УДК 004.85

АДАПТИВНЫЙ МЕТОД НАСТРОЙКИ ПАРАМЕТРОВ АКТИВАЦИОННОЙ ФУНКЦИИ ИСКУССТВЕННЫХ НЕЙРОНОВ

Анохин М.Н.

Институт сферы обслуживания и предпринимательства (филиал) ДГТУ (346500, ул. Шевченко, 147, г. Шахты, Ростовская обл.), email: mail@sssu.ru

В статье представлен метод адаптивной настройки нелинейной части искусственных нейронов. Согласно представленному методу активационная функция нейрона автоматически подстраивается под размерность входа нейрона таким образом, чтобы нейрон всегда оставался чувствительным к любым возможным входным воздействиям. Метод снимает необходимость выбора параметров активационной функции и упрощает применение конструктивных алгоритмов. Кроме того, в ходе экспериментов было показано, что он также увеличивает скорость обучения. Причём при применении специальных методик выбора параметров обучения положительный эффект представленного алгоритма значительно возрастает. Эффективность алгоритма показана на задаче распознавания символов MNIST с использованием одной ЭВМ. Разработанное решение основано на свёрточной нейронной сети. Представлен сравнительный анализ вычислительной эффективности её обучения с использованием предлагаемого алгоритма.

Ключевые слова: нейронные сети, адаптивные алгоритмы, конструктивные алгоритмы, машинное обучение, классификация.

ADAPTIVE METHOD FOR ADJUSTMENT SETTINGS OF ARTIFICIAL NEURON ACTIVATION FUNCTION

Anokhin M.N.

Institute of service and business (branch) Don State Technical University (346500, street Shevchenko, 147, Shakhty, Rostov Oblast), email: mail@sssu.ru

This paper presents a method for adaptive adjustment of the nonlinear artificial neurons. According to the present method of the neuron activation function automatically adjusts the dimension of the neuron input so that the neuron always stays sensitive to any possible inputs. The method eliminates the need for the choice of parameters and activation function simplifies the application design algorithms. In addition, the experiments have shown that it also increases the speed of learning. Also used with special techniques like algorithm of selecting learning parameters shows additional positive effect. The effectiveness of the algorithm is demonstrated on the task MNIST character recognition using a only just one computer. The developed solution is based on a convolutional neural network. The comparative analysis of the computational efficiency of its training using the proposed algorithm is presented.

Keywords: neural networks, adaptive algorithms, constructive algorithms, machine learning, classification.

В последнее время широкое распространения получила парадигма так называемого глубокого обучения. По существу, она представляет собой развитие классической идеи персептрона [6]. Известно, что сети прямого распространения позволяют решать широкий круг задач классификации, управления, аппроксимации [1]. Основное отличие современного подхода заключается в использовании не одно- или двухслойных сетей, а создание сетей с большим количеством слоёв. Этим объясняется использование приставки «глубокое» в термине *deeplearning*. Применение множества слоёв позволяет решать более сложные задачи. Например, такие, в которых предъявляются особенно жёсткие требования к инвариантности реакций сети при преобразованиях масштабирования и поворота входных образов.

При решении задачи разработки искусственной нейронной сети приходится сталкиваться с различными сложностями. Одна из главных проблем связана с тем, что увеличение количества слоёв (глубины) сети, с одной стороны, увеличивает обобщающий потенциал, а с другой, приводит к экспоненциальному росту количества синапсов, т.е. свободных параметров модели. Это в свою очередь существенно замедляет процесс обучения и приводит к проблеме переобучения. А это в свою очередь ведёт к экспоненциальному увеличению вычислительной сложности модели.

Решением является отказ от полносвязности слоёв. В классической схеме персептрона каждый нейрон связывался с каждым нейроном из предыдущего слоя, следовательно, количество синапсов было пропорционально количеству нейронов. Для решения этой проблемы была разработана концепция сверточных нейронных сетей [3].

