Noname manuscript No.
(will be inserted by the editor)

Fast Gradient Descent Algorithm for Image Classification with Neural Networks

Abdelkrim El Mouatasim

Received: date / Accepted: date

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 Nestrove 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] 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 approachs 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 it is used as a training algorithm of the classifier such as Nestrov 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 Nestrove step.

Abdelkrim El Mouatasim
Ibn Zohr University,
Faculty of Polydisciplinary Ouarzazate (FPO),
B.P. 284, Ouarzazate 45800, Morocco.
E-mail: a.elmouatasim@uiz.ac.ma

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 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 section 1 introduction. Section 2 notations and assumptions. Section 3 classification. In section 4 gradient algorithms. And finally the results are discussed in section 5 and section 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

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

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

- $E = \Re^n$ is a finite dimensional Euclidean space of input variables,
- x^t the transpose of $x = (x_1, x_2, ..., x_n) \in E$;
- $||x||_2 = \sqrt{x^t x} = \sqrt{(x_1^2 + ... + x_n^2)}$ is the Euclidean norm of x;
- $< x, y >= x^t y$ 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 $h : E \longrightarrow \Re$ 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

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

where $\mathbf{f} : E \longrightarrow \Re$ 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

$$\begin{aligned} \sigma : \mathbb{R} &\to (0,1) \\ x &\mapsto \tfrac{1}{1+e^{-1}} \end{aligned}$$

- the hyperbolic tangent function

$$\begin{aligned} \tanh : \mathbb{R} &\to (-1,1) \\ x &\mapsto \tfrac{e^{2x}-1}{1+e^{2x}+1} \end{aligned}$$

- the rectified linear unit function

$$\begin{aligned} ReLU : \mathbb{R} &\to \mathbb{R}^+ \\ x &\mapsto \max\{0, x\} \end{aligned}$$

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

$$\begin{aligned} \mathbb{R}^n &\to \mathbb{R}^m \\ x &\mapsto Wx + b \end{aligned}$$

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 $C = \{0, 1, \ldots, \mathbf{c} - 1\}$ the classification where $\mathbf{c}$ is number of classes, on input x, the neural network model must output $y \in \{\mathbb{R}^{\mathbf{c}} : \sum_i y_i = 1\}$.
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:

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

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 θ is parameter of regularization and $R(\mathbf{w}) = \|\mathbf{w}\|^2$ is ℓ_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 the equation (2) of classification error $f(w)$ by

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

where M_t is mini-batch.
The 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 Backpropagation 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

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

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

4 Gradient methods

4.1 Gradient descent (GD)

The neural network algorithm use an optimization method to evaluate global optimum. The optimization method used in the neural network algorithm was GD algorithm, represented by equation (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:

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

Here, η 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 Equation (5).

$$y(w) = f(w) \tag{5}$$

Note that the value of the step size η 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 η, 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

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

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

Let us suppose a learning rate η_{k-1} was used at iteration $k-1$ and let us identify at x^k the relation between η_{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

$$u_k =< g^k, g^{k-1} >$$

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

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

where:

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

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

$$< g^k, g^{k-1} > \ge c\|g^k\|^2$$

where c is real positive.

In this paper we used Armijo condition [5]:

$$f(\mathbf{x}^k) - f(\mathbf{x}^{k+1}) \ge \beta < g^k, (\mathbf{x}^k - \mathbf{x}^{k+1}) > . \tag{8}$$

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

$$\eta_{k+1} = \begin{cases} \eta_{decr}\eta_k, \text{ if the equation (8) satisfied,} \\ \eta_{incr}\eta_k, \text{ otherwise,} \end{cases} \tag{9}$$

4.3 Fast Gradient Descent (FGD) Algorithm

Gradient and Nesterov step

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. kth iteration.
   a) Compute $\mathbf{g_k} = \nabla \mathbf{f}(\mathbf{y}_k)$
   b) Put

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

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

Remark: We assume that

$$\eta_k \le \frac{1}{L} \tag{11}$$

4.3.1 Convergence rate of FGD

Lemma 1
For any $x, y \in I\!R^n$, we have

$$f(y) \le f(x) + < \nabla f(x), y - x > + \frac{L}{2}\|y - x\|^2.$$

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

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

Therefore, if $A = f(y) - f(x) - < \nabla f(x), y - x >$ then

$$\begin{aligned} A &\le |f(y)_f(x) - < \nabla f(x), y - x > | \\ &= |\textstyle\int_0^1 < \nabla f(x + t(y - x)) - \nabla f(x), y - x > dt| \\ &\le \textstyle\int_0^1 | < \nabla f(x + t(y - x)) - \nabla f(x), y - x > |dt \\ &\le \textstyle\int_0^1 \| < \nabla f(x + t(y - x)) - \nabla f(x)\|.\|y - x\|dt \\ &\le \textstyle\int_0^1 tL\|y - x\|^2 dt \\ &= \frac{L}{2}\|y - x\|^2. \end{aligned}$$

Then lemma 1 hold.

