УДК 004.85

РОБАСТНАЯ НЕЙРОННАЯ СЕТЬ С ПРОСТОЙ АРХИТЕКТУРОЙ

© 2021 В. С. Тимофеев[a], М. А. Сивак[b]

Новосибирский государственный технический университет, просп. К. Маркса, 20, г Новосибирск 630073, Россия

E-mails: [a]v.timofeev@corp.nstu.ru, [b]pepelyaeva@ami.nstu.ru

Поступила в редакцию 17.11.2020 г.; после доработки 04.10.2021 г.; принята к публикации 21.10.2021 г.

Рассматривается задача классификации и применение нейронных сетей простой архитектуры для её решения. Предлагается модификация алгоритма обратного распространения ошибки, который используется для обучения нейронной сети. Доказано утверждение, которое позволяет построить предложенную модификацию, используя робастную функцию потерь Хьюбера. С целью исследования свойств полученной нейронной сети был проведён ряд вычислительных экспериментов при различном количестве засоряющих наблюдений в выборке, уровне шума, различных объёмах обучающей и тестовой выборок и значениях параметра функции Хьюбера. Анализ результатов показал, что предложенная модификация способна значительно увеличить точность классификации при работе с зашумлёнными данными, а также скорость обучения нейронной сети.

Ключевые слова: задача классификации, нейронные сети, функция потерь Хьюбера, обратное распространение ошибки.

DOI: 10.33048/SIBJIM.2021.24.409

ВВЕДЕНИЕ

В настоящее время искусственные нейронные сети (ИНС) являются одним из наиболее популярных инструментов машинного обучения. Они применяются для решения широкого круга задач, например для классификации объектов, прогнозирования или управления. Обучение нейронных сетей происходит за счёт корректировки весов. Классическим алгоритмом обучения является алгоритм обратного распространения ошибки, который сводится к решению задачи нелинейной оптимизации. Как правило, в данном алгоритме используется квадратичная функция потерь. Из-за этого нейронные сети с простой архитектурой очень часто неудовлетворительно работают на реальных данных, для которых характерно частичное смешивание наблюдений, принадлежащих различным классам [1–3].

Первым вариантом решения этой проблемы является предобработка исследуемых данных и исключение нетипичных наблюдений (выбросов), что приводит к некоторой идеализации наблюдений и негативным образом сказывается на качестве работы сети, когда в дальнейшем на вход подаются реальные данные.

Второй вариант – усложнить архитектуру используемой сети, например использовать сверточную или рекурретную ИНС. Такой подход неизбежно повлечёт за собой увеличение затрат на вычислительные ресурсы.

Однако есть ещё третий вариант — модифицировать алгоритм обучения нейронной сети, взяв за основу робастный подход [4]. Введение робастной функции потерь позволит не

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проект 20-37-90077).

исключать выбросы из данных, а снизить их влияние при обучении. Подобный подход, хоть и является с точки зрения авторов весьма перспективным, используется не очень часто. Существующие работы демонстрируют возможность его применения только в отдельных случаях. Так, в [5] рассматривается применение функции потерь Хьюбера для реализации робастного алгоритма обучения с подкреплением. В [6] приводятся примеры различных функций потерь и рассматривается их использование в алгоритмах обучения без учителя; также выполняется построение адаптивной функции потерь и сравнение её с уже существующими. В [7] авторы предлагают модификацию алгоритма Левенберга — Марквардта с использованием функции потерь Хьюбера. Но при этом рассматривается только нейронная сеть с одним скрытым слоем и её применение для решения конкретной практической задачи — прогнозирования цен европейских опционов.

Однако классическим алгоритмом обучения нейронных сетей считается алгоритм обратного распространения ошибки, поэтому основная идея данной статьи состоит в построении его нового варианта с использованием робастных функций потерь. Принципиальное отличие данного алгоритма от алгоритма Левенберга — Марквардта заключается в подходе к корректировке весов: в алгоритме обратного распространения ошибки пересчёт весов сети происходит на каждой итерации, в то время как в алгоритме Левенберга — Марквардта корректировка весов выполняется уже после предъявления сети всех данных, используемых для обучения. Предложенная модификация излагается в наиболее общем виде и может применяться в случае нейронной сети с произвольным числом скрытых слоёв и с различными функциями потерь. В работе приводятся результаты исследований для функции потерь Хьюбера.

1. ПОСТАНОВКА ЗАДАЧИ

Одним из наиболее распространённых вариантов применения нейронных сетей является решение задачи классификации. Пусть имеется конечное множество классов $Q = \{q_1, \ldots, q_{|Q|}\}$, где q_k — непересекающиеся между собой классы, а также конечное множество объектов $X = \{X_1, \ldots, X_{|X|}\}$. При этом каждый объект $X_m, m = 1, \ldots, |X|$, можно описать вектором $x_m = \{x_{m1}, x_{m2}, \ldots, x_{mY}\}$, состоящим из значений признаков этого объекта x_{mi}, $i = 1, \ldots, Y$, где Y — количество признаков. Классифицировать объект — значит указать для каждого объекта X_m класс q_k, к которому этот объект относится.

При решении задачи классификации, как правило, всё множество объектов X разделяется на обучающую и тестовую подвыборки, которые представляют собой два непересекающихся подмножества [8]. Для этих подвыборок классы объектов известны.

