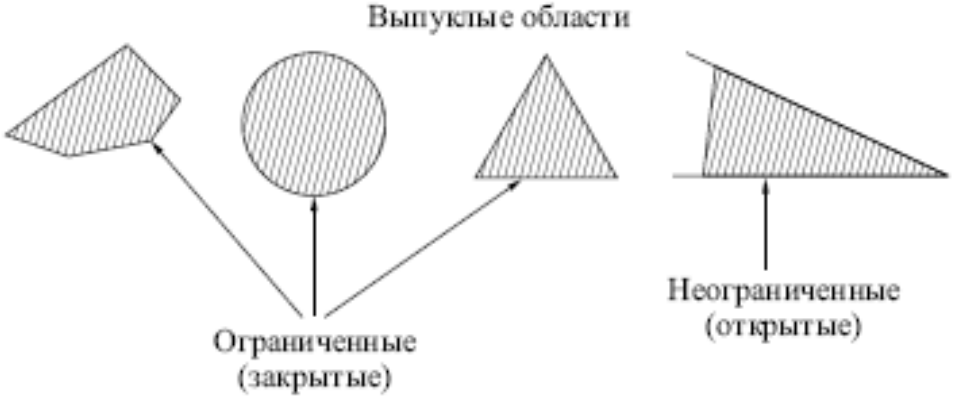

Рис. 4. Примеры выпуклых ограниченных и неограниченных областей

Чтобы уточнить требование выпуклости, рассмотрим простую двуслойную сеть с двумя входами, которые подведены к двум нейронам первого слоя, соединенными с единственным нейроном в слое 2 (рис. 5).

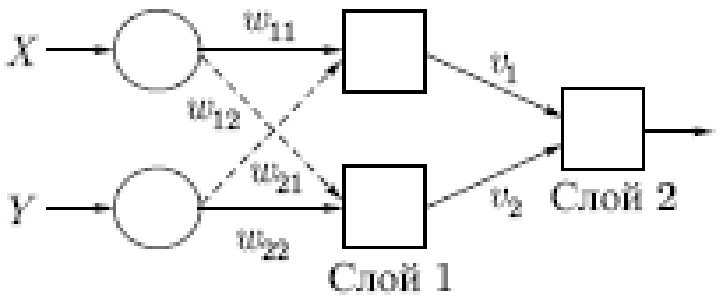

Рис. 5. Двуслойная сеть

Пусть порог выходного нейрона равен 0,75, а оба его веса равны 0,5. В этом случае для того, чтобы порог был превышен и на выходе появилась единица, требуется, чтобы оба нейрона первого уровня на выходе имели единицу. Таким образом, выходной нейрон реализует логическую функцию И. На рис. 5 каждый нейрон слоя 1 разбивает плоскость XOY на две полуплоскости, один обеспечивает единичный выход для входов ниже верхней линии, другой — для входов выше нижней линии. На рис. 6 показан результат такого двойного разбиения, где выходной сигнал нейрона второго слоя равен единице только внутри V-образной области.

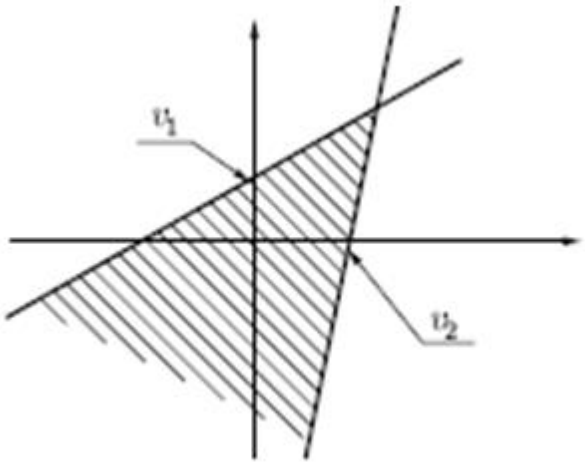

Рис. 6. Результат двойного разбиения

Аналогично, во втором слое может быть использовано три нейрона с дальнейшим разбиением плоскости и созданием области треугольной формы. Включением достаточного числа нейронов во входной слой может быть образован выпуклый многоугольник любой желаемой формы. Все такие многогранники выпуклы, так как они образованы с помощью операции *И* над областями, задаваемыми линиями: следовательно, только выпуклые области и возникают. Точки, не составляющие выпуклой области, не могут быть отделены от других точек плоскости двухслойной сетью.

Нейрон второго слоя не ограничен функцией *И*. Он может реализовывать многие другие функции при подходящем выборе весов и порога. Например, можно сделать так, чтобы единичный выход любого из нейронов первого слоя приводил к появлению единицы на выходе нейрона второго слоя, реализовав тем самым логическое *ИЛИ*. Например, имеется 16 двоичных функций от двух переменных. Если выбирать подходящим образом веса и порог, то можно воспроизвести 14 из них (все, кроме *исключающее ИЛИ* и *исключающее НЕТ*).

