DOI10.37882/2223–2966.2022.07.18

ЭФФЕКТИВНОСТЬ РАСПАРАЛЛЕЛИВАНИЯ МЕТОДА НА ОСНОВЕ РОЯ ЧАСТИЦ ПРИ ОПТИМИЗАЦИИ ОБУЧЕНИЯ НЕЙРОННЫХ СЕТЕЙ

EFFICIENCY OF PARTICLE SWARM BASED METHOD PARALLELIZATION IN OPTIMIZING TRAINING OF NEURAL NETWORKS

V. Larionov
O. Maleev

Summary. Subject of research: The possibility of using parallel computations of the particle swarm optimization algorithm (PSO) in classification problem using feedforward neural network. Three datasets with various lengths, different number of instances, input features and weights numbers were used for calculations; neural network training was carried out for 500 epochs in 50 independent program runs using different processors number for parallelization. Cross-entropy loss function was chosen for the penalty function in order to estimate performance of the proposed methods. The advantage of using parallel computations with the PSO was shown on the designed networks analysis results.

Keywords: neural network, optimization, PSO, PAPSO, Broadcast PPSO.

Ларионов Вячеслав Сергеевич
Санкт-Петербургский политехнический университет Петра Великого (СПбПУ)
larionov.vs@edu.spbstu.ru
Малеев Олег Геннадьевич
К.т.н., доцент, Санкт-Петербургский политехнический университет Петра Великого (СПбПУ)
maleev_og@spbstu.ru

Аннотация. Предмет исследования: возможность применения распараллеливания вычислений метода роя частиц (PSO) в задаче классификации с использованием нейронных сетей прямого распространения. Для расчетов использованы три датасета, отличающихся числом экземпляров, входных признаков и, соответственно, количеством весовых коэффициентов сети; обучение нейронной сети проводилось в течение 500 эпох в 50 независимых экспериментах с разным количеством процессоров для распараллеливания. Для оценки качества работы предлагаемых методов была выбрана штрафная функция в виде функции перекрестной энтропии. По результатам анализа спроектированных сетей было показано преимущество применения распараллеливания на оптимизации роя частиц.

Ключевые слова: нейронная сеть, оптимизация, PSO, PAPSO, Broadcast PPSO.

Введение

На данный момент времени оптимизация вычислительных методов является одним из самых актуальных задач современной прикладной математики. В ней изучается большое количество различных методов, позволяющих выбрать оптимальный способ решения тех или иных практических задач.

Для облегчения постепенно увеличивающейся сложности решаемых задач, в том числе разбиения сложных вычислений на более простые, были приняты попытки создания систем, имитирующих работу мыслительного процесса человека, с целью их дальнейшего улучшения и развития. [1] Подобные системы именуются "искусственным интеллектом" (ИИ).

Область, которую ИИ охватывает на данный момент, имеет множество направлений, из которых наиболее популярное — машинное обучение. [2] Одним из типов моделей машинного обучения являются искусственные нейронные сети (НС), задействованные практически во всех сферах человеческой жизни: задачи распознавания объектов, получаемых камерой, сегментация объектов на видеоизображении, контроль поведения объектов и т.д. Также для нейронных сетей было опробовано большое количество методов из раздела оптимизации, с помощью которых можно решить те или иные задачи. [2]

В настоящей работе проведен анализ использования методов, полученных применением распараллеливания на алгоритме оптимизации роя частиц PSO; приведены результаты расчетов их вычислительной точности в обучающей и тестовой выборках данных.

1. Постановка задачи

В качестве решаемой задачи в данной работе была выбрана задача классификации. Формулировка задачи звучит следующим образом: имеется некоторый первоначальный набор исходных данных (датасет), состоящий из множества объектов (экземпляров) и конечного множества классов, к которым они принадлежат. Перед началом обучения датасет требуется разделить, следуя одной из возможных стратегий:

- *На две части*: большая часть экземпляров выделяется для тренировочного датасета (обучающую выборку), часть для тестового датасета (например, 80:20);

