A Bayesian approach for initialization of weights in backpropagation neural net with application to character recognition

Nadir Murru[1], Rosaria Rossini

Department of Mathematics
University of Turin
Via Carlo Alberto 8/10, Turin, Italy
nadir.murru@unito.it
rossini@di.unito.it

Abstract

Convergence rate of training algorithms for neural networks is heavily affected by initialization of weights. In this paper, an original algorithm for initialization of weights in backpropagation neural net is presented with application to character recognition. The initialization method is mainly based on a customization of the Kalman filter, translating it into Bayesian statistics terms. A metrological approach is used in this context considering weights as measurements modeled by mutually dependent normal random variables. The algorithm performance is demonstrated by reporting and discussing results of simulation trials. Results are compared with random weights initialization and other methods. The proposed method shows an improved convergence rate for the backpropagation training algorithm.

Keywords: backpropagation algorithm; Bayesian statistics; character recognition; Kalman filter; neural network.

1. Introduction

In the last decades, neural networks have generated much interest both from a theoretical point of view and for their several applications in complex problems, such as function approximations, data processing, robotics, computer numerical control. Moreover, neural nets are particularly exploited in pattern recognition and consequently can be conveniently used in the realization of Optical Character Recognition (OCR) software.

An artificial neural network (ANN) is a mathematical model designed as the structure of the nervous system. The model was presented for the first time by McCulloch and Pitts [23] and involves four main components: a set of nodes (neurons), their connections (synapses), an activation function that determines the output of each node and a set of weights associated to the connections.

Initialization of weights heavily affects performances of feedforward neural networks [34], as a consequence many different initialization methods have been studied. Since neural nets are applied to many different complex problems, these methods have fluctuating performances. For this reason, random weight initialization is still

[1]Corresponding author: nadir.murru@gmail.com

the most used method also due to its simplicity. Thus, the study of new weight initialization methods is an important research field in order to improve application of neural nets and deepen their knowledge.

In this paper we focus on feedforward neural nets trained by using the Backpropagation (BP) algorithm, which is a widely used method of training. It is well–known that convergence of BP neural net is heavily affected by initial weights [4], [34], [21], [1].

Different initialization techniques have been proposed for feedforward neural nets, such as adaptive step size method [30] and partial least squares method [22]. Hsiao et al. [15] applied the partial least squares method to BP network. Duch et al. [9] investigated the optimal initialization of multilayered perceptrons by means of clusterization techniques. Varnava and Meade [35] constructed an initialization method for feedforward neural nets by using polynomial bases.

Kathirvalavakumar and Subavathi [18] proposed a method that improves convergence rate exploiting Cauchy inequality and performing a sensitivity analysis. An interval based weight initialization method is presented in [33], where authors used the resilient BP algorithm for testing. Adam et al. [2] treated the problem of initial weights in terms of solving a linear interval tolerance problem and tested their method on neural networks trained with BP algorithm.

Yam et al. [37] evaluated optimal initial weights by using a least squares method that minimizes the initial error allowing convergence of neural net by a reduced number of steps. The method is tested on BP neural net with application to character recognition. Other different approaches can be found in [10], [3], [20], [26], [24] where authors focused on BP artificial neural network.

A comparison among several weight initialization methods can be found in [27], where the authors tested methods on BP network with hyperbolic tangent transfer function.

In this paper, we propose a novel approach based on a Bayesian estimation of initial weights. Bayesian estimation techniques are widely used in many different contexts. For instance, in [7] authors developed a customization of the Kalman filter, translating it into Bayesian statistics terms. The purpose of this customization was to address metrological problems. Here, we extend such an approach in order to evaluate an optimized set of initial weights for BP neural net with sigmoidal transfer function. Through several simulations we show the effectiveness of our approach in the field of character recognition.

The paper is structured as follows. In Section 2, we briefly recall the BP training algorithm. In Section 3 we discuss a novel approach for weight initialization in BP neural nets using a Bayesian approach derived by a customization of the Kalman filter. In Section 4, we discuss the setting of some parameters and we show experimental results on the convergence of BP neural net in character recognition. Our Bayesian weight initialization method is compared with classical random initialization and other methods. A sensitivity analysis on some parameters is also presented here. Section 5 concludes the paper.

2. Overview of Backpropagation training algorithm

In this section we present an overview of the BP training algorithm introducing some notation.

Let us consider a feedforward neural network with L layers. Let $N(i)$ be the number of neurons in layer i, for $i = 1, ..., L$, and $w^{(k)}$ be the weight matrix $N(k) \times$

$N(k-1)$ corresponding to connections among neurons in layers k and $k-1$, for $k = 2, ..., L$. In other words, $w_{ij}^{(k)}$ is the weight of connection between i–th neuron in layer k and j–th neuron in layer $k-1$. In the following, we will consider biases equal to zero for seek of simplicity.

Artificial neural networks are trained over a set of inputs so that the neural net provides a fixed output for a given training input. Let us denote X the set of training inputs and $n = |X|$ the number of different training inputs. An element $\mathbf{x} \in X$ is a vector (e.g., a string of bit 0 and 1) whose length is usually equals to $N(1)$. In the following, bold symbols will denote vectorial quantities.

Let $a_i^{(k,\mathbf{x})}$ be the activation of neuron i in layer k given the input $\mathbf{x}$:

$$\begin{cases} a_i^{(1,\mathbf{x})} = \sigma(x_i) \\ a_i^{(k,\mathbf{x})} = \sigma\left(\sum_{j=1}^{N(k-1)} w_{ij}^{(k)} a_j^{(k-1,\mathbf{x})}\right), \quad k = 2, ..., L \end{cases},$$

where σ is the transfer function. In the following, σ will be the sigmoidal function. Moreover, let us denote $z_i^{(k,\mathbf{x})}$ the weighted input to the activation function for neuron i in layer k, given the input $\mathbf{x}$:

$$\begin{cases} z_i^{(1,\mathbf{x})} = x_i \\ z_i^{(k,\mathbf{x})} = \sum_{j=1}^{N(k-1)} w_{ij}^{(k)} a_j^{(k-1,\mathbf{x})}, \quad k = 2, ..., L \end{cases}.$$

Using vectorial notation, we have

$$\begin{cases} \mathbf{z}^{(1,\mathbf{x})} = \mathbf{x}, \quad \mathbf{a}^{(1,\mathbf{x})} = \sigma(\mathbf{z}) \\ \mathbf{z}^{(k,\mathbf{x})} = w^{(k)} \mathbf{a}^{(k-1,\mathbf{x})}, \quad \mathbf{a}^{(k,\mathbf{x})} = \sigma(\mathbf{z}^{(k,\mathbf{x})}), \quad k = 2, ..., L \end{cases}.$$

Finally, let $\mathbf{y}^{(\mathbf{x})}$ be the desired output of the neural network corresponding to input $\mathbf{x}$. In other words, we would like that $\mathbf{a}^{(L,\mathbf{x})} = \mathbf{y}^{(\mathbf{x})}$, when neural net processes input $\mathbf{x}$. Clearly, this depends on weights $w_{ij}^{(k)}$ and it is not possible to know their correct values a priori. Thus, it is usual to randomly initialize values of weights and use a training algorithm in order to adjust their values. In Algorithm 1, the BP training algorithm is described.

3. Bayesian weight initialization based on a customized Kalman filter technique

The Kalman filter [17] is a well–established method to estimate the state $\mathbf{w}_t$ of a dynamic process at each time t. The estimation $\tilde{\mathbf{w}}_t$ is obtained balancing prior estimations and measurements of the process $\mathbf{w}_t$ by means of the Kalman gain matrix. This matrix is constructed in order to minimize the mean–square–error $\mathbb{E}[(\tilde{\mathbf{w}}_t - \mathbf{w}_t)(\tilde{\mathbf{w}}_t - \mathbf{w}_t)^T]$. Estimates attained by Kalman filter are optimal under such diverse criteria, like least-squares or minimum-mean-square-error, and its practice is developed with application to several fields.

The Kalman filter has been successfully used with neural networks [13]. In this context, training of neural networks is treated as a non–linear estimating problem and consequently the extended Kalman filter is usually exploited in order to derive new training algorithms. Many modifications of the extended Kalman filter exist, thus different algorithms have been developed as, e.g., in [32], [36], [12], [28]. However, extended Kalman filter is computationally complex and needs tuning several parameters that makes its implementation a difficult problem (see, e.g., [16]).