Входы не обязательно должны быть двоичными. Вектор непрерывных входов может представлять собой произвольную точку на плоскости XOY. В этом случае мы имеем дело со способностью сети разбивать плоскость на

непрерывные области, а не с разделением дискретных множеств точек. Для всех этих функций, однако, линейная разделимость показывает, что выход нейрона второго слоя равен единице только в части плоскости XOY, ограниченной многоугольной областью. Поэтому для разделения плоскостей P и Q необходимо, чтобы все P лежали внутри выпуклой многоугольной области, не содержащей точек Q (или наоборот).

Трехслойная *сеть*, впрочем, есть более общий случай. Ее классифицирующие возможности ограничены лишь числом искусственных нейронов и весов. Ограничения на выпуклость отсутствуют. Теперь нейрон третьего слоя принимает в качестве входа набор выпуклых многоугольников, и их логическая комбинация может быть невыпуклой. На рис. 6 иллюстрируется ситуация, когда два треугольника A и B, скомбинированные с помощью функций "A и не B", задают невыпуклую область. При добавлении нейронов и весов число сторон многоугольников может неограниченно возрастать. Это позволяет аппроксимировать область любой формы с любой точностью. Вдобавок, не все выходные области второго слоя должны пересекаться. Возможно, следовательно, объединять различные области, выпуклые и невыпуклые, выдавая на выходе единицу всякий раз, когда входной вектор принадлежит одной из них.

Несмотря на то, что возможности многослойных сетей были известны давно, в течение многих лет не было теоретически обоснованного алгоритма для настройки их весов. Мы детально изучим многослойные обучающие алгоритмы, но сейчас достаточно понимать суть проблемы и знать, что исследования привели к определенным результатам.

Эффективность запоминания

Серьезные вопросы существуют относительно эффективности запоминания информации в персептроне (или любых других нейронных сетях) по сравнению с обычной компьютерной памятью и методами поиска информации в ней. Например, в компьютерной памяти можно хранить все входные образы вместе с классифицирующими битами. Компьютер должен найти требуемый образ и дать его классификацию. Многочисленные и хорошо известные методы могли бы применяться для ускорения поиска. Если точное соответствие не найдено, то для ответа может быть использовано правило ближайшего соседа.

Число битов, необходимое для хранения этой же информации в весах персептрона, может быть значительно меньшим по сравнению с методом обычной компьютерной памяти, если образы допускают экономичную запись.

Однако М.Л. Минский построил патологические примеры, в которых число битов, требуемых для представления весов, растет в зависимости от размерности задачи быстрее, чем экспоненциально. В этих случаях требования к памяти быстро становятся невыполнимыми. Если, как он предположил, эта ситуация не является исключением, то персептроны часто могут быть ограничены только малыми задачами. Насколько общими являются такие неподатливые множества образов? Вопрос остается открытым и относится ко всем нейронным сетям. Поиски ответа чрезвычайно важны для дальнейших исследований в этой области.

Пример решения задачи «исключающее ИЛИ»

Рассмотрим один из таких способов решение задачи. Модернизуем сеть на рисунке, добавив еще один скрытый слой нейронов ():

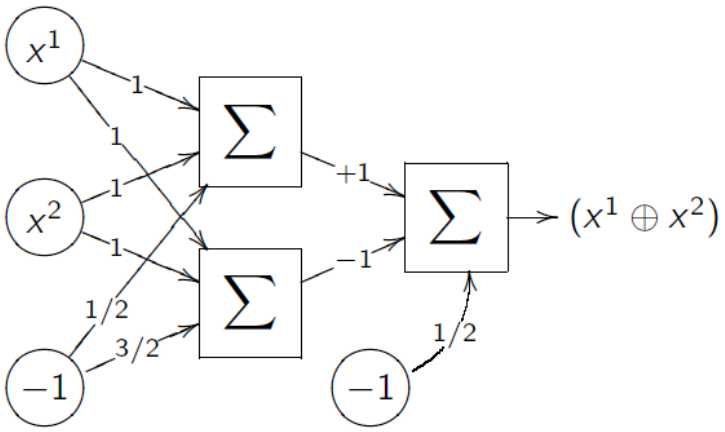

Рис. 7. Архитектура двухслойной сети

Тогда если посмотрите предыдущую лекцию. Выражение должно содержать *логические И , ИЛИ* (рис.8)

