Савенков К.Е., Николаева В.А. Алгоритм метода Нестерова для обучения сетей радиальных базисных функций. // Проблемы информатики в образовании, управлении, экономике и технике: Сб. статей XVII Междунар. научно-техн. конф. – Пенза: ПДЗ, 2017. – С. 87-93.

УДК 004.032.26

АЛГОРИТМ МЕТОДА НЕСТЕРОВА ДЛЯ ОБУЧЕНИЯ СЕТЕЙ РАДИАЛЬНЫХ БАЗИСНЫХ ФУНКЦИЙ

К.Е. Савенков, В.А. Николаева

ALGORITHM OF NESTEROV'S METHOD FOR LEARNING NETWORKS OF RADIAL BASIS FUNCTIONS

K.E. Savenkov, V.A. Nikolaeva

Аннотация. В обучении искусственных нейронных сетей, и в частности сетей радиальных-базисных функций, лежит проблема обучения какого-либо оптимизационного алгоритма. В данной работе были исследованы различные вариации метода градиентного спуска и метода Нестерова, рассмотрены их плюсы и минусы, сделаны выводы о целесообразности применения каждого из них.

Ключевые слова: машинное обучение, сети радиально-базисных функций, сети радиальных базисных функций, метод Нестерова, градиентные методы обучения, пакетное обучение, обучение на подвыборке, стохастический метод обучения.

Abstract. In the learning of artificial neural networks and, in particular, networks of radial-basis functions, there is the problem of learning of any optimization algorithm. In this article, various variants of the gradient descent method and the Nesterov method were investigated, their pros and cons were examined, conclusions were drawn on the appropriateness of using each of them.

Keywords: machine learning, radial basis functions network, Nesterov's accelerated gradient method, gradientlearningmethods, batch, mini-batch, stochastic method.

Искусственные нейронные сети – исключительно мощный инструмент, способный к моделированию нелинейных процессов, работе с зашумленными данными. В частности, описываемые в работе сети радиальных базисных функций (СРБФ) активно используются при решении задач приближения функций [1, 2]. Одна из проблем использования искусственных нейронных сетей была исследована и описана в текущей работе. Производится экспериментальное сравнение метода градиентного спуска с методом Нестерова.

СРБФ – искусственная нейронная сеть, которая использует радиальные базисные функции (РБФ) как функции активации. Пример архитектуры такой сети представлен на рис. 1, где r – Евклидова метрика, задающая расстояние между точкой $\mathbf{x}$ и центром РБФ $\mathbf{c}$. Структура типичной СРБФ включает три слоя [1]. Входной слой осуществляет распределение входных значений между нейронами скрытого слоя. Скрытый слой совершает нелинейное преобразование значений, полученных с входного слоя нейронов. Выходной слой производит линейное суммирование полученных значений из скрытого слоя нейронов с весами и подает значение на выход сети.

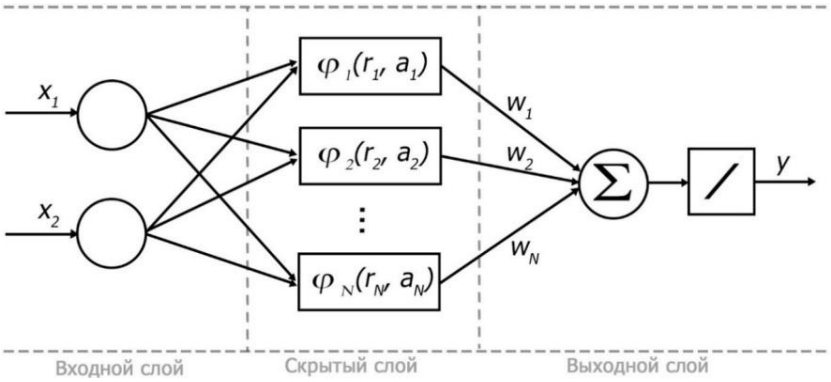

Рис. 1. Структура сети радиальных базисных функций

СРБФ осуществляет преобразование вида $F:\mathbf{x}\to y$, где $\mathbf{x}=[x_1,x_2,...,x_n]$. Выход сети вычисляется по формуле

$$f(\mathbf{x})=\sum_{i=1}^{N} w_i \varphi\left(\|\mathbf{x}-\mathbf{c}_i\|, a_i\right),$$

где N – количество нейронов скрытого слоя, $\mathbf{w}=[w_1,w_2,...,w_N]$ – веса линейного сумматора. В качестве РБФ использовалась функция Гаусса (Гауссиан)

$$\varphi\left(\|\mathbf{x}-\mathbf{c}_i\|, a_i\right)=\exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_i\|^2}{2a_i^{\,2}}\right),$$

где $\mathbf{a}=[a_1,a_2,...,a_N]$ является параметром ширины радиальных базисных функций. Выражение $\|\mathbf{x}-\mathbf{c}_i\|$ вычисляет расстояние между точкой $\mathbf{x}$ и центром $\mathbf{c}_i$ РБФ:

$$\|\mathbf{x}-\mathbf{c}_i\|=\sqrt{(x_1-c_{1i})-(x_2-c_{2i})-...-(x_n-c_{ni})}$$

Пусть $f(x)$ – функция, задающая отображение входных значений СРБФ в выходные, т.е. $f(x)$ и есть СРБФ. Тогда, при обучении сети, будем пытаться минимизировать функционал среднеквадратичной ошибки

$$MSE(\mathbf{o},\mathbf{t})=\frac{1}{n}\sum_{i=1}^{n}(o_i-t_i)^2,$$

где $\mathbf{o}=[o_1,o_2,...,o_n]$ являются выходами, а $\mathbf{t}=[t_1,t_2,...,t_n]$ – целевыми значениями $f(x)$, n равно количеству объектов обучающей выборки. Существует множество оптимизационных алгоритмов, мы рассмотрим **метод Нестерова**.

