Сравнительный анализ эффективности алгоритмов обучения многослойного персептрона на примере решения задачи классификации

С.А. Онисич[1], О.П. Солдатова[1]

[1]Самарский национальный исследовательский университет им. академика С.П. Королева, Московское шоссе 34А, Самара, Россия, 443086

Аннотация. В работе рассматривается процесс решения задачи классификации на основе модели многослойного персептрона с использованием различных алгоритмов обучения. Анализируются результаты трёх различных алгоритмов обучения: наискорейшего спуска и метода обратного распространения ошибки, генетического алгоритма и алгоритма роя частиц. Проводится сравнительный анализ погрешности классификации и затрат времени на одну эпоху обучения при обучении многослойного персептрона различными алгоритмами на модельных данных. В качестве исследуемых данных используются наборы данных ирисов Фишера и Red Wine Quality.

1. Введение

В настоящее время для решения задачи классификации широко используются нейронные сети различной архитектуры: сети с самоорганизацией на основе конкуренции, многослойный персептрон, сети с нечёткой логикой, рекуррентные сети.

Целью данной работы является исследование эффективности решения задачи классификации при помощи модели многослойного персептрона в зависимости от использования алгоритмов обучения различных типов: градиентного алгоритма наискорейшего спуска вместе с алгоритмом обратного распространения ошибки; генетического алгоритма и стохастического алгоритма роя частиц.

Впервые математическая модель персептрона была предложена Ф. Розенблаттом в 1957 году. Впоследствии работы Розенблатта подверглась критике со стороны различных ученых; в 1968 году Д. Румельхартом была представлена обновленная версия этой нейронной сети. В числе изменений, внесенных Румельхартом стоит отметить использование нелинейной функции активации и наличие нескольких обучаемых слоев [1,2].

Идея применения многослойного персептрона для решения задачи классификации не нова. Можно найти множество работ, в которых используется этот тип нейронной сети, опубликованных в течение последних 30 лет. В качестве примера можно взять работу «Classification of Iris data set» [3], в которой авторы рассматривают решение задачи классификации на примере наборе данных об ирисах Фишера с использованием различных типов нейронных сетей: многослойного персептрона, сети радиально-базисных функций и вероятностной нейронной сети. В результате проведенных экспериментов было установлено, что после предобработки данных многослойный персептрон показывает наибольшую точность

как на обучающем наборе данных, так и на тестовой выборке. Полученное в результате значение кросс-валидации оказывается близким к полученной в этой работе.
В качестве данных для исследования используются данные задачи об ирисах Фишера, как классический пример задачи классификации, и набор данных для классификации итальянских вин Red Wine Quality. Оба набора данных взяты из открытого источника Kaggle.

2. Обучение многослойного персептрона

Задача обучения нейронной сети состоит в обучении таким образом, чтобы достичь баланса между способностью давать верный отклик на входные данные, использовавшиеся в процессе обучения, и способностью выдавать правильные результаты в ответ на входные данные, схожие, но не идентичные тем, что использовались при обучении.
Одним из самых распространённых алгоритмов обучения является алгоритм обратного распространения ошибки – систематический метод расчета компонентов вектора градиента функции ошибки для обучения многослойных нейронных сетей градиентными алгоритмами. Как правило, при использовании алгоритма обратного распространения ошибки в качестве функции активации используется сигмоида. Выбор этого вида функций объясняется несколькими достоинствам: нормализации суммы входных сигналов до требуемого значения, автоматическому контролю усиления (большее усиление для слабых сигналов) и простое вычисление производной. В реализованной системе в качестве функции активации используется гиперболический тангенс [4].
Для описания алгоритма обратного распространения ошибки обозначим входы нейронной сети как x_1, …, x_n, а множество выходных узлов — Outputs. Пронумеруем все узлы, в том числе входные и выходные, числами от 1 до N. Тогда вес связи между i-м и j-м узлами можно обозначить $w_{i,j}$, а выход i-го узла — o_i. В таком случае, обозначив тестовое значение k-го нейрона в тестовом векторе d как $\{t_k^d\}$, при учете размера тестовой выборки m, функцию ошибки, полученную по методу наименьших квадратов, выглядит следующим образом:

$$E(\{w_{i,j}\}) = \frac{1}{2}\sum_{d=1}^{m}\sum_{k\in Outputs}\left(t_k^d - o_k\left(x_{1,}^d\ldots, x_n^d\right)\right)^2.$$