Обучающая выборка $L = \{X_1, \ldots, X_{|L|}\}$ представляет собой набор объектов, с помощью которого происходит обучение нейронной сети. Тестовая выборка $D = \{X_{|L|+1}, \ldots, X_{|X|}\}$ представляет собой набор объектов, на котором производится оценка эффективности нейронной сети. Каждый объект X_m из тестовой выборки подаётся на вход обученной сети, а затем сравнивается результат её работы с известным значением класса q_k для этого объекта. Считается, что нейронная сеть тем эффективнее, чем чаще эти значения совпадают.

Рассмотрим пример задачи классификации, в которой каждый объект X_m имеет четыре признака ($Y = 4$), а множество Q включает в себя три класса. Для решения этой задачи предлагается использовать нейронную сеть с простой архитектурой, представленную на рис. 1. Данная сеть состоит из трёх слоёв ($N = 3$). Входной слой сети, на который подаются признаки объекта X_m, включает в себя четыре нейрона, выходной — три (по одному для каждого класса q_k). Скрытый слой рассматриваемой сети состоит из четырёх нейронов. В качестве функции активации $\varphi = \varphi(z)$ желательно использовать монотонную и непрерывно дифференцируемую функцию. В данной работе была использована сигмоида [9]:

$$\varphi(z) = \frac{1}{1 + e^z}. \tag{1}$$

На рис. 1 через y_1, y_2, y_3 обозначены значения нейронов на выходном слое сети. Номер нейрона k, имеющего наибольшее значение, будет соответствовать номеру класса q_k для объекта, поданного нейронной сети на вход. Веса на рёбрах между нейронами первого и второго слоёв обозначены $w_{ij}^{(1)}$, $i, j = 1, 2, 3, 4$; веса на рёбрах между нейронами второго и третьего слоёв — $w_{jk}^{(2)}$, $j = 1, 2, 3, 4$, $k = 1, 2, 3$. Входные значения нейронов на втором слое обозначаются как $s_j^{(2)}$, $j = 1, 2, 3, 4$; входные значения на третьем слое — $s_k^{(3)}$, $k = 1, 2, 3$; выходные значения нейронов на первом слое — $o_i^{(1)}$, $i = 1, 2, 3, 4$, на втором слое — $o_j^{(2)}$, $j = 1, 2, 3, 4$.

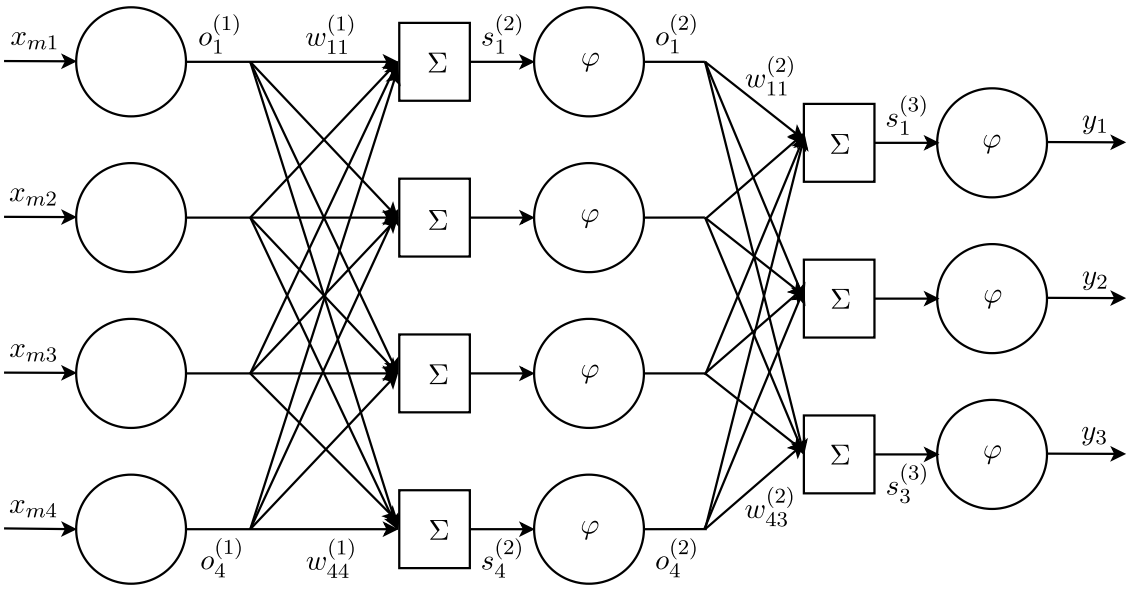

Рис. 1. Искусственная нейронная сеть с одним скрытым слоем

Данные обозначения очевидным образом обобщаются для более сложных нейронных сетей: y_k, $k = 1, \ldots, |T|$, — значения на выходном слое нейронной сети; $w_{ij}^{(n-1)}$, $i = 1, \ldots, l^{(n-1)}$, $j = 1, \ldots, l^{(n)}$, — вес между j-м нейроном слоя n и i-м нейроном слоя $n-1$ ($l^{(n)}$ — количество нейронов на слое n); $s_j^{(n)}$ — входное значение j-го нейрона на слое n, определяемое на основе $o_i^{(n-1)}$ — выходных значений нейронов на слое $n-1$: $s_j^{(n)} = \sum\limits_{i=1}^{l^{(n-1)}} w_{ij}^{(n-1)} o_i^{(n-1)}$, $n = 2, 3, \ldots, N$. Если же нейрон находится на входном слое, то для него будет выполняться $s_i^{(1)} = o_i^{(1)}$. При обучении нейронной сети каждый объект X_m из обучающей выборки L подаётся на вход сети. Поэтому будет справедливо следующее равенство: $s_i^{(1)} = o_i^{(1)} = x_{mi}$, $m = 1, \ldots, |L|$. При оценке точности работы сети аналогичным образом на вход сети подаются объекты тестовой выборки D:

$$s_i^{(1)} = o_i^{(1)} = x_{mi}, \quad m = |L+1|, \ldots, |X|.$$

Для каждого нейрона j на слое n, $n > 1$, его выходное значение $o_j^{(n)}$ определяется в соответствии с выражением

$$o_j^{(n)} = \varphi\big(s_j^{(n)}\big). \tag{2}$$

Тогда выходные значения нейронной сети будут вычисляться как

$$y_k = \varphi\big(s_k^{(N)}\big). \tag{3}$$

Таким образом, значения на выходном слое сети зависят от всех весовых коэффициентов.

2. РОБАСТНАЯ МОДИФИКАЦИЯ АЛГОРИТМА ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ

Согласно [9] существует два варианта корректировки весовых коэффициентов нейронной сети в рамках одной эпохи: после предъявления каждого объекта из обучающей выборки и после предъявления всех объектов обучающей выборки. В данной работе рассматривается первый вариант, в связи с чем суммарная функция потерь E может быть представлена как сумма значений функций потерь $f(t_j, y_j)$ на каждом выходе нейронной сети. Поэтому обучение сети сводится к решению следующей задачи оптимизации:

$$E = \sum_{j=1}^{l^{(N)}} f(t_j, y_j) \to \min_{w_{ij}^{(1)},\ldots,w_{ij}^{(N-1)}}, \tag{4}$$

где t_j — требуемый ответ на j-м выходе сети, определяемый следующим образом:

$$t_j = \begin{cases} 1, & X_m \in q_j, \\ 0 & \text{иначе.} \end{cases}$$

Как правило, в качестве функции потерь используется квадратичная функция

$$f(t_j, y_j) = \frac{1}{2}(y_j - t_j)^2. \tag{5}$$

Для минимизации суммарной функции потерь необходимо вычислить её производную по весам нейронной сети. Исходя из (2) и (3), частная производная (4) вычисляется по следующему цепному правилу [9]:

$$E'^{(n)}_{ji} = \frac{\partial E}{\partial w_{ij}^{(n-1)}} = \frac{\partial E}{\partial o_j^{(n)}} \frac{\partial o_j^{(n)}}{\partial \mathrm{s}_j^{(n)}} \frac{\partial \mathrm{s}_j^{(n)}}{\partial w_{ij}^{(n-1)}}. \tag{6}$$

Во входном значении нейрона $s_j^{(n)}$ от $w_{ij}^{(n-1)}$ зависит только одно слагаемое, таким образом:

$$\frac{\partial s_j^{(n)}}{\partial w_{ij}^{(n-1)}} = \frac{\partial}{\partial w_{ij}^{(n-1)}} \left(\sum_{i=1}^{l^{(n-1)}} w_{ij}^{(n-1)} o_i^{(n-1)} \right) = o_i^{(n-1)}. \tag{7}$$

Производная выходного значения нейрона $o_j^{(n)}$ по его входному значению $s_j^{(n)}$ — это производная функции активации (1):

$$\frac{\partial o_j^{(n)}}{\partial \mathrm{s}_j^{(n)}} = \frac{d\varphi(\mathrm{s}_j^{(n)})}{d\mathrm{s}_j^{(n)}}. \tag{8}$$

Если нейрон находится в выходном слое, то $n = N$ и первый множитель в (6) можно легко вычислить, поскольку $o_j^{(N)} = y_j$:

$$\frac{\partial E}{\partial o_j^{(N)}} = \frac{\partial E}{\partial y_j} = \frac{\partial f(y_j, t_j)}{\partial y_j}. \tag{9}$$

Теперь получим выражение для производной E по $o_j^{(n)}$ в случае, когда n — произвольный внутренний слой сети. Для этого рассмотрим E как функцию от входных значений нейронов следующего слоя: $E = E\big(s_k^{(n+1)}\big)$, $k = 1, \ldots, l^{(n+1)}$, а затем возьмём производную по $o_j^{(n)}$ [9]:

$$\frac{\partial E}{\partial o_j^{(n)}} = \sum_{k=1}^{l^{(n+1)}} \left(\frac{\partial E}{\partial s_k^{(n+1)}} \frac{\partial s_k^{(n+1)}}{\partial o_j^{(n)}} \right) = \sum_{k=1}^{l^{(n+1)}} \left(\frac{\partial E}{\partial o_k^{(n+1)}} \frac{\partial o_k^{(n+1)}}{\partial s_k^{(n+1)}} w_{jk}^{(n)} \right). \tag{10}$$

Производную (10) можно вычислить, если известны все производные по выходным значениям для следующего слоя.

Таким образом, производная суммарной функции потерь будет вычисляться по формуле

$$E'^{(n)}_{ji} = \delta^{(n)}_j o^{(n-1)}_i, \tag{11}$$

где $\delta^{(n)}_j$, исходя из (6)–(10), вычисляется следующим образом:

$$\delta^{(n)}_j = \frac{\partial E}{\partial o^{(n)}_j}\frac{\partial o^{(n)}_j}{\partial s^{(n)}_j} = \begin{cases} \dfrac{\partial f(y_j,t_j)}{\partial y_j}\varphi'(y_j), & n = N, \\ \left(\sum\limits_{k=1}^{l^{(n+1)}} w^{(n)}_{jk}\delta^{(n+1)}_k\right)\varphi'(s^{(n)}_j) & \text{иначе.} \end{cases} \tag{12}$$

При этом полученное соотношение справедливо для произвольной функции активации $\varphi = \varphi(z)$, а не только для сигмоиды, использованной в данной работе. Поскольку соотношение для вычисления производной известно, для решения задачи оптимизации можно воспользоваться методом градиентного спуска [10].