В обычном методе градиентного спуска на каждом шаге итерации следующее приближение получается из предыдущего вычитанием вектора градиента, умноженного на шаг η_k

$$\boldsymbol{\theta}^{(k)}=\boldsymbol{\theta}^{(k-1)}-\eta^{(k)}\nabla_{\theta}Q\left(\boldsymbol{\theta}^{(k-1)},\mathbf{X}\right),$$

где $Q\left(\boldsymbol{\theta}^{(k-1)},\mathbf{X}\right)$ – функционал ошибки алгоритма. В матричной форме выражение для градиента имеет вид

$$\nabla_{\theta}Q(\boldsymbol{\theta},\mathbf{X})=\frac{2}{n}\mathbf{X}^T(\mathbf{X}\boldsymbol{\theta}-y).$$

Таким образом, выражение для j-ой компоненты градиента содержит суммирование по всем объектам обучающей выборки

$$\frac{\partial Q}{\partial \theta_j}=\frac{2}{n}\sum_{i=1}^{n}x_i^j\left(\langle\boldsymbol{\theta},x_i\rangle-y_i\right).$$

В этом и состоит основной недостаток стандартного градиентного спуска – для корректировки параметров модели на одном шаге обучения в случае большой выборки даже одна итерация метода градиентного спуска будет производиться долго.

Идея стохастического градиентного спуска основана на том, что в сумме в выражении для j-ой компоненты градиента i-ое слагаемое указывает то, как нужно поменять параметр θ_j, чтобы качество увеличилось для i-го объекта выборки. Вся сумма при этом задает, как нужно изменить этот вес, чтобы повысить качество для всех объектов выборки. В стохастическом методе градиентного спуска градиент функционала качества вычисляется только на одном случайно выбранном объекте обучающей выборки. Это позволяет обойти вышеупомянутый недостаток обычного градиентного спуска.Таким образом, алгоритм стохастического градиентного спуска следующий. Сначала выбирается начальное приближение, например, $\mathbf{w}^0 = (0,0,...,0)$.Далее, последовательно вычисляются итерации $\boldsymbol{\theta}^{(k)}$: сначала случайным образом выбирается объект $\mathbf{x}_i$ из обучающей выборки $\mathbf{X}$ и вычисляется вектор градиента функционала качества на этом объекте, а следующее приближение получается из предыдущего вычитанием умноженного на шаг η_θ полученного вектора

$$\boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}^{(k-1)} - \eta_\theta \nabla_\theta Q(\boldsymbol{\theta}^{(k-1)}, \mathbf{x}_i).$$

Итерации продолжаются до достижения определенного условия, например,

$$\left\| \boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)} \right\| < \varepsilon.$$

Между этими двумя видами градиентного спуска существует компромисс, называемый иногда «**mini-batch**». В этом случае обновления происходят по некоторому числу обучающих примеров. Это, во-первых, ускоряет вычисление градиента, во-вторых, позволяет (теоретически) избежать локальных минимумов.

Метод Нестерова представляет собой вариант градиентного спуска с моментом [3, 4]. Правила обновления следующие:

$$\mathbf{v}^{(k)} = \gamma \mathbf{v}^{(k-1)} + \eta_\theta \nabla_\theta Q\left(\boldsymbol{\theta}^{(k-1)} - \gamma \mathbf{v}^{(k-1)}\right),$$

$$\boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}^{(k-1)} - \mathbf{v}^{(k)},$$

где $\mathbf{v}^{(k)}$ – вектор изменения параметров сети (скорость движения), γ – коэффициент сохранения скорости. Разница между методом Нестерова и спуском с моментом заключается в том, как определяется градиент. В методе Нестерова градиент вычисляется по уже измененным параметрам, а не исходным. Очевидные плюсы в том, что возможно "перепрыгивать" некоторые локальные минимумы.

