УДК 004.032.26

Использование адаптивных методов оптимизации при обучении нейронной сети для задачи генерации текста

А.В. Боднар, Я.Р. Парсаданян
Донецкий национальный технический университет
кафедра программной инженерии им. Л.П. Фельдмана
Email: linabykova13@ya.ru, ypars00@mail.ru

Аннотация
В статье рассматривается принцип работы сети для решения задачи генерации текста, а также способ ее реализации. Для этого изучена архитектура LSTM сетей, а для более продуктивного обучения обращено внимание на адаптивные методы обучения. В результате работы проведено сравнение обучения сети с применением адаптивных методов и без использования таковых, которое позволило сделать вывод, что использование адаптивных методов значительно улучшает работу нейронной сети на этапе обучения.

Введение

Актуальность применения нейронных сетей заключается в необходимости решения плохо формализованных задач. Потенциальные области применения искусственных нейронных сетей являются – те, где человек малоэффективен, а привычные расчеты трудоемки [1].

Актуальность выбранной темы связана с активным развитием данной технологии в сфере информационных технологий. Нейронные сети – перспективное направление и решение большого количества задач человека, в их числе и задача генерации текста.

Нейронные сети для решения такой задачи достаточно объемные и требуют затраты ресурсов. Поэтому, чтобы оптимизировать работу нейронной сети на этапе обучения (при этом не должна пострадать конечная производительность), рассмотрим и применим к разрабатываемой системе адаптивные методы обучения.

Общий принцип работы нейронной сети для задачи генерации текста

Для решения задачи моделирования языка нейронная сеть должна вычислить и проанализировать вероятность последовательности символов (w1, w2, … , wT) в языке L.

$$P(w_1, \ldots, w_T) = P(w_1, \ldots, w_{T-1})P(w_1, \ldots, w_{T-1}) = \prod_{t=1}^{T} P(w_1, \ldots, w_{T-1}). \quad (1)$$

Вычисление вероятности P (wt |w1, …, wt−1) для случайных t невозможно, поэтому используют приближение с помощью P (wt |wt−n, …, wt−1) — вероятности следующей информации, при длине предыдущего контекста n [3].

И всё же данное решение несет за собой проблему в виде большого объема памяти, занимаемой данными сети. Описанное приближение сводит всё к n-граммным моделям (частотные распределения всех цепочек длины n, встречаемых в обучающем тексте).

Использование рекурсивных нейронных сетей – более оптимальное решение задачи моделирования языка. Данный тип сети может выучить последовательности слов без непосредственного запоминания всех слов в контексте.

Относительно n-граммных моделей рекурсивные сети экономичнее используют память, однако проблема большого места остаётся значимой. Поэтому есть смысл попытки сжатия сети для более гибкого применения на различных устройствах (с разным уровнем аппаратного обеспечения) [3].

Описание принципа работы нейронной сети архитектуры LSTM

LSTM – подвид рекуррентных нейронных сетей. В данной модели учитываются входные данные, при этом принимается во внимание предыдущий выход. При “развертке” выбранной модели мы получим последовательность повторяющихся блоков нейронной сети (рис. 1). Блок состоит из четырех слоев.

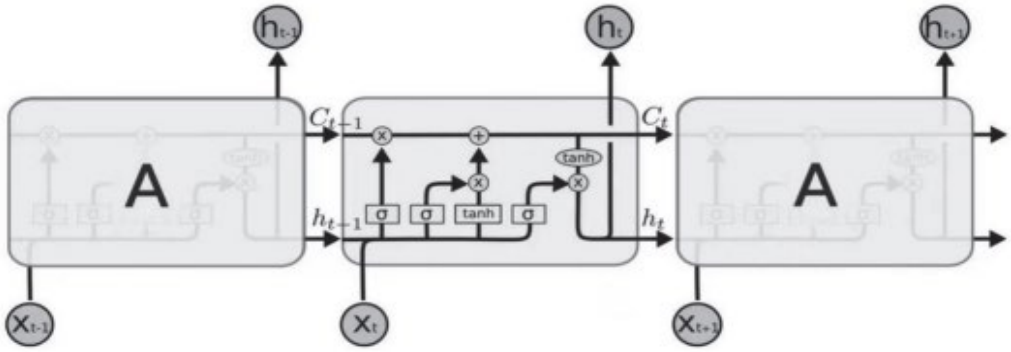

Рисунок 1 – Принцип работы и содержимое блоков LSTM [4]

На указанном рисунке:

A – блок;

C – состояние блока;

x – входные данные;

h – данные скрытого состояния (выходные данные);

σ – слой нейронной сети с сигмоидальный функцией активации;

X – поэлементное умножение;

+ – поэлементное сложение;

tanh – слой нейронной сети с гиперболическим тангенсом.