Algorithm 1: Backpropagation training algorithm

Data:
L number of layers
$N(k)$ number of neurons in layer k, for $k = 1, ..., L$
$w_{ij}^{(k)}$ initial weights, for $i = 1, ..., N(k)$, $j = 1, ..., N(k-1)$, $k = 2, ..., L$
X set of training inputs, $n = |X|$
$\mathbf{y}^{(\mathbf{x})}$ desired output for all training inputs $\mathbf{x} \in X$
η learning rate
Result: $w_{ij}^{(k)}$ final weights, for $i = 1, ..., N(k)$, $j = 1, ..., N(k-1)$, $k = 2, ..., L$, such that $\mathbf{a}^{(L,\mathbf{x})} = \mathbf{y}^{(\mathbf{x})}$, $\forall \mathbf{x} \in X$
begin
  while $\exists \boldsymbol{x} \in X : \boldsymbol{a}^{(L,\boldsymbol{x})} \neq \boldsymbol{y}^{(\boldsymbol{x})}$ **do**
    for $\boldsymbol{x} \in X$ **do** `// for each training input`
      $\mathbf{a}^{(1,\mathbf{x})} = \sigma(\mathbf{x})$
      for *k=2,...,L* **do**
        $\mathbf{z}^{(k,\mathbf{x})} = w^{(k)}\mathbf{a}^{(k-1,\mathbf{x})}, \quad \mathbf{a}^{(k,\mathbf{x})} = \sigma(\mathbf{z}^{(k,\mathbf{x})})$
      $\mathbf{d}^{(L,\mathbf{x})} = (\mathbf{a}^{(L,\mathbf{x})} - \mathbf{y}^{(\mathbf{x})}) \odot \sigma'(\mathbf{z}^{(L,\mathbf{x})})$, `// ⊙ componentwise product`
      for *k=L-1,...,2* **do**
        $\mathbf{d}^{(k,\mathbf{x})} = ((w^{(k+1)})^T \mathbf{d}^{(k+1,\mathbf{x})}) \odot \sigma'(\mathbf{z}^{(k,\mathbf{x})})$ `// right superscript` T `stands for transpose operator`
    for *k=L,...,2* **do**
      $w^{(k)} = w^{(k)} - \frac{\eta}{n} \sum_{\mathbf{x} \in X} \mathbf{d}^{(k,\mathbf{x})} (\mathbf{a}^{(k-1,\mathbf{x})})^T$

In this section, we show that classical Kalman filter could be used in place of the extended version, constructing a simplified Kalman filter used in combination with BP algorithm in order to reduce computational costs. The motivations about using Kalman filter and proposing a novel approach can be summarized as follows: Kalman filter is widespread in several applied fields in order to optimize performances (including neural networks); it produces optimal estimations under diverse and well–established criteria; it has been used with neural networks mainly in the extended version with the problems above specified.

Let the dynamic of the process be described by the following equation:

$$\mathbf{w}_{t+1} = A_t \mathbf{w}_t + B_t \mathbf{u}_t + \mathbf{p}_t \tag{1}$$

where $\mathbf{u}_t, \mathbf{p}_t$ are the optional control input and the white noise, respectively. Matrices A_t, B_t are used to relate the process state at the step $t+1$ to the t–th process state and to the t–th control input, respectively.

We now introduce the (direct) measurement values of the process $\mathbf{m}_t$ as:

$$\mathbf{m}_t = \mathbf{w}_t + \mathbf{r}_t$$

where $\mathbf{r}_t$ represents measurements uncertainty. Given that, a simplified version of the estimation $\tilde{\mathbf{w}}_t$ produced by the Kalman filter can be represented as follows:

$$\tilde{\mathbf{w}}_t = \mathbf{w}_t^- + K_t(\mathbf{m}_t - \mathbf{w}_t^-) \tag{2}$$

where K_t is the Kalman gain matrix and

$$\mathbf{w}_t^- = A_{t-1}\tilde{\mathbf{w}}_{t-1} + B_{t-1}\mathbf{u}_{t-1}$$

for a given initial prior estimation $\mathbf{w}_0^-$ of $\mathbf{w}_0$.

As stated in the introduction, the Kalman filter has been applied to dimensional metrology by D'Errico and Murru in [7]. The aim of the authors was to minimize the error of measurement instrumentations deriving a simplified version of the Kalman gain matrix by using the Bayes theorem and considering components of each state of the process $\mathbf{w}_t$ as mutually independent normal random variables.

In this section, we extend such an approach in order to optimize weights initialization of neural networks. In particular, we introduce a possible correlations among components of $\mathbf{w}_t$ and we consider the weights as processes whose measurements are provided by random sampling. Furthermore, in the following section, we will specify the construction of some covariance matrices necessary to apply the Kalman filter in this context.

Using the above notation, let $\mathbf{W}_t$ and $\mathbf{M}_t$ be multivariate random variables such that

$$f(\mathbf{W}_t) = \mathcal{N}(\mathbf{w}_t^-, Q_t), \quad f(\mathbf{M}_t|\mathbf{W}_t) = \mathcal{N}(\mathbf{m}_t, R_t), \quad 0 \leq t \leq t_{max} \tag{3}$$

where $\mathcal{N}(\mu, \Sigma)$ is a Gaussian multivariate probability density function with mean μ and covariance matrix Σ. In (3), the random variable $\mathbf{W}_t$ models prior estimations and Q_t is the covariance matrix whose diagonal entries represent their uncertainties and non–diagonal entries are correlations between components of $\mathbf{w}_t^-$. Similarly, $\mathbf{M}_t|\mathbf{W}_t$ models measurements and R_t is the covariance matrix whose entries describe same information of Q_t related to $\mathbf{m}_t$.

The Bayes theorem states that

$$f(\mathbf{W}_t|\mathbf{M}_t) = \frac{f(\mathbf{M}_t|\mathbf{W}_t)f(\mathbf{W}_t)}{\int_{-\infty}^{+\infty} f(\mathbf{M}_t|\mathbf{W}_t)f(\mathbf{W}_t)d\mathbf{W}_t}$$

where $f(\mathbf{W}_t|\mathbf{M}_t)$ is called the posterior density, $f(\mathbf{W}_t)$ the prior density and $f(\mathbf{M}_t|\mathbf{W}_t)$ the likelihood. We have

$$f(\mathbf{W}_t|\mathbf{M}_t) \propto \mathcal{N}(\mathbf{w}_t^-, Q_t)\mathcal{N}(\mathbf{m}_t, R_t) = \mathcal{N}(\tilde{\mathbf{w}}_t, P_t)$$

where

$$\tilde{\mathbf{w}}_t = (Q_t^{-1} + R_t^{-1})^{-1}(Q_t^{-1}\mathbf{w}_t^- + R_t^{-1}\mathbf{m}_t), \quad P_t = (Q_t^{-1} + R_t^{-1})^{-1}.$$

In metrological terms, diagonal entries of P_t can be used for type B uncertainty treatment (see the guide [6]) and the expected value of the posterior Gaussian $f(\mathbf{W}_{t_{max}}|\mathbf{M}_{t_{max}})$ is the final estimate of the process.

We can apply this technique to weights initialization considering processes $\mathbf{w}_t(k)$, for $k = 2, ..., L$, as non–time–varying quantities, i.e.,

$$\mathbf{w}_{t+1}(k) = \mathbf{w}_t(k) + \mathbf{p}_t(k) \tag{4}$$

whose components are the unknown values of weights $w^{(k)}$, for $k = 2, ..., L$, of the neural net such that $\mathbf{a}^{(L,\mathbf{x})} = \mathbf{y}^{(\mathbf{x})}$. Eq. (4) is the simplified version of (1), i.e., describes the dynamics of our processes.

The goal is to provide an estimation of initial weights to reduce the number of steps that allows convergence of BP neural net.

Thus, for each set $w^{(k)}$ we consider initial weights as unknown processes and we optimize randomly generated weights (which we consider as measurements of the processes) with the above approach. In these terms, we derive an optimal initialization of weights by means of the following equations:

$$\begin{cases} \tilde{\mathbf{w}}_t = (Q_t^{-1} + R_t^{-1})^{-1}(Q_t^{-1}\mathbf{w}_t^- + R_t^{-1}\mathbf{m}_t) \\ Q_{t+1} = (Q_t^{-1} + R_t^{-1})^{-1} \\ \mathbf{w}_{t+1}^- = \tilde{\mathbf{w}}_t \end{cases} \tag{5}$$

for t varying from 0 to t_{max} and for each set of weights $w^{(k)}$. For the sake of simplicity we omitted dependence from k in the above equations. In Equations (5), the initial state $\mathbf{w}_0^-$ of $\mathbf{w}_t$ is a prior estimation of $\mathbf{w}_0$ that should be provided. Moreover, covariance matrices Q_0 and R_t must be set in a convenient way. First equation in (5) is the metrological realization of the Kalman–based equation (2). From previous equations, we derive the Kalman gain matrix as

$$K_t = (Q_t^{-1} + R_t^{-1})^{-1}R_t^{-1}.$$

Indeed, we have that $I - K_t = (Q_t^{-1} + R_t^{-1})^{-1}Q_t^{-1}$, where I is the identity matrix.

In the following section we discuss the setting of these parameters and we also provide the results about the comparison of our approach to random initialization with application to character recognition.

