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)

5. ЛЕКЦИЯ. 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:

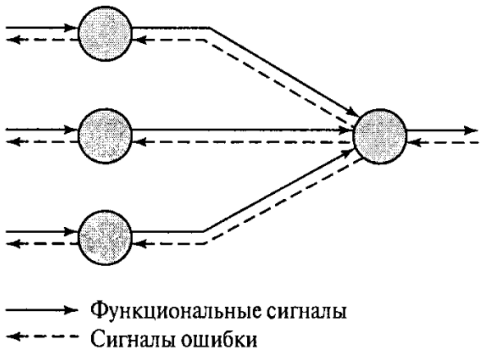

Рис. 4.2. Направление двух основных потоков сигнала для многослойного персептрона: прямое распространение функционального сигнала и обратное распространение сигнала ошибки.

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:

$$\boldsymbol{OUT' = OUT(1 - OUT)}$$

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 $\boldsymbol{d_j}$ – целевое значение входного сигнала; $\boldsymbol{y_j}$ – 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

$\boldsymbol{\delta_0 = y_j(1 - y_j)(d_j - y_j)}$ 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 $\boldsymbol{j}$**;**

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

Для весов соединений между скрытым слоем и входом

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

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

Для весов смещений

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

где i – номер нейрона; 0 – cvtotybt; α – коэффициент обучения; $f(U_0)$ – 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

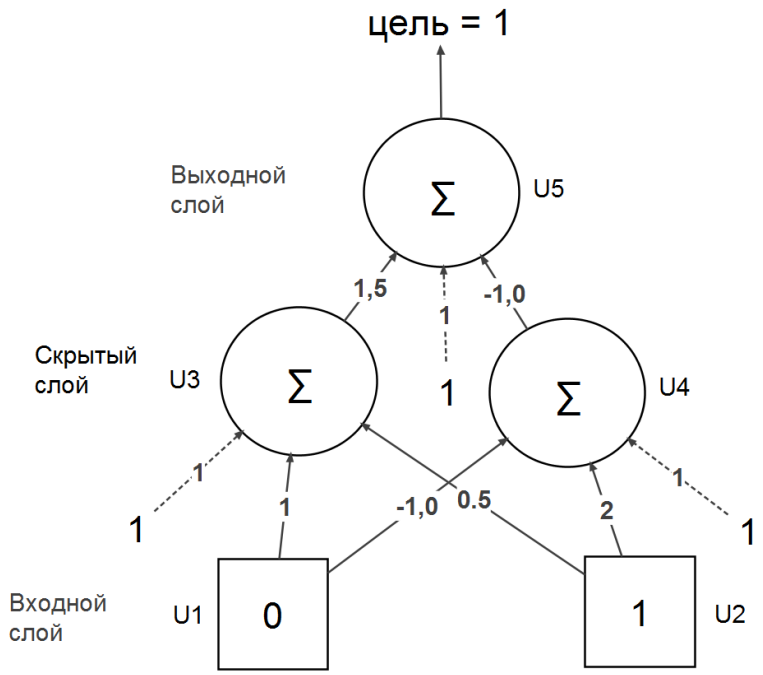

Рис.2. Архитектура сети

Движение вперед

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:

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

Алгоритм обратного распространения для ошибки

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

Рассчитаем ошибку в выходном узле.

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

$\delta_5 =$ **0,0373**

Теперь следует рассчитать ошибку для двух скрытых узлов.

$\delta_{U3} = OUT(1 - OUT)(\omega_{53}\delta_5) = f(U_3)\big(1 - f(U_3)\big)(\omega_{53}\delta_5) =$

$= 0{,}81757(1 - 0{,}81757)(1{,}5 * 0{,}0373) = 0{,}0083449$

$\delta_{U4} = OUT(1 - OUT)(\omega_{54}\delta_5) = f(U_4)\big(1 - f(U_4)\big)(\omega_{54}\delta_5) =$

$= 0{,}95274(1 - 0{,}95274)(-1{,}0 * 0{,}0373) = -0{,}0016851$

Изменения весов соединений

Когда рассчитаны значения ошибок для выходного и скрытого слоев, можно с помощью уравнений дельта-правил изменить веса. Используем коэффициент обучения ($\boldsymbol{\alpha}$), равный 0,5.

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

$\mathrm{w}_{\mathrm{ij}}^{*} = \mathrm{w}_{\mathrm{ij}} + \alpha\,\delta_0\,f(U_i)$

Рассматриваем связь между выходом $\boldsymbol{U_5}$ и элементом скрытого слоя $\boldsymbol{U_4}$.

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

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

Рассматриваем связь между выходом $\boldsymbol{U_5}$ и элементом скрытого слоя $\boldsymbol{U_3}$.

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

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

Теперь нужно обновить смещение для выходной ячейки $\boldsymbol{U_5}$:

$\mathrm{w}_{\mathrm{ij}}^{*} = \mathrm{w}_{\mathrm{ij}} + \alpha\,\delta_0\,f(U_0)$

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

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:

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

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***.