Как отмечалось ранее, алгоритм обратного распространения ошибки является неустойчивым к выбросам. В связи с этим авторы предлагают его модификацию, которая заключается в использовании робастной функции потерь $f_R(y_j, t_j)$ вместо квадратичной функции (5): $f(y_j, t_j) = f_R(y_j, t_j)$, где функция $f_R(y_j, t_j)$ должна быть непрерывно дифференцируемой. Этому ограничению удовлетворяют, например, функции потерь Уэлша, Рамсея, Коши, рассмотренные в [6], а также функция потерь Хьюбера [11], используемая для построения робастной нейронной сети в рамках данной работы:

$$f_R(y_j,t_j) = \begin{cases} \dfrac{1}{2}(y_j - t_j)^2, & |y_j - t_j| \leqslant \beta, \\ \beta|y_j - t_j| - \dfrac{1}{2}\beta^2, & |y_j - t_j| > \beta, \end{cases} \tag{13}$$

где $\beta > 0$ — параметр функции. Частная производная (13) по выходу j-го нейрона на выходном слое будет вычисляться следующим образом:

$$\frac{\partial f_R(y_j,t_j)}{\partial y_j} = \begin{cases} y_j - t_j, & |y_j - t_j| \leqslant \beta, \\ -\beta, & y_j - t_j < -\beta, \\ \beta, & y_j - t_j > \beta. \end{cases}$$

Из приведённых соотношений видно, что поведение функции потерь Хьюбера на отрезке $[-\beta, \beta]$ совпадает с квадратичной, а на интервалах $(-\infty, -\beta)$ и $(\beta, +\infty)$ оно имеет линейный характер, что обеспечит снижение влияния выбросов.

Утверждение. *Использование робастной функции потерь* (13) *вместо квадратичной функции потерь* (5) *в алгоритме обратного распространения ошибки приведёт к изменению только тождества* (12).

Доказательство. Подставим в (4) вместо квадратичной функции потерь соотношение (13). В этом случае задача оптимизации будет следующей:

$$E = \sum_{j=1}^{l^{(N)}} f_R(t_j, y_j) \to \min_{w^{(1)}_{ij},\ldots,w^{(N-1)}_{ij}}.$$

Замена квадратичной функции потерь на робастную не влияет на структуру нейронной сети, поэтому, повторяя приведённую выше логику рассуждений, заметим, что соотношения

(6)–(8) не изменятся, так как результат их вычисления от функции потерь не зависит. При этом для нейрона выходного слоя первый множитель в (6) теперь примет вид

$$\frac{\partial E}{\partial o_j^{(N)}} = \frac{\partial E}{\partial y_j} = \frac{\partial f_R(y_j, t_j)}{\partial y_j}. \tag{14}$$

Цепное правило для нейрона на произвольном внутреннем слое сети (10) также зависит только от структуры сети и поэтому не изменится. Следовательно, останется прежним и соотношение для вычисления производной суммарной функции потерь (11). Однако в силу (14) множитель $\delta_j^{(n)}$ в (11) теперь примет вид

$$\delta_j^{(n)} = \frac{\partial E}{\partial o_j^{(n)}} \frac{\partial o_j^{(n)}}{\partial s_j^{(n)}} = \begin{cases} \dfrac{\partial f_R(y_j, t_j)}{\partial y_j} \varphi'(y_j), & n = N, \\ \left(\sum\limits_{k=1}^{l^{(n+1)}} w_{jk}^{(n)} \delta_k^{(n+1)} \right) \varphi'(s_j^{(n)}) & \text{иначе}, \end{cases} \tag{15}$$

что согласуется с утверждением. Таким образом, утверждение доказано. □

Доказанное утверждение позволяет построить модификацию алгоритма обратного распространения ошибки путём замены (12) на (15), сохраняя остальные тождества алгоритма. При этом будет получена совершенно новая нейронная сеть. Для исследования свойств этой сети был проведён ряд вычислительных экспериментов.

3. РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЙ

Для проведения исследований был использован один из популярных наборов данных для классификации — ирисы Фишера [12]. Этот набор часто используется именно для иллюстрации работы различных алгоритмов классификации, поскольку является достаточно компактным и позволяет построить классификатор при минимуме признаков объекта. Набор состоит из 150 объектов, которые представляют собой экземпляры ирисов трёх видов (iris setosa, iris versicolor, iris virginica). Каждый ирис X_m, $m = 1, \ldots, 150$, характеризуется четырьмя признаками x_{mi} (длина и ширина чашелистика, длина и ширина лепестка) и классом y_k, $k = 1, 2, 3$. Столь малое количество признаков позволяет наглядно отслеживать существующие закономерности и работу алгоритма.