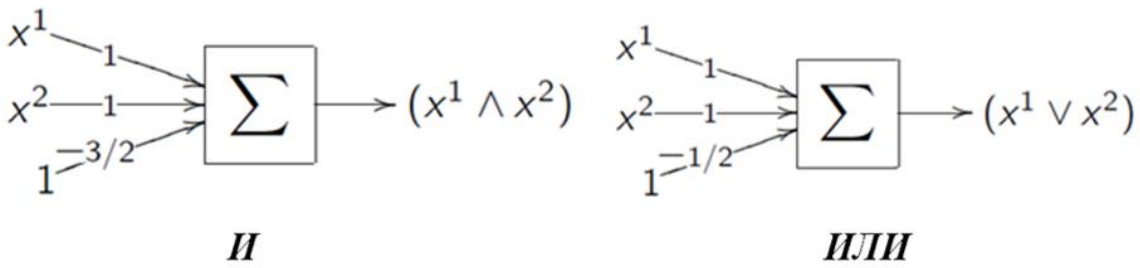

Рис.8. Элементы сети

Тогда можно записать следующее выражение

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

Логическое И имеет выходы

OUT₁″ = 0, OUT₂″ = 0, OUT₃″ = 0, OUT₄″ = 1

Логическое ИЛИ имеет выходы

$\boldsymbol{OUT_1^{или} = 0, OUT_2^{или} = 1, OUT_3^{или} = 1, OUT_4^{или} = 1}$

Таким образом, при $OUT = \begin{cases} 1, \text{если } NET > 0 \\ 0, \text{если } NET \leq 0 \end{cases}$ *имеем*

$\boldsymbol{OUT_1^{или} = 0; OUT_1^{и} = 0.}$ → 0*1 - 0*1 - 0,5 = - 0,5 → **0**

$\boldsymbol{OUT_2^{или} = 1, OUT_2^{и} = 0}$ → 1*1 - 0*1 - 0,5 = 0,5 → **1**

$\boldsymbol{OUT_3^{или} = 1, OUT_3^{и} = 0}$ → 1*1 - 0*1 - 0,5 = 0,5 → **1**

$\boldsymbol{OUT_4^{или} = 1, OUT_4^{и} = 1}$ → 1*1 - 1*1 - 0,5 = - 0,5 → **0**

Если у *логических И* и *ИЛИ* графическая интерпретация следующая (рис.9)

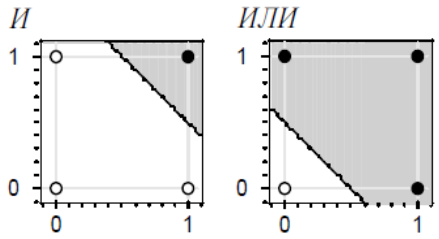

Рис. 9. Интерпретация логических И и ИЛИ

то соответственно для *исключающей ИЛИ* (рис. 10)

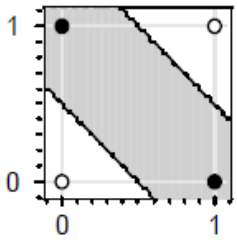

Рис.10. Интерпретация исключающей ИЛИ

Выбор архитектуры искусственной нейронной сети определяется задачей. Для некоторых классов задач уже существуют оптимальные конфигурации. Если же задача не может быть сведена ни к одному из известных классов, разработчику приходится решать задачу синтеза новой конфигурации. Проблема синтеза искусственной нейронной сети сильно зависит от задачи, дать общие подробные рекомендации затруднительно. В большинстве случаев оптимальный вариант искусственной нейронной сети получается опытным путем.

Многослойные искусственные нейронные сети

Многослойные сети (персептроны) обладают большими возможностями, чем однослойные, и в последние годы были разработаны алгоритмы для их обучения. Многослойные персептроны имеют три отличительных признака.

1. Каждый нейрон сети имеет нелинейную функцию активации. Важно подчеркнуть, что данная нелинейная функция является гладкой (т.е. всюду дифференцируемой), в отличие от жесткой пороговой функции, используемой в персептроне Розенблатта. Самой популярной формой функции,

удовлетворяющей этому требованию, является сигмоидальная.

2. Сеть содержит один или несколько слоев скрытых нейронов, не являющихся частью входа или выхода сети. Эти нейроны позволяют сети обучаться решению сложных задач, последовательно извлекая наиболее важные признаки из входного образа (вектора).

3. Сеть обладает высокой степенью связности, реализуемой посредством синаптических соединений. Изменение уровня связности сети требует изменения множества синаптических соединений или их весовых коэффициентов.

Комбинация всех этих свойств наряду со способностью к обучению на собственном опыте обеспечивает вычислительную мощность многослойного персептрона. Однако эти же качества являются причиной неполноты современных знаний о поведении такого рода сетей. Во-первых, распределенная форма нелинейности и высокая связность сети существенно усложняют теоретический анализ многослойного персептрона. Во-вторых, наличие скрытых нейронов делает процесс обучения более трудным для визуализации. Именно в процессе обучения необходимо определить, какие признаки входного сигнала следует представлять скрытыми нейронами. Тогда процесс обучения становится еще более сложным, поскольку поиск должен выполняться в очень широкой области возможных функций, а выбор должен производиться среди альтернативных представлений входных образов.

На рис. 11 показан архитектурный граф многослойного персептрона с двумя скрытыми слоями и одним выходным слоем.