- *На три части*: большая часть выделяется на тренировочную выборку, две оставшиеся части делятся на небольшую валидационную выборку, которая используется для оценки точности классификации данных после окончания эпохи обучения, и тестовую выборку, на которой будет проведена итоговая оценка точности классификации обученной нейронной сети (например, процентное соотношение выборок 70:10:20, где 70 — тренировочная, 10 — обучающая, 20 — тестовая)

Обучающий и валидационный датасет имеют информацию о каждом классе, к которому принадлежит определенный набор признаков. Используя эти данные, при обучении НС устанавливается зависимость между входными признаками объекта (экземпляра) и его выходными значениями (классами), что в процессе формирует модель, весовые коэффициенты которой рассчитываются для определенного набора данных и позволяют в дальнейшем классифицировать любой другой экземпляр из данного датасета.

Математическая модель данной задачи может быть описана таким образом: имеется два набора данных — X и Y, где X — множество признаков (свойств) объектов, Y — множество классов, определяемых свойствами объекта. Необходимо найти зависимость, существующую между значениями X и Y; при этом известны только те её значения, которые определяются конечным множеством обучающей выборки длиной k:

$$X^k = \{(x_1, y_1), \dots, (x_k, y_k)\} \quad (1)$$

где X^k — набор данных; x_i — множество свойств объектов из набора X, y_i — класс, к которому они относятся.

Таким образом, чтобы решить поставленную задачу, требуется найти решение задачи оптимизации следующего вида:

$$min\, f(z) = f(z_1, z_2, \dots, z_n) \quad (2)$$

где f — оптимизируемая функция; $z_1, z_2, \dots, z_n$ — её входные параметры. Применяя формулу (2) к задаче классификации, можно заметить, что параметрами функции f являются переменные, представляющие собой зависимость между наборами данных X и Y, а сама функция представляет из себя некоторую штрафную функцию, результат которой показывает, насколько определённый набор входных параметров хорошо минимизирует вероятность ложной классификации данных на некотором датасете.

В качестве метода машинного обучения, выбранного для решения данной проблемы, является нейронная сеть прямого распространения. [3] Выбранная архитектура нейронной сети — многослойная нейронная сеть прямого распространения (feed forward network, FFN) [4–5], использующаяся в задачах кластеризации, классификации, регрессии, прогнозирования и т.д. [6] Для лучшего результата вычислений в данной НС также будут использованы нейроны смещения.

В данной работе рассматриваются методы, основанные на распараллеливании вычислений оптимизации роя частиц в задачах, требующих расчетов в функциях от нескольких переменных:

- Распараллеленная асинхронная оптимизация роя частиц (PAPSO).
- Распараллеленная оптимизация роя частиц с использованием трансляции данных (Broadcast PPSO).

1.1 Оптимизации роя частиц (PSO)

Подробное описание оптимизации роя частиц было опубликовано в 1995 году. [7] Авторы данного алгоритма, Кеннеди и Элберхарт, разработали его на основе наблюдений за социальным поведением птиц при поиске пищи. На данный момент было создано множество модификаций и улучшений метода, проведено огромное количество исследований с применением данного алгоритма в различных задачах, помимо тех, что относятся к машинному обучению.

Суть данного алгоритма состоит в следующем: в качестве возможных решений задачи оптимизации PSO использует конечное множество частиц количества N — рой, или же, популяцию. Данная популяция располагается в пространстве случайным образом, используя закон равномерного распределения при генерации начальных положений и скоростей частиц. Эти значения меняются каждую итерацию алгоритма, основываясь на двух следующих значениях: собственное лучшее положение частицы в пространстве и глобальное лучшее положение частиц роя на i-й итерации.

Каждая частица в популяции имеет размерность D, что позволяет описать ее положение в пространстве как $x_i = (x_{i1}, x_{i2}, \dots, x_{iD})$, где i — порядковый номер частицы в рое от 1 до N. Лучшее положение частицы за все время работы алгоритма обозначается как p_i или p_{best_i}. Лучшее положение среди всех частиц в рое обозначается как g_{best} по итогу всех пройденных итераций или g^t_{best}, где t — номер текущей итерации. Скорость частицы, с которой она перемещается в пространстве, обозначается как $v_i = (v_{i1}, v_{i2}, \dots, v_{iD})$. Значения, которые скорости частиц могут принимать, варьируются между следующими границами:

$$[\overline{v_{min}};\ \overline{v_{max}}] = \left[-б * \left(\overline{b_{up}} - \overline{b_{low}}\right); б * \left(\overline{b_{up}} - \overline{b_{low}}\right)\right] \quad (3)$$