При выполнении исследований проводилось зашумление значений третьего и четвёртого признаков x_{m3}, x_{m4}: $\tilde{x}_{mi} = x_{mi} + \varepsilon_{mi}$, $i = 3, 4$, где ε_{mi} — случайные ошибки, которые моделировались независимыми и одинаково распределёнными. Функция распределения ε_{mi} следующая:

$$F_i(x) = (1 - \lambda) F_1(x, 0, \sigma_{i1}) + \lambda F_2(x, 0, \sigma_{i2}), \quad i = 3, 4,$$

где $F_j(x, 0, \sigma_{ij})$, $j = 1, 2$, — функция нормального распределения с нулевым математическим ожиданием и дисперсией σ_{ij}^2, $\lambda \in [0, 1]$ — параметр смеси, который здесь играет роль доли засоряющих наблюдений. В проведённых экспериментах полагалось, что $\sigma_{i1}^2 < \sigma_{i2}^2$. При этом в ходе моделирования задавались не сами значения дисперсий σ_{i1}^2 и σ_{i2}^2, а соответствующие им значения уровня шума ρ_{i1} и ρ_{i2}. Значения ρ_{i1} соответствовали уровню фонового шума для каждого признака, а ρ_{i2} — уровню шума выбросов, т. е. их удалённости от общей группы наблюдений. Уровень шума, введённый в [13], определяется следующим образом:

$$\rho_{ij} = \frac{\sigma_{ij}}{c} \cdot 100\%,$$

где c^2 — дисперсия незашумлённой выборки.

Используемые в работе данные являются результатом реальных измерений, поэтому авторами было выдвинуто предположение, что в них уже содержатся погрешности (шум). Поскольку основной интерес состоял в изучении влияния на работу робастной сети именно выбросов, в ходе исследований варьировались значения уровня шума ρ_{32} и ρ_{42}. Значения уровня фонового шума во всех экспериментах было установлено на минимальном уровне: $\rho_{31} = \rho_{41} = 0{,}05\%$.

Для оценки точности работы нейронных сетей использовалось значение доли объектов, которые сеть отнесла к правильному классу: $\alpha = \dfrac{D_{\text{corr}}}{|D|} \cdot 100\%$, где D_{corr} — количество объектов, отнесённых к правильному классу, а $|D|$ — объём тестовой выборки.

Алгоритм обучения нейронной сети итерационный: за одну эпоху в процессе обучения последовательно предъявляются все объекты из обучающей выборки. В данной работе весовые коэффициенты корректируются после предъявления каждого объекта, а точность классификации на тестовой выборке оценивается после предъявления всех объектов, т. е. по окончании эпохи. Поскольку в ходе исследования использовались зашумлённые данные, на всех этапах для каждого теста проводилось по 200 вычислительных экспериментов, результаты которых затем усреднялись. На первом этапе исследований был проведён сравнительный анализ точности работы обычной и робастной нейронных сетей при разной доле засоряющих наблюдений.

Для робастной сети рассматривались значения параметра β на интервале (0, 1). Поскольку t_j принимает только значения 0 и 1, а y_j, в силу используемой функции активации, может принимать значения только на полуинтервале (0, 1], то $|y_j - t_j| \leqslant 1$. При $\beta \geqslant 1$ функция потерь Хьюбера на отрезке [-1, 1] будет совпадать с квадратичной — это фактически означает, что такие значения параметра рассматривать нецелесообразно.

Результаты исследования точности классификации для робастной и обычной нейронных сетей, полученные на этом этапе, представлены в табл. 1. Доля засоряющих наблюдений λ варьировалась от $0,05$ до $0,40$ с шагом в $0,05$. Значения уровня шума ρ_{i2}, которым соответствовали дисперсии ошибок σ_{i2}^2, изменялись в пределах от 48 до 149% и от 67 до 162% для третьего и четвёртого признаков соответственно. Приводятся значения параметра функции Хьюбера, при которых была достигнута наилучшая точность классификации $\alpha_{\max}$, число эпох, по прошествии которого она была достигнута, а также точность работы сети по прошествии 50-ти эпох обучения α_{50}.

Таблица 1

Сравнение точности классификации для робастной и обычной сетей при различной доле выбросов

ρ_{32}	ρ_{42}	Доля засоряющих наблюдений	Робастная НС				Обычная НС		
			$\alpha_{\max}, \%$	β	Число эпох	$\alpha_{50}, \%$	$\alpha_{\max}, \%$	Число эпох	$\alpha_{50}, \%$
48	67	0,05	100,00	0,5	400	72,50	98,95	400	71,1
74	70	0,10	97,31	0,8	500	41,61	97,18	600	39,13
91	104	0,15	94,00	0,3	1000	64,49	93,20	1000	34,50
102	120	0,20	89,10	0,2	600	66,55	87,99	1000	61,59
117	138	0,25	86,77	0,2	1000	51,28	76,43	1000	66,25
138	147	0,30	89,73	0,5	1000	66,55	84,80	1000	53,11
147	153	0,35	85,88	0,4	1000	66,67	79,38	1000	45,04
149	162	0,40	92,71	0,4	900	66,67	84,73	1000	65,78

Из таблицы видно, что точность классификации при использовании робастной сети во всех случаях либо выше, чем при использовании обычной, либо сопоставима с ней. В половине случаев робастная сеть обучилась быстрее обычной. Такой вывод можно сделать, во-первых,

из того, что максимальная точность классификации была достигнута за меньшее количество эпох обучения, а во-вторых, что точность обучения по прошествии 50-ти эпох была выше. Кроме того, нетрудно заметить, что при некоторых значениях доли выбросов робастная сеть даёт выигрыш в точности от 6 до 10%. Рассмотрим результат работы двух сетей при доле засоряющих наблюдений 40%, представленный на рис. 2.