Оптимальное проектирование и обучение нейронных сетей

Теорема существования

Из предыдущего изложения следуют два важных вывода.

1. В жизни встречается множество практически важных задач, решить которые можно методом математического моделирования, т. е. путем построения некоторой сложной функции, осуществляющей преобразование вектора входных параметров ***X*** в вектор выходных параметров ***D***.

2. Универсальным инструментом построения такой функции являются нейросетевые технологии.

Естественно, возникают вопросы: всегда ли можно построить нейронную сеть, выполняющую преобразование, заданное любым множеством обучающих примеров, и каким требованиям должна удовлетворять эта нейронная сеть?

С одной из таких трудностей, названной «Проблемой исключающего ИЛИ», мы уже столкнулись при изучении и теперь знаем, что персептрон должен иметь скрытый слой нейронов, но осталось два вопроса.

1. Всегда ли можно спроектировать и обучить многослойный персептрон, обеспечивающий решение любой задачи?

2. Каким образом лучше задавать количество внутренних нейронных слоев и количество нейронов в них? Может быть, как в мозге,—10^{11} нейронов? Может, чем их будет больше, тем лучше?

Ответы на эти вопросы мы выясним, познакомившись с теоретической базой нейронных сетей. Важнейшее место в теории нейронных сетей занимает теорема *Арнольда—Колмогорова—Хехт-Нильсена*, доказательство которой достаточно сложно и поэтому в нашем курсе не рассматривается.

С физической точки зрения персептрон—это устройство, моделирующее человеческий мозг на структурном уровне. Однако, анализируя формулы, по которым он преобразует сигналы, можно заметить, что с математической точки зрения персептрон—это всего лишь аппроксиматор, заменяющий функцию многих аргументов суммой функций, каждая из которых зависит только от одного аргумента.

Аппроксимация – научный метод, состоящий в замене одних объектов другими, в каком-то смысле близкими к исходным, но более простыми. Аппроксимация позволяет исследовать числовые характеристики и качественные свойства объекта, сводя задачу к изучению более простых или более удобных объектов (например, таких, характеристики которых легко вычисляются или свойства которых уже известны)

Вопрос о том, всегда ли можно любую функцию многих аргументов представить в виде суммы функций меньшего количества аргументов, интересовал математиков на протяжении нескольких столетий.

В 1900 г. на Всемирном математическом конгрессе в Париже знаменитый немецкий математик Давид Гильберт сформулировал 23 проблемы, которые он предложил решать математикам начинающегося XX в. Одна из этих проблем (под номером 13) как раз и декларировала невозможность такого представления.

Таким образом, приговор новой области искусственного интеллекта был вынесен за полвека до ее появления. Получалось, что персептрон, сколько бы нейронов он ни имел, не всегда мог построить нужную математическую функцию.

Сомнения относительно возможностей персептронов развеяли советские математики—академики В. И. Арнольд и А. Н. Колмогоров. Им удалось доказать, что любая непрерывная функция *n* аргументов $f(x_1, x_2, \ldots, x_n)$ всегда может быть представлена в виде суммы непрерывных функций одного аргумента:

$$f_1(x_1) + f_2(x_2) + \ldots + f_n(x_n).$$

Тем самым гипотеза Гильберта была опровергнута, а нейринформатике был открыт «зеленый свет».

В 1987–1991 гг. профессор Калифорнийского университета (США) Р. Хехт-Нильсен переработал теорему Арнольда—Комогорова применительно к нейронным сетям. Он доказал, что *для любого множества различающихся между собой пар векторов* $\boldsymbol{X}q$ и $\boldsymbol{D}q$ *произвольной размерности существует двухслойный персептрон с сигмоидными активационными функциями и с конечным числом нейронов, который для каждого входного вектора* $\boldsymbol{X}q$ *формирует соответствующий ему выходной вектор* $\boldsymbol{D}q$.

Таким образом, была доказана принципиальная возможность построения нейронной сети, выполняющей преобразование, заданное *любым* множеством различающихся между собой обучающих примеров, и установлено, что такой универсальной нейронной сетью является двухслойный персептрон – персептрон с одним скрытым слоем, причем активационные функции его нейронов должны быть сигмоидными.

Теорема Арнольда—Колмогорова—Хехт-Нильсена имеет очень важное для практики следствие в виде формулы, с помощью которой можно определять необходимое количество синаптических весов нейронной сети:

$$\frac{N_y Q}{1 + log_2(Q)} \leq N_w \leq N_y \left(\frac{Q}{N_x} + 1\right)(N_x + N_y + 1) + N_y$$

где *Nx* —количество нейронов входного слоя; *Ny* —количество нейронов выходного слоя; *Q*—количество элементов множества обучающих примеров, т. е. количество пар входных и выходных векторов ***Xq*** и ***Dq***; *Nw* —необходимое число синаптических связей.