где $\overline{v_{min}};\ \overline{v_{max}}$ — нижние и верхние границы скорости для каждого измерения; $\overline{b_{low}}, \overline{b_{up}}$ — векторы нижней и верхней границы области поиска для каждого измерения; б — пороговый ограничитель в промежутке от 0 до 1, значение которого обычно выбирается небольшим. Наиболее частые константы для порогового ограничителя — 0,2 или 0,25.

Оптимизация роя частиц работает на протяжении заданного количества итераций. На каждой итерации производится обновление значений скорости и положения каждого измерения для каждой частицы, описывающееся следующими уравнениями:

$$v_{id}^{t+1} = w * v_{id}^{t} + c_1 * r_1 * (p_{id}^{t} - x_{id}^{t}) + c_2 * r_2 * \quad (4)$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1}$$

где t — номер текущей итерации, d — номер измерения, w — вес инерции; c_1, c_2 — познавательный и социальный факторы обучения; r_1, r_2 — случайные числа, располагающиеся на отрезке [0; 1]. После вычисления значения скорости по формуле (4) требуется проверить его на попадание в допустимый предел, обозначенный в (3), и, в случае выхода за границы, присвоить значение, располагающееся на ближайшей по значению границе.

После полного обновления измерений скорости и положения частицы производится подсчет функции оптимизации (2) в данной точке и сравнение с её с лучшим положением p_{best_i} за все время, и, если новое значение минимизирует функцию оптимизации более успешно, происходит замена лучшего положения частицы на новое. Далее по тому же принципу проводится проверка, является ли новая позиция частицы лучшей в рое (g_{best}) на данный момент.

Для формулы (3) также требуется подбор параметров, используемых при подсчётах (w, c_1, c_2). Для когнитивного и социального факторов c_1 и c_2 экспериментальным путем были найдены оптимальные варианты значений для обоих значений: 1.49617 [8] и 1,49445 [9]. Для веса инерции w была выбрана формула линейного убывания:

$$w = w_0 + (w_0 - w_1) * t \ / \ t_{max}$$

где w_0 — верхняя граница изменения веса инерции; w_1 — нижняя граница изменения веса инерции; t — номер текущей итерации; t_{max} — максимальное количество итераций алгоритма. [10–11] Значениями для улучшения сходимости алгоритма являются константы w_0=0,9 и w_1=0,4.

1.2 Распараллеленная асинхронная оптимизация роя частиц (PAPSO)

Уравнения данного метода аналогичны тем, которые применяются в PSO, за исключением того, что работа данного алгоритма опирается на параллельные вычисления целевого функционала. В основе PAPSO [12] лежит следующая парадигма — ведущий / ведомый. Главный процессор хранит очередь частиц, готовых к отправке подчиненным процессорам, и выполняет все процессы принятия решений, такие как обновление скорости / положения и проверки сходимости, таким образом не выполняя вычислений функционала напрямую. Ведомые процессоры же многократно вычисляют функцию потерь, используя присвоенные им частицы на каждой итерации.

Главный процессор делает следующие задачи:

1.1. Инициализирует все параметры оптимизации, положения и скорости частиц;

1.2. Содержит очередь частиц для оценки подчиненными процессорами;

1.3. Обновляет положения и скорости частиц на основе имеющейся в настоящее время информации p_i, g_{best};

1.4. Инициализирует все параметры оптимизации, положения и скорости частиц;

1.5. Содержит очередь частиц для оценки подчиненными процессорами;

1.6. Обновляет положения и скорости частиц на основе имеющейся в настоящее время информации p_i, g_{best};

1.7. Отправляет позицию x_i следующей частицы в очереди доступному ведомому процессору;

1.8. Получает значения целевого функционала от подчиненных процессоров;

1.9. Проверяет сходимость.