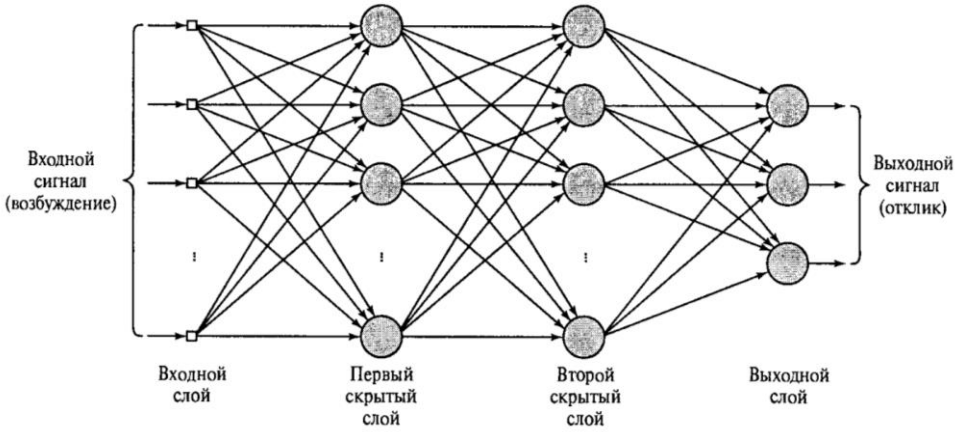

Рис. 11. Архитектурный граф персептрона с двумя скрытыми слоями

Показанная на рисунке сеть является полностью связанной, что характерно для многослойного персептрона общего вида. Это значит, что каждый нейрон в любом слое сети связан со всеми нейронами (узлами) предыдущего слоя. Сигнал передается по сети исключительно в прямом направлении, слева направо, от слоя

к слою. Выход одного слоя является входом для последующего слоя. ***Многослойные сети не могут привести к увеличению вычислительной мощности по сравнению с однослойной сетью, если активационная функция между слоями линейна.*** Вычисление выхода слоя заключается в умножении входного вектора на первую весовую матрицу с последующим умножением (если отсутствует нелинейная активационная функция) результирующего вектора на вторую весовую матрицу

$$OUT = (XW_1)\ W_2$$

Так как умножение матриц ассоциативно (сочетательное свойство), то

$$(XW_1)\ W_2 = X(W_1W_2)$$

Это показывает, что двухслойная линейная сеть эквивалентна одному слою с весовой матрицей, равной произведению двух весовых матриц. Следовательно, любая многослойная линейная сеть может быть заменена эквивалентной однослойной сетью. Однако однослойные сети весьма ограничены по своим вычислительным возможностям. Таким образом, для расширения возможностей сетей по сравнению с однослойной сетью необходима нелинейная активационная функция.

Метод обратного распространения ошибок

Введение в процедуру обратного распространения

Среди различных структур нейронных сетей (НС) одной из наиболее известных является многослойная структура, в которой каждый нейрон произвольного слоя связан со всеми аксонами нейронов предыдущего слоя или, в случае первого слоя, со всеми входами НС. Такие НС называются полносвязными. Когда в сети только один слой, алгоритм ее обучения с учителем довольно очевиден, так как правильные выходные состояния нейронов единственного слоя заведомо известны и подстройка синаптических связей идет в направлении, минимизирующем ошибку на выходе сети. По этому принципу строится, например, алгоритм обучения однослойного персептрона. В многослойных же сетях оптимальные выходные значения нейронов всех слоев, кроме последнего, как правило, не известны, и двух- или более слойный персептрон уже невозможно обучить, руководствуясь только величинами ошибок на выходах НС.

Один из вариантов решения этой проблемы — разработка наборов выходных сигналов, соответствующих входным, для каждого слоя НС, что, конечно, является очень трудоемкой операцией и не всегда осуществимо. Второй вариант — динамическая подстройка весовых коэффициентов синапсов, в ходе которой

выбираются, как правило, наиболее слабые связи и изменяются на малую величину в ту или иную сторону, а сохраняются только те изменения, которые повлекли уменьшение ошибки на выходе всей сети. Очевидно, что данный «метод тыка», несмотря на свою кажущуюся простоту, требует громоздких рутинных вычислений. И, наконец, третий, более приемлемый вариант — 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. Разработка алгоритма обратного распространения сыграла важную роль в возрождении интереса к искусственным нейронным сетям. Обратное распространение – это систематический метод для обучения многослойных искусственных нейронных сетей. Он имеет солидное математическое обоснование. Несмотря на некоторые ограничения, процедура обратного распространения сильно расширила область проблем, в которых могут быть использованы искусственные нейронные сети, и убедительно продемонстрировала богатые возможности этой методики.

На рис. 1 показан фрагмент многослойного персептрона. Для этого типа сети выделяют два типа сигналов:

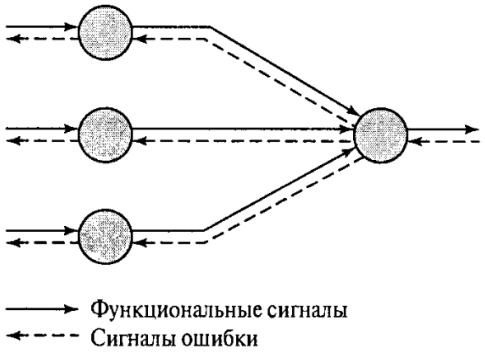

Рис. 4.2. Направление двух основных потоков сигнала для многослойного персептрона: прямое распространение функционального сигнала и обратное распространение сигнала ошибки.