Основной элемент блока – состояние, которое по своей сути является конвейерной лентой, проходящей через всю цепочку [2]. Именно эта часть блока позволяет сохранять важный контент на протяжении работы сети, добавляя или удаляя из нее данные (см. рис. 2). Блок с поэлементным умножением отвечает за, так называемое, забвение данных, блок с поэлементным сложением – за запоминание новой важной информации.

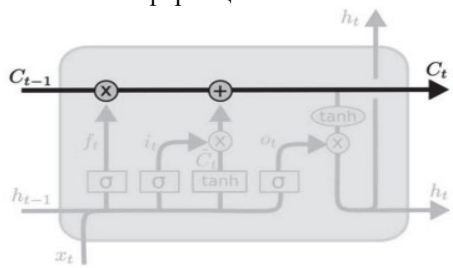

Рисунок 2 – Основная структура состояния блоков LSTM [4]

Сеть LSTM реализована таким образом, который позволяет удалять информацию из состояния блока. Настоящий процесс регулируется фильтрами. Фильтры дают возможность пропускать информацию в зависимости от определенных условий.

Фильтры в целом позволяют защищать и контролировать состояние блока. Так сигмоидальный слой возвращает значение от 0 до 1. Это показатель того, какую информацию запускать на новую итерацию, а какую исключить. Отметим, что значение 0 говорит о том, что никакую информацию сеть не пропускает, 1 – пропускает всё [5].

На рисунке 3 изображен процесс забывания информации. На данном этапе работы блок объединяет входные данные и данные скрытого состояния.

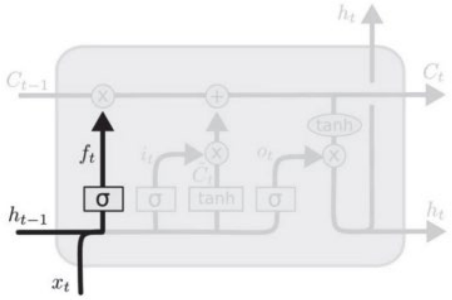

Рисунок 3 – Процесс выявления забываемых значений [4]

Объединенный вектор подается на вход сигмоидального полносвязного слоя, на выходе формируются значения f, размерность которого совпадает с размерностью вектора состояния на предыдущем шаге. После поэлементного умножения, в зависимости от значений вектора f, блок понимает какие данные необходимо забыть. По аналогии происходит обновление контента (см. рис. 4). Настоящий шаг включает в себя две части.

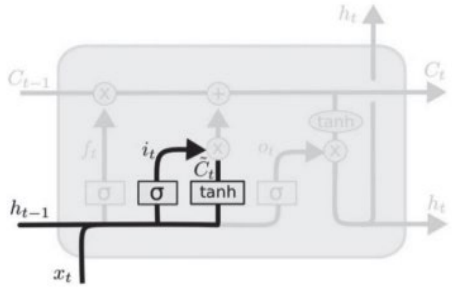

Рисунок 4 - Процедура добавления новой информации [4]

Происходит объединение входных данных и данные скрытого состояния. Объединенный вектор подается на вход сигмоидального полносвязного слоя. Этот же

вектор подается на слой нейронной сети с гиперболическим тангенсом. Полученные на двух этапах векторы поэлементно умножаются, результат этой операции поэлементно складывается с вектором состояния.

В результате получаем обновленный вектор состояния, в котором после описанных действий удалено ненужное, сохранено важное (см. рис. 5, [4]).

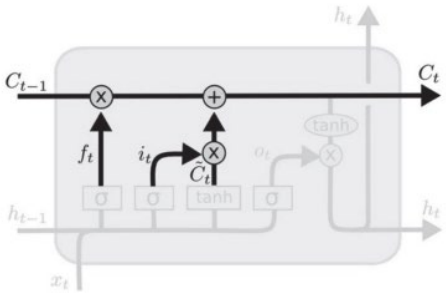

Рисунок 5 – Формирование вектора состояния

Финальный этап работы блока – формирование скрытого выходного вектора состояния (рис. 6). Для его формирования вычисляем оценочный вектор. Вектор состояния пропускаем через функцию гиперболического тангенса и поэлементно умножаем на оценочный.

