Исследование методов настройки гиперпараметров нейросетей на высокопроизводительных кластерах с процессорами POWER8 [1]

К.Г. Хамитов, Н.Н. Попова

[1] МГУ им. М.В. Ломоносова

В настоящее время Искуственные нейронные сети(ИНС) обучаются с помощью различных адаптивных методов, использующих производные высших порядков. Помимо параметров обучения нейронные сети имеют большое количество гиперпараметров различных типов, относящихся как ко всей сети (выбор метода оптимизации, коэффициента пространственного масштабирования, коэффициента скорости обучения), так и к конкретному нейрону или слою (количество нейронов в слое, dropout rate, метод свёртки), которые сильно зависят от типов ячеек. При настройке моделей ИНС для решения конкретных задач или улучшения их качества при проведении экспериментов чаще всего применяются экспертные оценки и ручная настройка гиперпараметров. Разработка новых ячеек нейросетей также ведётся в основном аналитически вручную. Однако существуют и автоматические методы настройки гиперпараметров, позволяющие повысить различные метрики нейросетевой модели(время работы прямого хода, точность и пр.). Одним из подходов к автоматической настройке являются методы нейроэволюции в пространстве гиперпараметров ИНС. Одним из таких подходов является нейроэволюция через аугментацию топологий(NEAT) и её модификации для глубоких сетей(DeepNEAT, CoDeepNEAT). Другим популярным подходом является использование эволюции кодирующих сетей(CPPN, HyperNEAT). В данной работе рассмотрено применение вышеперечисленных подходов для эволюции ИНС для решения практических задач.

1. Введение

Для настройки гиперпараметров нейросетей могут применяться различные методы. Как правило, на сегодняшний день множество гиперпараметров, таких, как коэффициента скорости обучения, метод оптимизации при обучении, настраиваются эмпирически или на основе экспертных оценок, что позволяет достигать воспроизводимых результатов, но не всегда может представлять наиболее оптимальные конфигурации. Гиперпараметры, отвечающие за структуру сети, например, связанность слоёв, топология сети и другие, настраиваются на основе экспертных оценок и теоретических оценок применимости различных типов сетей для конкретных практических задач. В рамках адаптации сети под задачу происходит, как правило, приведенный выше экспертный выбор архитектуры и подстройка входных слоёв или комбинация сети с сетью кодирования с целью конвертации и адаптации входных данных под ожидаемый формат входа для выбранной архитектуры. В данной работе будет рассматриваться применение эволюционных методов для одновременной настройки всех типов гиперпараметров ИНС в задачах различного типа.

2. Обзор нейроэволюционных методов для настройки гиперпараметров

Нейроэволюционные методы, как следует из названия, основываются на применении эволюционных алгоритмов. В качестве гиперпараметров при этом выступают различные характеристики сети, такие, как веса связей, топология, методы оптимизации, количество

[1]Работа выполнена при частичной поддержке РФФИ (проект № 17-07-01562).

слоёв и др. Для оптимизации гиперпараметров, таких как веса, широко применяются эволюционные методы - генетические алгоритмы, ковариационная матричная стратегия эволюционной адаптации(CMA-ES), позволяющая решать задачи глобальной оптимизации, поскольку может учитывать взаимодействия между параметрами. Другие подходы - CoSyNE, ESP, SANE, позволяют применять эволюционные методы для эволюции блоков и слоёв сетей с целью дальнейшей их комбинации в полную ИНС. Другие методы, такие как NEAT и клеточное кодирование, применяются для эволюции топологий ИНС, с использованием которой можно значительно повысить эффективность сети.
Все вышеперечисленные технологии применяются для решения самых различных задач, но из-за большой размерности пространства гиперпараметров, их возможности ограничиваются только сетями прямого распространения с небольшим количеством слоёв. Для повышения производительности эволюционные методы могут быть совместимы с градиентными методами для настройки гиперпараметров глубоких нейросетей. В таких методах градиентные методы обычно используются для оптимизации сходимости поиска оптимальных гиперпараметров. Многие методы работают за счёт того, что обучение происходит только в отобранных объектах популяции (так называемый, Baldwin effect [1]).
Применение нейроэволюции для глубоких сетей немного отходит от описанных методов. В сетях, применяемых на практике, градиенты для всех слоёв доступны и вычислимы за адекватное время и применяются для оптимизации топологии. Глубокая нейроэволюция является многоуровневым эволюционным процессом. На разных уровнях оптимизируются гиперпараметры различных типов. Например, при двухуровневой нейроэволюции на верхнем уровне оптимизируется топология и связанность блоков или слоёв сети. На нижнем уровне оптимизируются сами блоки. Необходимость применения такого подхода вытекает естественным образом из постановок задач, для решения которых применяются глубокие сети. Для примера можно рассмотреть известную задачу контролирования манипулятора с помощью ИНС. Одним из подходов к решению этой задачи является нейроэволюция. Однако, если применять одноранговую нейроэволюцию, то получится, что восемь гиперпараметров (таких, как размер популяции, тип мутации, вероятность отбора) самой нейроэволюции, которая в свою очередь оптимизирует восемь нижележащих гиперпараметров сети, приходится оптимизировать вручную. Количество гиперпараметров нижнего уровня может быть увеличено до 15, что позволяет повысить качество решения задачи. На верхнем уровне нейроэволюции в рассматриваемой задаче применяются модификации NEAT и SANE. Оптимизация гиперпараметров сети проводится методами CMA-ES/GA.