При помощи сверточных нейронных сетей наиболее эффективно решаются задачи распознавания образов. По этой причине многие крупные компании уже представили свои программные библиотеки для решения задач глубокого обучения свёрточных нейронных сетей [4,5]. При этом основной упор делается на реализации использующих технологию вычислений общего назначения на графических ускорителях (GPGPU). Структура этих ускорителей превосходно подходит для решения задач *deeplearning* [4].

Применение сверточных слоёв позволяет решить вопросы, связанные с управлением вычислительной сложностью модели. Однако она не решает множества сопутствующих вопросов. Выбор количества нейронов, выбор параметров активационной функции, выбор параметров обучения. Для этих целей используются так называемые конструктивные алгоритмы. Эти алгоритмы позволяют автоматизировать процесс выбора архитектуры сети. Это приводит к неоправданному росту размерности модели, и, как следствие, снижает её эффективность. В статье представлен метод, позволяющий повысить эффективность конструктивных алгоритмов. Благодаря предлагаемому методу нейрон может адаптировать параметры активационной функции к размерности входных воздействий таким образом, чтобы его выходное значение всегда оставалось вне зоны насыщения. Таким образом, независимо от того, как конструктивный алгоритм изменит топологию сети, все нейроны гарантированно будут принимать участие в работе сети.

Алгоритм адаптации параметров активационной функции

При разработке структуры сети приходится сталкиваться с проблемой выбора параметров активационной функции. Для сигмоидальной функции нужно указать два параметра: смещение и угол наклона. Если параметры будут заданы неверно, то превышение входными значениями определённого порога приведёт к тому, что выходное значение нейрона будет равным максимально возможному выходу, следовательно, нейрон

становится нечувствительным к любым изменениям входа за пределами этого порога. Это особенно актуально при использовании конструктивных алгоритмов, в которых конфигурация сети постоянно изменяется, и, следовательно, невозможно заранее определить параметры активационной функции.

Для решения проблемы предлагается встроить в нейроны адаптационный алгоритм выбора параметров активационной функции, который бы предотвращал проблему переполнения нейрона, автоматически определяя корректные параметры. Для этого параметры активационной функции сами определяются как функции от параметра. В данной работе рассматривается работа с сигмоидальной функцией, но аналогичные рассуждения могут быть проведены для других видов активационных функций.

$$y = \frac{1}{1+ e^{-z}} \qquad (1)$$

В работе [2] даются общие рекомендации по настройке параметров сигмоидальной функции

$$y = \frac{1}{1+ e^{-a(s-b)}}, \qquad (2)$$

где a – определяет угол наклона сигмоиды, a b – определяет смещение сигмоиды относительно начала координат, s – суммарный вход нейрона (индуцированное локальное поле). В работе [2] указывается, что значения параметров получаются экспертным путём. Коэффициент наклона можно вычислить на основе оценки значения выхода в точке, соответствующей максимальному выходному значению сигнала. Например, 0.99. Аналогично смещение получается из значения входа при выходном сигнале равном 0.5.

Таким образом, вид активационной функции зависит от максимально возможного индуцированного локального поля. От его значения зависит как коэффициент наклона, так и смещение сигмоиды. Поскольку

$$s = \sum_{i=1}^{n} w_i x_i,$$

где x_i – i-й вход нейрона, w_i – вес i-го входа. И учитывая, что

$$0 < w \leq 1$$

$$0 \leq x \leq 1$$

не сложно видеть, что максимальный вход определяется формулой

$$\max = \sum_{i=1}^{n} w_i,$$

где max – максимально возможное значение индуцированного локального поля нейрона.

Исходя из вышеизложенного, формулы для расчёта параметров активационной функции имеют вид: угол наклона

$$a = \frac{Ln(99)}{0.5\, max},$$

смещение

$b = \frac{max}{2}$.

Таким образом, алгоритм работы нейрона принимает вид:

1. Рассчитать максимальный вход max.
2. Рассчитать индуцированное локальное поле s.
3. Определить параметры активационной функции a и b.
4. Вычислить выходное значение нейрона y.