Для модификации весов будем применять стохастический градиентный спуск, для чего необходимо изменять веса после каждого тестового примера. Необходимо двигаться в сторону, противоположную направлению градиента, для чего к весу w_{ij} прибавляется величина:

$$\Delta w_{i,j} = -\eta\frac{\delta E^d}{\delta w_{i,j}},$$

где $E^d(\{w_{i,j}\}) = \frac{1}{2}\sum_{k\in Outputs}(t_k^d - o_k^d)^2$. Производная считается следующим образом. Пусть сначала $j \in Outputs$, то есть изменяемый вес относится к нейрону выходного слоя. Таким образом, w_{ij} влияет на выход только как часть суммы $S_j = \sum_i w_{i,j}\,x_{i,j}$, где сумма берется по входам j-го узла. Поэтому

$$\frac{\delta E^d}{\delta w_{i,j}} = \frac{\delta E^d}{\delta S_j}\frac{\delta S_j}{\delta w_{i,j}} = x_{i,j}\frac{\delta E^d}{\delta S_j}.$$

Аналогично, S_j влияет на общую ошибку только в рамках выхода j-го узла o_j. Поэтому

$$\frac{\delta E^d}{\delta S_j} = \frac{\delta E^d}{\delta o_j}\frac{\delta o_j}{\delta S_j} = \left(\frac{\delta}{\delta o_j}\frac{1}{2}\sum_{k\in Outputs}(t_k - o_k)^2\right)\left(\frac{\delta\sigma(S_j)}{\delta S_j}\right) = \left(\frac{1}{2}\frac{\delta}{\delta o_j}\left(t_j - o_j\right)^2\right)\left(o_j(1-o_j)\right) = -c.$$

Если же j-й узел находится не в последнем слое, то есть нейроны в следующем слое, соединенные с ним. Обозначим такие нейроны как Children(j). В таком случае:

$$\frac{\delta E^d}{\delta S_j} = \sum_{k\in Children(j)}\frac{\delta E^d}{\delta S_k}\frac{\delta S_k}{\delta S_j};\frac{\delta S_k}{\delta S_j} = \frac{\delta S_k}{\delta o_j}\frac{\delta o_j}{\delta S_j} = w_{i,j}\frac{\delta o_j}{\delta S_j} = w_{i,j}o_j(1-o_j),$$

где $\frac{\delta E^d}{\delta S_k}$ - это та же поправка, но вычисленная для узла следующего слоя. Обозначим ее δ_k – она отличается от Δ_k отсутствием множителя $\eta x_{i,j}$. Свое название алгоритм получил именно из-за вычисления поправок на основе поправок следующего слоя — т. е. значение ошибки влияет на нейронную сеть от выходного слоя к входному.

Преимущество использования генетического алгоритма и алгоритма роя частиц состоит в отсутствии требований к используемой функции активации. Их действие основано на одновременной обработке нескольких потенциальных решений и применении простых арифметических операций к множество обрабатываемых значений, что требует больше времени на обработку.

Основа теории генетических алгоритмов была разработана в 1975 году Дж. Г. Холландом, а впоследствии это направление развивалось и другими исследователями. Генетические алгоритмы заимствуют многие принципы и термины из генетики. Каждая особь в генетическом алгоритме представляет собой потенциальное решение некоторой задачи. Множество используемых особей называют популяцией. Поиск оптимального решения происходит в процессе эволюции популяции, т. е. преобразования одного множества решений в другое при помощи генетических операторов: репродукции, рекомбинации и мутации.