Ведомый процессор делает следующие задачи:

1.1. Получает позицию частицы от главного процессора;

1.2. Оценивает функцию анализа $f(x_i)$ при заданном положении частицы x_i;

1.3. Отправляет значение функции стоимости на главный процессор.

После того, выполнена инициализация, проводимая главным процессором, частицы отправляются в подчиненные процессоры для оценки функции анализа. [13] Для работы с ведомыми процессами алгоритмом используется централизованная очередь задач по принципу «первым поступил — первым обслужен», тем самым, определяется порядок поступления частиц рабочим процессорам. Когда подчиненный процессор завершает оценку функции, то он отправляет главному процессору значение целевого функционала и соответствующий номер частицы, для которой был произведён расчёт,

Таблица 1. Общая характеристика датасетов

№ датасета	Наименование применяемого датасета	Количество экземпляров	Количество входных признаков	Количество выходных классов
1	Vehicles	846	18	4
2	Wireless Indoor Localization	2000	7	4
3	Wine Quality — White	4898	11	7

и ведущий процессор помещает номер частицы в конец очереди задач.

Поскольку порядок, в котором частицы сообщают о своих результатах, меняется, возникает случайность в порядке частиц. Как только частица достигает начала очереди задач, главный процессор обновляет ее позицию и отправляет ее следующему доступному подчиненному процессору. Если количество подчиненных процессоров равно количеству частиц, то следующим доступным процессором всегда будет тот же процессор, который первоначально обрабатывал частицу. Если количество подчиненных процессоров меньше количества частиц, то следующим доступным процессором будет тот процессор, который окажется свободным, когда частица достигнет начала очереди задач.

1.3 Распараллеленная оптимизация роя частиц с использованием трансляции данных (Broadcast PPSO)

Как и в предыдущем методе, в модели широковещательной распараллеленной оптимизации роя частиц происходит разбивка частиц на несколько отдельных подгрупп. Однако, в отличии от предыдущего алгоритма, процессоры не синхронизуют всю свою дальнейшую работу с некоторым ведущим. Они обмениваются данными друг с другом и выполняют свои вычисления целевого функционала параллельно. [14] Информация о g_{best_k} ($k = 1, \ldots, N$, где N — количество отдельных роев), получаемая каждой группой, транслируется остальным группам для вычисления наилучшего значения во всем рое, вследствие чего все узлы после определения лучшего значения роя g_{best} могут продолжать поиск оптимального значения.

Данный алгоритм выполняет следующую последовательность действий:

1. Главный процессор инициализирует начальные параметры и делится ими ведомым устройствам. К этим параметрам относится количество итерации, вес инерции, период связи (например, t = 4 итерации), популяция размеры и коэффициенты ускорения;
2. Каждая подгруппа развивается отдельно и получает свои p_i и g_{best_k};
3. Все подгруппы делят свои лучшие позиции g_{best_k} друг с другом для получения g_{best} роя в определенный период общения t;
4. С обновленными p_i для отдельных роев и уже новым подсчитанным глобальным g_{best} по всему рою, каждый рой обновляет позиции и скорости своих частиц;
5. Повторение шага 3 до тех пор, пока не будет достигнут критерий остановки (достигнута необходимая точности), либо пока не пройдут все итерации алгоритма.

2. Используемые функции обработки данных и метрики

2.1 Подготовка датасетов

Для данной работы были выбраны следующие датасеты, информация о которых имеется в таблице 1.

Количество экземпляров по каждому из классов в каждом из датасетов следующее:

1. *Vehicles*: 1 класс — 199, 2 класс — 217, 3 класс — 218, 4 класс — 212
2. *Wireless Indoor Localization*: 1 класс — 500, 2 класс — 500, 3 класс — 500, 4 класс — 500
3. *Wireless Indoor Localization*: 1 класс — 20, 2 класс — 163, 3 класс — 1457, 4 класс — 2198, 5 класс — 880, 6 класс — 175, 7 класс — 5

Таким образом, в последнем датасете также имеется проблема несбалансированных данных.

Входные признаки в наборах данных нормализованы для упрощения работы НС в заданной границами области. Начальные значения весов (измерений частиц) генерируются по закону равномерного распределения, используя для положений частиц границы $\overline{b_{low}}$ и $\overline{b_{up}}$, а для скоростей — границы из формулы $\overline{v_{min}}$ и $\overline{v_{max}}$. Каждый датасет будет разбит в соотношении 80 на 20 для обучающей и тестовой выборок.