1. Функциональный сигнал. Это входной сигнал (стимул), поступающий в сеть и передаваемый вперед от нейрона к нейрону по всей сети. Такой сигнал достигает конца сети в виде выходного сигнала. Будем называть этот сигнал функциональным по двум причинам. Во-первых, он предназначен для выполнения некоторой функции на выходе сети. Во-вторых, в каждом нейроне, через который передается этот сигнал, вычисляется некоторая функция с учетом весовых коэффициентов.

2. Сигнал ошибки. Сигнал ошибки берет свое начало на выходе сети и распространяется в обратном направлении (от слоя к слою). Он получил свое

название благодаря тому, что вычисляется каждым нейроном сети на основе функции ошибки, представленной в той или иной форме.

Выходные нейроны (вычислительные узлы) составляют выходной слой сети. Остальные нейроны (вычислительные узлы) относятся к скрытым слоям. Таким образом, скрытые узлы не являются частью входа или выхода сети отсюда они и получили свое название. Первый скрытый слой получает данные из входного слоя, составленного из сенсорных элементов (входных узлов). Результирующий сигнал первого скрытого слоя, в свою очередь, поступает на следующий скрытый слой, и т.д., до самого конца сети.

Любой скрытый или выходной нейрон многослойного персептрона может выполнять два типа вычислений.

1. Вычисление функционального сигнала на выходе нейрона, реализуемое в виде непрерывной нелинейной функции от входного сигнала и синаптических весов, связанных с данным нейроном.

2. Вычисление оценки вектора градиента (т.е. градиента поверхности ошибки по синаптических весам, связанным со входами данного нейрона), необходимого для обратного прохода через сеть.

Вывод алгоритма обратного распространения слишком громоздок.

Обучение методом обратного распространения ошибок

Для обучения многослойной сети в 1986 г. Руммельхартом и Хинтоном был предложен алгоритм обратного распространения ошибок (error back propagation). Многочисленные публикации о промышленных применениях многослойных сетей с этим алгоритмом обучения подтвердили его принципиальную работоспособность на практике.

Основная идея обратного распространения состоит в том, как получить оценку ошибки для нейронов скрытых слоев. Заметим, что известные ошибки, делаемые нейронами выходного слоя, возникают вследствие неизвестных пока ошибок нейронов скрытых слоев. Чем больше значение синоптической связи между нейроном скрытого слоя и выходным нейроном, тем сильнее ошибка первого влияет на ошибку второго. Следовательно, оценку ошибки элементов скрытых слоев можно получить, как взвешенную сумму ошибок последующих слоев. При обучении информация распространяется от низших слоев иерархии к высшим, а оценки ошибок, делаемые сетью - в обратном направлении, что и отражено в названии метода.

Таким образом, входные сигналы двигаются в прямом направлении, в результате чего мы получаем выходной сигнал, из которого мы получаем значение ошибки. Величина ошибки двигается в обратном направлении, в

результате происходит корректировка весовых коэффициентов связей сети.

Общая структура алгоритма аналогична, алгоритму обучения Розенблатта (дельта-правило) с усложнением формул подстройки весов. В качестве активационной функции в многослойных персептронах, как правило, используется сигмоидальная активационная функция, в частности логистическая:

$$\boldsymbol{f(U) = {1}/{(1 + e^{-x})}} \text{ или } \boldsymbol{OUT = {1}/{(1 + e^{-NET})}}$$

Вспомним, что производная этой функции равна:

$$\boldsymbol{OUT^{/}} = \text{OUT}(1 - \text{OUT})$$

Рассмотрим алгоритм обратного распространения ошибки, который требует выполнения следующих операций:

Шаг 1. Выбрать очередную обучающую пару из обучающего множества; подать входной вектор на вход сети. Берется пример входного сигнала с соответствующим правильным значением выхода.

Шаг 2. Вычислить выход сети. При этом нейроны последовательно от слоя к слою функционируют по следующим формуле.

Операции, выполняемые шагами 1 и 2, сходны с теми, которые выполняются при функционировании уже обученной сети, – подается входной вектор и вычисляется получающийся выход. Вычисления выполняются послойно. На шаги 1 и 2 можно смотреть как на «движение вперед», так как сигнал распространяется по сети от входа к выходу

Шаг 3. Рассчитывается прямое распространение ошибки через сеть. Берется, функционал суммарной квадратичной ошибки сети для одного входного образа имеет вид:

$$E = 0{,}5 \sum_{n} (d_j - y_j)^2$$

где dj – целевое значение входного сигнала; yj – реальное значение входного сигнала.