Процесс отбора обусловлен механизмами естественной эволюции: выживания наиболее приспособленных особей, т. е. решений с наилучшими значениями целевой функции, скрещивание хромосом-родителей для получения хромосомы-потомка и наличия изменяющей хромосомы мутации [5].

Основной алгоритм работы для популяции из n особей выглядит следующим образом:

1 генерация начальной популяции из n особей;
2 вычисление значения целевой функции для каждой из особей;
3 если критерий останова не выполняется, то выбор нескольких особей для создания новой популяции и переход к шагу 3;
4 отбор пары особей-родителей с помощью одного из способов отбора;
5 проведение рекомбинации двух родителей для производства двух потомков;
6 проведение мутации среди потомков с заранее определенной вероятностью;
7 повторение шагов 4-6 до тех пор, пока не будет получено новое поколение из n особей;
8 переход к шагу 2.

Существует множество различных вариаций операторов отбора, рекомбинации и мутации, описание которых выходит за границы данной работы. Однако необходимо рассказать об используемых в данной работе операторах.

Оператор отбора реализован на основе панмиксии, т. е. случайного выбора родительских особей из популяции. Такой подход не обладает специфическими особенностями, поэтому подходит для решения любых задач. Тем не менее, эффективность генетического алгоритма с использованием панмиксии будет снижаться при увеличении размера популяции.

Оператор рекомбинации основан на дискретной рекомбинации или на кроссинговере. В данной работе используется однородный кроссинговер, суть которого заключается в следующем: для создания потомков используется бинарная маска с длиной, равной длине хромосом-родителей. Каждый бит маски в вероятность p0 (в данной работе p0 = 0,5) выставляется равным 1, иначе он равен 0. По одной маске можно получить двух потомков, аналогично операции одноточечного кроссинговера, для чего первому потомку те гены, в которых в маске выставлено значение 1, берутся от первого родителя, а там, где в маске стоит 0, в хромосому берутся гены второго родителя. Второй потомок получается тем же путем, за исключением того, что первому родителю соответствует 0, а второму – 1.

Оператор мутации используется, чтобы избежать попадания в локальный максимум, что достигается за счет случайного изменения хромосомы особи. Чаще всего используют операторы мутации для вещественных особей, бинарную мутацию, мутацию плотности и некоторые другие виды. В данной работе используется мутация для вещественных особей. Суть данного метода в том, что ген сдвигается на определенную величину, называемую шагом мутации. Определение этой величины вызывает некоторые трудности, т.к. при малом значении шага мутации популяция может «застрять» в локальном минимуме, а при большом – «перелететь» его. В данной работе значение величины шага мутации δ определяется случайным образом в каждом отдельном случае по формуле $\delta = \sum_{i=1}^{m} a(i)2^{-i}$, где a(i) = 1 с вероятностью 1/m, в противном случае a(i) = 0, а m – параметр оператора.

Для выбор особей в новую популяцию используются отбор отсечением и элитарный отбор. В ходе элитарного отбора в новую популяцию попадают только лучшие особи из старой популяции, что позволяет избежать потери наилучших решений. Для отбора усечением выбирается порог T ϵ [0;1]. Этот порог обозначает долю особей, участвующих в отборе. Среди этих особей выбирается случайная особь и добавляется к новой популяции. Этот процесс повторяется до тех пор, пока размер новой популяции не достигнет требуемой величины [6].

Алгоритм роя частиц был предложен в 1995 году Джеймсом Кеннеди и Расселом Эберхартом на основе модели стаи птиц Boids, а также работ Хеппнера и Гренадера на схожую тему. Кеннеди и Эберхарт отметили, что обе модели основаны на управлении дистанциями между птицами, поэтому можно считать синхронность стаи функцией от усилий, которые птицы прикладывают для сохранения оптимальной дистанции.