4. Numerical results

In this section, we explain the process of our weights initialization and the involved parameters with particular attention to the structure of the covariance matrices, Section 4.1. To evaluate performances of the BP algorithm with random weights initialization (RI) against Bayesian weights initialization (BI) provided by Algorithm 2, we apply neural nets in character recognition. In particular, we discuss the results of our experimental evaluation about the comparison of our approach with a random approach initialization conduct in a field of printed character recognition, taking into account convergence rate, Section 4.2. In this section we use a neural net with 3 layers and sigmoidal activation function. Afterwards, we train BP neural nets (with 3 and 5 layers, using both sigmoidal and hyperbolic tangent activation functions) on the MNIST database for the recognition of handwritten digits, Section 4.3. In these simulations, we also take into account classification accuracy. Finally, we compare BI method with other methods in Section 4.4. These experiments show the advantage that our approach provides in terms of number of steps used to train the artificial neural network.

4.1. Parameters of weights initialization algorithm

The method of weights initialization described in Section 3 is presented in Algorithm 2.

Since we do not have any prior knowledge about processes $\mathbf{w}(k)$, the random variable $\mathbf{W}_0(k)$, which models initial prior estimation, is initialized with the normal distribution $\mathcal{N}(\mathbf{0}, \frac{1}{\epsilon}I)$, where ϵ is a small quantity. In our simulations, we will use a fixed $\epsilon = 10^{-5}$. Note that such an initialization is a standard [32].

Measurements $\mathbf{m}_t(k)$ are obtained by randomly sampling in the real interval $(-h, h)$, for all t. Usually the value of h depends on the specific problem where neural net is applied. Then, we provide a sensitivity analysis on this parameter in the discussion of the results.

The covariance matrix $R_t(k)$ is a symmetric matrix whose entries outside the main diagonal are set equal to 0.7. This choice is based on a sensitivity analysis involving the Pearson coefficient (about correlations of weights) that improves performance of our algorithm. In [7], diagonal entries of covariance matrices were used to describe uncertainty of measurements. In our context, high values of $(R_t(k))_{ii}$ reflect bad accuracy of $(\mathbf{m}_t(k))_i$, i. e., this weight affects output of the neural net being far from the desired output. Thus, we can use values of $\mathbf{d}^{(k,\mathbf{x})}$ to measure inaccuracy of $\mathbf{m}_t(k)$ as follows:

$$(R_t(k))_{ii} = \frac{1}{N(k)N(k-1)} \sum_{\mathbf{x}\in X} \|\mathbf{d}^{(k,\mathbf{x})}\|^2, \quad \forall i, \forall k,$$

where $\|\cdot\|$ stands for the Euclidean norm. Quantity $\|\mathbf{d}^{(k,\mathbf{x})}\|^2$ expresses distance from output and desired output of k–th layer, given the input $\mathbf{x}$. The sum over all $\mathbf{x} \in X$ measures the total inaccuracy of the output of k–th layer. We divide by the number of weights connecting neurons in layers $k-1$ and k so that $(R_t(k))_{ii}$ represents in mean the inaccuracy of a single weight connecting a neuron in layer $k-1$ with a neuron in layer k.

Finally, we iterate Eqs. (5) for a small number of times. Indeed, entries of Q_t rapidly decrease with respect to R_t by means of second equation in (5). Consequently after a few steps, in first equation of (5), $\mathbf{w}_t^-$ has much greater weight than $\mathbf{m}_t$ so that improvements of $\tilde{\mathbf{w}}_t$ could not be significative. In our simulations, we fixed a threshold of $t_{max} = 2$ in order to reduce number of iterations of our algorithm (and consequently number of operations) but obtaining a significant reduction of the step number in the BP algorithm.

4.2. Experiments on latin printed characters

In this section we train the neural network in order to recognize latin printed characters using both BI and RI methods and we compare these results.

The set X of training inputs is composed by 26 characters of the alphabet for 5 different fonts (Arial, Courier, Georgia, Times New Roman, Verdana) with 12 pt. Thus, we have $n = 130$ different inputs. The characters are considered as binary images contained in 15×12 rectangles. Thus, an element $\mathbf{x} \in X$ is a vector of length $15 \cdot 12 = 180$ with components 0 or 1. Figure 1 shows an example of characters of our dataset. A white pixel is coded with 0, a black pixel is coded with 1. The corresponding vector is constructed reading the matrix row–by–row (from left to right, from down to top).

For the experiment presented here, we use a neural net with $L = 3$ layers, $N(1) = 15 \cdot 12 = 180$, $N(3) = 26$. Conventionally, size of first layer is equal to size of training inputs and size of last layer is equal to the number of different desired outputs. In our case, last layer has 26 neurons, as the characters of the latin alphabet. The desired output $\mathbf{y}^{(\mathbf{x})}$ is the vector $(1, 0, 0, ..., 0)$, of length 26, when input $\mathbf{x}$ is the character a (for any font), is the vector $(0, 1, 0, ..., 0)$ when the input is the character b, etc.

For comparison purposes, simulations are performed for different values of parameters $N(2)$, h, and η. We recall that $N(2)$ is the number of neurons in layer 2,

Algorithm 2: Weights initialization algorithm based on Kalman filter

Data:
L number of layers
$N(k)$ number of neurons in layer k, for $k = 1, ..., L$
X set of training inputs, $n = |X|$
$\mathbf{y}^{(\mathbf{x})}$ desired output for all training inputs $\mathbf{x} \in X$
$Q_0(k)$, for $k = 2, ..., L$
$\mathbf{w}_0^-(k)$ prior estimation of $\bar{\mathbf{w}}^{(k)}$, for $k = 2, ..., L$
$\mathbf{m}_0(k)$ measurement of $\bar{\mathbf{w}}^{(k)}$, for $k = 2, ..., L$
$R_0(k)$, for $k = 2, ..., L$
Result: $\tilde{\mathbf{w}}_2(k)$, optimized initial weights for backpropagation algorithm, for $k = 2, ..., L$
begin
  for *k=2,...,L* **do** `// for each set of weights`
    for $t = 0, 1, 2$ **do**
      $\tilde{\mathbf{w}}_t(k) = (Q_t^{-1}(k) + R_t^{-1}(k))^{-1}(k)(Q_t^{-1}(k)\mathbf{w}_t^-(k) + R_t^{-1}(k)\mathbf{m}_t(k))$
      $Q_{t+1}(k) = (Q_t^{-1}(k) + R_t^{-1}(k))^{-1}$
      $w_{t+1}^-(k) = \tilde{\mathbf{w}}_t(k)$
      $\mathbf{m}_{t+1}(k) = Rnd(-h, h)$ `//` $Rnd(-h, h)$ `random sampling in the interval` $(-h, h)$
      $(R_{t+1}(k))_{ii} = \frac{1}{N(k)N(k-1)} \sum_{\mathbf{x} \in X} \|\mathbf{d}^{(k,\mathbf{x})}\|^2, \forall i$
      $(R_{t+1}(k))_{lm} = 0.7, \forall l, m$

$(-h, h)$ is the interval where weights are sampled, and η is the learning rate. To the best of our knowledge these parameters have not a standard initialization, see, e.g., [29].

For each combination of $N(2), h, \eta$, we train the neural net with RI for 1000 different times and we evaluate the mean number of steps necessary to terminate the training. Similarly, we evaluate the mean number of steps when weights are initialized by the Bayesian weights initialization in Algorithm 2.

Figures 2 and 3 depict behavior of the mean number of steps that determine convergence of the BP algorithm with RI, for $N(2) = 70, 80$, respectively. Each figure reports on the abscissa different values of h and we show the behavior for $\eta = 0.6, 0.8, 1, 1.2, 1.4$. Figures 4 and 5 show same information for the BP algorithm with BI.

By figures 2 and 3 (RI), we can observe that for $0.5 \leq h \leq 1$ number of steps, which determine convergence of BP algorithm, generally decreases (with some fluctuation) given any η. Moreover, increasing values of η produce an improvement in the performances. However, such an improvement is less and less noticeable.

By figures 4 and 5 (BI), we can observe that for $0.5 \leq h \leq 1$ performances of BP algorithm improve, similarly to random initialization. For $h > 1$, number of steps, which determine convergence of BP algorithm, increases but slower than the random initialization case. Moreover, increasing values of η produce an improvement in the performances, but it is less noticeable than the case of random initialization.

The improvement in convergence rate due to BI is noticeable at a glance in these

figures. In particular, we can see that BI approach is more resistant than RI with respect to high values of h, in the sense that number of steps increases slower. In fact, for large values of h, weights can range over a large interval. Consequently, RI produces weights scattered on a large interval causing a slower convergence of BP algorithm. On the other hand, BI seems to set initial weights on regions that allow a faster convergence of BP algorithm, despite the size of h. This could be very useful in complex problems where small values of h do not allow convergence of BP algorithm and large intervals are necessary.

Moreover, these figures provide some information about optimal values for h and η that should be reached around 1 and 1.4, respectively.