Шаг 4. Начиная с выходов, выполняется обратное движение через ячейки выходного и промежуточного слоя, при этом программа рассчитывает значения:

Для выходной ячейки

$$\boldsymbol{\delta_0 = y_j(1 - y_j)(d_j - y_j)} \text{ или } \boldsymbol{\delta_0 = OUT(1 - OUT)(OUT^{*} - OUT)};$$

где OUT^{*} - целевое значение.

OUT – реальное значение

$OUT(1 - OUT)$ – производная от сигмоида.

Выход нейрона слоя, вычитаемый из целевого значения, дает сигнал ошибки.

Он умножается на производную сжимающей функции $OUT(1-OUT)$, вычисленную для этого нейрона, давая, таким образом, величину обратной ошибки в узле.

Для скрытых ячеек

$$\boldsymbol{\delta_j = OUT(1-OUT)\sum_k \omega_{kj}\,\delta_k}$$

$\boldsymbol{\delta_j}$ – ошибка элемента с индексом j;

k – индекс, соответствующий слою, который посылает ошибку «обратно»;

$\boldsymbol{\sum_k \omega_{kj}\,\delta_k}$ – обозначает все ячейки, связанные со скрытым слоем, $\boldsymbol{\omega_{ij}}$ – заданный вектор веса в скрытом слое, $\boldsymbol{\delta_k}$ – обратная ошибка от слоя, который ее посылает.

Шаг 5. Рассчитываем величину, на которую необходимо изменить значения весовых коэффициентов. При этом используется дельта-праило.

Тогда для весов соединений между скрытым слоем и выходом

$$\boldsymbol{w_{ij}^* = w_{ij} + \alpha\,\delta_0 f(U_i)}$$

где i – номер нейрона в выходном слое; j – номер нейрона в скрытом слое; α – коэффициент обучения; $\boldsymbol{f(U_i)}$– входной сигнал; $\boldsymbol{\delta_0}$ – ошибка обучения на выходе.

Для весов соединений между скрытым слоем и входом

$$\boldsymbol{w_{ij}^* = w_{ij} + \alpha\,\delta_i f(U_i)}$$

где i – номер нейрона в скрытом слое; j – номер нейрона в входном слое; α – коэффициент обучения; $\boldsymbol{f(U_i)}$ – сигнал от ячейки скрытого слоя.

Для весов смещений

$$\boldsymbol{w_{i0}^* = w_{i0} + \alpha\,\delta_0\, f(U_0)}$$

где i – номер нейрона; 0 – cvtotybt; α – коэффициент обучения; $\boldsymbol{f(U_0)}$ – сигнал смещения.

Шаги 4, 5 составляют «обратный проход», здесь вычисляемый сигнал ошибки распространяется обратно по сети и используется для подстройки весов

Таким образом, продвижение вперед по сети рассчитывает активации ячеек и выход, продвижение назад - градиент (по отношению к данному примеру). Затем веса обновляются таким образом, чтобы минимизировать ошибку для данного входного сигнала. Коэффициент обучения минимизирует процент изменения, которое может произойти с весами. Хотя при небольшом коэффициенте процесс может занять больше времени, мы минимизируем возможность пропуска правильной комбинации весов. Если коэффициент обучения слишком велик, сеть может никогда не сойтись, то есть не будут найдены правильные леса связей.

Шаг 5. Повторять шаги с 2 по 4 для каждого вектора обучающего множества до тех пор, пока ошибка на всем множестве не достигнет приемлемого уровня.

Как видно из описания шагов 2 - 4, обучение сводится к решению задачи оптимизации функционала ошибки градиентным методом. Вся «соль» обратного распространения ошибки состоит в том, что для ее оценки для нейронов скрытых слоев можно принять взвешенную сумму ошибок последующего слоя.

О сходимости *необходимо сделать несколько дополнительных замечаний. Во-первых, практика показывает, что сходимость метода обратного распространения весьма медленная. Невысокий темп сходимости является «генетической болезнью» всех градиентных методов, так как локальное направление градиента отнюдь не совпадает с направлением к минимуму. Во-вторых, подстройка весов выполняется независимо для каждой пары образов обучающей выборки. При этом улучшение функционирования на некоторой заданной паре может, вообще говоря, приводить к ухудшению работы на предыдущих образах. В этом смысле, нет достоверных (кроме весьма обширной практики применения метода) гарантий сходимости.*

