



# Fast gradient descent algorithm for image classification with neural networks

Abdelkrim El Mouatasim<sup>1</sup>

Received: 11 November 2019 / Revised: 6 March 2020 / Accepted: 18 April 2020 / Published online: 20 May 2020  
© Springer-Verlag London Ltd., part of Springer Nature 2020

## Abstract

Any optimization of gradient descent methods involves selecting a learning rate. Tuning the learning rate can quickly become repetitive with deeper models of image classification, does not necessarily lead to optimal convergence. We proposed in this paper, a modification of the gradient descent algorithm in which the Nesterov step is added, and the learning rate is update in each epoch. Instead, we learn learning rate itself, either by Armijo rule, or by control step. Our algorithm called fast gradient descent (FGD) for solving image classification with neural networks problems, the quadratic convergence rate  $o(k^2)$  of FGD algorithm are proved. FGD algorithm are applicate to a MNIST dataset. The numerical experiment, show that our approach FGD algorithm is faster than gradient descent algorithms.

**Keywords** Gradient algorithm · Nesterov algorithm · Learning rate control · Image classification · Neural networks

## 1 Introduction

Computer vision helps machines to view and comprehend digital images, something that humans can do autonomously to high accuracy levels. Image processing is an important computer vision field, with major real-world applications such as autonomous vehicles [2], industry [15], medical diagnosis [19], and face recognition [21]. Image processing has many tasks, such as: regularization [7,8], clustering [13], localization [6] and classification [9]. In this paper we concerned by image classification that is the process of assigning a class label to an image.

Typically, the current state of the art solution to image classification uses artificial neural network [19], convolutionary neural network [17] and deep neural network [11,14], but this approaches has limitations such as the need for carefully designed structures and poor interpretability. The training algorithms used for solving image classification are gradient descent algorithms [4,15,19] and genetic algorithm [9].

However, the gradient descent algorithms and stochastic algorithms are easy to converge into the local minimum slowly. In fact, it is possible to divide the image classification

into two phases: extraction and classification of the feature. It is possible to perform two stages in the order. It can avoid simultaneously adjusting the parameters of the entire network and reducing the parameter adjustment difficulty.

Based on the above considerations, an improved gradient descent method are proposed, called stochastic gradient descent algorithm (SGD). SGD combines the advantages of the gradient descent algorithms, back propagation [4] and stochastic strategy, and it is used as a training algorithm of the classifier such as Nesterov accelerate gradient (NAG) [3].

In this paper, we propose a fast iterative algorithm for image classification with neural network called fast control gradient algorithm (FGD), combined SGD algorithm with controlled learning rate by Armigo rule and accelerated the algorithm by Nesterov step.

Neural network is composed of the stacked restricted boltzmann machines [10], or auto-encoder [1], which is used to extract the image features and then classify those extracted features by the softmax classifier.

FGD is proposed as the softmax classifier's training algorithm and is used to find the optimum of the softmax classifier parameters.

We discuss the complexity convergence of FGD the proposed algorithm and present some promising results of numerical experiments application to MNIST dataset, show that the proposed classification method FGD has better

✉ Abdelkrim El Mouatasim  
a.elmouatasim@uiz.ac.ma

<sup>1</sup> Faculty of Polydisciplinary Ouarzazate (FPO), Ibn Zohr University, B.P. 284, Ouarzazate 45800, Morocco

accuracy and antiover-fitting than other image classification methods such as SGD and NAG.

These algorithms are implemented in this paper using python programming tool for analyzing MNIST dataset [12].

This paper is organized in as drafted below in Sect. 1 introduction. Section 2 notations and assumptions. Section 3 classification. In Sect. 4 gradient algorithms. And finally the results are discussed in Sects. 5 and 6 concludes.

## 2 Notations and assumptions

First, we establish notation for future use:

- features  $x_i$  is the input variables,
- target  $y_i$  is the output variable that we are trying to predict,
- training example is a pair  $(x_i, y_i)$ ,
- training set is the dataset that we'll be using to learn a list of  $n$  training examples

$$\{(x_i, y_i) : i = 1, \dots, n\},$$

Note that the superscript ‘ $i$ ’ in the above notation is simply an index into the dataset. We will also use the following notation:

- $E = \mathfrak{R}^n$  is a finite dimensional Euclidean space of input variables,
- $x^t$  the transpose of  $x = (x_1, x_2, \dots, x_n) \in E$ ;
- $\|x\|_2 = \sqrt{x^t x} = \sqrt{(x_1^2 + \dots + x_n^2)}$  is the Euclidean norm of  $x$ ;
- $\langle x, y \rangle = x^t y$  is the inner product and corresponding norm  $\|x\|_2$ .