In Figures 6 and 7 performances of BP algorithm with BI and RI are compared, varying η on the x–axis and using two different values for h, for $N(2) = 70, 80$, respectively. Similarly, figures 8 and 9 compare BI and RI, varying h on the x–axis and using two different values for η, for $N(2) = 70, 80$, respectively.

These figures show that generally BI determines an improvement of the convergence rate of the BP algorithm.

In these simulations, the best performance of BP algorithm with RI is obtained with $h = 0.9$ and $\eta = 1.2$, where the number of steps to terminate the training is 463. The best performance of BP algorithm with BI is obtained with $h = 1.6$ and $\eta = 1.4$, where the number of steps to terminate the training is 339.

We can observe that for $0.4 \leq \eta \leq 1$, BI improves the convergence rate with respect to RI, given any value of h. Furthermore, the improvement of convergence rate is more significant when h increases. For $\eta = 1.2$, BI produces improvements only for $h \geq 1.2$, but in this case we can observe that such improvements are significant. For $\eta = 1.4$ and $\eta = 1.6$, BI produces improvements only for $h = 1.4$ and $h = 1.6$. Such improvements are very significant both compared to corresponding results obtained by RI and compared to results generally obtained by BI.

4.3. Experiments on handwritten digits

In this section, we train neural networks in order to recognize handwritten digits in several cases. The benchmark is composed by handwritten digits of the MNIST database. The MNIST database is composed by 60000 handwritten digits usually used as training data and by 10000 handwritten digits usually used as validation data. A handwritten digit is an image with 28 by 28 pixels (gray scale).

In the following our neural networks have $N_1 = 28 \cdot 28 = 784$ and $N_L = 10$. The desired output $\mathbf{y}^{(\mathbf{x})}$ is the vector $(1, 0, 0, ..., 0)$, of length 10, when input $\mathbf{x}$ is the digit 0; it is the vector $(0, 1, 0, ..., 0)$ when the input is the digit 1; etc.

For the experiments here presented, we use different neural nets. Specifically, we perform experiments for the following neural nets: $L = 3$ and the sigmoidal activation function, $L = 3$ and the hyperbolic tangent activation function, $L = 5$ and the sigmoidal activation function, $L = 5$ and the hyperbolic tangent activation function.

In the above situations, we compare convergence rate of BP algorithm with BI and RI. The convergence rate is evaluated performing 100 different experiments (for each method and situation) and computing the mean value of the steps necessary to achieve the convergence. Moreover, we will also take into account accuracy obtained by these methods testing the trained neural networks on the recognition of 10000 handwritten digits in the MNIST validation test.

In Figures 10, 11, 12 and 13 performances of BP algorithm with BI and RI methods are compared training a neural net with 3 layers, for $N_2 = 70$, on the

first 20000 images contained in the MNIST training set. Specifically, in Figures 10 and 11, we have set $h = 1$, varying η on the x–axis, and we have used sigmoidal and hyperbolic tangent function, respectively. In Figures 12 and 13, we have set $\eta = 3.5$, varying h on the x–axis, and we have used sigmoidal and hyperbolic tangent function, respectively.

Figures 14, 15 and 16 show behavior of BP algorithm with BI and RI methods for a neural network with 5 layers. We have set $N_2 = 50$, $N_3 = 40$, $N_4 = 80$ (note that this parameters have not been optimized, thus different deep neural nets could obtain better performances). In Figures 14, 15, we vary h on the x–axis for $\eta = 1.4$ and $\eta = 2.5$, respectively. In Figure 16, we vary η on the x–axis for $h = 1.4$. For all the above situations we have used the hyperbolic tangent as activation function.

These experiments confirm performances observed in the previous sections. Indeed, BI generally determines an improvement of the convergence rate of the BP algorithm with respect to RI.

In Figures 10 and 11, BI has a worst performance than RI only for $\eta = 3.5$ and we can observe that we have significant improvement of convergence rate for low values of η. Thus, in Figures 12 and 13 we have tested our method in situations where it seems to have poor performances (i.e, for high values of η). Specifically we used $\eta = 3.5$, varying h on x–axis from 0.6 to 1.4. In these simulations, the results are good: for $h \leq 1.2$ BI determines a faster convergence than RI. Moreover, let us observe that best performances are generally obtained when $\eta \leq 3$ and $h \leq 1.2$ for both BI and RI. Thus, the use of high values of η and h is not suitable in this context.

Figures 14, 15 and 16 show that BI method generally improves convergence rate of BP algorithm also when deep neural networks are used. In this cases, we see that when h increases, the distance between number of steps to achieve convergence with BI and RI is more marked in favor of BI method.

Finally, in Tables 1 and 2 we have analyzed classification accuracy of neural networks trained using the BI against RI. In Table 1, we have tested neural networks in the recognition of the 10000 digits of the MNIST validation set, when the training on the first 20000 digits of the MNIST training set is terminated. In Table 2, we have tested neural networks in the recognition of the 10000 digits of the MNIST validation set, after 300 steps of training on the 60000 digits of the MNIST training set. We can observe that percentage of recognized digits is generally greater when BI is used. This result could be expected for simulations reported in Table 2, since after the same number of steps the neural network with BI recognizes a greater number of digits of the training step than neural network with RI (since, neural net with BI converges faster than neural net with RI). Moreover, these results are also confirmed in Table 1 where both neural net with BI and RI have terminated the training.

4.4. Comparison with other initialization methods

In this section, we compare performances of BI method with other ones. We use results provided in [27], where several methods have been tested and compared on different benchmarks from the UCI repository of machine learning databases. Specifically, we perform tests on the following problems: Balance Scale (BAL), Cylinders Bands (BAN), Liver Disorders (LIV), Glass Identification (GLA), Heart Disease (HEA), Imagen Segmentation (IMA). The methods tested in [27] have been developed by Drago and Ridella [8] (Method A), Kim and Ra [19] (Method B),

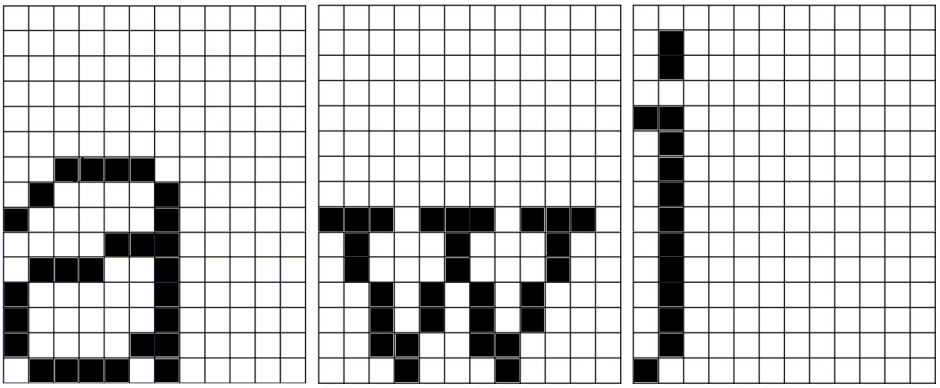

Figure 1: Example of characters of the dataset: letter "a", font arial, pt 12; letter "w", font times new roman, pt 12; letter "j", font georgia, pt 12

Table 1: Percentage of recognized digits in the MNIST validation set. Neural networks trained on first 20000 digits of the MNIST training set.

	$L = 5$, $\eta = 1.5$		$L = 3$, $\eta = 3$	
h	Random in.	Bayes in.	Random in.	Bayes in.
0.7	85	87	91	92
0.8	81	85	92	92
0.9	81	82	90	92
1	78	79	88	91
1.1	86	86	92	92
1.2	80	80	90	91
1.3	82	81	85	89
1.4	77	76	81	81

Table 2: Percentage of recognized digits in the MNIST validation set. Neural networks trained for 300 steps on the MNIST training set.

	$L = 5$, $h = 0.8$		$L = 3$, $h = 1$	
η	Random in.	Bayes in.	Random in.	Bayes in.
0.5	92	92	90	91
1	92	95	91	94
1.5	95	96	94	93
2	93	95	93	93
2.5	92	96	95	94
3	90	88	95	96

Palubinskas [25] (Method C), Shimodaira [31] (method D), Yoon et al. [38] (Method E). Note that in [27] these methods are labeled in a different way.

In Table 3, we report the mean number of steps to achieve convergence with the BP algorithm (30 different trials are performed). Results of Methods A, B, C, D, E and RI are reported from [27]. The BI method is tested with $h = 0.05$ (since in [27] weights are sampled in the interval $[-0.05, 0.05]$) and $\eta = 2$.

We can see that methods B and D have better performances than BI method in Balance Scale problem, whereas only method D converges faster than BI in Cylinder Bands and Liver Disorders problems. In the remaining problems, BI method provides the best performances. We can observe that, as stated in [27], Method D needs determining several parameters by a trial and error procedure. Indeed, here we only reported the best performances of Method D obtained in [27], where the method is tested with several different values of the parameters. On the contrary, BI method does not need tuning extra parameters.