Рассмотрим пример функционирования сети в процессе обучения

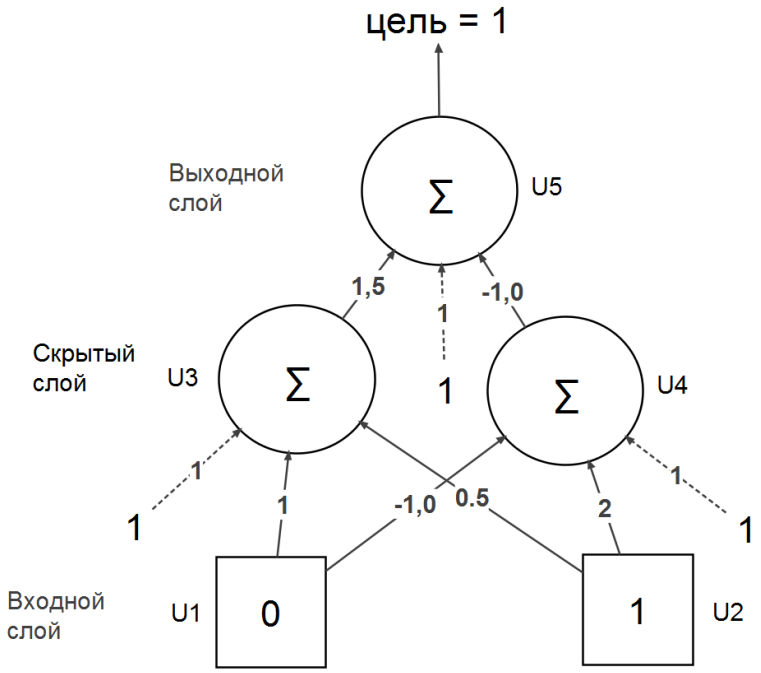

Рис.2. Архитектура сети

<u>Движение вперед</u>

Сначала выполняется расчет движения входного сигнала по сети. Рассмотрим значения для скрытого слоя (рис.2):

$U_3 = f(W_{31}U_1, + W_{32}U_2 + W_0X_0$ *(смещение)*)

U_3=f(l*0 + 0,5*l + l*l) = f(l,5)

f(x) – является активационной функцией, то есть функцией сигмоида

$$f(U_3) = {}^{1}/_{(1+e^{-NET})} = {}^{1}/_{(1+e^{-1,5})} = {}^{1}/_{(1+0,22313)} = 0,81757$$

$U_4 = f(W_{41}U_1, + W_{42}U_2 + W_0X_0)$

U_4=f(-l*0 + 2*l + l*l) = f(3)

Тогда $f(U_4) = {}^{1}/_{(1+e^{-3})} = {}^{1}/_{(1+0,04978)} = 0,95274$

Теперь сигнал дошел до скрытого слоя. Конечный шаг заключается в том, чтобы переместить сигнал из скрытого слоя в выходной слой и рассчитать значение на выходе из сети:

$U_5 = f(W_{51}U_1, + W_{52}U_2 + W_0X_0)$

U_5= f(l,5*0,81757 + (-1,0*0,952574) + l*l) = f(l,2195)

Тогда $f(U_5) = 0,78139$

Правильной реакцией нейронной сети на тестовый входной сигнал является **1,0** *(цель)*. Однако выходное значение, рассчитанное сетью, составляет **OUT = 0,78139**. Это не так уж и плохо, но можно уменьшить значение ошибки, применив для сети алгоритм обратного распространения ошибки.

Для коррекции весовых коэффициентов в сети используется суммарная квадратичная ошибка. Поэтому ошибка составляет:

$$E = 0,5\sum_{n}(d_j - y_j)^2 = 0,5(1-0,78139)^2 = 0,023895$$

<u>Алгоритм обратного распространения для ошибки</u>

Применим обратное распространение, начиная с определения ошибки в выходном и скрытых узлах.

Рассчитаем ошибку в выходном узле.

$\delta_5 = y_j(1-y_j)(d_j - y_j) = 0,78139\,(1-0,78139\,)(1-0,78139\,);$

δ_5 = **0,0373**

Теперь следует рассчитать ошибку для двух скрытых узлов.

$$\delta_{U3} = OUT(1-OUT)(\omega_{53}\delta_5) = f(U_3)\big(1-f(U_3)\big)(\omega_{53}\delta_5) =$$
$$= 0,81757(1-0,81757)(1,5*0,0373) = 0,0083449$$
$$\delta_{U4} = OUT(1-OUT)(\omega_{54}\delta_5) = f(U_4)\big(1-f(U_4)\big)(\omega_{54}\delta_5) =$$
$$= 0,95274(1-0,95274)(-1,0*0,0373) = -0,0016851$$

<u>Изменения весов соединений</u>

Когда рассчитаны значения ошибок для выходного и скрытого слоев, можно с помощью уравнений дельта-правил изменить веса. Используем коэффициент обучения (***α***), равный 0,5.

Сначала следует обновить веса для соединений между выходным и скрытым слоями:

$$w_{ij}^{*} = w_{ij} + \alpha \delta_0 f(U_i)$$

Рассматриваем связь между выходом U_5 и элементом скрытого слоя U_4.

Тогда $w_{54} = -1$, $\delta_5 = 0{,}0373$, $f(U_4) = 0{,}95274$

$$w_{54}^{*} = w_{54} + \alpha \delta_5 f(U_4) = -1 + (0{,}5 * 0{,}0373 * 0{,}95274\ ) = \mathbf{-0{,}9882}$$

Рассматриваем связь между выходом U_5 и элементом скрытого слоя U_3.