To describe the classification problem slightly more formally, given a training set, our objective is to learn a function  $h : E \rightarrow \mathfrak{R}$  called a hypothesis, so that  $h(x)$  is an optimal predictor for the corresponding value of  $y$ .

We consider the following basic problem of convex optimization

$$\min_{\mathbf{x} \in E} \mathbf{f}(\mathbf{x}), \quad (1)$$

where  $\mathbf{f} : E \rightarrow \mathfrak{R}$  is a smooth convex function of type  $C^{1,1}$ , i.e., continuously differentiable with Lipschitz continuous gradient  $L$ :

$$\|\nabla \mathbf{f}(\mathbf{x}) - \nabla \mathbf{f}(\mathbf{y})\| \leq L \|\mathbf{x} - \mathbf{y}\| \quad \forall \mathbf{x}, \mathbf{y} \in E,$$

and  $\nabla \mathbf{f}(\mathbf{x})$  is the gradient of  $\mathbf{f}(\mathbf{x})$ .

It is assumed that solution  $\mathbf{x}^*$  of (1) exists.

Typically computational algorithms for (1) rely on the use of gradient oracles which provide at arbitrary point  $\mathbf{x}$  the value of objective function  $\mathbf{f}(\mathbf{x})$  and some gradient  $\mathbf{g}$  of  $\mathbf{f}(\mathbf{x})$ , then  $\mathbf{g} = \nabla \mathbf{f}(\mathbf{x})$ .

## 3 Image classification with neural networks

### 3.1 Model

We define the following activation functions:

- the sigmoid function

$$\sigma : \mathbb{R} \rightarrow (0, 1) \\ x \mapsto \frac{1}{1+e^{-x}}$$

- the hyperbolic tangent function

$$\tanh : \mathbb{R} \rightarrow (-1, 1) \\ x \mapsto \frac{e^{2x}-1}{1+e^{2x}+1}$$

- the rectified linear unit function

$$ReLU : \mathbb{R} \rightarrow \mathbb{R}^+ \\ x \mapsto \max\{0, x\}$$

Neural networks is machine learning models [18], which as functions of their input, are the alternating composition of linear transformations, i.e.,

$$\mathbb{R}^n \rightarrow \mathbb{R}^m \\ x \mapsto Wx + b$$

where  $b \in \mathbb{R}^m$  and  $W$  is a  $m \times n$  matrix, with activation functions, applied to each coordinate.  $b$  and  $W$  are usually parameters of the model to be estimated.

In all of the layers to be trained, we denote the biases  $b$  and weights  $W$  by parameters  $\mathbf{w}$ .

We denote  $C = \{0, 1, \dots, c-1\}$  the classification where  $c$  is number of classes, on input  $x$ , the neural network model must output  $y \in \{\mathbb{R}^c : \sum_i y_i = 1\}$ . For  $k \in C$ , the softmax function

$$\sigma(y)_k = \frac{e^{y_k}}{\sum_i e^{y_i}}$$

is applicable.

The typically the crossentropy, a smooth approximation of classification error is objective function in our minimization problem:

$$f(\mathbf{w}) = -\frac{1}{N} \sum_{j=1}^N \sum_{q=0}^{c-1} z_{jq} \log(\hat{z}_q(x_j; \mathbf{w})) \quad (2)$$

where  $z_{jq} = \begin{cases} 1 & \text{if } x_j \text{ is in class } q \\ 0 & \text{otherwise} \end{cases}$  and  $\hat{z}_q(x_j; \mathbf{w})$  is the predicted probability that  $x_j$  belongs to class  $q$ .

We define the training loss function  $L(\mathbf{w})$  as :

$$L(\mathbf{w}) = f(\mathbf{w}) + \theta R(\mathbf{w})$$

where  $\theta$  is parameter of regularization and  $R(\mathbf{w}) = \|\mathbf{w}\|^2$  is  $\ell_2$  regularization.

The popular algorithms for estimated the parameters  $w$  of neural networks is gradient descent algorithms with back propagation.

For each step  $t$ , the technical used in SGD algorithms is replacing Eq. (2) of classification error  $f(w)$  by

$$f(\mathbf{w}, \mathbf{t}) = -\frac{1}{|M_t|} \sum_{j: x_j \in M_t} \sum_{q=0}^{c-1} z_{jq} \log(\hat{z}_q(x_j; \mathbf{w})) \quad (3)$$

where  $M_t$  is mini-batch. Equation (3) satisfy,

$$\mathbb{E}[f(w, t)] = f(w)$$

and

$$\mathbb{E}[\nabla_w f(w, t)] = \nabla_w f(w)$$

However, once before testing, the sample is shuffled arbitrarily, separated into batches of a fixed size, and these batches are used one at a time until they are all collected. This is then repeated from the first batch in the same order again. Each epoch then consists of sequential batches covering all training data, starting with the first batch and ending with the last one.