Разработанный алгоритм прост; он моделирует многоагентную систему, в которой агенты-частицы двигаются к оптимальным решениям, обмениваясь собранной информацией с другими агентами. Текущее состояние частицы характеризуется координатами в пространстве решений, вектором скорости перемещения (эти параметры выбираются случайным образом на этапе инициализации), а также координаты лучшего из найденных ей решений и лучшее из пройденных всеми частицами решений, благодаря чему реализуется мгновенный обмен информацией между агентами.

На каждой итерации алгоритма направление и длина вектора скорости изменяется в соответствие с найденными оптимумами:

$$\vec{V} = \vec{V} + \alpha_1 * rnd(\quad) * \left(\overrightarrow{pbest_i} - \vec{X}\right) + \alpha_2 * rnd(\quad) * \left(\overrightarrow{gbest_i} - \vec{X}\right),$$

где $\vec{V}$ - вектор скорости частицы, α_1, α_2 — постоянные ускорения, pbest – лучшая найденная частицей точка, gbest – лучшая точка из пройденных всеми частицами системы, $\vec{X}$ - текущее положение частицы, rnd() - функция, возвращающая случайное число от 0 до 1.

3. Исследование зависимости значения погрешности от используемого алгоритма обучения на примере набора данных «Ирисы Фишера»

Ирисы Фишера – это набор данных, описывающих 150 экземпляров ирисов, по 50 экземпляров каждого из трех сортов: iris setosa, iris viginica и iris versicolor. Этот набор данных был опубликован в 1936 году Р.А. Фишером в статье «The Use of Multiple measurements in Taxonomic Problems». Один из видов ирисов линейно отделим от двух других, но оставшиеся два линейно не отделимы друг от друга [7]. Для обучения выборка разбивалась на две части: обучающую, включающая в себя данные о 120 ирисах, и тестовую, с данными о 30 ирисах. Обучаемая нейронная сеть представляет собой многослойный персептрон с одним скрытым слоем; в первом слое персептрон содержит 4 нейрона, в скрытом слое – также 4 нейрона и 3 нейрона в выходном слое. Постоянная обучения для алгоритма наискорейшего спуска равна 0,05. Популяция в генетическом алгоритме состояла из 50 особей; столько же частиц использовалось при работе алгоритма роя частиц. Результаты измерений включают погрешность классификации и время, затрачиваемое на проведение одной эпохи обучения. Они представлены на рисунках 1 и 2 соответственно.

Набор данных Red Wine Quality – это набор, описывающий сорта португальского вина «Vinho Verde». В него входят различные характеристики вина: постоянная и летучая кислотность, содержание лимонной кислоты, pH, содержание алкоголя и др. Всего параметров 11, на их основе определяется качество вина как целое число от 0 до 10 [8]. Набор данных включает более полутора тысяч обучающих векторов; для обучения нейронной сети он был разбит в соотношении 2:1. Параметры алгоритмов обучения те же, что и в предыдущем эксперименте. В данном случае использовался персептрон с двумя скрытыми слоями, каждый из которых содержит по 8 нейронов. Полученные результаты представлены на рисунках 3 и 4.

На основании полученных данных можно прийти к следующим выводам:

1) Алгоритм наискорейшего спуска вместе с алгоритмом обратного распространения ошибки в большинстве случаев сводит значение погрешности к достаточно близкому к

минимальному значению после продолжительного обучения. Стоит также отметить наименьшее время обучения среди всех рассматриваемых алгоритмов.

2) Алгоритм роя частиц проводит обучение за минимальное количество эпох. Достигнутая погрешность немногим больше погрешности, достигнутой алгоритмом наискорейшего спуска вместе с алгоритмом обратного распространения ошибки. Таким образом, несмотря на время обучения одной эпохи почти в два раза большее, чем у алгоритма наискорейшего спуска вместе с алгоритмом обратного распространения ошибки, алгоритм имеет большой потенциал благодаря разнообразным его модификациям.
3) Генетический алгоритм в используемой его реализации показал наихудшие результаты из всех алгоритмов. Время обучения существенно больше, чем у двух предыдущих алгоритмов, и значение погрешности, достигнутое им, также самое большое. Тем не менее, как алгоритм роя частиц, генетический алгоритм имеет множество вариаций, обусловленных различными операторами селекции, воспроизведения и мутации.