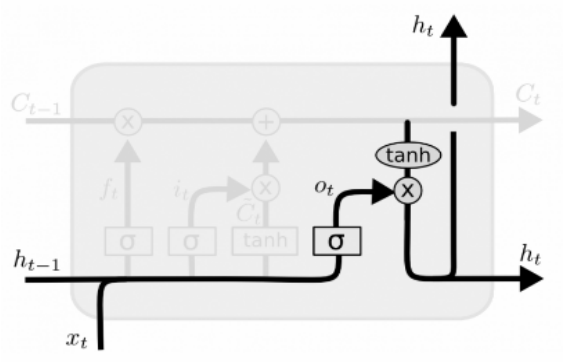

Рисунок 6 - Процедура добавления новой информации [5]

Реализация нейронной сети для решения задачи генерации текста на языке программирования C#

Проанализировав архитектуру LSTM-сети, можем сделать вывод о необходимых составляющих системы.

Проект состоит из нескольких ключевых классов (рис. 7). Каждый класс выполняет специфические функции, необходимые для работы сети LSTM. Основными классами являются LSTMRNN, Block, Gate и LinearRegressionLayer.

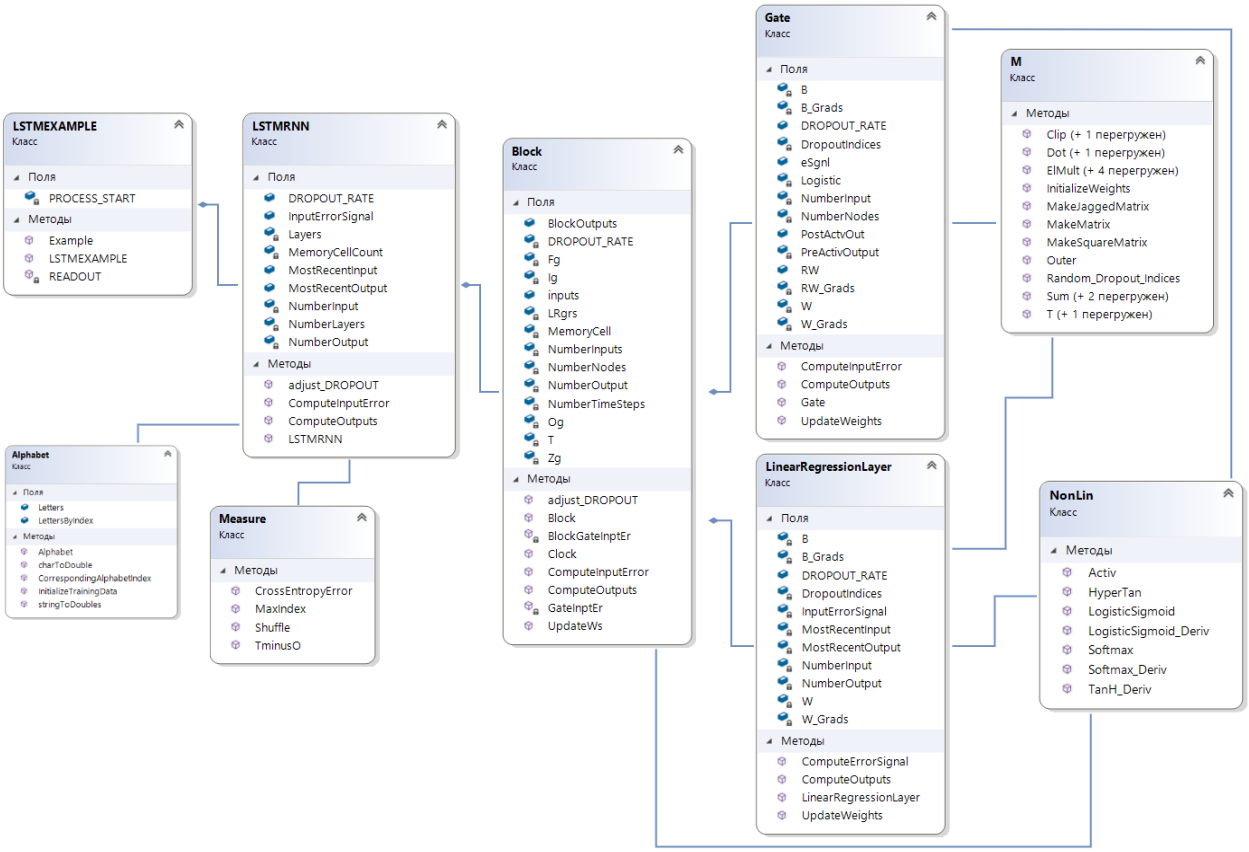

Рисунок 7 – Диаграмма классов сети

Эти классы взаимодействуют между собой для создания и обучения многослойной нейронной сети, способной решать задачи, связанные с генерацией текста (см. рис. 7). Класс Block представляет собой основную структуру LSTM-блока, который содержит несколько фильтров (gate) и слой линейной регрессии для вычисления выходных данных. Этот блок выполняет все основные операции, необходимые для вычисления выходных данных и обновления весов во время обучения.

Класс Gate представляет собой ворота (фильтры) LSTM-блока, которые выполняют операции, необходимые для управления потоком данных и состояния памяти внутри LSTM-блока.

Класс LinearRegressionLayer представляет собой слой линейной регрессии, который используется для преобразования выходных данных LSTM-блока в окончательные предсказания.

Классы Block, Gate и LinearRegressionLayer представляют собой основные строительные блоки для реализации LSTM-сети. Они обеспечивают функциональность, необходимую для управления состоянием памяти, вычисления выходных данных, обучения модели и обновления весов. Эти классы агрегируются в классе LSTMRNN, который в свою очередь агрегируются в LSTMEXAMPLE.

Класс LSTMRNN представляет собой реализацию нейронной сети LSTM. Он содержит методы для инициализации, обучения и использования нейронной сети LSTM.

Класс LSTMEXAMPLE представляет пример использования модели LSTM. Он содержит методы для обучения и использования нейронной сети LSTM.