### 3.2 Back propagation

The updating weight parameters and fixed errors should be doing through the layers in the neural network, by combining GD algorithm and back propagation algorithm.

Let  $\ell_{w,b}(x)$  the activation function as:

$$\ell_{W,b}(x) = \sigma(Wx + b)$$

where  $x$  is input of training sample. The loss function is:

$$g(W, b, x, y) = \frac{1}{2} \|y - \ell_{W,b}(x)\|^2$$

Since the property of sigmoid function is:

$$\alpha = \sigma(\alpha), \quad \frac{\partial \alpha}{\partial z} = \alpha(1 - \alpha).$$

where  $z = Wx + b$ . Using chain rule:

$$\frac{\partial}{\partial z} g(W, b, x, y) = \frac{\partial g}{\partial \alpha} \frac{\partial \alpha}{\partial z} = -(y - \alpha)\alpha(1 - \alpha)$$

Then

$$\begin{aligned} \frac{\partial}{\partial W} g(W, b, x, y) &= \frac{\partial g}{\partial z} \frac{\partial z}{\partial W} = -(y - \alpha)\alpha(1 - \alpha)x^T \\ \frac{\partial}{\partial b} g(W, b, x, y) &= \frac{\partial g}{\partial z} \frac{\partial z}{\partial b} = -(y - \alpha)\alpha(1 - \alpha). \end{aligned}$$

## 4 Gradient methods

### 4.1 Gradient descent (GD)

The neural network algorithm uses an optimization method to evaluate global optimum. The optimization method used in the neural network algorithm was GD algorithm, represented by Eq. (4), a well referenced artificial intelligence function to model a first order optimization algorithm that helps us to find a local minimum. GD is an algorithm that is used to minimize a function, in this case, the cost function. The aim was to find a value of  $w$  which renders the lowest error and enables the cost function to reach a local minimum. Each iteration in this method aims to find a new value of  $w$  that yields a slightly lower error than the previous iteration. A learning rate is also used to control how large of a step we take downhill during each iteration. Since GD algorithm starts with some initial  $w$ , and repeatedly performs the update:

$$w_i = w_i - \eta \frac{\partial}{\partial w_i} f(w), \quad i = 0, \dots, n. \quad (4)$$

Here,  $\eta$  is a constant learning rate. This is a very natural algorithm that repeatedly takes a step in the direction of steepest decrease of  $f$ . Since differentiation and combination are interchangeable we can calculate the gradient as a sum of discrete components thus avoiding unnecessary complications of analytical formulas for neural network. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient at the current point. If instead one takes steps proportional to the positive of the gradient, one approaches a local maximum of that function; the procedure is then known as gradient ascent. In our case, we are looking for minimum value of the cost function, represented by Eq. (5).

$$y(w) = f(w) \quad (5)$$

Note that the value of the step size  $\eta$  is allowed to change at every iteration of the neural network algorithm, known as learning rate, of the optimization process. With certain

assumptions on the function  $y$  and particular choices of  $\eta$ , convergence to a local minimum can be guaranteed. When the function  $y$  is convex, all local minimum are also global minimum, so in this case gradient descent can converge to the global solution.

### 4.2 Control gradient algorithms

The simplest algorithms for (1) are gradient methods of the kind

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \eta_k \mathbf{g}_k, \quad \mathbf{g}^k = \nabla \mathbf{f}(\mathbf{x}^k), \quad k = 0, 1, \dots \quad (6)$$

which were under intensive study since 1960’s. It was shown that (6) converges under very mild conditions for learning rate  $\eta_k$  satisfying “divergence series” condition  $\sum_k \eta_k = \infty, \eta_k \rightarrow +0$ .

Let us suppose a learning rate  $\eta_{k-1}$  was used at iteration  $k - 1$  and let us identify at  $x^k$  the relation between  $\eta_{k-1}$  and the learning rate providing minimization of  $f$  in the direction  $-g^{k-1}$ . To this end Uryasev used in [20] scalar products

$$u_k = \langle g^k, g^{k-1} \rangle$$

It was heuristically suggested in [20] to correct learning rate according to the formula

$$\eta_{k+1} = \begin{cases} \eta_{\text{decr}} \eta_k, & \text{if } \mathbf{u}_{k+1} \leq 0, \\ \eta_{\text{incr}} \eta_k, & \text{otherwise,} \end{cases} \quad (7)$$

where:

- $\eta_{\text{decr}}$  is a decreasing coefficient,
- $\eta_{\text{incr}}$  is a increasing coefficient,
- $0 < \eta_{\text{decr}} < 1 < \eta_{\text{incr}}$  and  $\eta_{\text{incr}} \cdot \eta_{\text{decr}} < 1$ .