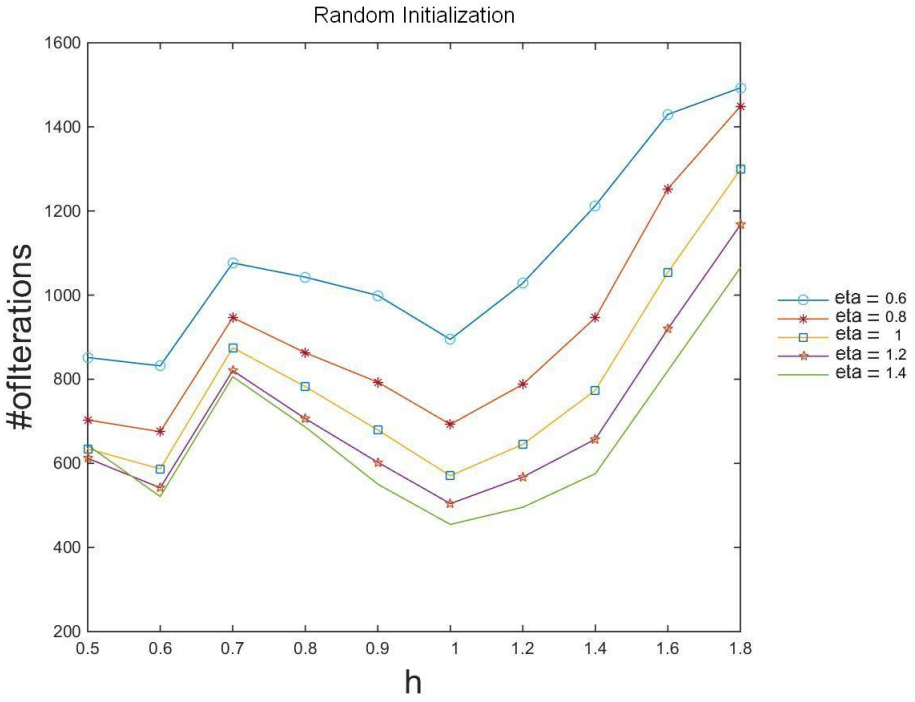

Figure 2: Convergence rate of backpropagation algorithm with random weight initialization with $N(2) = 70$ applied to recognition of latin printed characters

Table 3: Mean number of steps of backpropagation algorithm to converge with different initialization methods applied to different problems.

Method \ Problem	**BAL**	**BAN**	**LIV**	**GLA**	**HEA**	**IMA**
Meth. RI	120	800	1300	111	220	710
Meth. A	130	600	1600	230	200	1090
Meth. B	80	720	1300	150	320	1010
Meth. C	120	700	2800	160	430	950
Meth. D	80	470	500	91	290	970
Meth. E	270	800	2100	300	500	1040
Meth. BI	89	523	925	84	161	459

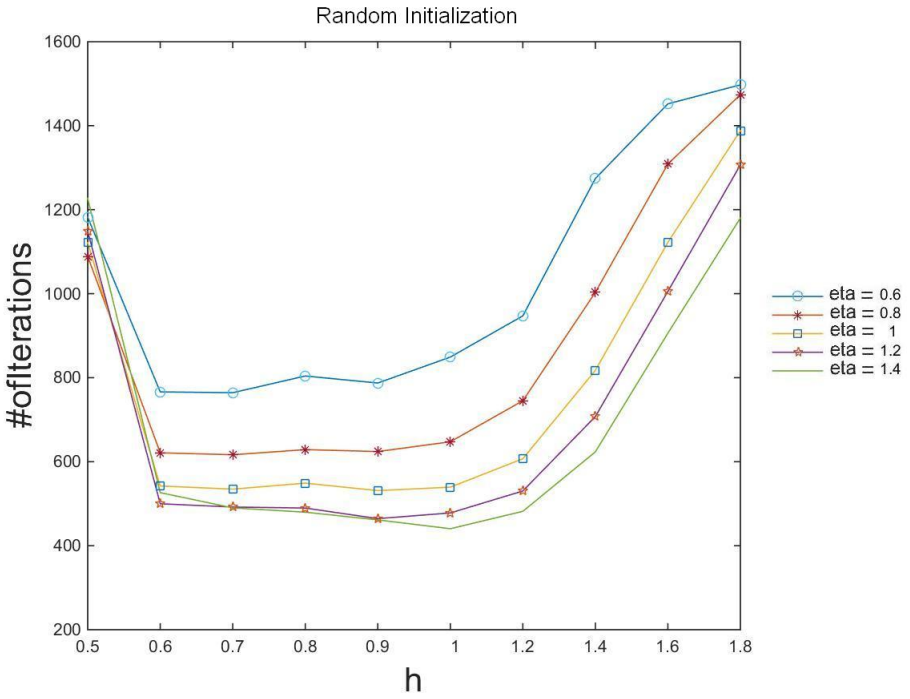

Figure 3: Convergence rate of backpropagation algorithm with random weight initialization with $N(2) = 80$ applied to recognition of latin printed characters

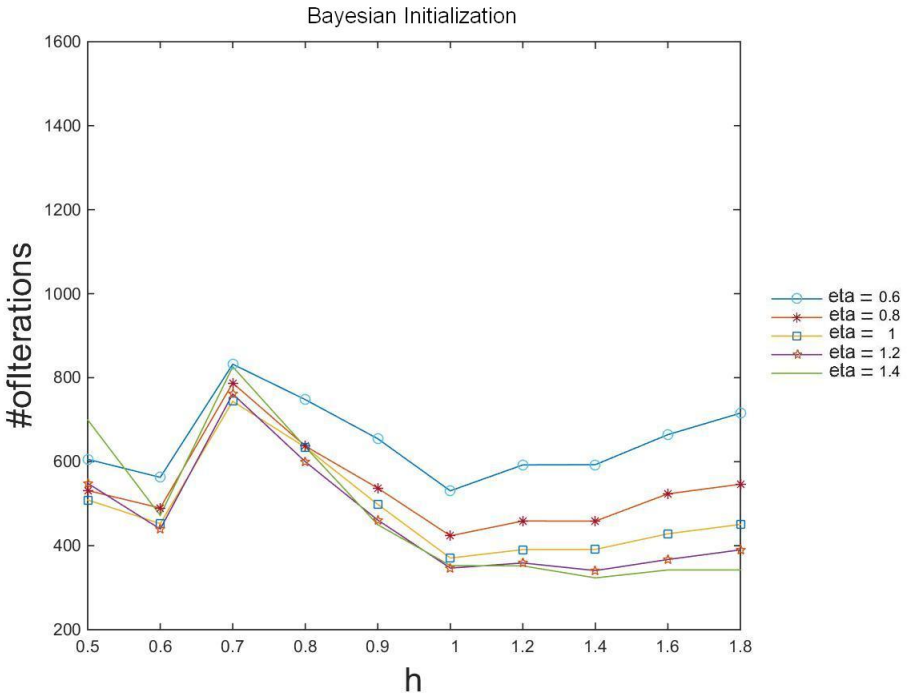

Figure 4: Convergence rate of backpropagation algorithm with Bayesian weight initialization with $N(2) = 70$ applied to recognition of latin printed characters

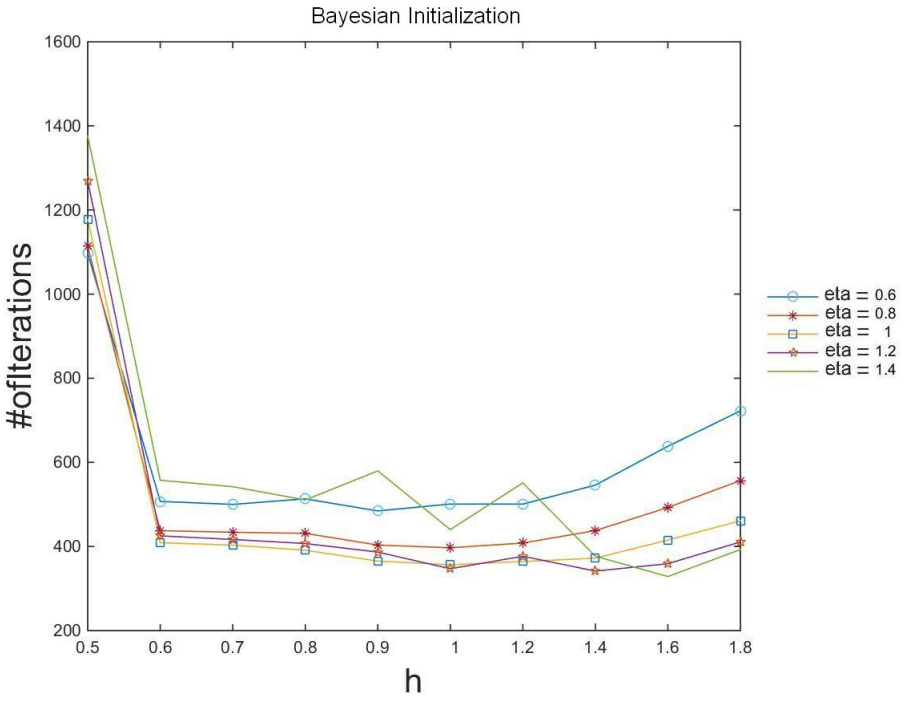

Figure 5: Convergence rate of backpropagation algorithm with Bayesian weight initialization with $N(2) = 80$ applied to recognition of latin printed characters