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

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

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 $x \in \Re^n$ 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

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

By lemma 1

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

Combining (14) and (15), we have

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

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

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

Using (11), we obtain

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

Since

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

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 $\frac{1}{b_{k+1}} \in (0, 1], \quad \forall k \geq 0$, 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

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

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

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

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

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

and

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

Using equations (27) and (28) in (26), we have

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

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.

5 Computational experiment

All the experiments were carried out on a personal puter with an HP i3 CPU processor 1.80GHz, RAM, x64, using Python 3.7 for Windows 8.1.

In this section, the dataset used is MNIST consisting of 60,000 training and 10,000 test 28 grayscale images of handwritten digits, 0 to 9, as s in Figure 1, so 10 classes. MNIST dataset is perfc to verify the performance of the FGD algorithm.

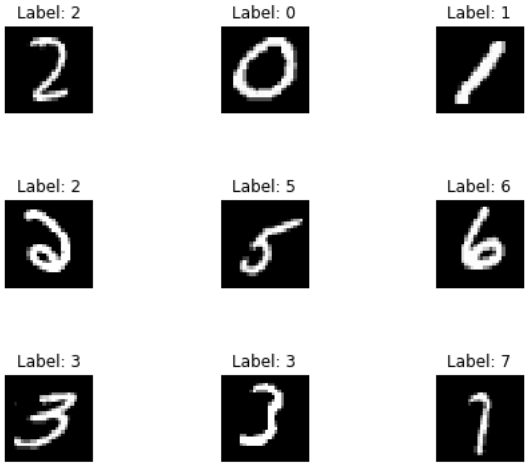

Fig. 1 Samples from the MNIST dataset

To train the network, we present digits fror 60,000 MNIST training set to the network. The (ent 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 5, Figure 2 and Figure 3, this results show that the FGD is faster with best classification accuracy.

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
Adadelta	92.87%	92.38%	0.67	132.75

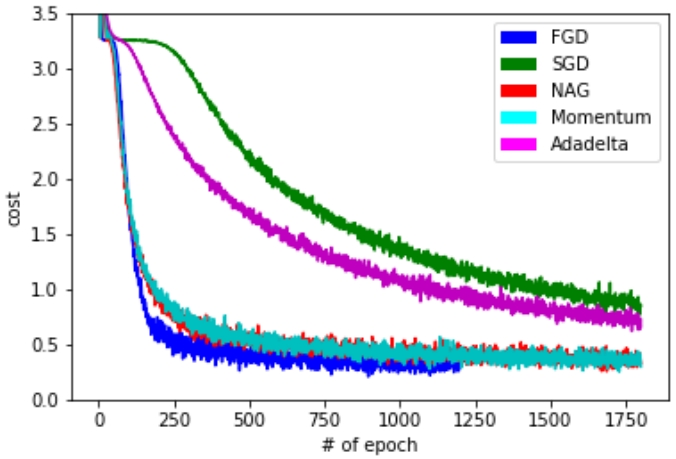

Fig. 2 Comparing epoch of FGD with SGD, NAG, Moumentum and Adaselta

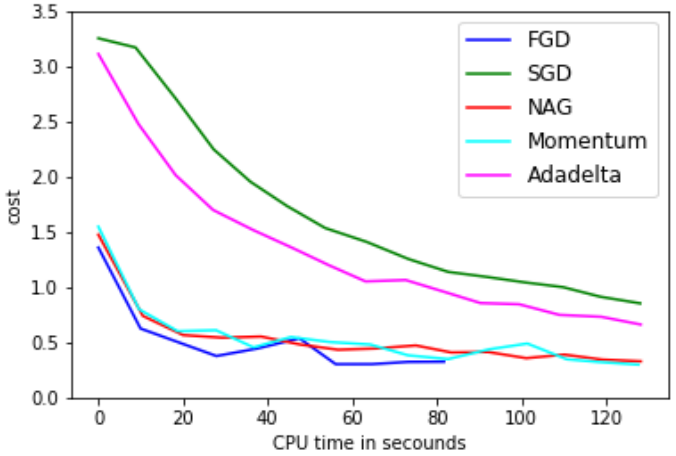

Fig. 3 Comparing CPU time of FGD with SGD, NAG, Moumentum and Adaselta

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 Figure 4 and Figure 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 nu-