**Remark** The control learning rate of Uryasev is equivalent to the second conditions of Wolfe:

$$\langle g^k, g^{k-1} \rangle \geq c \|g^k\|^2$$

where  $c$  is real positive.

In this paper, we used Armijo condition [5]:

$$f(\mathbf{x}^k) - f(\mathbf{x}^{k+1}) \geq \beta \langle g^k, (\mathbf{x}^k - \mathbf{x}^{k+1}) \rangle. \quad (8)$$

where  $0 < \beta < 0.5$ . Then, the control learning rate can be calculate by this equation:

$$\eta_{k+1} = \begin{cases} \eta_{\text{decr}} \eta_k, & \text{if Eq. (8) satisfied,} \\ \eta_{\text{incr}} \eta_k, & \text{otherwise,} \end{cases} \quad (9)$$

### 4.3 Fast gradient descent (FGD) algorithm

Gradient and Nesterov step [16]

1. Select a point  $\mathbf{y}_0 \in \mathbf{E}$ . Put

$$k = 0, \quad \mathbf{b}_0 = 1, \quad \mathbf{x}^{-1} = \mathbf{y}_0.$$

2.  $k$ th iteration.

a) Compute  $\mathbf{g}_k = \nabla \mathbf{f}(\mathbf{y}_k)$

b) Put

$$\begin{aligned} \mathbf{x}^k &= \mathbf{y}_k - \eta_k \mathbf{g}_k, \\ \mathbf{b}_{k+1} &= 0.5(1 + \sqrt{4\mathbf{b}_k^2 + 1}), \\ \mathbf{y}_{k+1} &= \mathbf{x}^k + \left(\frac{\mathbf{b}_k - 1}{\mathbf{b}_{k+1}}\right)(\mathbf{x}^k - \mathbf{x}^{k-1}). \end{aligned} \quad (10)$$

The recalculation of the point  $\mathbf{y}_k$  in(10) is done using a “ravine” step, and  $\eta_k$  is the control learning rate (7).

**Remark** We assume that

$$\eta_k \leq \frac{1}{L} \quad (11)$$

#### 4.3.1 Convergence rate of FGD

**Lemma 1** For any  $x, y \in \mathbb{R}^n$ , we have

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|y - x\|^2.$$

**Proof** For all  $x, y \in \mathbb{R}^n$ , we have

$$\begin{aligned} f(y) &= f(x) + \int_0^1 \langle \nabla f(x + t(y - x)), y - x \rangle dt \\ &= f(x) + \langle \nabla f(x), y - x \rangle \\ &\quad + \int_0^1 \langle \nabla f(x + t(y - x)) - \nabla f(x), y - x \rangle dt. \end{aligned}$$

Therefore, if  $A = f(y) - f(x) - \langle \nabla f(x), y - x \rangle$  then

$$\begin{aligned} A &\leq |f(y) - f(x) - \langle \nabla f(x), y - x \rangle| \\ &= \left| \int_0^1 \langle \nabla f(x + t(y - x)) - \nabla f(x), y - x \rangle dt \right| \\ &\leq \int_0^1 | \langle \nabla f(x + t(y - x)) - \nabla f(x), y - x \rangle | dt \\ &\leq \int_0^1 \| \nabla f(x + t(y - x)) - \nabla f(x) \| \cdot \|y - x\| dt \\ &\leq \int_0^1 tL \|y - x\|^2 dt \\ &= \frac{L}{2} \|y - x\|^2. \end{aligned}$$

Then, Lemma 1 hold. □

**Lemma 2** The sequence  $(b_k)_{k \geq 1}$  generated by the scheme (10) satisfies the following:

$$\frac{k + 1}{2} \leq b_k \quad \forall k \geq 1. \quad (12)$$

**Proof** We have

$$0.5(1 + \sqrt{4b_{k-1}^2 + 1}) \geq 0.5(1 + 2b_{k-1}^2)$$

then

$$b_k \geq \frac{1}{2} + b_{k-1}$$

Therefore, we have  $b_k \geq \frac{k+1}{2}$ . □

**Lemma 3** Let  $\{x^k, y_k\}$  be the sequence generated by the FGD, then for any  $x \in \mathbb{R}^n$  we have

$$f(x) \geq f(x^k) + \frac{L}{2} \|x - x^k\|^2 - \frac{L}{2} \|x - y^k\|^2. \tag{13}$$

**Proof** By the convexity of  $f$  we have

$$f(x) \geq f(y_k) + \langle \nabla f(y_k), x - y_k \rangle. \tag{14}$$

By Lemma 1

$$f(x^k) \leq f(y_k) + \langle \nabla f(y_k), x^k - y_k \rangle + \frac{L}{2} \|x^k - y_k\|^2 \tag{15}$$

Combining (14) and (15), we have

$$f(x) \geq f(x^k) + \langle \nabla f(y_k), x - x^k \rangle - \frac{L}{2} \|y_k - x^k\|^2. \tag{16}$$