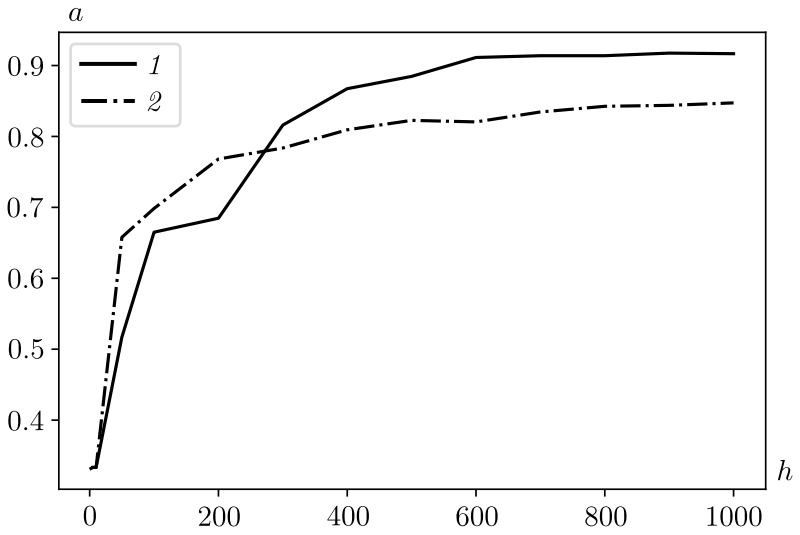

Рис. 2. Сравнение точности работы робастной и обычной нейронных сетей при доле выбросов 40%; h — число эпох, a — точность работы НС; *1* — робастная НС, *2* — обычная НС

Рассматриваемые нейронные сети обучались в течение различного количества эпох (от 2 до 1000), причём значения точности фиксировались на отметках в 2, 5, 10, 50, 100 эпох и далее с шагом в 100. Из рис. 2. видно, что на малом количестве эпох обучения (до 200) точность работы робастной сети сопоставима с точностью работы обычной. Однако с увеличением количества эпох робастная сеть начинает работать точнее, из чего можно сделать вывод, что она обучается быстрее обычной. В конечном итоге точность классификации превышает 90%, что является достаточно хорошим результатом при столь сильном зашумлении данных.

На следующих двух этапах было зафиксировано значение параметра $\beta = 0{,}2$, так как именно для него был получен наибольший выигрыш в точности, доля засоряющих наблюдений составляла 0,25. На втором этапе исследования проводились для различных значений уровня шума ρ_{32} (значение варьировалось в пределах от 28 до 278%) и ρ_{42} (значение варьировалось в пределах от 24 до 236%), для третьего и четвёртого признаков соответственно. Такие значения уровня шума являются экстремально высокими и вряд ли могут быть практически реализованными, поэтому они рассматривались авторами лишь в исследовательских целях. Полученные результаты представлены в табл. 2.

Нетрудно заметить, что при меньших значениях уровня шума робастная сеть, хоть и не всегда обучается быстрее обычной, даёт значительный выигрыш в точности: в общем не менее 9 и до $24,52\%$ при значениях $\rho_{32} = 85\%$, $\rho_{42} = 63\%$. При более высоких значениях уровня шума робастная сеть начинает работать несколько хуже обычной: разница в точности составляет от 2,37 до 6,42%. Это связано с тем, что при увеличении дисперсии выбросов наблюдения из разных классов становятся менее смешанными, что фактически эквивалентно меньшей степени зашумлённости данных.

На последнем этапе исследований изучалось влияние объёмов обучающей и тестовой выборок на точность классификации. Рассматривались три варианта разделения исходной выборки на обучающую и тестовую подвыборки:

— вариант 1: $|L| = 105$ наблюдений (70%), $|D| = 45$ наблюдений (30%);
— вариант 2: $|L| = 120$ наблюдений (80%), $|D| = 30$ наблюдений (20%);
— вариант 3: $|L| = 135$ наблюдений (90%), $|D| = 15$ наблюдений (10%).

Таблица 2

Сравнение точности классификации робастной и обычной сетей при различном уровне шума, доле выбросов 25% и параметре $\beta = 0{,}2$

ρ_{32}	ρ_{42}	Робастная НС			Обычная НС		
		$\alpha_{\max}, \%$	Число эпох	$\alpha_{50}, \%$	$\alpha_{\max}, \%$	Число эпох	$\alpha_{50}, \%$
28	24	97,38	1000	61,62	84,38	1000	67,73
59	45	91,65	500	58,02	82,40	1000	53,70
85	63	89,97	1000	56,07	65,45	1000	51,22
111	98	88,07	1000	54,40	76,30	1000	62,02
131	127	86,28	1000	49,92	77,45	900	65,35
145	181	89,35	1000	48,45	89,05	900	66,95
175	157	90,03	1000	46,78	92,40	900	67,40
190	213	89,62	1000	50,02	95,73	1000	67,65
211	228	90,73	1000	43,65	97,15	900	67,35
278	236	94,02	1000	45,58	97,65	900	67,35

Значение параметра β робастной нейронной сети полагалось равным $0,2$, доля засоряющих наблюдений $0,25$. Значение уровня шума ρ_{32} изменялось в пределах от 109 до 122%, а ρ_{42} — от 129 до 330%. Появление столь экстремальных значений уровня шума связано с увеличением объёма обучающей выборки, что приводит к увеличению фактически реализованного уровня шума. Как и ранее, такие значения рассматривались в исследовательских целях. Результаты исследований для данного этапа представлены в табл. 3.