Теперь при определении структуры сети инженер может не беспокоиться о выборе параметров активационной функции. Как бы не изменялась размерность входного вектора, нейрон автоматически подстроится к новой размерности, и выход нейрона всегда будет оставаться корректным. Это особенно удобно при использовании парадигмы растущих сетей, поскольку заранее оценить максимальную размерность входного вектора каждого нейрона невозможно. При этом выход нейрона не только будет оставаться корректным, но также будет поддерживаться достаточно высокое его значение. Таким образом, сеть одновременно предохраняется как от перенасыщения, так и от вырождения.

Модификация алгоритма обучения

Основным методом обучения сетей прямого распространения является метод обратного распространения ошибки. Он в свою очередь основывается на стохастическом градиентном спуске. Если обозначить обновление параметра x в момент времени t как x_t формула обновления параметра будет иметь вид:

$$x_{t+1} = x_t + \Delta x_t$$

Алгоритм градиентного спуска пошагово оптимизирует параметры модели, используя формулу

$$\Delta x_t = -n g_t \qquad (3)$$

где$g_t = \frac{\partial f(x_t)}{\partial x_t}$,n – параметр скорости обучения. От этого параметра зависит размер шага модификации параметров. Если шаг будет малым, процесс обучения будет происходить медленно, если шаг будет слишком большим, обучение не сможет обнаруживать локальные минимумы.

Существует ряд модификаций классического алгоритма градиентного спуска. Они основываются на методе Ньютона

$$\Delta x_t = H_t^{-1} g_t, \qquad (4)$$

где H_t^{-1} –матрица обратная матрице Гессе вторых производных, вычисленных для итерации t. В чистом виде на практике данный метод не применим из-за высокой вычислительной стоимости.

Поскольку для управления обучением невозможно использовать точные методы, выбор квазиоптимального параметра обучения является частью инженерной сложности в задаче обучения нейронной сети.

Существуют различные алгоритмы аппроксимирующие формулу (4). В работе [7] рассматривается один из таких алгоритмов, под названием Adadelta. Он позволяет:

- отказаться от ручного выбора параметра обучения;
- обеспечить нечувствительность к значениям параметров обучения;
- производить обучение с различной скоростью по каждому из параметров;
- уменьшить количество итераций при градиентном спуске;
- снизить зависимость обучения от начальных условий;
- возможность использовать один алгоритм для глобального и локального поиска.

Алгоритм имеет следующий вид. Сначала рассчитывается экспоненциальное среднее градиента:

$$E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho) g_t^2,$$

$$RSM[g]_t = \sqrt{E[g^2]_t + \epsilon}$$

где ρ – коэффициент затухания во времени, ϵ – параметр для улучшения свойств знаменателя [8]. В [7] указывается, что работа алгоритма не зависит существенным образом от выбора этих параметров.

После чего рассчитывается текущее приращение

$$\Delta x_t = -\frac{RSM[\Delta x]_{t-1}}{RSM[g]_t} g_t$$

И затем рассчитывается экспоненциальное среднее приращений

$$E[\Delta x^2]_t = \rho E[\Delta x^2]_{t-1} + (1-\rho)\Delta x_t^2$$

Более подробно алгоритм рассмотрен в работе [7]. Выбор параметра n происходит в автоматическом режиме в зависимости от хода процесса обучения, и при решении задачи многомерной оптимизации процесс обучения управляется по всем размерностям в отдельности.

Введение адаптационных поправок в алгоритм работы нейрона приводит к необходимости изменения стандартного алгоритма обратного распространения ошибки. Модификация весовых коэффициентов нейронов производится по формуле (3). В качестве f(x) выступает функция ошибки E(w), где w это значение весов синапсов. По правилу дифференцирования сложной функции

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial s}\frac{\partial s}{\partial w}$$

Таким образом, при использовании классической сигмоидальной функции

$$\Delta w_{ij} = -n\delta_j y_i^{q-1}$$

где q – номер слоя, а δ_j определяется в зависимости от номера слоя.