3. Актуальность

Ранее подобные методы эволюции сетей были давно известны для сетей прямого распространения, но их применение ограничивалось лишь расширением методов эволюционного обучения ИНС и изначально были неприменимы для модификации глубоких нейронных сетей, лишь с развитием GPU-вычислений [3] удалось эффективно обобщить эволюционные методы, для оптимизации гиперпараметров больших размерностей с использованием модификации генетического алгоритма. В связи с развитием и широким распространениемибридных кластеров, узлы которых обладают процессорами на архитектуре IBM POWER и технологиями NVLink, которые теперь занимают лидирующие позиции списка Top-500 и актуальности задач адаптации сетей. В данной работе рассматривается применение методов настройки гиперпараметров нейронных сетей различных типов с помощью эволюционных алгоритмов для решения различных практических задач, таких как обучения с подкреплением и задаче улучшения топологии конкретных ячеек для конкретных данных(задачах адаптации сетей), оценено влияние типа связей между GPU и CPU в гибридных кластерах на скорость настройки гиперпараметрами и точности работы на задачах настроенных сетей.

3.1. Методы нейроэволюции

3.1.1. NEAT

NEAT - метод нейроэволюции посредством аугментации (расширения) топологии сетей [5]. Метод базируется на расширении идей клеточного кодирования. Основой метода является генетический алгоритм, в котором в качестве генотипа выступают последовательно закодированные связи нейронов. Параметрами связи являются признак активности связи, значение веса и уровень - так называемый номер "инновации"нейрона в сети. Номер используется для отслеживания истории эволюции и поиска соответствующих друг другу генов при скрещивании различных сетей с разным количеством нейронов. Метод использует два новых типа мутаций. Кроме гауссова зашумления для мутации вещественных параметров используется добавление связей между уже имеющимся нейронами и добавление промежуточных нейронов. Пример скрещивания сетей с различным количеством генов, представлен на Рис. [1]. При скрещивании вещественные гиперпараметры (веса), копируются из одного

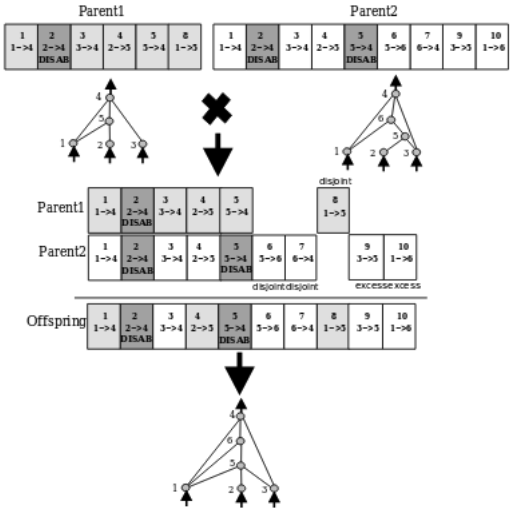

Рис. 1: Схема скрещивания сетей с различным количеством нейронов в методе NEAT

из родителей. Отбор производится по обучению сети на конкретной задаче с определенным числом эпох. Значением фитнеса является значение функции качества. При отборе при прочих равных отбирается сеть с наименьшим количеством нейронов в генотипе. В случае совпадения количества происходит случайный отбор. В [5] показана хорошая применимость NEAT для полносвязных сетей с небольшим количеством слоёв.