Также для выполнения вспомогательных операций и расчетов реализованы классы M, NonLin, Alphabet, Measure. Класс M отвечает за выполнение различных математических операций, таких как сложение, умножение, транспонирование матриц, он предоставляет удобные методы для работы с числами и массивами. В классе NonLin реализованы различные нелинейные функции активации, такие как сигмоидальная функция и гиперболический тангенс. Эти функции применяются в нейронных сетях для добавления нелинейности и расширения возможностей модели. Класс Alphabet предназначен для обработки текстовых данных. Он содержит методы для представления текстовой информации в виде числовых векторов, а также для работы с алфавитом и символами. Класс Measure занимается измерениями и вычислениями, в основном используется для различных операций и вычислений внутри нейронной сети, таких как вычисление ошибки и других метрик.

Адаптивные методы обучения нейронной сети

Адаптивные методы обучения представляют собой семейство оптимизационных алгоритмов, которые динамически изменяют скорость обучения в процессе обучения модели. Эти методы часто улучшают сходимость и эффективность обучения нейронных сетей, обеспечивая более быстрое и стабильное обновление весов [6].

AdaGrad (Adaptive Gradient Algorithm) – это адаптивный метод оптимизации, который изменяет скорость обучения для каждого параметра на основе накопленных исторических градиентов.

Основная идея метода заключается в том, чтобы адаптировать скорость обучения для каждого параметра отдельно. Параметры, которые редко обновляются, получают более высокий темп обучения, в то время как часто обновляемые параметры получают более низкий темп. Это достигается за счет учета накопленной суммы квадратов градиентов для каждого параметра [7]. Способ вычисления представлен в формулах 3 – 4. Для каждой итерации t необходимо вычислить градиент (формула 2), обновить накопленный градиент (формула 3), обновить параметры (формула 4).

$$g_t = \nabla_{t-1} f(O_{t-1}), \quad (2)$$

$$G_t = G_{t-1} + g_t * g_t, \quad (3)$$

$$O_t = O_{t-1} - \frac{n}{\sqrt{G_t + eps}} * g_t. \quad (4)$$

Преимуществом данного метода является подстраиваемая для каждого значения скорость обучения, что помогает более эффективно обновлять коэффициенты модели. Также за счет уменьшения скорости обучения для часто обновляемых параметров, AdaGrad помогает избежать больших колебаний и переобучения модели. Однако есть и минус у описанного метода в части затухания скорости обучения. Дело в том, что накопленный градиент постоянно растет, что приводит к уменьшению скорости обучения до очень низких значений и остановке обучения в целом.