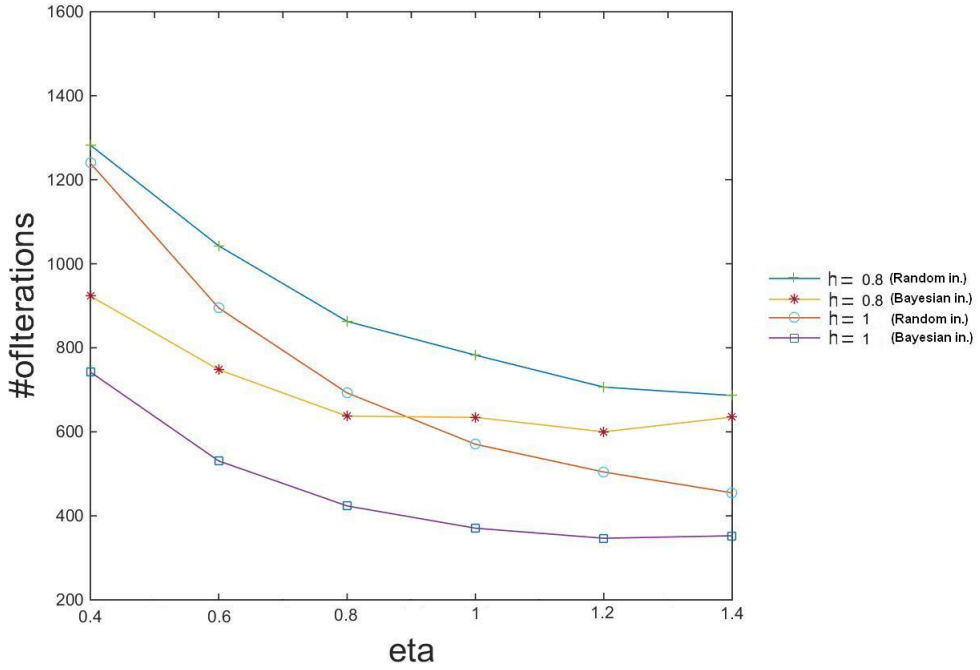

Figure 6: Comparison between Bayesian and random weights initialization with $N(2) = 70$ and η varying on x–axis applied to recognition of latin printed characters

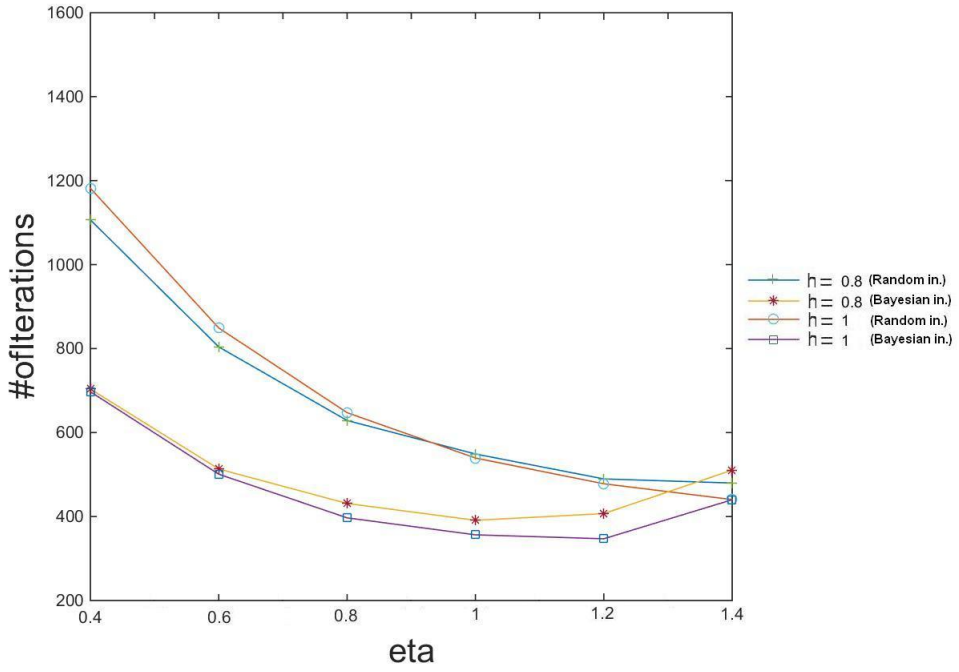

Figure 7: Comparison between Bayesian and random weights initialization with $N(2) = 80$ and η varying on x–axis applied to recognition of latin printed characters

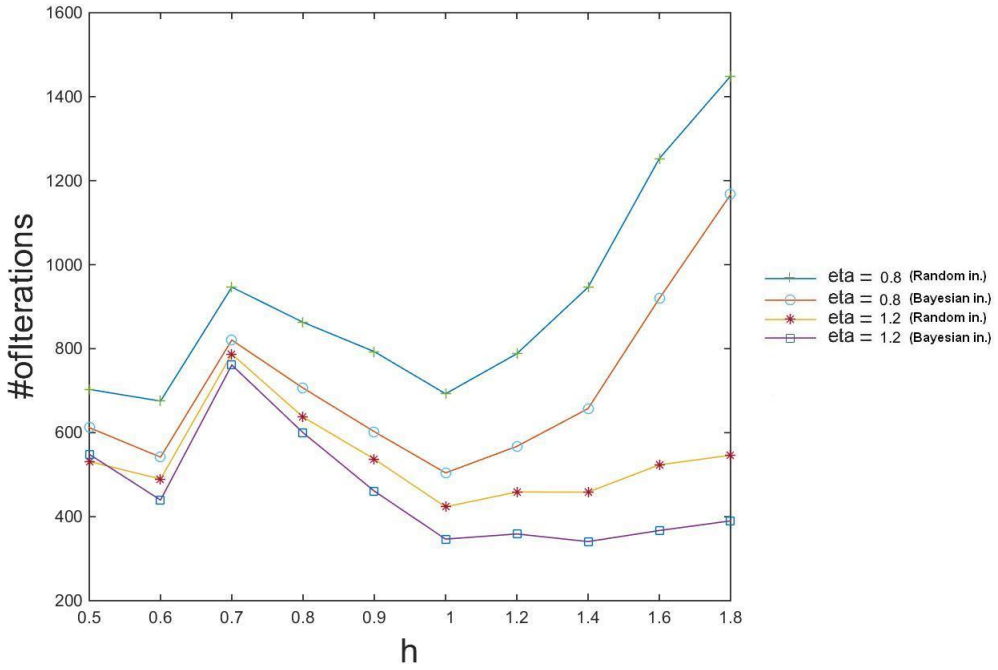

Figure 8: Comparison between Bayesian and random weights initialization with $N(2) = 70$ and h varying on x–axis applied to recognition of latin printed characters

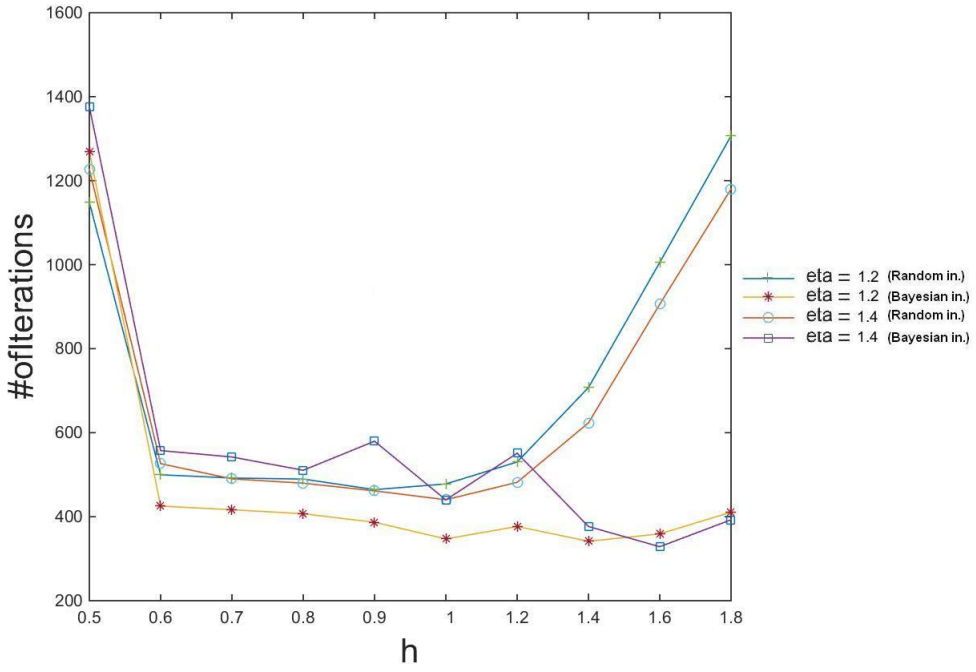

Figure 9: Comparison between Bayesian and random weights initialization with $N(2) = 80$ and h varying on x–axis applied to recognition of latin printed characters