Таблица 2. Распределение нейронов на каждом из слоев нейронной сети

№ датасета	Количество входных нейронов	Количество нейронов скрытого слоя			Количество выходных нейронов
		1 слой	2 слой	3 слой	
1	18	20	15	10	4
2	7	10	8	6	4
3	11	15	11	9	7

Количество слоев в НС равно пяти: входной, три скрытых и выходной. Количество нейронов на каждом из слоев сети представлено в таблице 2.

В соответствии с известным количеством нейронов на каждом слое НС можно определить количество весовых коэффициентов сети для каждого из датасетов:

1. *Vehicles*: 899
2. *Wireless Indoor Localization*: 250
3. *Wireless Indoor Localization*: 534

2.2 Выбор используемых функций обработки данных и метрик

Так как в каждом из датасетов представлена задача множественной классификации, то в данной статье в качестве функции потерь рассматривается перекрестная энтропия (Cross-Entropy) [15]:

$$H_{(p_n,q_n)} = -\sum_{i=1}^{N} p_i * log(q_i)$$

где p_i — известное значение i-го выхода для соответствующего набора входных признаков, q_i — значение, подсчитанное функцией активации для i-го нейрона выходного слоя.

В соответствии с выбранной функцией потерь требуется определить функцию активации для скрытых слоев и для последнего слоя, результаты которого и будут анализировать:

1.1. Для слоёв скрытого уровня — гиперболический тангенс:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

где x — взвешенная сумма, поступающая на вход нейрона следующего слоя.

1.2. Для нейронов выходного слоя — Softmax:

$$f(\overline{x}) = \frac{e^{x_i}}{\sum_{i=1}^{N} e^{x_i}}$$

где x_i — значение выходного нейрона, отвечающего за i-й класс.

Таким образом, значения, преобразованные гиперболическим тангенсом, будут преобразовывать выходные значения в пределе, допустимом для Softmax, а значения, поступающие на выходной слой, будут преобразованы функцией Softmax в связанные вероятности, наглядно показывающие наличие того или иного класса на определенном наборе входных признаков.

2.3 Выбор оптимальных параметров

Для PSO, PAPSO и Broadcast PPSO были выбраны следующие общие параметры:

4. N = 10 / 25;
5. D = количество весовых коэффициентов нейронной сети;
6. t_{max} = 500 итераций (эпох);
7. b_{lo} = –0.3;
8. b_{up} = 0.3;
9. $c_1 = c_2$ = 1.49445;
10. w_0 = 0.9;
11. w_1 = 0.4;
12. $alpha$ = 0.2.

Для Broadcast PPSO был выбран период общения m = 4 эпохи.

Количество независимых запусков обучения и тестирования нейронной сети для каждого из датасетов = 50.

Количество процессоров для PAPSO и Broadcast PPSO — 4 / 8.

3 Обсуждение результатов

3.1 Результаты первого эксперимента

Для первого эксперимента начальное количество эпох обучения выбрано равным 500, количество частиц (наборов весовых коэффициентов) N — равным 10, количество процессоров для распараллеленных методов — 4. После усреднения результатов, подсчитанных

в независимых запусках, была получена следующая информация:

1. Для датасета 1 при использовании PSO точность классификации на тренировочной выборке перешла порог в 72% на обучающей выборке и 71% на тестовой, для PAPSO — 71% на обоих выборках, для Broadcast PPSO — результат, аналогичный PSO. Среднее время работы метода PAPSO составило 39,41% от времени PSO; у Broadcast PPSO — 51,06%.
2. Для датасета 2 при использовании PSO точность классификации на тренировочной и тестовой выборке перешла порог в 97%, для PAPSO — 97% на обоих выборках, для Broadcast PPSO — 97% на обучающей выборке и 96% на тестовой. Среднее время работы метода PAPSO составило 42,29% от времени PSO; Broadcast у PPSO — 47,13%.
3. Для датасета 3 при использовании PSO точность классификации на тренировочной и тестовой выборке перешла порог в 54%, для PAPSO и Broadcast PPSO — 53% на обоих выборках. Среднее время работы метода PAPSO составило 49,30% от времени PSO; у Broadcast PPSO — 54,75%.