Since  $x^k = y_k - \eta_k \nabla f(y_k)$ , then

$$f(x) \geq f(x^k) + \frac{1}{\eta_k} \langle y_k - x^k, x - x^k \rangle - \frac{L}{2} \|y_k - x^k\|^2. \tag{17}$$

Using (11), we obtain

$$f(x) \geq f(x^k) + L \langle y_k - x^k, x - x^k \rangle - \frac{L}{2} \|y_k - x^k\|^2. \tag{18}$$

Since

$$\langle y_k - x^k, x - x^k \rangle = \frac{1}{2} (\|y_k - x^k\|^2 + \|x - x^k\|^2 - \|x - y_k\|^2)$$

then from (18) the inequality (13) hold. □

**Theorem 1** If the sequence  $\{x_k\}_{k \geq 0}$  is constructed by FGD, then there exist a constant  $C$  such that

$$f(x_k) - f(x^*) \leq C/k^2. \tag{19}$$

**Proof** We denote

$$z_k = \frac{1}{b_{k+1}} x^* + (1 - \frac{1}{b_{k+1}}) x^k. \tag{20}$$

We know that  $\frac{1}{b_{k+1}} \in (0, 1], \forall k \geq 0$ , by the convexity of  $f$  we have

$$f(z_k) \leq \frac{1}{b_{k+1}} f(x^*) + (1 - \frac{1}{b_{k+1}}) f(x^k). \tag{21}$$

(21)  $\times b_{k+1}^2$ , we get

$$b_{k+1}^2 f(z_k) \leq b_{k+1} f(x^*) + (b_{k+1}^2 - b_{k+1}) f(x^k) \tag{22}$$

With  $b_{k+1}^2 - b_{k+1} = b_k^2$  yield

$$b_{k+1}^2 f(z_k) \leq b_{k+1}^2 f(x^*) - b_k^2 f(x^*) + b_k^2 f(x^k). \tag{23}$$

(23)  $+ b_{k+1}^2 f(x^{k+1})$ , we get

$$b_{k+1}^2 (f(x^{k+1}) - f(z_k)) \geq b_{k+1}^2 (f(x^{k+1}) - f(x^*)) - b_k^2 (f(x^k) - f(x^*)). \tag{24}$$

By Lemma 3, we have

$$f(x^{k+1}) - f(z_k) \leq \frac{L}{2} \|z_k - y_{k+1}\|^2 - \frac{L}{2} \|z_k - x^{k+1}\|^2. \tag{25}$$

Then,

$$b_{k+1}^2 \frac{L}{2} (\|z_k - y_{k+1}\|^2 - \|z_k - x^{k+1}\|^2) \geq b_{k+1}^2 (f(x^{k+1}) - f(x^*)) - b_k^2 (f(x^k) - f(x^*)). \tag{26}$$

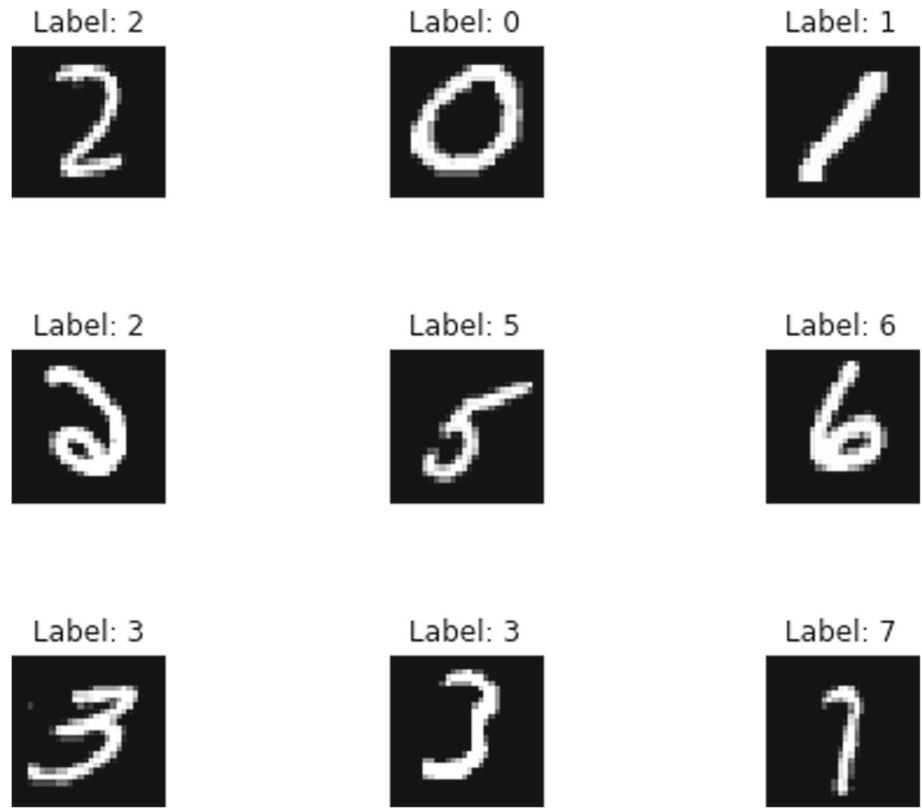
Let  $p_k = x^{k-1} + b_k(x^k - x^{k-1})$ , then