Таблица 3

Сравнение точности классификации робастной и обычной сетей для различных вариантов разбиения выборки при доле выбросов 25% и параметре $\beta = 0{,}2$

Вариант разбиения	Робастная НС			Обычная НС		
	$\alpha_{\max}, \%$	Число эпох	$\alpha_{50}, \%$	$\alpha_{\max}, \%$	Число эпох	$\alpha_{50}, \%$
Вариант 1	96,31	800	61,50	99,96	1000	66,72
Вариант 2	88,03	1000	59,75	78,17	1000	56,92
Вариант 3	95,17	400	48,33	67,87	1000	34,60

Можно заметить, что на втором и третьем вариантах разбиения робастная сеть даёт прирост в точности классификации в 10,3 и в 27,3% соответственно. Помимо этого видно, что на третьем варианте разбиения обычная сеть начинает работать гораздо хуже, чем на остальных. Это можно объяснить тем, что с увеличением объёма обучающей выборки также увеличивается количество выбросов в ней (в соответствии с рассматриваемой долей выбросов). Поэтому точность классификации для обычной сети снижается, зато робастная сеть обучается быстрее за счёт большего количества наблюдений, что в конечном итоге позволяет получить более высокую точность классификации.

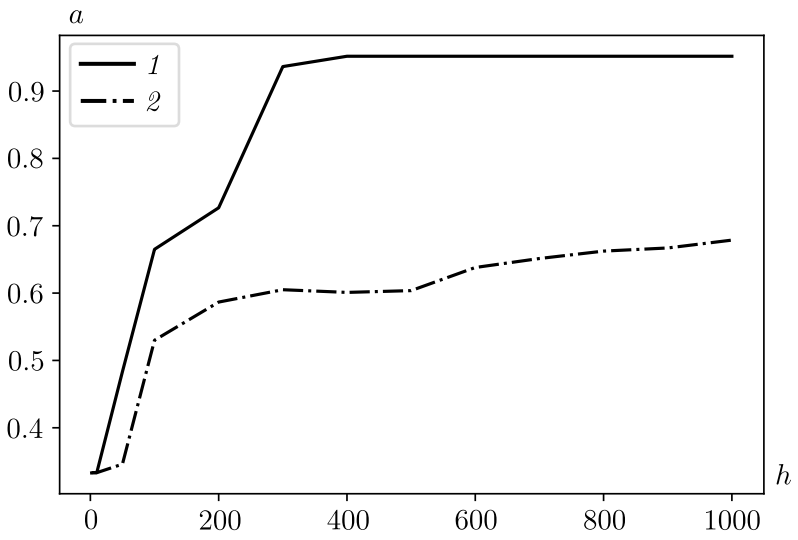

Рис. 3. Сравнение точности работы робастной и обычной нейронных сетей для третьего варианта разбиения выборки;
h — число эпох, a — точность работы НС; *1* — робастная НС, *2* — обычная НС

На рис. 3 представлен результат работы двух нейронных сетей для третьего варианта разбиения выборки, на котором был получен наибольший выигрыш в точности. Число эпох варьировалось так же, как ранее. Видно, что даже при малом числе эпох робастная сеть обучается значительно быстрее обычной и уже за 400 эпох достигается максимальная точность классификации (95,17%). В то же время даже по прошествии 1000 эпох точность классификации обычной нейронной сети остаётся гораздо ниже, чем у робастной сети (67,87%).

Полученные результаты ещё раз подтверждают снижение влияния выбросов при использовании робастной функции потерь. Чем больше нетипичных наблюдений в выборке и/или чем дальше эти наблюдения расположены, тем качество работы робастной нейронной сети становится выше по сравнению с классической нейронной сетью. Особую роль играет значение параметра β, определяющее, насколько сильно выбросы будут оказывать негативное влияние на результат обучения и, как следствие, на точность работы сети.

ЗАКЛЮЧЕНИЕ

В ходе работы была предложена модификация алгоритма обратного распространения ошибки, заключающаяся в замене функции потерь на робастную функцию Хьюбера. Было доказано утверждение, позволяющее построить предложенную модификацию. В результате была получена нейронная сеть, обладающая новыми свойствами, которые были обнаружены и исследованы посредством ряда вычислительных экспериментов.

Изменение параметра функции Хьюбера β позволило в среднем увеличить точность классификации более чем на $8,6\%$, а в одном случае — на $27,3\%$. Результаты исследований показали, что робастная нейронная сеть является более эффективной при работе именно с зашумлёнными данными, тогда как на незашумлённых данных обычная сеть работает лучше. Кроме того, робастная сеть обучается гораздо быстрее обычной: наибольший выигрыш в скорости обучения составил 2,5 раза. При увеличении объёма обучающей выборки скорость обучения и точность работы робастной нейронной сети значительно возрастают.