RMSprop (Root Mean Square Propagation) – это метод оптимизации, используемый для обучения нейронных сетей. Этот метод является вариацией стохастического градиентного спуска, который адаптирует скорость обучения для каждого параметра на основе истории градиентов. RMSprop является модификацией Adagrad, которая включает экспоненциальное

сглаживание градиентов [8]. Это помогает избежать затухания скорости обучения и сохраняет преимущества Adagrad (алгоритм вычисления представлен на формулах 5 – 6) [9].

$$v_t = \beta * v_{t-1} + (1 - \beta)(\nabla w_t)^2, \quad (5)$$

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_t + eps}}(\nabla w_t). \quad (6)$$

Несмотря на явные преимущества, метод RMSprop имеет некоторые недостатки. Необходимо тщательно подбирать параметры метода, такие как скорость обучения и коэффициент сглаживания, чтобы обеспечить эффективное обучение сети. В некоторых случаях метод может столкнуться с проблемой потери градиента, когда скорость обучения слишком мала или слишком велика.

Метод Adam (Adaptive Moment Estimation) комбинирует идеи Adagrad и RMSprop, отслеживая как средние значения, так и дисперсии градиентов. Свое название алгоритм получил ввиду оценки первого момента (среднюю величину) и второго момента (дисперсию) [10].

ADAM автоматически адаптирует скорость обучения для каждого параметра на основе оценки второго момента градиента. В методе используются моменты первого и второго порядка для ускорения сходимости и стабилизации обучения. В начале обучения оценки градиента и квадрата градиента могут быть сильно смещены к нулю, для борьбы с этим ADAM применяет коррекцию смещения, чтобы сделать оценки более точными.

Формулы обновления параметров в методе ADAM выглядят следующим образом (см. формулы 7 – 11).

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t, \quad (7)$$

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2, \quad (8)$$

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad (9)$$

$$\widehat{v}_t = \frac{v_t}{1-\beta_2^t}, \quad (10)$$

$$O_{t+1} = O_t - n\frac{\widehat{m}_t}{\sqrt{\widehat{v}_t + eps}}. \quad (11)$$

где m_t и v_t – оценки первого и второго моментов соответственно;

β_1 и β_2 – коэффициенты затухания для оценки моментов;

n – скорость обучения;

g_t– текущий градиент;

eps – небольшое число для численной стабильности;

$\widehat{m}_t$ и $\widehat{v}_t$ – исправленные оценки моментов;

O_t и O_{t+1} – параметры на шагах t и t+1 соответственно.

Коэффициенты затухания для оценки моментов и значение численной стабильности разработчик задает самостоятельно. Проанализировав статьи, приходим к выводу, что оптимальные значения параметров 0.9, 0.999 и 10^{-8}соответственно [9, 10, 11]. Именно такие параметры зададим при реализации метода в нашем проекте.

Реализация метода ADAM, анализ работы измененной нейронной сети

Для реализации метода оптимизации Adam нужно внести изменение в классах Gate, Block, LSTMRNN, а именно функции обновления весов и добавить вспомогательные переменные для хранения моментов (m) и скоростей (v) для весов и смещений (bias).

Проанализируем процесс обучения нейронной сети до применения адаптивных методов. Чтобы обучить сеть, нужны данные, которые могут быть обработаны и понятны модели. В нашем случае используем текстовые данные, а именно строки из стихотворений А.С. Пушкина. Отметим, что текстовые данные – набор символов.

Принцип обучения следующий – текст разбивается на небольшие блоки, где входом является фрагмент текста, а выходом – следующий символ. Таким образом, нейросеть получает на входе фрагменты текста, а на выходе символы, которые она должна сформировать. Например: вход: "совершенствование" выход: "р", вход: "овершенствование р" выход: "а", вход: "вершенствование ра" выход: "б" и так далее.

Для обучения зададим 500 эпох. Посмотрим сгенерированный текст на 10, 350 и 500 итерациях (см. табл. 1). Результат обучения нейронной сети виден невооруженным взглядом. Конечно, нейронная сеть не всегда корректно соотносит слова в контексте и всё же допускает грамматические или механические ошибки. Однако на 500 эпохе мы видим вполне разборчивый текст.