$$\begin{aligned} z_k - y_{k+1} &= \frac{1}{b_{k+1}} x^* + (1 - \frac{1}{b_{k+1}}) x^k - x^k \\ &\quad + (\frac{b_k - 1}{b_{k+1}}) (x^k - x^{k-1}) \\ &= \frac{1}{b_{k+1}} (x^* - x^{k-1} + b_k(x^{k-1} - x^k)) \\ &= \frac{1}{b_{k+1}} (x^* - p_k) \end{aligned} \tag{27}$$

and

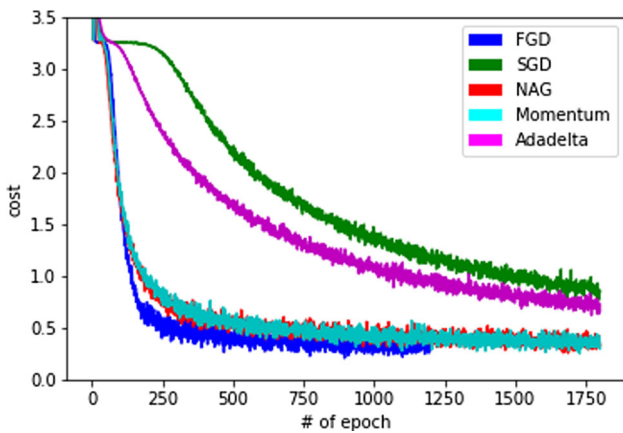
$$\begin{aligned} z_k - x^{k+1} &= \frac{1}{b_{k+1}} x^* + (1 - \frac{1}{b_{k+1}}) x^k - x^{k+1} \\ &= \frac{1}{b_{k+1}} (x^* - x^k + b_{k+1}(x^k - x^{k+1})) \\ &= \frac{1}{b_{k+1}} (x^* - p_{k+1}). \end{aligned} \tag{28}$$

**Fig. 1** Samples from the MNIST dataset



**Table 1** Comparison of the classification accuracy between FGD and four classified methods

| Methods   | Training accuracy (%) | Test accuracy (%) | Loss cost | CPU time/s |
|-----------|-----------------------|-------------------|-----------|------------|
| FGD       | 97.54                 | 94.59             | 0.31      | 83.17      |
| SGD       | 91.87                 | 91.40             | 0.88      | 132.66     |
| NAG       | 96.50                 | 94.58             | 0.35      | 127.46     |
| Momentum  | 96.287                | 94.43             | 0.31      | 128.31     |
| Adadelata | 92.87                 | 92.38             | 0.67      | 132.75     |



**Fig. 2** Comparing epoch of FGD with SGD, NAG, Moumentum and Adaselata

Using Eqs. (27) and (28) in (26), we have

$$\begin{aligned} & \frac{L}{2} \|x^* - p_k\|^2 - \frac{L}{2} \|x^* - p_{k+1}\|^2 \\ & \geq b_{k+1}^2 (f(x^{k+1}) - f(x^*)) - b_k^2 (f(x^k) - f(x^*)). \end{aligned}$$