3.1.2. DeepNEAT

DeepNeat - это известное расширение NEAT для работы с глубокими ИНС. Также, как и NEAT, он является разновидностью генетического алгоритма. Главное отличие DeepNEAT от предшественника заключается в способе кодирования. Каждый ген в хромосоме кодирует не отдельный нейрон, а слой целиком со всеми гиперпараметрами слоя, что несколько ограничивает геном. Хромосома хранит в себе большое количество гиперпараметров различных типов, как целочисленных, так и вещественных. Для них применяются мутации вида random bit-flip. В DeepNEAT информация о весах связей не хранится. Вместо этого сохраняются структурные данные о том, как слои связаны друг с другом. Чтобы воссоздать информацию о полной сети, при вычислении фитнеса достаточно пройти по хромосоме и собрать полную топологию сети из блоков слоёв. При этом необходимо отметить, что если

разрешена произвольная связность между блоками слоёв,то необходимо применять слои, изменяющие размерность. Если повышение размерности можно выразить через умножение на единичный тензор нужной размерности, то процедура понижения сильно зависит от структуры исходных данных и задачи. В [3] отмечено, что хотя DeepNEAT и применим к современным глубоким сетям, таким как ResNET, полученные изменения чаще всего получаются переусложненными и неструктурированными, что затрудняет их анализ и их последующее применение и адаптацию для других задач.

3.1.3. CoDeepNEAT

Дальнейшая развитие идея многоуровневой нейроэволюции получила в методе CoDeepNEAT (Coevolution DeepNEAT). В данном подходе используется гибридная эволюция, в которой общая популяция состоит из двух непересекающихся групп. Первая группа отвечает за топологию сети (так называемый шаблон сети). Шаблон определяет блоки и слои, использующиеся в сети, и группу, отвечающую за сами слои или блоки, из которых состоит глубокая сеть. Сеть тоже представляется в виде графа нейронов, как в оригинальном NEAT. Во время подсчёта значения фитнес-функции шаблоны и блоки соединяются в большую глубокую сеть. В отличие от DeepNEAT значение фитнес-функции распространяется в популяции модулей и шаблонов как среднее среди всех сетей, содержащих данный модуль или построенных по данному шаблону. Приблизительная структурная схема CoDeepNEAT

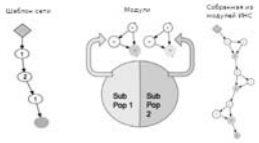

Рис. 2: Метод коэволюции, использу в CoDeepNEAT [3]

представлена на Рис. 2. Сходимость у CoDeepNEAT лучше, чем у своих предшественников, так как в этом методе имеется больше простора для мутаций, а малейшие мутации среди популяции шаблонов или модулей приводят к значительному изменению получаемых сетей. Операторы мутации используются, как в NEAT, для блоков, и DeepNEAT для шаблонов соответственно. Метод применим для рекуррентных сетей. Метод позволяет как улучшать сами ячейки, так и модифицировать их между различными элементами одной временной эпохи. В частности, в случае LSTM, сеть разворачивается на нужное количество итераций. Также вводятся две мутации: одна отвечает за создание новых рекуррентных связей между слоями, вторая аналогична процедуре dropout нейронов в свёрточных сетях. В различных поколениях шаблоны состоят из вариантов блоков LSTM.

3.1.4. HyperNEAT

Немного другой подход к нейроэволюции представлен в методе HyperNEAT [4]. Данный подход использует CPPN - сети, производящие сложные образцы, которые используются для кодирования архитектуры сетей по точкам в N-мерном пространстве. CPPN генерирует значение веса и других гиперпараметров по двум точкам N-мерного пространства ($CPPN(x_1, \ldots, x_n, y_1, \ldots, y_n) = w_{x,y}$). Таким образом, CPPN позволяет кодировать гиперпараметры для любых типов сетей. Конфигурация сети описывается гиперкубом размерности N, для которого и будет определяться топология сети методом перебора всех соседей по данному гиперкубу с помощью CPPN. Выбор гиперкуба, описывающего конфигурацию, очень важен. Он зависит от исходных данных задачи, так как позволяет выявлять явным образом геометрические признаки и использовать геометрическую локальность во входных данных для вычисления соответствующих признаков. Сами алгоритм можно описать

с помощью следующих шагов:

- Выбрать конфигурацию
- Инициализировать популяцию минимальными CPPN, сетями со случайными весами
- Для всех элементов популяции получить веса для любых возможных соединений внутри конфигурации
- Посчитать значение фитнеса на полученной архитектуре
- Выполнить шаги мутации скрещивания из NEAT для CPPN - популяции

. Для определения входов и выходов сети используются разметки в самих гиперкубах. Главное отличие в использовании CPPN от методов, используемых в DeepNEAT и NEAT, заключается в том, что CPPN не впрямую задает архитектуру сети, а обучается общему принципу соединимости элементов, что позволяет, масштабируя гиперкубы, получать сети различных размеров. В [4] показано применение такого метода для агентного моделирования поведения робота, собирающего ресурсы, для которого достаточно двумерных сеток размером 55x55. При этом отмечается высокая степень параллелизма метода.