ЛИТЕРАТУРА

1. *Ланкин Ю. П., Басканова Т. Ф., Лобова Т. И.* Нейросетевой анализ сложноорганизованных экологических данных // Современные проблемы науки и образования. 2012. № 4;
URL: https://www.science-education.ru/ru/article/view?id=6754

2. *Манжула В. Г., Федяшов Д. С.* Нейронные сети Кохонена и нечёткие нейронные сети в интеллектуальном анализе данных // Фундаментальные исследования. 2011. № 4. С. 108–115; URL: https://www.fundamental-research.ru/ru/article/view?id=21239
3. Глубокие нейросети. Часть I: Подготовка данных; URL: https://www.mql5.com/ru/articles/3486
4. *Fan J., Gijbels I.* Local Polynomial Modelling and Its Applications. London: Chapman & Hall, 1996; DOI: 10.1201/9780203748725
5. *Fujimoto S., Meger D., Precup D.* An Equivalence between Loss Functions and Non-Uniform Sampling in Experience Replay. 2020; URL: https://papers.nips.cc/paper/2020/file/a3bf6e4db673b6449c2f7d13ee6ec9c0-Paper.pdf
6. *Barron J. T.* A General and Adaptive Robust Loss Function. 2017; URL: https://arxiv.org/abs/1701.03077
7. *Andreou P., Charalambous C., Martzoukos S.* Robust Artificial Neural Networks for Pricing of European Options // Computational Economics. 2006. V. 2, N 27. P. 329–351; DOI: 10.1007/s10614-006-9030-x
8. *Sebastiani F.* Text Categorization // Text mining and Its Applications. Southampton: WIT Press, 2005. P. 109–129; DOI: 10.1007/978-0-387-39940-9_414
9. *Bishop C.* Neural Networks for Pattern Recognition. N. Y.: Oxford Univ. Press, 1995.
10. *Химмельблау Д.* Прикладное нелинейное программирование. М.: Мир, 1975.
11. *Huber J. P.* Robust Statistics. N. J.: Wiley, Hoboken, 2009; DOI:10.1002/9780470434697
12. *UCI Machine Learning Repository*; URL: http://www.ics.uci.edu/ mlearn/MLRepository.html
13. *Ивахненко А. Г., Степашко В. С.* Помехоустойчивость моделирования. Киев: Наук. думка, 1985.

SIBERIAN JOURNAL OF INDUSTRIAL MATHEMATICS

UDC 004.85

SIMPLE ROBUST NEURAL NETWORK

© 2021 **V. S. Timofeev**[a], **M. A. Sivak**[b]

Novosibirsk state technical university,
prosp. K. Marksa 20, Novosibirsk 630073, Russia

E-mails: [a]v.timofeev@corp.nstu.ru, [b]pepelyaeva@ami.nstu.ru

Received 17.11.2020, revised 04.10.2021, accepted 21.10.2021

Abstract. The classification problem and applying simple neural networks for solving it are considered. The robust modification of the error backpropagation algorithm that is used for training neural networks is proposed. The proclaim that allows building the proposed modification with the Huber loss-function is proved. In order to study the properties of the obtained neural network, a number of computational experiments has been carried out. The different values of outliers' fraction, noise level, and training and test samples size have been considered. The result analysis shows that the proposed modification can significantly increase classification accuracy and learning rate of a neural network when working with noisy data.

Keywords: classification problem, neural network, Huber loss-function, error backpropagation algorithm.

DOI: 10.33048/SIBJIM.2021.24.409

REFERENCES

1. Lankin Yu. P., Baskanova T. F., Lobova T. I. Нейросетевой анализ сложноорганизованных экологических данных [Neural network analysis of complicated ecological data]. Sovremennye problemy nauki i obrazovaniya, 2012, No. 4 (in Russian); URL: https://www.science-education.ru/ru/article/view?id=6754
2. Manzhula V. G., Fedyashov D. S. Neironnye seti Kohonena i nechetkie neironnye seti v intellekdtualnom analize dannyh [Kohonen neural networks and fuzzy neural networks in data mining]. *Fundamentalnye issledovaniya*, 2011, No. 4, pp. 108–115;
URL: https://www.fundamental-research.ru/ru/article/view?id=21239
3. Glubokie neiroseti (Chast' 1). Podgotovka dannyh [Deep neural networks (Part 1). Data preprocessing];
URL: https://www.mql5.com/ru/articles/3486
4. Fan J., Gijbels I. Local Polynomial Modelling and Its Applications. London: Chapman & Hall, 1996;
DOI: 10.1201/9780203748725
5. Fujimoto S., Meger D., Precup D. An Equivalence between Loss Functions and Non-Uniform Sampling in Experience Replay. 2020;
URL: https://papers.nips.cc/paper/2020/file/a3bf6e4db673b6449c2f7d13ee6ec9c0-Paper.pdf
6. Barron J. T. A General and Adaptive Robust Loss Function. 2017;
URL: https://arxiv.org/abs/1701.03077
7. Andreou P., Charalambous C., Martzoukos S. Robust artificial neural networks for pricing of european options. *Computational Economics*, 2006, Vol. 2, No. 27, pp. 329–351; DOI: 10.1007/s10614-006-9030-x
8. Sebastiani F. Text Categorization. Text mining and Its Applications. Southampton: WIT Press, 2005. P. 109–129; DOI: 10.1007/978-0-387-39940-9_414
9. Bishop C. Neural Networks for Pattern Recognition. N. Y.: Oxford Univ. Press, 1995.

English translation is published in Journal of Applied and Industrial Mathematics, 2021, Vol. 15, No. 4.

10. Himmelblau D. M. Applied Nonlinear Programming. McGraw-Hill, 1972.
11. Huber J. P. Robust Statistics. N. J.: Wiley, Hoboken, 2009; DOI:10.1002/9780470434697
12. UCI Machine Learning Repository; URL: http://www.ics.uci.edu/ mlearn/MLRepository.html
13. Ivahnenko A. G., Stepashko V. S. Pomehoustojchivost' modelirovanija [Noise-resistant modelling]. Kiev: Naukova Dumka, 1985 (in Russian).