УДК 004.032.26

СРАВНЕНИЕ РАЗЛИЧНЫХ ОПТИМИЗАТОРОВ НА ДАТАСЕТЕ MNIST И ЕГО ВАРИАЦИЯХ ПРИ ИСПОЛЬЗОВАНИИ ФУНКЦИИ LEAKYRELU

Монтик Николай Сергеевич
ст. преподаватель, аспирант
Научный руководитель: **Головко Владимир Адамович**
д. т. н., профессор, зав. кафедрой
УО «Брестский государственный технический университет»

Аннотация: В статье рассмотрено сравнение различных оптимизаторов при функции активации LeakyReLU. Правильный выбор алгоритма оптимизации имеет большое значение для точности классификации и скорости обучения нейронной сети. В качестве датасета для проверки будет использоваться классический датасет MNIST, а также его вариации: KMNIST и FashionMNIST.

Ключевые слова: нейронная сеть, LeakyReLU, MNIST, задача классификации, алгоритмы оптимизации, Adadelta, Adam, Adamax, AdamW, NAdam, RAdam.

COMPARISON OF DIFFERENT OPTIMIZERS ON MNIST DATASET AND ITS VARIATIONS USING LEAKYRELU FUNCTION

Montik Nikolay Sergeevich
Scientific adviser: **Golovko Vladimir Adamovich**

Abstract: The paper deals with the comparison of different optimizers using LeakyReLU activation function. The correct choice of the optimization algorithm is important for the classification accuracy and training speed of the neural network. The classic MNIST dataset and its variations: KMNIST and FashionMNIST will be used as validation datasets.

Key words: neural network, LeakyReLU, MNIST, classification problem, optimization algorithms, Adadelta, Adam, Adamax, AdamW, NAdam, RAdam.

MNIST — это один из наиболее широко используемых наборов данных в машинном обучении и компьютерном зрении. Он состоит из 70 000 изображений рукописных цифр от 0 до 9, написанных от руки. Набор данных разделен на обучающий набор (60 000 изображений) и тестовый набор (10 000 изображений). Каждое изображение представляет собой черно-белое изображение размером 28x28 пикселей. MNIST широко используется для оценки и сравнения производительности алгоритмов классификации и распознавания образов, особенно в начальных этапах обучения и исследований в области машинного обучения.

KMNIST — это набор данных, аналогичный MNIST, но содержащий японские рукописные символы вместо цифр. "K" в KMNIST означает "Kuzushiji" (кудзуси), что является старым японским письмом. Как и MNIST, KMNIST содержит 70 000 изображений рукописных символов, разделенных на обучающий и тестовый наборы по 60 000 и 10 000 изображений соответственно. Этот датасет был создан для того, чтобы помочь в развитии и оценке алгоритмов распознавания рукописных символов для японского письма.

FashionMNIST — это альтернативный набор данных для тестирования алгоритмов машинного обучения и нейронных сетей. Он аналогичен по структуре наборам данных MNIST и KMNIST, но вместо рукописных цифр или символов японской письменности содержит изображения различных предметов одежды и аксессуаров.

FashionMNIST включает в себя 10 классов, каждый из которых представляет собой определенный тип одежды или аксессуара, такие как футболка, платье, кроссовки и т. д.

FashionMNIST часто используется в качестве альтернативы MNIST для проверки и сравнения алгоритмов классификации изображений, так как он предоставляет более сложные и разнообразные данные по сравнению с набором данных MNIST.

Функция активации нейрона — это математическая функция, которая определяет выходное значение нейрона на основе входных данных. Она добавляет нелинейность в вычисления нейрона и помогает нейронной обобщать закономерности в данных. Функции активации обычно применяются к сумме взвешенных входов нейрона и определяют, активируется ли нейрон и какое значение он выдаст.