3.2 Результаты второго эксперимента

Для второго эксперимента были выбран те же параметры, что и в первом, за исключением количества процессоров для распараллеленных методов, которое в данном случае равно 8. После усреднения результатов, подсчитанных в независимых запусках, была получена следующая информация по распараллеленным методам:

1. Для датасета 1 при использовании PAPSO точность на обоих увеличилась на 0,07% на обучающей выборке и на 0,04% на тестовой выборке, при использовании Broadcast PPSO — упала на 0,065% на обучающей выборке и на 0,05% на тестовой выборке. Среднее время работы метода PAPSO составило 39,68% от времени PSO; у Broadcast PPSO — 43,21%.
2. Для датасета 2 при использовании PAPSO точность увеличилась на 0,05% на обоих выборках, для Broadcast PPSO — упала на 0,06% на обучающей выборке и 0,02% на тестовой. Среднее время работы метода PAPSO составило 42,48% от времени PSO; у Broadcast PPSO — 44,84%.
3. Для датасета 3 при использовании PAPSO точность на обоих увеличилась на 0,03% на обучающей выборке и на 0,06% на тестовой выборке, при использовании Broadcast PPSO — упала на 0,03% на обучающей выборке и на 0,07% на тестовой выборке. Среднее время работы метода PAPSO составило 49,40% от времени PSO; у Broadcast PPSO — 52,07%.

3.3 Результаты третьего эксперимента

Для третьего эксперимента начальное количество эпох обучения выбрано равным 500, количество частиц N — равным 25, количество процессоров для распараллеленных методов — 4. После усреднения результатов, подсчитанных в независимых запусках, была получена следующая информация:

1. Для датасета 1 при использовании PSO точность классификации на тренировочной выборке перешла порог в 75% на обучающей и 74% на тестовой выборке, для PAPSO — 74% на обоих выборках, для Broadcast PPSO — 74% на обучающей и 73% на тестовой выборке. Среднее время работы метода PAPSO составило 45,96% от времени PSO; у Broadcast PPSO — 45,24%.
2. Для датасета 2 при использовании PSO точность классификации на тренировочной выборке перешла порог в 98% на обучающей и тестовой выборке, для PAPSO — 98% на обучающей и 97% на тестовой, для Broadcast PPSO — результат, аналогичный PSO. Среднее время работы метода PAPSO составило 51,97% от времени PSO; у Broadcast PPSO — 50,8%.
3. Для датасета 3 при использовании PSO точность классификации на тренировочной и тестовой выборке перешла порог в 54%, для PAPSO — 53% на обоих выборках, для Broadcast PPSO — 53% на тренировочной и 52% на обучающей. Среднее время работы метода PAPSO составило 49,98% от времени PSO; у Broadcast PPSO — 50,12%.

3.4 Результаты четвертого эксперимента

Для четвертого эксперимента были выбран те же параметры, что и в третьем, за исключением количества процессоров для распараллеленных методов, которое в данном случае равно 8. После усреднения результатов, подсчитанных в независимых запусках, была получена следующая информация по распараллеленным методам:

1. Для датасета 1 при использовании PAPSO точность на обоих уменьшилась на 0,05% на обучающей выборке и на 0,01% на тестовой выборке, при использовании Broadcast PPSO — уменьшилась на 0,01% на обучающей выборке и на 0,05% на тестовой выборке. Среднее время работы метода PAPSO составило 45,93% от времени PSO; у Broadcast PPSO — 47,84%.
2. Для датасета 2 при использовании PAPSO точность увеличилась на 0,05% на обоих выборках, для Broadcast PPSO — упала на 0,06% на обучающей выборке и 0,02% на тестовой. Среднее время работы метода PAPSO составило 51,87% от времени PSO; у Broadcast PPSO — 51,79%.