$$\frac{\partial E}{\partial y_j} = \sum_r \frac{\partial E}{\partial y_r}\frac{\partial y_r}{\partial s_r}\frac{\partial s_r}{\partial y_j} = \sum_r \frac{\partial E}{\partial y_r}\frac{\partial y_r}{\partial s_r} w_{jr}^{(q+1)}$$

$$\delta_j = \begin{cases} \text{если } q = k, \text{то} \left(y_j^q - d_j\right)\dfrac{\partial y_j}{\partial s_j} \\ \text{если } q < k, \text{то } \left[\sum_r \delta_r^{(q+1)} w_{jr}^{(q+1)}\right]\dfrac{\partial y_j}{\partial s_j} \end{cases}$$

$\frac{\partial y_j}{\partial s_j} = y_j\left(1 - y_j\right) = (1 + e^{-x})^{-2}e^{-x},$

где k –количество слоёв нейронов, q – номер слоя.

Однако адаптивная сигмоида (2) имеет более сложную формулу по сравнению с исходной (1). Её значение зависит в частности от значения весов нейрона. Поэтому требуется внести изменения в расчеты сложной производной от функции ошибки.

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial c}\frac{\partial c}{\partial w}$$

$$c = -a(s - b)$$

$$\frac{\partial E}{\partial y_j} = \sum_r \frac{\partial E}{\partial y_r}\frac{\partial y_r}{\partial s_r}\frac{\partial s_r}{\partial y_j} = \sum_r \frac{\partial E}{\partial y_r}\frac{\partial y_r}{\partial s_r}(-a_r) w_{jr}^{(q+1)}$$

$$\delta_j = \begin{cases} \text{если } q = k, \text{то} \left(y_j^q - d_j\right)\dfrac{\partial y_j}{\partial w_j} \\ \text{если } q < k, \text{то } \left[\sum_r \delta_r^{(q+1)} w_{jr}^{(q+1)}(-a_r)\right]\dfrac{\partial y_j}{\partial w_j} \end{cases}$$

$$\frac{\partial y_j}{\partial w_j} = \frac{Ln(99)}{0.5} max_j^{-2}\left(s_j - b_j\right) - a_j(y_j - 0.5)$$

Приведённые формулы включают изменения, вносимые адаптационным алгоритмом в процесс прямого распространения сигнала по сети, и учитывают их в реализации алгоритма градиентного спуска.

Тестирование рассмотренных алгоритмов

Описанные алгоритмы требуют большего объема памяти по сравнения с традиционными, поскольку вынуждены сохранять дополнительные данные между разными этапами передачи сигнала и обучения. Кроме того, они имеют большую вычислительную сложность, поскольку вводят дополнительные параметры, часть из которых рассчитывается динамически. Для оценки реального изменения сложности был проведён ряд экспериментов. При тестировании использовались данные из [10]. Данные представляют собой изображения

цифр 28x28 пикселов. Во время теста использовалась сеть прямого распространения, архитектурно аналогичная сети из [9]. Средние значения, полученные в результате тестовых запусков, сведены в таблицу 1.

Таблица 1

Результаты тестирования

Алгоритмы	Скорость сходимости алгоритма (в эпохах)	Скорость сходимости алгоритма (по времени в минутах)	Средняя максимальная ошибка	Затраты памяти(Мб)
A1	3553	56	0,098	**293**
A2	2544	46.48	0,09	293
A3	2289	40.88	**0,076**	367
A4	**1397**	**34.73**	0,089	367

Для оценки работы рассмотренных алгоритмов использовались 4 реализации:

A1 – без применения рассмотренных методов. Параметры нейронов и обучения выбирались вручную. В таблице 1 приводятся лучшие результаты.

A2 – без адаптации нейронов, но с алгоритмом Adadelta.

A3 – с алгоритмом адаптации нейронов, но без Adadelta

A4 – с алгоритмом адаптации нейронов и с алгоритмом Adadelta.