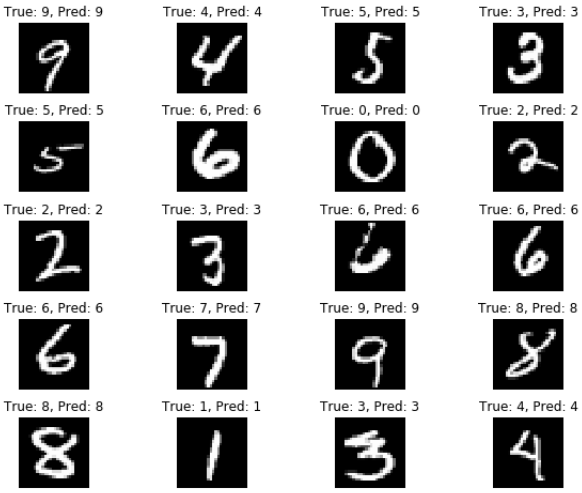

Fig. 4 predict sample using FGD in MNIST

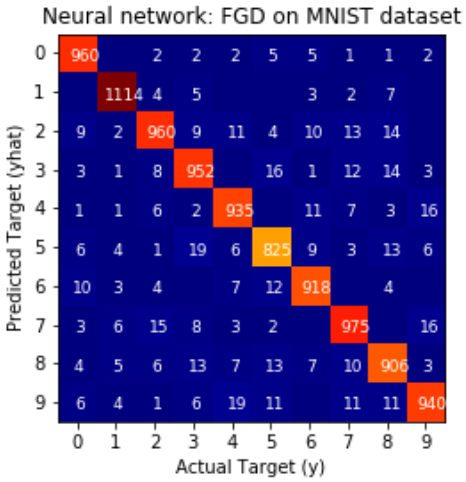

Fig. 5 Average confusion matrix of the testing results overt en presentations of the 10,000 MNIST test set digits using FGD algorithm

merical results of FGD with four 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

1. Y. Bengio. Learning deep architectures for AI. Foundations and trends in Machine Learning, vol. 2, no. 1, pp. 1-27, 2009.
2. M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al., End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316, 2016.
3. A. Botev, G. Lever, D. Barber. Nesterov's accelerated gradient and momentum as approximations to regularised update descent. Neural Networks (IJCNN) 2017 International Joint Conference on, pp. 1899-1903, 2017.
4. N. Cui,. Applying Gradient Descent in Convolutional Neural Networks. Journal of Physics: Conf. Series, 1004, 012027, 2018.
5. A. Chen, X. Xu, S. Ryu and Z. Zhou. A self-adaptive Armijo stepsize strategy with application to traffic assignment models and algorithms. Transportmetrica A: Transport Science, Vol. 9, No. 8, 695-712, 2013.
6. A. El Mouatasim, R. Ellaia and J.E. Souza de Cursi. Stochastic perturbation of reduced gradient & GRG methods for nonconvex programming problems. Journal of Applied Mathematics and Computation, 226, 198–211, 2014.
7. El Mouatasim, A. and Wakrim, M., Control Subgradient Algorithm for Image Regularization. Journal of Signal, Image and Video Processing (SIViP), 9, 275–283, 2015.
8. El Mouatasim, A., Control Proximal Gradient Algorithm for ℓ_1 Regularization Image. Journal of Signal, Image and Video Processing (SIViP), to appear, 2019.
9. B. Evans, H. Al-Sahaf, B. Xue, M. Zhang. Evolutionary deep learning: A genetic programming approach to image classification. IEEE Congress on Evolutionary Computation, pp. 1–6, 2018.
10. G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, vol. 313, no. 5786, pp. 504-507, 2006.
11. Huang K., Hussain A., Wang Q. and Zhang R., Deep learning: fundamentals, theory and applications. Springer, 2019.
12. Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.
13. P. Li, S. Lee and J. Park. Development of a global batch clustering with gradient descent and initial parameters in colour image classification. IET Image Process., Vol. 13 Iss. 1, pp. 161–174, 2019.
14. G. Liu, L. Xiao, C. Xiong. Image classification with deep belief networks and improved gradient descent. International Conference on Embedded and Ubiquitous Computing, 2017.
15. J. MacLean, J. Tsotsos. Fast Pattern Recognition Using Gradient-Descent Search in an Image Pyramid. 0–7695–0750–6/00 IEEE 2000.
16. Nesterov, Y.: Introduction lectures on convex optimization : A basic course. Vol. 87. Springer (2004).
17. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge. International Journal of Computer Vision, vol. 115, no. 3, pp. 211-252, 2015.
18. Shi B. and Lyengar S.S., Mathematical Theories of Machine Learning: Theory and Applications. Springer, 2020.
19. Singh B.K., Verma K., and Thoke A.S., Adaptive gradient descent backpropagation for classification of breast tumors in ultrasound imaging. Procedia Computer Science 46, 1601 – 1609, 2015
20. Uryas'ev S.P., New Variable-Metric Algorithms for Non-differentiable Optimization Problems. Journal of optimization theory and applications: 71(2): 359–388, 1991.
21. Xinhua L. and Qian Y., Face Recognition based on Deep Neural Network. International Journal of Signal Processing, Image Processing and Pattern Recognition Vol.8, No.10, pp:29–38, 2015.