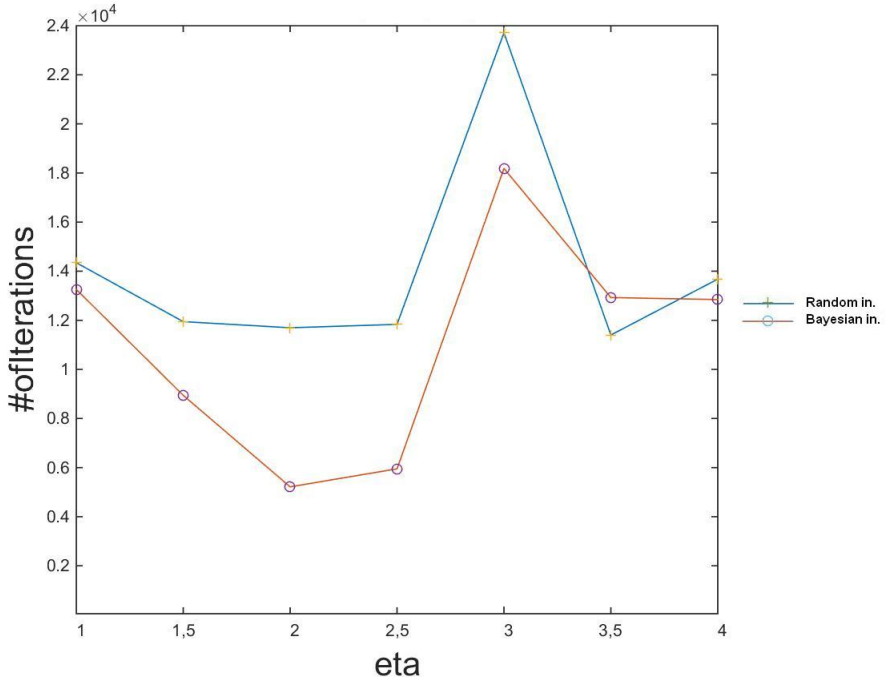

Figure 10: Comparison between Bayesian and random weights initialization with $L = 3$, $N(2) = 70$, $h = 1.5$, η varying on x–axis, sigmoidal activation function, applied to recognition of handwritten digits of the MNIST database

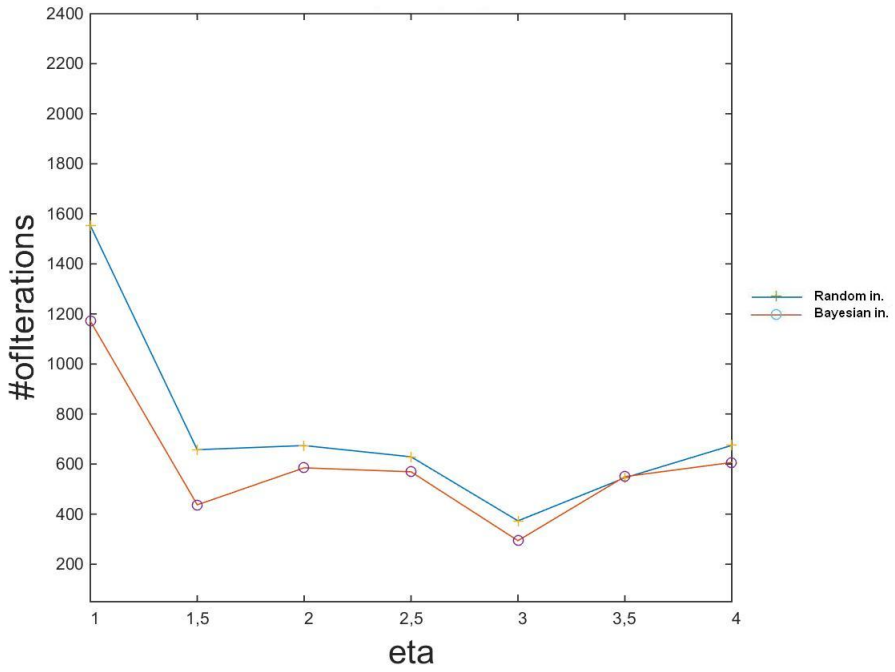

Figure 11: Comparison between Bayesian and random weights initialization with $L = 3$, $N(2) = 70$, $h = 1.5$, η varying on x–axis, hyperbolic tangent activation function, applied to recognition of handwritten digits of the MNIST database

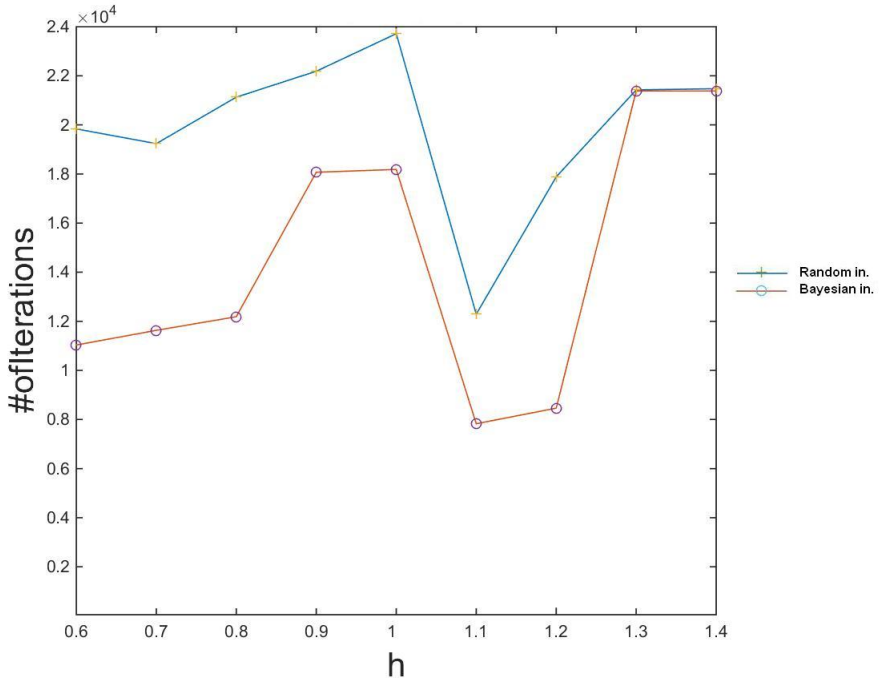

Figure 12: Comparison between Bayesian and random weights initialization with $L = 3$, $N(2) = 70$, $\eta = 3.5$, h varying on x–axis, sigmoidal activation function, applied to recognition of handwritten digits of the MNIST database

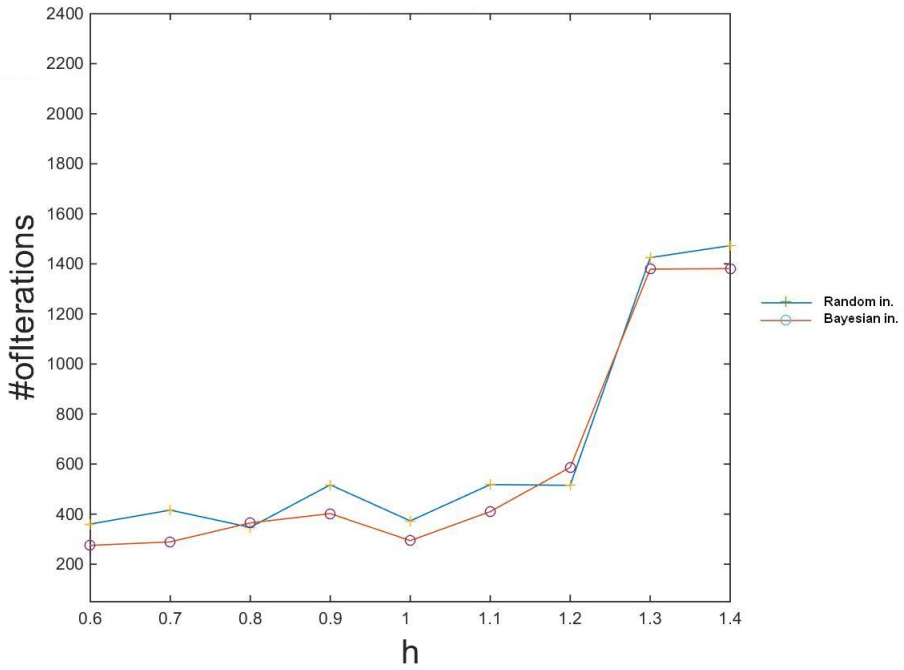

Figure 13: Comparison between Bayesian and random weights initialization with $L = 3$, $N(2) = 70$, $\eta = 3.5$, h varying on x–axis, hyperbolic tangent activation function, applied to recognition of handwritten digits of the MNIST database