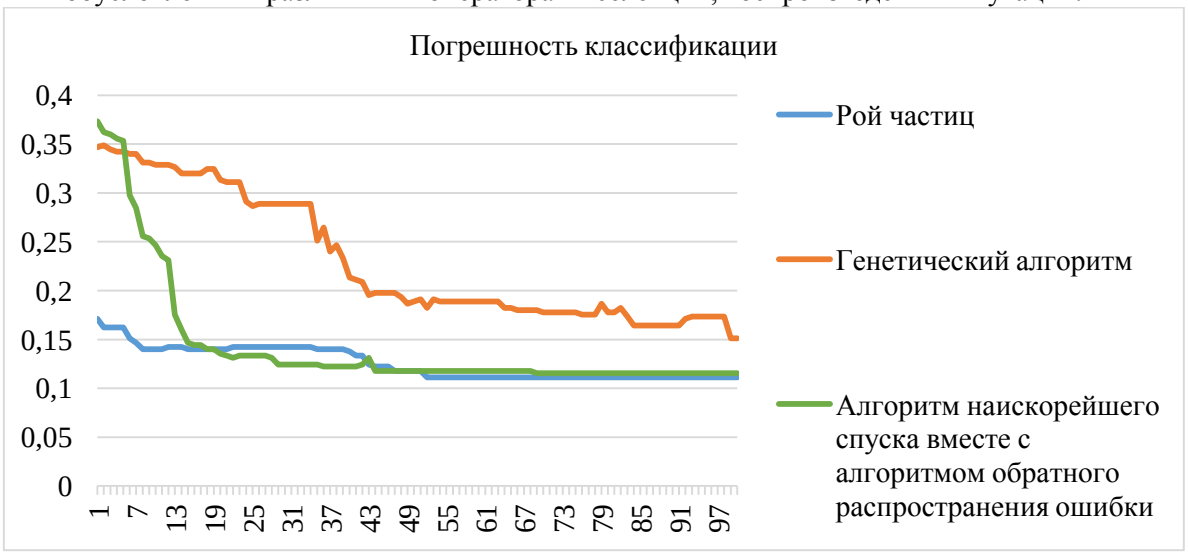

Рисунок 1. Погрешность классификации ирисов Фишера для различных алгоритмов обучения.