После реализации метода ADAM и внесения изменений в соответствующие структуры. Переобучим нейронную сеть с помощью того же обучающего множества, по тому же принципу, как и не модернизированную LSTM сеть.

Для обучения зададим 500 эпох. Посмотрим сгенерированный текст на 10, 350 и 500 итерациях (см. табл. 2).

Таблица 1 – Результат работы нейронной сети

Входные данные	10 эпоха	350 эпоха	500 эпоха
Ясный день	Ясный день кк ноо т уечч д уууббр злелеее нй дйцццы ъъеподдд онм нч пееенн ь дуе то ии дн коо	Ясный день коот уеччный дубр зелеоный децыцы поддд онм нч пень дубе то ии дн кот учеенный дц злон дб а	Ясный день кот ученыый дуб зеленный девицы у окном ночью песни дубе то м и днем и дуб зеленый кот ходит
Друг чудесный	Друг чудесныйдрррргаа ьяяя зннны ывтво пафвв пд луулл ой а нч впввмуб иии идн ииие отые ыы а в лн р	Друг чудесный дрг злеенннный кт двцы пд луой нч пн дууббуб тм и дн виииe каооты а в лн разз адцц пд ок	Друг чудесный друг злеваль каак девиц под лунный ночь у дуб там день и ночью видеение кар под окном царь с

Таблица 2 – Результат работы сети после переобучении с помощью метода ADAM

Входные данные	10 эпоха	350 эпоха	500 эпоха
Ясный день	Ясный деньльтнтьнии д нляясс снтььян дт н нясля д нннтн ляссс н я а о дн какпр крррад оо кттт чйй	Ясный день дно оч чудное мгнова поздно вечер ночьь пееснился сонн была цариейй мимолетной краaасот учеенный	Ясный день чудесныйй явилась ты как друг верный ккрасты а гониимый золотая цепь на дубе томиился хлеб по о
Друг чудесный	Друг чудесныйй кп рк гуркгу луул ллукаа пр ср чъг гг выф апрт о оооо ввщл ды а ч впмм миикполл ок ккн и хоо ид	Друг чудесный еще ии днем пооздно вечркоо м мооpоз на дубе томящ сон внемлет и о поет голос злотая цепь на	Друг чудесный ден ни печали девицы под оконья вырос рос я быылины расскажу о гений чистой совести моей судьб

Как видим, модернизированная сеть обучилась быстрее. Стоит отметить, что время самой итерации во втором случае не сократилось ввиду дополнительных вычилений в алгоритме. Однако мы выигрываем время за счет получения удовлетворяющего результата на более ранних эпохах. Как результат нейронная сеть с адаптивным методом ADAM обучилась быстрее, конечный результат – сгенерированный текст – в качестве в части смысловой нагрузки и корректности не потерял.

Вывод

В статье рассмотрена нейронная сеть архитектуры LSTM, которая характерна для решения задач генерации текста: изучена архитектура LSTM RNN, в рамках настоящей работы представлена диаграмма классов разработанной нейронной сети.

Ввиду необходимости сокращения времени обучения была осуществлена попытка применения адаптивного метода обучения к разработанной сети. Адаптивные методы обучения, такие как Adam, могут значительно улучшить процесс обучения нейронной сети, ускоряя сходимость и улучшая стабильность. Реализация Adam требует дополнительных вспомогательных переменных для моментов и скоростей, а также корректировки процесса обновления весов.

В ходе исследования проведены эксперименты на сетях с применением адаптивного метода и без. Сравнительный анализ результата работы LSTM сетей оказался положительным. Использование адаптивных методов значительно улучшает работу нейронной сети на этапе обучения.

Литература

1. Парсаданян, Я. Р. Теоретический анализ принципа работы нейронных сетей / Я. Р. Парсаданян, А. В. Боднар // Программная инженерия: методы и технологии разработки информационно-вычислительных систем (ПИИВС-2022): сборник научных трудов IV научно-практической конференции (студенческая секция), 29-30 ноября 2022 г. – Донецк, ДонНТУ, 2022. - Том 2.

2. Грачёв, А. М. Методы сжатия рекуррентных нейронных сетей для задач

обработки естественного языка // Национальный исследовательский институт «Высшая школа экономики» - Москва, 2019.