В данной статье будет использоваться функция активации LeakyReLU. Leaky ReLU (Leaky Rectified Linear Unit) — это вариант функции активации

ReLU (Rectified Linear Unit), который позволяет небольшой, но постоянный градиент для отрицательных входов.

Графически функция LeakyReLU представлена на рисунке 1.

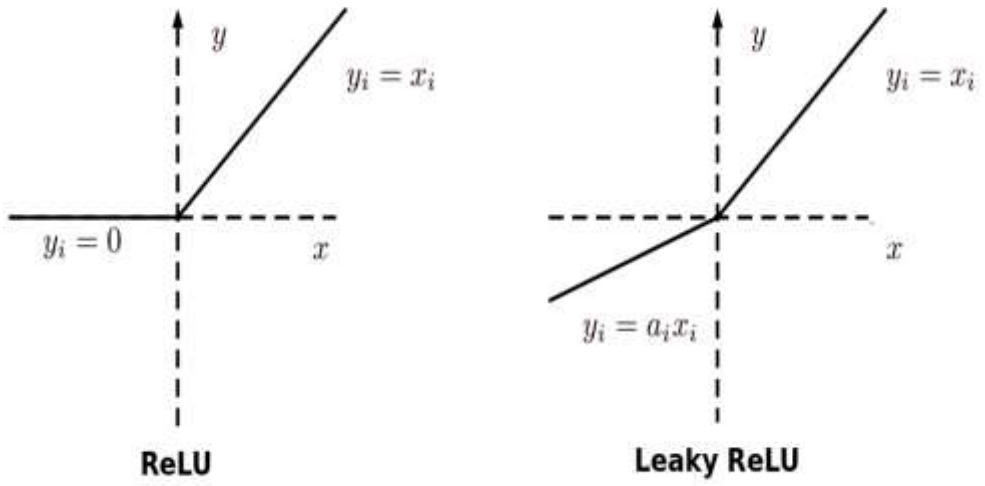

Рис. 1. Графическое изображение функции ReLU и LeakyReLU

Функция потерь в нейронной сети — это функция, которая измеряет расхождение между предсказанными значениями модели и фактическими значениями на обучающем наборе данных. Она является ключевым компонентом в процессе обучения нейронной сети.

Функция потерь позволяет алгоритму оптимизации корректировать параметры модели таким образом, чтобы улучшить качество прогнозов модели. В данной работе будет использоваться функция потерь MSE (Mean Squared Error).

Алгоритм оптимизации нейронной сети — это алгоритм, который используется для обновления параметров нейронной сети в процессе обучения с целью минимизации функции потерь.

В процессе обучения нейронной сети алгоритм оптимизации адаптирует веса сети на основе градиента функции потерь по параметрам сети. Он использует этот градиент, чтобы определить направление, в котором следует обновить веса с тем, чтобы минимизировать значение функции потерь.

В данной статье будут использоваться следующие алгоритмы оптимизации Adadelta, Adan и его модификации: Adamax, AdamW, NAdam, RAdam.

Алгоритм оптимизации Adadelta — это метод стохастической оптимизации, разработанный для обучения нейронных сетей. Этот алгоритм является вариацией метода адаптивного градиентного спуска (AdaGrad).

Adadelta решает две проблемы, с которыми сталкиваются AdaGrad: затухание скорости обучения и ручная настройка скорости обучения.

Adam (Adaptive Moment Estimation) — это адаптивный метод оптимизации, который комбинирует два ключевых аспекта: адаптивное изменение скорости обучения и адаптивное изменение момента градиента. Adam был предложен в 2014 году и стал одним из наиболее популярных алгоритмов оптимизации для обучения нейронных сетей.

Основная идея Adam заключается в том, чтобы адаптировать скорость обучения для каждого параметра независимо на основе двух оценок первого и второго моментов градиента. Это позволяет Adam эффективно обновлять веса модели в разных направлениях и на разных скоростях, что ускоряет процесс обучения и повышает его стабильность.