4. Актуальность тематики

В связи с широкой применимостью ИНС, оптимизация гиперпараметров сетей в автоматическом режиме является важной задачей. Использование большого количества гиперпараметров,что приводит к необходимости использования высокопроизводительных кластеров. В данной работе рассматривается применение методов настройки гиперпараметров нейронных сетей различных типов с помощью эволюционных алгоритмов для решения различных практических задач, таких как обучения с подкреплением и задаче улучшения топологии конкретных ячеек для конкретных данных(задачах адаптации сетей) на кластере Polus процессорами IBM POWER8 и NVLink связями как между CPU-GPU так и между GPU-GPU. Данная архитектура узлов кластера, сходна с архитектурой кластеров, занимающих лидирующие позиции списка Top-500.

5. Задачи

Основной задачей данной работы было исследование методов нейроэволюции для решения задач адаптации нейронных сетей как с помощью оптимизации архитектуры, так и с использованием настройки слоевых/ячеечных гиперпараметров. В работе рассмотрена задача оптимизации архитектуры для решения задач обучения с подкреплением на различных разделах набора данных Atari©.

6. Метод исследования

Были исследованы методы нейроэволюции CoDeepNEAT и HyperNEAT на кластере Polus для решения задач адаптации нейронных сетей как с помощью оптимизации архитектуры, так и с использованием настройки слоевых/ячеечных гиперпараметров. В работе рассмотрена задача оптимизации архитектуры для решения задач обучения с подкреплением на различных разделах набора данных Atari© и задача оптимизация архитектуры LSTM сети предсказания следующего слова из контекста на наборе данных [2]. Для анализа ускорения процесса нейроэволюции были произведены серии запусков с определенными критериями, перечисленными далее.

6.1. Задача адаптации LSTM–сети на наборе данных Problem Report Corpus

Рассмотрим нейроэволюцию рекуррентных сетей с помощью метода CoDeepNEAT на наборе данных Problem Report Corpus [2] в задаче "Language Modelling заключающейся в предсказании следующего слова по контексту в текстовой информации. Задачи такого вида являются классическими задачами для рекуррентных сетей, в частности, для LSTM. Опишем выбранные параметры рассматриваемой задачи. Начальная популяция составляла 50 сетей с равномерно распределенными весами в интервале $[-0.05, 0.05]$. В каждом слое было 300 LSTM–ячеек в слое с числом разворота при обучении методом TBPTT 35. Количество эпох при вычислении фитнеса равнялось 10, со скоростью 0.8. Число поколений нейроэволюции до останова равнялось 25. Правилом останова служило число поколений равное 500. Для исследования масштабируемости параллельной нейроэволюции были проделаны эксперименты с распределённым обучением в процессе подсчёта фитнес-функции из PyTorch.

6.2. Задача адаптации сети для обучения с подкреплением на наборе данных Atari

Вторая задача адаптации сетей, рассмотренная в работе, – поиск оптимальной сети в задаче обучения с подкреплением на одном из популярных наборов данных Atari Games. Использовался не весь набор, а только его части asteroids и enduro. Нейроэволюция осуществлялась методом HyperNEAT со входной решёткой $128x128$. Количество эпох для обучения составляло 30. Вычисление фитнес-функции и скрещивание проводились аналогично методу CoDeepNEAT. Было рассчитано 200 поколений эволюционного алгоритма.

6.3. Практическая реализация

Практическая реализация вышеперечисленных методов была выполнена на фреймворке PyTorch, метод CoDeepNEAT был реализован полностью с использованием пакетов deep-neuroevolution и neat-python. Для реализации HyperNEAT была произведена адаптация реализации, представленой в [6] на фреймворк PyTorch и использование другого типа коммуникаций. Отключение NVLink для оценки его использования осуществлялось путем запрета Peer-to-peer коммуникаций в конфигурациях PyTorch, что также приводит к изменению количества и интенсивности вычислений на CPU. Параллельность обучения достигалась встроенными средствами пакета PyTorch, дополнительных шагов к увеличению степени параллелизма стандартных алгоритмов не проводилось. Для реализаций проанализировано ускорение параллельной реализации алгоритмов эволюционной оптимизации.