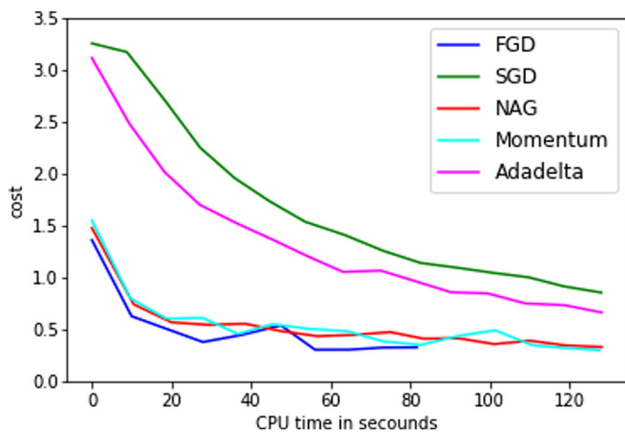
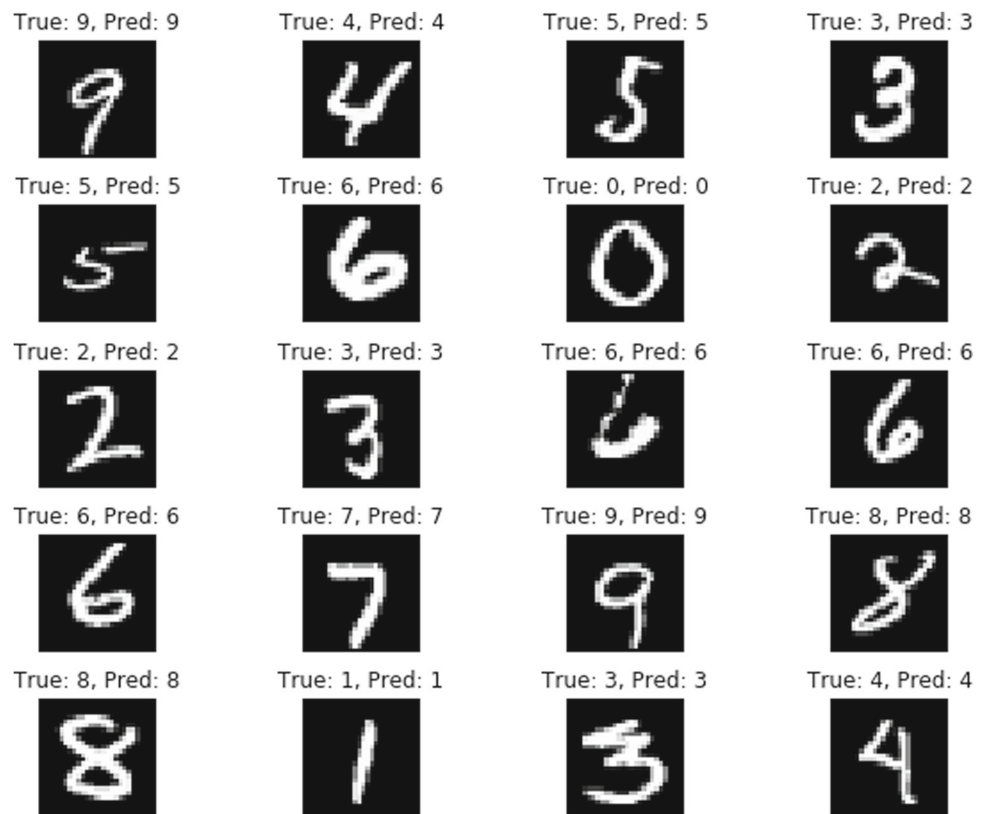
Since  $b_0 = 0$  and  $p_0 = x^{-1} = x^0$ , then we get

$$b_k^2 (f(x^k) - f(x^*)) \leq \frac{L}{2} \|x^* - x^0\|^2.$$

Let  $C = 2L \|x^* - x^0\|^2$ , using Lemma 2 then theorem hold.

□

**Fig. 4** predict sample using FGD in MNIST



**Fig. 3** Comparing CPU time of FGD with SGD, NAG, Moutementum and Adaselta

### 5 Computational experiment

All the experiments were carried out on a personal computer with an HP i3 CPU processor 1.80 GHz, 4 Go RAM, ×64, using Python 3.7 for Windows 8.1.

In this section, the dataset used is MNIST [12], consisting of 60,000 training and 10,000 test 28 × 28 grayscale images of handwritten digits, 0 to 9, as shown in Fig. 1, so 10 classes.

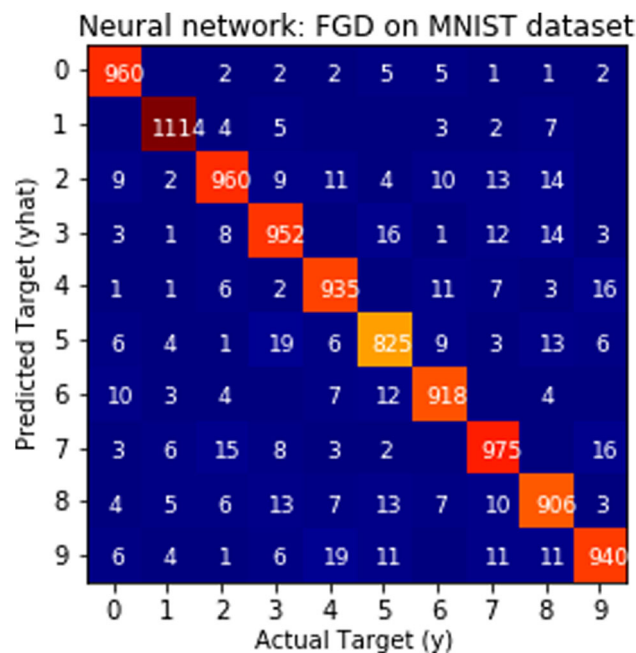
MNIST dataset is performed to verify the performance of the FGD algorithm.