Adamax — это вариация алгоритма оптимизации Adam. Основное отличие Adamax от Adam заключается в том, что Adamax использует бесконечную вместо второго момента градиента.

Алгоритм оптимизации AdamW является вариантом алгоритма оптимизации Adam, который сочетает в себе метод стохастической оптимизации с адаптивным изменением скорости обучения и моментом.

Основное отличие AdamW от Adam заключается в том, что AdamW добавляет регуляризацию весов непосредственно в процесс оптимизации, тогда как Adam обычно применяет регуляризацию отдельно. Регуляризация весов предназначена для предотвращения переобучения модели и улучшения её обобщающей способности.

NAdam (Nesterov-accelerated Adaptive Moment Estimation) — это вариант алгоритма оптимизации, который объединяет два ключевых метода: метод ускоренного градиентного спуска Нестерова и алгоритм Adam (Adaptive Moment Estimation).

Основная идея NAdam заключается в том, чтобы использовать адаптивную оценку моментов для адаптивного изменения скорости обучения и добавить метод Нестерова для ускоренного градиентного спуска.

Преимущества NAdam включают в себя эффективное использование адаптивного изменения скорости обучения, а также улучшенную сходимость благодаря методу Нестерова.

RAdam (Rectified Adam) - это адаптивный алгоритм оптимизации, который представляет собой вариант оптимизации Adam с дополнительным механизмом, который решает проблему начальной нестабильности, связанную с алгоритмом Adam.

Проблема, с которой сталкиваются некоторые реализации Adam, заключается в том, что в начале обучения скорость обновления весов может быть слишком высокой из-за высоких значений первых моментов (момента градиента). Это может привести к нестабильности и плохому качеству обучения в начальных эпохах.

RAdam вводит дополнительный механизм, который регулирует скорость обновления весов в начале обучения. Этот механизм позволяет динамически корректировать скорость обновления весов в зависимости от статистики градиентов и позволяет обеспечить более стабильное и эффективное обучение.

Остальные оптимизаторы не будут сравниваться, т.к. продемонстрировали слишком низкую точность относительно вышеописанных алгоритмов.

Архитектура нейронной сети взята из [1] и является следующей: 28x28(784)-500-500-2000-10 нейронов в каждом слое, сеть полносвязная, на слоях используется функция активации LeakyReLU. Размер батча – 500, коэффициент для функции LeakyReLU – 0.025.

Для реализации поставленной задачи будет использоваться язык программирования python, библиотека pytorch. Инициализация весов стандартная, все алгоритмы оптимизации также используют стандартные параметры, для сида было установлено значение 68690064390100. В качестве значения точности выбирается максимальное значение за 100 эпох.

Результаты сравнения представлены в таблице 1.

Таблица 1

Сравнение алгоритмов оптимизации

Датасет	Точность, %					
	Adadelta	**Adam**	**Adamax**	**AdamW**	**NAdam**	**RAdam**
MNIST	98,63	98,4	98,49	98,36	98,23	98,71
KMNIST	91,38	92,74	93,1	92,58	92,41	92,85
FashionMNIST	89,81	89,65	89,98	89,7	89,23	90,04
Среднее	93,273	93,597	93,857	93,547	93,29	93,867
Максимум	98,63	98,4	98,49	98,36	98,23	98,71

Как видно из сравнения, наибольшая средняя и максимальная точность классификации на 2/3 датасетах была продемонстрирована оптимизатором RAadam. Исходя из этого, можно сделать вывод, что наибольшую точность для данной архитектуры демонстрирует оптимизатор RAdam, и его следует использовать для сетей схожих архитектур.

Список литературы

1. A fast learning algorithm for deep belief nets / Hinton G. E., Osindero S., The Y.-W.// University of Toronto [Электронный ресурс]. – 2006. – Режим доступа: https://www.cs.toronto.edu/~hinton/absps/fastnc.pdf/. – Дата доступа: 01.04.2024.