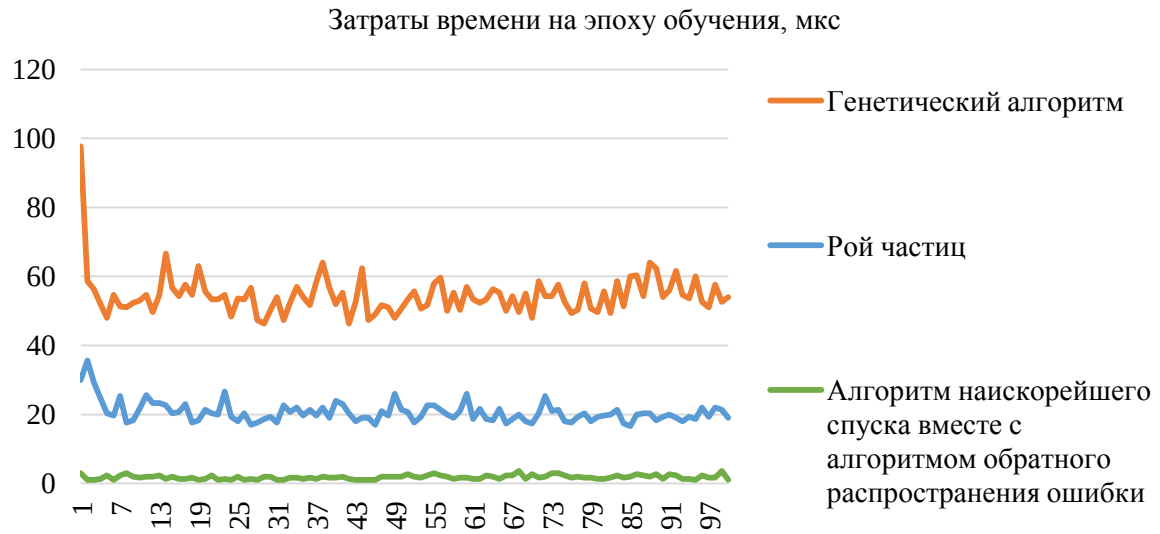

Рисунок 2. Затраты времени на эпоху обучения при классификации ирисов Фишера.

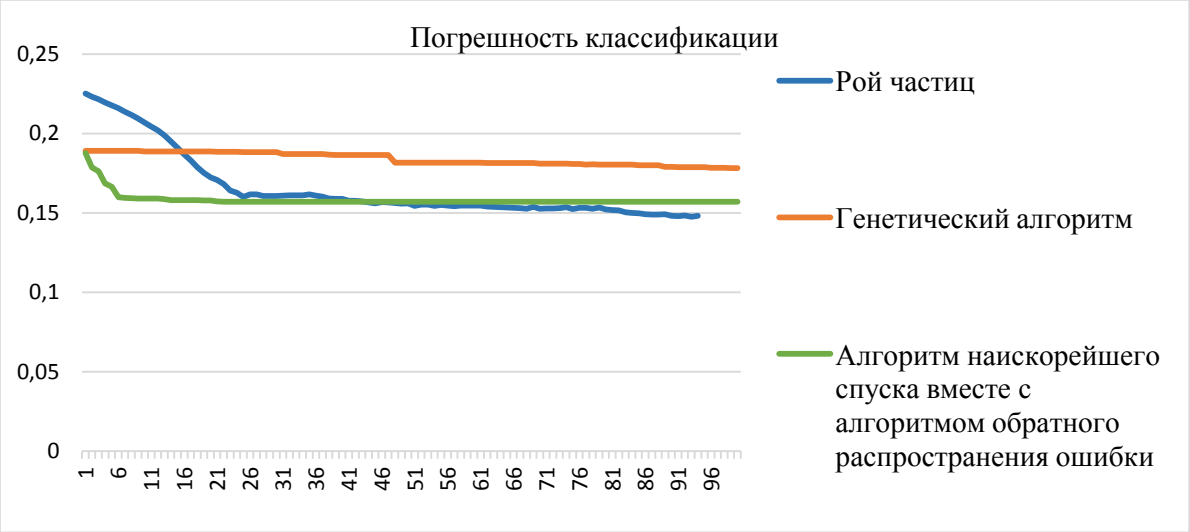

Рисунок 3. Погрешность классификации набора данных Red Wine Quality для различных алгоритмов обучения.