7. Вычислительный эксперимент

Вычислительноый эксперимент для решения вышеперечисленных задач проводился с использованием параллельных нейроэволюционных методов на моделях из материалов из статей [3] [6] с использованием фреймворка PyTorch. Эксперименты проводились на 4-ех вычислительных узлах кластера Polus, построенного на базе узлов с процессорами IBM POWER8®, 256GB RAM, и 2 NVIDIA Tesla® P100. Таким образом на один узел приходится 2 CPU POWER8 и 2 карты NVIDIA Tesla P100. В качестве шаблонной реализации нейроэволюции для задачи обучения с подкреплением на наборе Atari была взята референсная модель и выборка данных из [6]. В эксперименте замерялось время эволюции по достижению заданного количества эпох.

Результаты замеров времени CPU и GPU, а также полученные ускорение представлены Рис. 3, Рис. 4. Из рисунков видно, что метод нейроэволюции обладает хорошим ресурсом параллелизма, а также использование NVLink позволяет уменьшить не только время на

Таблица 1: Время работы алгоритма нейроэволюции в с.

Кол-во CPU	Atari asteroids NVLink	Atari asteroids	Atari:enduro NVLink	Atari enduro	LSTM Problem Report Corpus NVLink	LSTM Problem Report Corpus
1	29370	33551	14685	16775	22345	26752
2	18401	25713	9184	12856	13281	19003
4	10682	15726	5340	7863	7941	10843
8	8227	10207	4113	5103	5984	7265

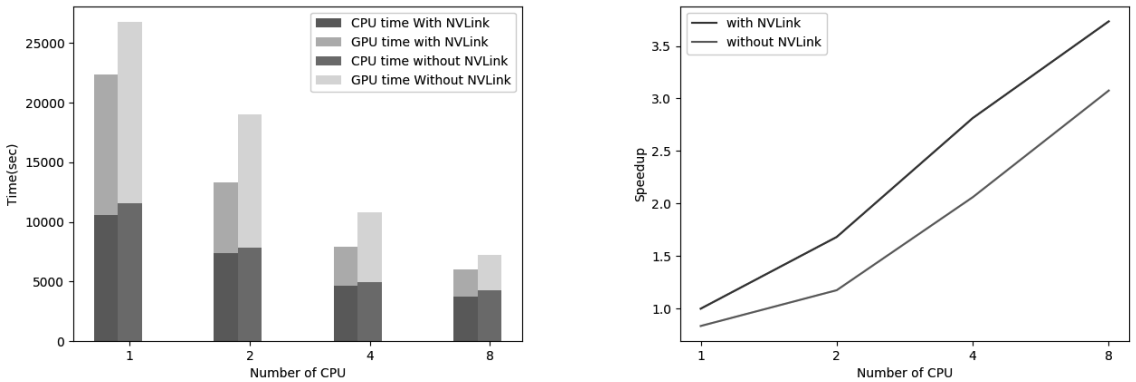

Рис. 3: Время работы и ускорение в задаче адаптации LSTM сети на данных Problem report data bank

GPU, но и на CPU. Для расчёта точности выполнялась эволюция с 10% количеством эпох и оценивалось точность работы на входных данных. Графики достигаемой на данном числе эпох точности представлены на Рис. 5

По результатам эксперемента можно сделать вывод о положительном влиянии использования технологии NVLink на точность работы сети в зависимости от номера эпохи на котором была получена ИНС. Результаты показывают повышение уровня точности, в зависимости от числа эпох, по сравнению с таковыми в [3]. Точность на сетях, полученных в результате эволюции на кластере с POWER8 превосходит более чем на 7% на задаче адаптации на наборе данных Problem Report Corpus и 9% и 6.5% на задаче обучения с подкреплением Atari соответственно, что подтверждает хорошую применимость гибридных кластеров на архитектуре POWER для задач нейроэволюции.