Оценив с помощью этой формулы необходимое число синаптических связей *Nw*, можно рассчитать и необходимое количество нейронов в скрытых слоях. Например, количество нейронов скрытого слоя двухслойного персептрона будет равно:

$$N = \frac{N_w}{N_x + N_y}$$

Последняя формула становится очевидной, если ее левую и правую части умножить на (*Nx* + *Ny*) и нарисовать схему двухслойного персептрона (т. е. персептрона с одним скрытым слоем).

Практические рекомендации по проектированию персептронов

Как следует из теорем Арнольда—Колмогорова—Хехт-Нильсена, для построения нейросетевой модели любого сколь угодно сложного объекта достаточно использовать персептрон с одним скрытым слоем сигмоидных нейронов. Однако в практических реализациях персептронов оптимальное количество как слоев, так и нейронов в каждом из них нередко отличается от теоретических. К тому же, иногда бывает целесообразно использовать персептроны с большим количеством скрытых слоев.

Строгой теории выбора оптимального количества скрытых слоев и нейронов в скрытых слоях пока не существует. На практике чаще всего используются персептроны, имеющие один или два скрытых слоя, причем количество нейронов в скрытых слоях обычно колеблется от *Nx*/2 до 3*Nx*.

Рассмотрим факторы, от которых зависит успешность обучения нейронной сети правильному решению задачи. В первую очередь, сеть должна быть достаточно гибкой, чтобы научиться правильно решать все примеры обучающей выборки. Поэтому в нейронной сети должно быть достаточное количество нейронов и связей.

На основании обучающей выборки достаточно сложно определить, сколько слоев и нейронов необходимо. Поэтому поступают обычно так. Обучают сеть со структурой, предлагаемой программой-нейроимитатором по умолчанию, а в дальнейшем, если сеть не может обучиться, пробуют обучить сеть большего или меньшего размера.

Свойство нейросети терять способность к обобщению при чрезмерном увеличении количества скрытых нейронов (степеней свободы) называют переобучением, или гиперразмерностью. Переобучение – явление, когда построенная модель хорошо объясняет примеры из обучающей выборки, но относительно плохо работает на примерах, не участвовавших в обучении (на примерах из тестовой выборки).

Однако, даже несмотря на все усилия по обучения НС, сеть все же может не обучаться давать много ошибок на тестовой выборке. Причины этого могут заключаться в следующем:

1) *противоречивость ОВ*, т.е. в обучающей выборке присутствуют задачи с одинаковыми условиями, но разными ответами (одинаковыми входными векторами данных, но разными выходными $A_i=A_j$, но $D_i \neq D_j$);

2) *нерепрезентативность ОВ*, т.е. обучающая выборка не охватывает всего множества ситуаций (выборка мала или просто узкоспециализирована);

3) *неравномерность ОВ*, т.е. в обучающей выборке может быть неодинаковое число примеров для разных классов. При этом при тестировании НС будет достаточно хорошо распознавать примеры класса, для которого в обучающей выборке было большинство примеров, и относить к этому же классу много примеров другого класса. Поэтому желательно, чтобы в обучающей выборке было примерно одинаковое число примеров для каждого класса, или, по крайней мере, не было отличия на порядок и более.

После обучения нейронной сети необходимо провести ее тестирование на *тестовой выборке* для определения точности решения не входивших в обучающую выборку задач. Точность правильного решения очень сильно зависит от репрезентативности обучающей выборки. Обычно при решении различных неформализованных задач в разных проблемных областях точность в 70-90% правильных ответов на тестовой выборке соответствует проценту правильных ответов при решении этих же задач специалистом-экспертом.

Нет строго определенной процедуры для выбора количества нейронов и количества слоев в сети. Чем больше количество нейронов и слоев, тем шире возможности сети, тем медленнее она обучается и работает и тем более нелинейной может быть зависимость вход-выход.

Количество нейронов и слоев связано:

1) со сложностью задачи;

2) с размерностью обучающей выборки;

3) с количеством данных для обучения;

4) с требуемым количеством входов и выходов сети;

5) с имеющимися ресурсами: памятью и быстродействием машины, на которой моделируется сеть;

Были попытки записать эмпирические формулы для числа слоев и нейронов, но применимость формул оказалась очень ограниченной.

Если в сети слишком мало нейронов или слоев:

1) сеть не обучится, и ошибка при работе сети останется большой (ошибка обобщения);

2) на выход сети не будут передаваться резкие колебания аппроксимируемой функции ***y(x)***.

Превышение требуемого количества нейронов тоже мешает работе сети.

Если нейронов или слоев слишком много:

1) быстродействие будет низким, а памяти потребуется много (на фон-неймановских ЭВМ);

2) сеть переобучится: выходной вектор будет передавать незначительные и несущественные детали в изучаемой зависимости *y(x)*, например, шум или ошибочные данные;

3) зависимость выхода от входа окажется резко нелинейной: выходной вектор будет существенно и непредсказуемо меняться при малом изменении входного вектора *x*;

4) сеть будет неспособна к обобщению: в области, где нет или мало известных точек функции *y(x)* выходной вектор будет случаен и непредсказуем, не будет адекватен решаемой задаче.