3. Парсаданян, Я. Р. Теоретический анализ методов сжатия рекурсивных нейронных сетей и их практических результатов / Я. Р. Парсаданян, А. В. Боднар // Современные информационные технологии в образовании и научных исследованиях (СИТОНИ-2023) : сб. материалов VIII Всерос. науч.-техн. конф., г. Донецк, 29 ноября 2023 г. / отв. ред. В.Н. Павлыш. – Донецк : ДонНТУ, 2023.

4. Хабиб, Ж. М. Т. Сравнение методов анализа настроений глубокого обучения, включая LSTM и машинное обучение / Ж. М. Т. Хабиб, А. А. Погуда // Открытое образование. – 2023. – Т. 27. - № 4. – С. 60-71. – DOI 10.21686/1818-4243-2023-4-60-71. – EDN QERCPQ.

5. Зоткина, А. А. Решение проблем рекуррентной нейронной сети при помощи модели "долговременной кратковременной памяти" / А. А. Зоткина, Н. С. Шиндина // Современные информационные технологии. – 2023. – № 37(37). – С. 18-20. – EDN BMPDGV.

6. Khafaga D. S. Improved Prediction of Metamaterial Antenna Bandwidth Using Adaptive Optimization of LSTM / Doaa Sami Khafaga, AmelAliAlhussan, El-SayedM.El-kenawy, Abdelhameed Ibrahim, Said H. Abd Elkhalik, Shady Y. El-Mashad, Abdelaziz A. Abdelhamid // Computers, Materials & Continua, 2022. – 73(1). - P. 865-881.

7. Строим градиентные алгоритмы оптимизации Adam, RMSProp, Adagrad, Adadelta [Электронный ресурс]. – Режим доступа: https://proproprogs.ru/tensorflow/tf-stroim-gradientnye-algoritmy-optimizacii-adam-rmsprop-adagrad-adadelta

8. Британов, В. С. Обзор и сравнение методов оптимизации применяемых в машинном обучении / В. С. Британов, А. И. Мартышкин, Е. А. Данилов // Тенденции развития науки и образования. – 2023. – № 97-12. – С. 45-49. – DOI 10.18411/trnio-05-2023-655. – EDN WPBHBM.

9. How using adaptive methods can help your network perform better [Электронный ресурс]. – Режим доступа: https://medium.com/bedrockdbd/how-using-adaptive-methods -can-help-your-network-perform-better-bcdd36b9214e

10. Пелин, В. О. Исследовательский анализ алгоритмов оптимизации искусственных нейронных сетей для задач прогнозирования / В. О. Пелин, Н. А. Акпаралиев // Сборник трудов VII Конгресса молодых ученых, Санкт-Петербург, 17–20 апреля 2018 года. – Санкт-Петербург: Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, 2018. – Т. 1. - С. 164-167. – EDN NEUUOE.

11. Kingma, D. P. Adam: a Method for Stochastic Optimization / D. P. Kingma, J. L. Ba // International Conference on Learning Representations. – 2016. – V. 53. – P. 1–13.

Парсаданян Я.Р., Боднар А.В. Использование адаптивных методов оптимизации при обучении нейронной сети для задачи генерации текста. *В статье рассматривается принцип работы сети для решения задачи генерации текста, а также способ ее реализации. Для этого изучена архитектура LSTM сетей, а для более продуктивного обучения обращено внимание на адаптивные методы обучения. В результате работы проведено сравнение обучения сети с применением адаптивных методов и без использования таковых, которое позволило сделать вывод, что использование адаптивных методов значительно улучшает работу нейронной сети на этапе обучения.*

Ключевые слова: *нейронная сеть, LSTM, генерация текста, обучение, адаптивные методы, скорость обучения ADAM.*

Parsadanyan Y.R., Bodnar A.V. The use of adaptive optimization methods in training a neural network for the task of text generation. *The article discusses the principle of the network for solving the problem of text generation, as well as the method of its implementation. To do this, the architecture of LTE networks has been studied, and for more productive learning, attention has been paid to adaptive learning methods. As a result of the work, we obtained a comparison of network training using adaptive methods without using them.*

Keywords: *neural network, LSTM, text generation, learning, adaptive methods, learning rate ADAM*

Статья поступила в редакцию 01.65.2024
Рекомендована к публикации профессором Зори С. А.