To train the network, we present digits from the 60,000 MNIST training set to the network. The different gradient descent optimization sch as, SGD, NAG, Momentum, Adadelta and our approach FGD are used in this experiments. For the results of comparing methods see Table 1, Figs. 2 and 3, this results show that the FGD is faster with best classification accuracy.

The class-assigned neuron response is then used in the 10,000 MNIST test set to measure the network’s classification accuracy. The estimated number is calculated by multiplying each neuron’s responses by class and then choosing the class with the highest average firing rate see Figs. 4 and 5.

### 6 Concluding remarks

In this paper, the gradient descent algorithm are accelerated by Nestrov step and control learning rate via Armijo condition called FGD algorithm for solving minimum loss function of image classification. The convergence rate of FGD algorithm proved that it is quadratic convergence, and the comprising of numerical results of FGD with four gra-



**Fig. 5** Average confusion matrix of the testing results over ten presentations of the 10,000 MNIST test set digits using FGD algorithm

gradient descent algorithms in dataset MNIST, show that FGD algorithm is robust and faster than others methods.

**Acknowledgements** We indebted to the anonymous Reviewers and Editors for many suggestions and stimulating comments to improve the original manuscript.

## References

- Bengio, Y.: Learning deep architectures for AI. *Found. Trends Mach. Learn.* **2**(1), 1–27 (2009)
- Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al.: End to End Learning for Self-driving Cars (2016). [arXiv:1604.07316](https://arxiv.org/abs/1604.07316)
- Botev, A., Lever, G., Barber, D.: Nesterov's accelerated gradient and momentum as approximations to regularised update descent. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp. 1899–1903 (2017)
- Cui, N.: Applying gradient descent in convolutional neural networks. *J. Phys. Conf. Ser.* **1004**, 012027 (2018)
- Chen, A., Xu, X., Ryu, S., Zhou, Z.: A self-adaptive Armijo stepsize strategy with application to traffic assignment models and algorithms. *Transp. A Transp. Sci.* **9**(8), 695–712 (2013)
- El Mouatasim, A., Ellaia, R., Souza de Cursi, J.E.: Stochastic perturbation of reduced gradient & GRG methods for nonconvex programming problems. *J. Appl. Math. Comput.* **226**, 198–211 (2014)
- El Mouatasim, A., Wakrim, M.: Control subgradient algorithm for image regularization. *J. Signal Image Video Process. (SIViP)* **9**, 275–283 (2015)

- El Mouatasim, A.: Control proximal gradient algorithm for  $\ell_1$  regularization image. *J. Signal Image Video Process. (SIViP)* **13**(6), 1113–1121 (2019)
- Evans, B., Al-Sahaf, H., Xue, B., Zhang, M.: Evolutionary deep learning: a genetic programming approach to image classification. In: *IEEE Congress on Evolutionary Computation*, pp. 1–6 (2018)
- Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. *Science* **313**(5786), 504–507 (2006)
- Huang, K., Hussain, A., Wang, Q., Zhang, R.: *Deep Learning: Fundamentals, Theory and Applications*. Springer, Berlin (2019)
- LeCun, Y., Cortes, C.: MNIST Handwritten Digit Database. AT&T Labs, vol. 2. (2010). <http://yann.lecun.com/exdb/mnist/>
- Li, P., Lee, S., Park, J.: Development of a global batch clustering with gradient descent and initial parameters in colour image classification. *IET Image Process* **13**(1), 161–174 (2019)
- Liu, G., Xiao, L., Xiong, C.: Image classification with deep belief networks and improved gradient descent. In: *International Conference on Embedded and Ubiquitous Computing* (2017)
- MacLean, J., Tsotsos, J.: Fast Pattern Recognition Using Gradient-Descent Search in an Image Pyramid. *IEEE* (2000)
- Nesterov, Y.: *Introduction Lectures on Convex Optimization: A Basic Course*, vol. 87. Springer, Berlin (2004)
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.* **115**(3), 211–252 (2015)
- Shi, B., Lyengar, S.S.: *Mathematical Theories of Machine Learning: Theory and Applications*. Springer, Berlin (2020)
- Singh, B.K., Verma, K., Thoke, A.S.: Adaptive gradient descent backpropagation for classification of breast tumors in ultrasound imaging. *Proc. Comput. Sci.* **46**, 1601–1609 (2015)
- Uryas'ev, S.P.: New variable-metric algorithms for nondifferentiable optimization problems. *J. Optim. Theory Appl.* **71**(2), 359–388 (1991)
- Xinhua, L., Qian, Y.: Face recognition based on deep neural network. *Int. J. Signal Process. Image Process. Pattern Recognit.* **8**(10), 29–38 (2015)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.