Тогда $w_{53} = 1{,}5$, $\delta_5 = 0{,}0373$, $f(U_3) = 0{,}81757$

$$w_{53}^{*} = w_{53} + \alpha \delta_5 f(U_3) = 1{,}5 + (0{,}5 * 0{,}0373 * 0{,}81757\ ) = \mathbf{1{,}51525}$$

Теперь нужно обновить смещение для выходной ячейки U_5:

$$w_{ij}^{*} = w_{ij} + \alpha \delta_0 f(U_0)$$

$$w_{50}^{*} = w_{50} + \alpha \delta_5 f(U_{50}) = 1 + (0{,}5 * 0{,}0373 * 1\ ) = \mathbf{1{,}01865}$$

Таким образом, для w_{54}^{*} и w_{53}^{*} вес увеличен. Смещение, было обновлено для повышения возбуждения.

Теперь нужно показать изменение весов для скрытого слоя (для входа к скрытым ячейкам)

$$w_{ij}^{*} = w_{ij} + \alpha \delta_i f(U_i)$$

Рассматриваем связь между входом U_2 и элементом скрытого слоя U_4.

Тогда $w_{42} = 2$, $\delta_{U4} = -0{,}0016851$, $f(U_2) = 1$

$$w_{42}^{*} = w_{42} + \alpha \delta_{U4} f(U_2) = 2 + (0{,}5 * -0{,}0016851 * 1) = \mathbf{1{,}99916}$$

Рассматриваем связь между входом U_1 и элементом скрытого слоя U_4.

Тогда $w_{41} = -1$, $\delta_{U4} = -0{,}0016851$, $f(U_1) = 0$

$$w_{41}^{*} = w_{41} + \alpha \delta_{U4} f(U_1) = -1 + (0{,}5 * -0{,}0016851 * 0) = \mathbf{-1{,}0}$$

Рассматриваем связь между входом U_2 и элементом скрытого слоя U_3.

Тогда $w_{32} = 0{,}5$, $\delta_{U3} = 0{,}0083449$, $f(U_2) = 1$

$$w_{32}^{*} = w_{32} + \alpha \delta_{U3} f(U_2) = 0{,}5 + (0{,}5 * 0{,}0083449 * 1) = \mathbf{0{,}50417}$$

Рассматриваем связь между входом U_1 и элементом скрытого слоя U_3.

Тогда $w_{31} = 1$, $\delta_{U3} = 0{,}0083449$, $f(U_1) = 0$

$$w_{31}^{*} = w_{31} + \alpha \delta_{U3} f(U_1) = 1 + (0{,}5 * 0{,}0083449 * 0) = \mathbf{1{,}0}$$

Последний шаг - это обновление смещений для ячеек:

$$w_{ij}^{*} = w_{ij} + \alpha \delta_0 f(U_0)$$

$$w_{40}^{*} = w_{40} + \alpha \delta_{U4} f(U_0) = 1 + (0{,}5 * -0{,}0016851 * 1\ ) = \mathbf{0{,}99915}$$

$$w_{30}^{*} = w_{30} + \alpha \delta_{U3} f(U_0) = 1 + (0{,}5 * 0{,}0083449 * 1\ ) = \mathbf{1{,}00417}$$

Обновление весов для выбранного обучающего сигнала завершается. Проверим, что алгоритм действительно уменьшает ошибку на выходе, введя тот

же обучающий сигнал еще раз:

$$U_3^* = f(w_{31}^* U_1 + w_{32}^* U_2 + w_{30}^* X_0)$$
$$= f(l, 0 * 0 + 0{,}50417 * l + 1{,}00417 * l) = f(l, 50834)$$
$$U_3^* = {}^{1}/_{(1+e^{-1,50834})} = \mathbf{0,8188}$$
$$U_4^* = f(w_{41}^* U_1 + w_{42}^* U_2 + w_{40}^* X_0)$$
$$= f(-l, 0 * 0 + 1{,}99916 * l + 0{,}99915 * l) = f(2{,}99831)$$
$$U_4^* = {}^{1}/_{(1+e^{-2,99831})} = \mathbf{0,952497}$$
$$U_5^* = f(w_{53}^* U_3^* + w_{54}^* U_4^* + w_{50}^* X_0)$$
$$= f(1{,}51525 * 0{,}8188 - 0{,}9882 * 0{,}952497 + 1{,}01865 * l)$$
$$= f(1{,}32379)$$
$$U_5^* = {}^{1}/_{(1+e^{-1,32379})} = \mathbf{0,7898}$$

Ошибка составляет:

$$E = 0{,}5 \sum_n (d_j - y_j)^2 = 0{,}5(1 - 0{,}7898)^2 = 0{,}022$$

Начальная ошибка была равна ***0,023895***. Текущая ошибка составляет ***0,022***, а это значит, что одна итерация алгоритма обратного распространения позволила уменьшить среднюю ошибку на ***0,001895***.