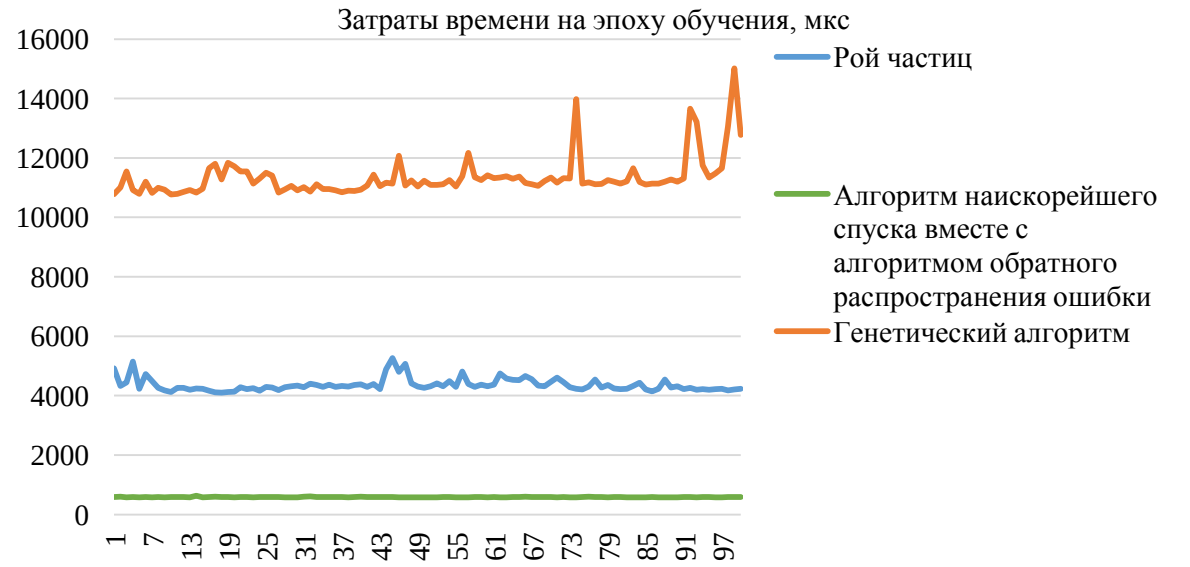

Рисунок 4. Затраты времени на эпоху обучения при использовании данных Red Wine Quality для различных алгоритмов обучения.

4. Заключение

В соответствии с обозначенной задаче было проведено исследование влияния алгоритма обучения на погрешность решения задачи классификации.

В ходе выполнения работы исследованы различные алгоритмы обучения: алгоритм наискорейшего спуска вместе с алгоритмом обратного распространения ошибки, генетический алгоритм и алгоритм роя частиц -, рассмотрены особенности их параметров.

В данной работе приведены результаты исследования зависимости погрешности обучения от алгоритма обучения. Рассмотрены наборы данных и алгоритмы обучения. Результаты подробно изложены во второй и третьей главах.

5. Литература

[1] Розенблатт, Ф. Принципы нейродинамики. Перцептроны и теория механизмов мозга. – М.: Мир, 1965. – 481 с.

[2] Rumelhart, D.E. Parallel Distributed Processing: Explorations in the Microstructures of Cognition / D.E. Rumelhart, J.L. McClelland. – Cambridge, MA: MIT Press, 1986.
[3] Potočnik, P. Classification of Iris data set / P. Potočnik, V. Borovinskiy. – Любляна, 2009.
[4] Уоссермен, Ф. Нейрокомпьютерная техника: Теория и практика. – М.: Мир, 1992. – 184 с.
[5] Эволюционные вычисления. Лекция 1: Введение. Основы генетических алгоритмов. [Электронный ресурс]. – Режим доступа: https://www.intuit.ru/studies/courses/14227/1284/lecture/24168 (04.06.18).
[6] Панченко, Т.В. Генетические алгоритмы. – Астрахань: Издательский дом «Астраханский университет», 2007. – 87 с.
[7] Iris Species | Kaggle [Электронный ресурс]. – Режим доступа: https://www.kaggle.com/uciml/iris/home (10.11.18).
[8] Red Wine Quality | Kaggle [Электронный ресурс]. – Режим доступа: https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009/home (10.11.18).

Comparative analysis of the effectiveness of the multilayer perceptron learning algorithm for solving the classification problem

S.A. Onisich[1], O.P. Soldatova[1]

[1]Samara National Research University, Moskovskoe Shosse 34A, Samara, Russia, 443086

Abstract. The process of solving the classification problem based on the multilayer perceptron model using different learning algorithms is presented. The steepest descent and the back propagation method, the genetic algorithm, and the particle swarm algorithm results are analyzed. A comparative analysis of the time spent on one era of training and classification errors in the training of a multilayer perceptron by different algorithms on model and real data is carried out. Fisher's iris and Red Wine Quality datasets are used as the test datasets.