3. Для датасета 3 при использовании PAPSO точность на обоих увеличилась на 0,02% на обучающей выборке и на 0,01% на тестовой выборке, при использовании Broadcast PPSO — упала на 0,01% на обучающей выборке и на 0,02% на тестовой выборке. Среднее время работы метода PAPSO составило 49,88% от времени PSO; у Broadcast PPSO — 51,11%.

Заключение

Была решена задача классификации, проверяющая качество работы методов, основанных на оптимизации роя частиц, а также являющихся его распараллеленными версиями. В процессе были продемонстрированы результаты их применения, разница в сравнении со стандартным алгоритмом. Данные методы обладают хорошей сходимостью на сбалансированных датасетах, а также требуют меньше времени на вычисление целевого функционала каждой частицы. При этом как время, так и точность классификации с использованием данных методов варьируются в зависимости от общего количества частиц в рое, а также от синхронных, либо асинхронных операций в алгоритмах. В случае несбалансированного датасета все методы показали невысокие результаты точности классификации данных, в связи с чем для решения данной проблемы может потребоваться выбрать другие, более подходящие метрики для оценки точности предсказания и использовать соответствующие взвешенные функции потерь.

Исходя из вышесказанного, можно сделать вывод о том, что данные методы могут быть использованы в задачах классификации наборов данных, имеющих сбалансированное количество классов, в качестве обучающего метода НС. Вместе с тем, эти методы требуют значительных вычислительных мощностей, а также большего объема памяти для вычислений при увеличении количества входных признаков и размера датасета.

ЛИТЕРАТУРА

1. Sarangi S., Sharma P. Artificial Intelligence. Routledge India, 2018.
2. Kubat M. Artificial Neural Networks // An Introduction to Machine Learning. Cham: Springer International Publishing, 2021. P. 117–143.
3. Rojas R. Neural Networks — A Systematic Introduction. 1996.
4. McClarren R.G. Feed-Forward Neural Networks // Machine Learning for Engineers. Cham: Springer International Publishing, 2021. P. 119–148.
5. Ketkar N., Moolayil J. Feed-Forward Neural Networks // Deep Learning with Python. Berkeley, CA: Apress, 2021. P. 93–131.
6. Rafisovich Gapsalamov A. et al. Approaches to Information Security in Educational Processes in the Context of Digitalization // TEM Journal. 2020. P. 708–715.
7. Kennedy J., Eberhart R.C. Particle swarm optimization // Proceedings of ICNN'95 — International Conference on Neural Networks. 1995. Vol. 4. P. 1942–1948 vol.4.
8. Van den Bergh F., Engelbrecht A.P. Cooperative learning in neural networks using particle swarm optimizers // South Afr. Comput. J. 2000. Vol. 26. P. 84–90.
9. Eberhart R.C., Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization // Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512). 2000. Vol. 1. P. 84–88 vol.1.
10. Liang J.J. et al. Particle swarm optimization algorithms with novel learning strategies // 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583). 2004. Vol. 4. P. 3659–3664 vol.4.
11. Li Z. et al. PMSM parameter identification based on improved PSO // Journal of Physics: Conference Series. 2021. Vol. 1754, № 1. P. 012235.
12. Venter G., Sobieszczanski-Sobieski J. Parallel Particle Swarm Optimization Algorithm Accelerated by Asynchronous Evaluations. Journal of Aerospace Computing Information and Communication — J AEROSP COMPUT INF COMMUN. 2006. Vol. 3, P. 123–137.
13. Koh B.I., George A.D., Haftka R.T., Fregly B.J. Parallel asynchronous particle swarm optimization // International journal for numerical methods in engineering. 2006. Vol. 67, № 4. P. 578–595.
14. Lalwani S., Sharma H., Satapathy S.C. A Survey on Parallel Particle Swarm Optimization Algorithms // Arabian Journal for Science and Engineering. 2019. Vol. 44, P. 2899–2923.
15. Shie M., Dori P., Reuven R. The cross entropy method for classification. // Proceedings of the 22nd international conference on Machine learning (ICML'05). Association for Computing Machinery. 2005. P. 561–568.

© Ларионов Вячеслав Сергеевич (larionov.vs@edu.spbstu.ru), Малеев Олег Геннадьевич (maleev_og@spbstu.ru).

Журнал «Современная наука: актуальные проблемы теории и практики»