Применительно к нашей структуре для изменения параметров будем вычислять градиенты по каждому из параметров сети. Опуская несложные, но громоздкие вычисления, запишем формулы вычисления компонентов градиента функционала сразу. Компонент градиента функционала по весу вычисляется следующим образом:

$$\frac{\partial Q^{(k)}}{\partial w_i^{(k)}} = \sum_{j=1}^{N} \left(f\left(\mathbf{p}_j\right) - t_j \right) \varphi_i\left(\mathbf{p}_j\right).$$

Компонент градиента функционала по центрам

$$\frac{\partial Q^{(k)}}{\partial c_{i1}^{(k)}} = w_i \sum_{j=1}^{N} \left(f\left(p_{j1}\right) - t_j \right) \varphi_i\left(\mathbf{p}_j\right) \frac{p_{j1} - c_{i1}}{a_i^2}.$$

Аналогично,

$$\frac{\partial Q^{(k)}}{\partial c_{i2}^{(k)}} = w_i \sum_{j=1}^{N} \left(f\left(p_{j2}\right) - t_j \right) \varphi_i\left(\mathbf{p}_j\right) \frac{p_{j2} - c_{i2}}{a_i^2}.$$

Компонент градиента функционала по ширине

$$\frac{\partial Q^{(k)}}{\partial a_i^{(k)}} = w_i \sum_{j=1}^{N} \left(f\left(\mathbf{p}_j\right) - t_j \right) \varphi_i\left(\mathbf{p}_j\right) \frac{\left\| \mathbf{p}_j - \mathbf{c}_i \right\|^2}{a_i^3}.$$

Используя вычисленные градиенты функционала корректируются веса

$$\mathbf{w}^{(k)} = \mathbf{w}^{(k-1)} - \left(\gamma_w \mathbf{w}^{(k-1)} + \eta_w \frac{\partial Q^{(k)}}{\partial w^{(k-1)}} \right),$$

центры

$$\mathbf{c}^{(k)} = \mathbf{c}^{(k-1)} - \left(\gamma_c \mathbf{c}^{(k-1)} + \eta_c \frac{\partial Q^{(k)}}{\partial c^{(k-1)}} \right),$$

и ширина РБФ

$$\mathbf{a}^{(k)} = \mathbf{a}^{(k-1)} - \left(\gamma_a \mathbf{a}^{(k-1)} + \eta_a \frac{\partial Q^{(k)}}{\partial a^{(k-1)}} \right).$$

Каждый вектор параметров при вычислении градиента должен участвовать в виде $\boldsymbol{\theta} - \gamma_\theta \mathbf{v}^{(k-1)}$, где $\boldsymbol{\theta}$ – текущий вектор параметров, γ_θ – коэффициент сохранения скорости, $\mathbf{v}^{(k-1)}$ – вектор значений на прошлой итерации.

В ходе исследовательской работы решалась задача приближения функции

$$f(x, y) = x^2 + y^2, x, y \in [-3; 3].$$

Задача решалась, используя по 3 вариации метода градиентного спускаи Нестерова (batch, mini-batch, stochastic). Обучение осуществлялось на одинаковых для всех методов начальных параметрах СРБФ.

Результаты тестирования алгоритмов сведены в таблицу ниже.

Результаты тестирования алгоритмов.

Алгоритм	Время обучения	Кол-во итераций	Достигнутая ошибка
batch	4min 59s	696	0.0999428312405
mini-batch	7min 5s	3513	0.0999271623457
stochastic	7min 50s	9999	1.40443564805
batch_nesterov	4min 10s	490	0.0990589642334
mini-batch_nesterov	2min 16s	709	0.0565060947887
stochastic_nesterov	7min 32s	9999	14.0565056489

Обучение осуществлялось на машине с OS:Windows 10, RAM: 6 GB, CPU: Intel(R)Core(TM) i3-4030UCPU @ 1.90GHz.

Из таблицы видно, что хорошие результаты были достигнуты всеми алгоритмами, за исключением стохастической их реализации. Для решения проблемы останова методов было решено использовать 10000 итераций как предельное количество, также обучение останавливалось, если алгоритм достигал значения среднеквадратичной ошибки в 0.1. Можно отметить, что метод Нестерова обучается чуть быстрее и использует меньшее количество итераций. Они оба нуждаются в тщательной настройке параметров, что требует некоторого терпения и времени. Также существует проблема выбора начальных параметров. Метод Нестерова намного чувствительнее к начальному приближению.

Авторы выражают благодарность доктору технических наук, профессору Горбаченко Владимиру Ивановичу за оказанную помощь и направление при написании данной работы.

Библиографический список

1. Kriesel D.A BriefIntroductiontoNeuralNetworks / D. Kriesel. 2007. URL: http://www.dkriesel.com.

2. Хайкин С. Нейронные сети: полный курс. М.: Вильямс, 2006. 1104 с.

3. Ruder S. An overview of gradient descent optimization algorithms [Электронный ресурс] / S. Ruder. 2017. URL: http://ruder.io/optimizing-gradient-descent/index.html#gradientdescentvariants

4. Goodfellow I. Deep Learning (Adaptive Computation and Machine Learning series). [Электронный ресурс] /I. Goodfellow, Y. Bengio, A. Courville. 2016. URL: http://www.deeplearningbook.org/

Савенков Константин Евгеньевич
Пензенский государственный университет, г. Пенза, Россия
E-mail: sav1996k@gmail.com

Savenkov K.E.
Penza State University, Penza, Russia

Николаева Валерия Андреевна
Пензенский государственный университет, г. Пенза, Россия
E-mail: insassin@gmail.com

Nikolaeva V.A.
Penza State University, Penza, Russia