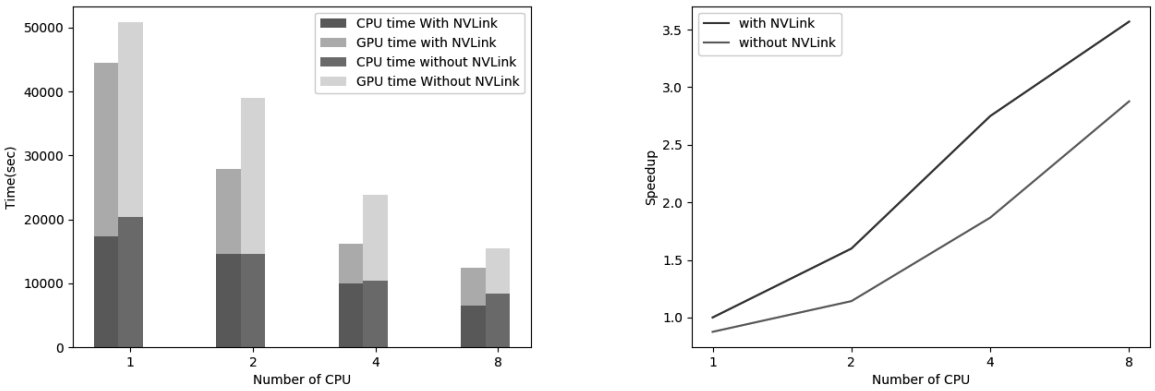

Рис. 4: Время работы и ускорение нейроэволюции в задаче обучения с подкреплением на наборе данных Atari

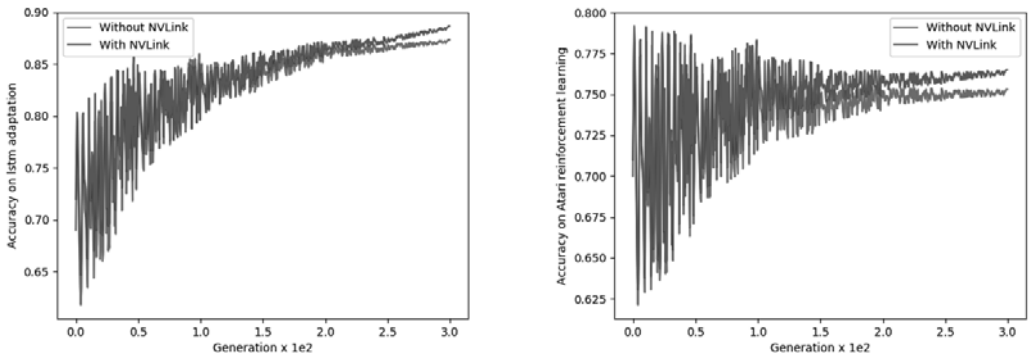

Рис. 5: Точность в задачах адаптации LSTM сети на данных Problem report data bank и обучения с подкреплением на "данных"Atari:asteroids

8. Выводы

Анализ полученных результатов позволяет сделать вывод о том, что рассмотренные методы нейроэволюции имеют хороший ресурс параллелизма в представленных задачах. Технология NVLink в гибридных кластерах на процессорах POWER8 ускоряет задачи нейроэволюции различных типов, а также косвенным образом повышает точность, что позволяет сделать вывод о хорошоей применимости гибридных кластеров данного типа для решения задач данного типа.
В дальнейшем планируется исследовать влияние выбора эволюционного алгоритма на ускорение и точность в задачах адаптации ИНС.

Литература

1. Hinton G., Nowlan S. How learning can guide evolution // Adaptive individuals in evolving populations: models and algorithms. 1996. Vol. 26. P. 447–454. ISBN: 0-201-48369-6

2. Andrew Ko. Problem Reports data.
URL: http://www.cs.cmu.edu/ marmalade/reports.html (accessed: 25.05.2019).

3. Miikkulainen R., Liang J., Meyerson E., et al. Evolving deep neural networks // Artificial Intelligence in the Age of Neural Networks and Brain Computing. Elseiver 2019.
P. 293–312. DOI: 10.1016/B978-0-12-815480-9.00015-3.

4. Stanley K.O., D'Ambrosio D.B., Gauci J. A hypercube-based encoding for evolving large-scale neural networks // Artificial life. MIT Press. 2009. Vol. 15 P. 185–212.
DOI: 10.1162/artl.2009.15.2.15202.

5. Stanley K.O., Miikkulainen R. Evolving neural networks through augmenting topologies // Evolutionary computation. MIT Press. 2002. Vol. 10 P.99–127.
DOI: 10.1162/artl.2009.15.2.15202.

6. Petroski S.F., Vashisht M., Edorado. C., et al. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning // arXiv preprint. arXiv: 1712.06567.