Как видно, скорость сходимости в эпохах у A4 значительно превосходит результаты других алгоритмов. Поскольку условия тестирования всех алгоритмов были идентичными и результаты, занесённые в таблицу, являются статистическими, можно предположить, что это связано с синергетическим эффектом, который возникает при взаимодействии алгоритма Adadelta с адаптационным алгоритмом. В то же время скорость сходимости по времени всего на 15 % меньше, чем у A3. Это связано с большей вычислительной сложностью алгоритма A4 по сравнению с остальными. Расход памяти при применении адаптационного алгоритма заметно больше, однако в условиях быстрого роста объёмов памяти современных ЭВМ этот параметр нельзя назвать существенным. Средняя ошибка A4 лишь незначительно хуже, чем у алгоритма A3. Это означает, что адаптационный алгоритм снижает качество локального поиска, однако, учитывая малость абсолютного значения ошибки, данный недостаток не представляется существенным.

Заключение

Предложенный алгоритм адаптации параметров сигмоидальной функции, а также алгоритм Adadelta, уменьшают сложность задачи проектирования и реализации

нейросетевых моделей. Это позволяет в частности снизить требование к инженерной квалификации специалиста, разрабатывающего нейросетевое решение, а, следовательно, позволяет расширить сферу применения нейросетевых алгоритмов, включив в неё специалистов из смежных отраслей знаний.

Синергетический эффект, возникающий при совместном применении двух алгоритмов, позволяет существенно повысить технические характеристики получаемых нейросетевых решений. Это стимулирует к поиску других алгоритмов, которые также позволят повысить эффективность сетей прямого распространения. В частности, открытым остаётся вопрос о выборе оптимального количества нейронов для решения конкретной задачи. Применение конструктивных алгоритмов требует значительных вычислительных ресурсов. Однако предложенный адаптационный алгоритм повышает степень автоматизации процесса синтеза структуры сети, предлагая механизм автоматического учета новых синаптических связей, без ограничения их количества на нейрон. При этом каждый нейрон сети избегает и вырождения, и перенасыщения. Таким образом, предложенный алгоритм является органической частью более широкой концепции, решающей не только технические, но инженерные вопросы проектирования нейронных сетей.

Списоклитературы

1. Хайкин, Саймон. Нейронные сети: полный курс, 2nd ed. /С. Хайкин. – М.: Вильямс, 2006. – 1104 с.
2. Пегат, А. Нечеткое моделирование и управление / А. Пегат. – М.: БИНОМ Лаборатория знаний, 2013. – 798 с.
3. LeCun, Y., and Bengio, Y. 1998. The handbook of brain theory and neural networks. Cambridge, MA, USA: MIT Press. Chapter Convolutional networks for images, speech, and time series, 255–258.
4. Accelerate Machine Learning with the cuDNN Deep Neural Network Library [Электронныйресурс] // NVIDIA – World Leader in Visual Computing Technologies [сайт]. [2015]. URL: http://devblogs.nvidia.com/parallelforall/accelerate-machine-learning-cudnn-deep-neural-network-library/ (дата обращения 11.02.2015).
5. Building high-level features using large scale unsupervised learning. Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pp. 8595 – 8598.
6. Deep Learning of Representations: Looking Forward. First International Conference, SLSP 2013, Tarragona, Spain, July 29-31, 2013. Proceedings,pp 1-37.

7. M. D. Zeiler. Adadelta: An adaptive learning rate method. arXiv preprint arXiv:1212.5701, 2012.
8. S. Becker and Y. Le Cun, “Improving the convergence of back-propagation learning with second order methods,” Tech. Rep., Department of Computer Science, University of Toronto, Toronto, ON, Canada, 1988.
9. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning arXiv preprint arXiv:1312.5602, 2013.
10. Le Cun, Y. The MNIST database of handwritten digits [Электронныйресурс] // MNIST handwritten digit database, YannLeCun and Corinna Cortes [сайт]. [2015]. URL: http://yann.lecun.com/exdb/mnist (дата обращения: 05.03.2015).

Рецензенты:

Галушкин Д.Н., д.т.н., доцент, профессор кафедры «Информатика», институт сферы обслуживания и предпринимательства (филиал) ДГТУ, г. Шахты.

Лебедев Б.К., д.т.н., профессор, профессор кафедры «Системы автоматизации проектирования», ФГОУ ВПО «Южный федеральный университет», г. Таганрог.