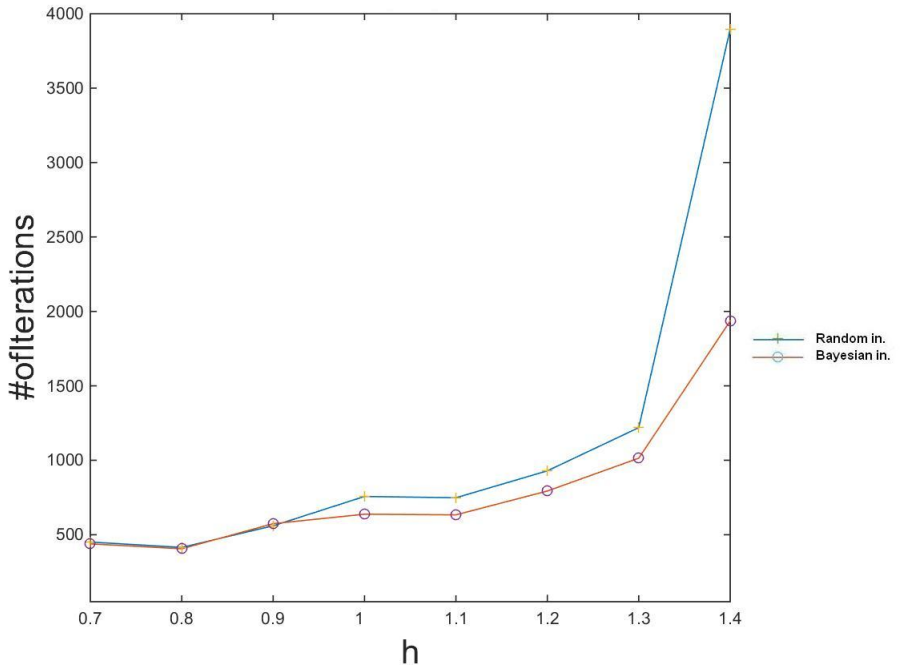

Figure 14: Comparison between Bayesian and random weights initialization with $L = 5$, $N(2) = 50, N(3) = 40, N(4) = 80$, $\eta = 1.4$, h varying on x–axis, hyperbolic tangent activation function, applied to recognition of handwritten digits of the MNIST database

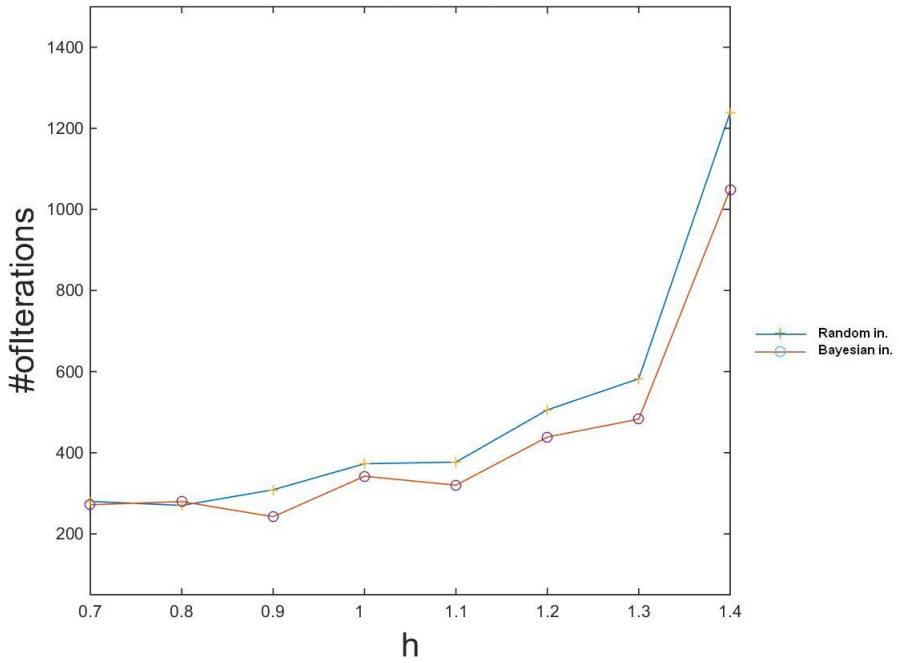

Figure 15: Comparison between Bayesian and random weights initialization with $L = 5$, $N(2) = 50, N(3) = 40, N(4) = 80$, $\eta = 2.5$, h varying on x–axis, hyperbolic tangent activation function, applied to recognition of handwritten digits of the MNIST database

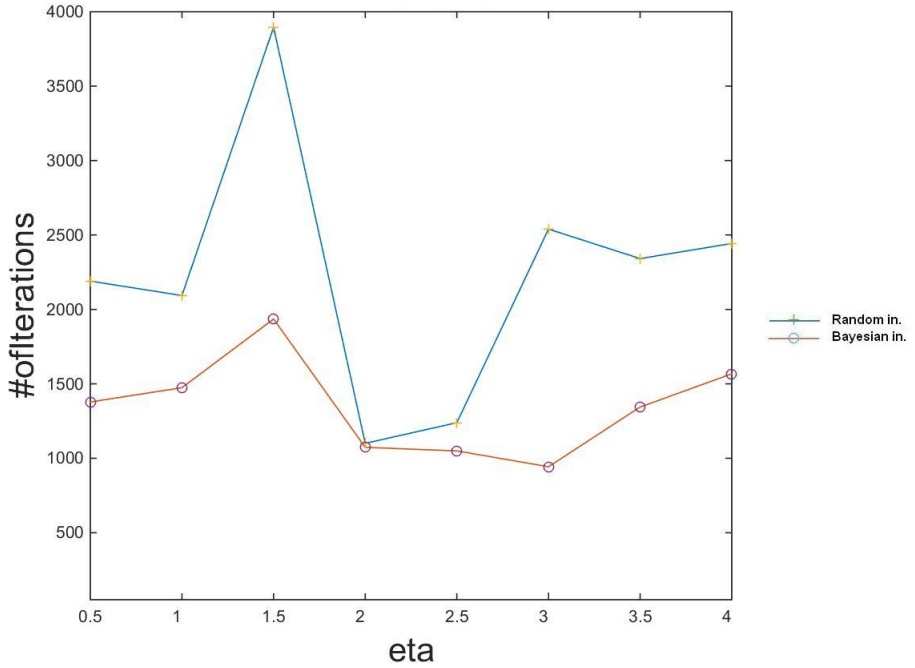

Figure 16: Comparison between Bayesian and random weights initialization with $L = 5$, $N(2) = 50, N(3) = 40, N(4) = 80$, $h = 1.4$, η varying on x–axis, hyperbolic tangent activation function, applied to recognition of handwritten digits of the MNIST database

5. Conclusion and future work

In this paper, the problem of convergence rate of the backpropagation algorithm for training neural networks has been treated. A novel method for the initialization of weights in the backpropagation algorithm has been proposed. The method is mainly based on an innovative use of the Kalman filter with an original metrological approach. A simulation study has been carried on to show the benefits of the proposed method with respect to random weights initialization, applying the neural net in the field of the character recognition. Some comparisons with other initialization methods have been performed. The obtained results are encouraging, and we expect that the new features we introduced are actually relevant in a variety of application contexts of neural nets. In particular, the Bayesian weights initialization could be very useful to solve complex problems where weights need large values of h to ensure convergence of BP algorithm. Looking at perspective advancements, the following issues could be addressed in future works:

- values of entries of covariance matrix $R_t(k)$ should be further optimized by means of a deeper study on correlations among weights of neural networks;
- theoretical analysis of the convergence of the BP algorithm with BI, evaluating and comparing the initial expected error of the neural network whose weights are initialized with the Bayesian approach against the expected error due to random initialization;
- application of the BI method to complex problems needing large values of h;
- recently, the greedy layer–wise unsupervised pre–training has been introduced in order to achieve fast convergence for backpropagation neural networks [14], [5], [11]; it could be interesting to compare this method with BI. Moreover, BI could be exploited in order to improve the pre–training of this method. In fact, the greedy layer–wise unsupervised pre–training involves several operations to initialize the weights in the final/overall deep network. Moreover, the random initialization of weights of neural nets is still the more widespread method. Thus, the study of simple methods that improves random initialization, like the Bayesian approach proposed here, is still an active research field.

6. Acknowledgments

This work has been developed in the framework of an agreement between IRIFOR/UICI (Institute for Research, Education and Rehabilitation/Italian Union for the Blind and Partially Sighted) and Turin University.

Special thanks go to Dott. Tiziana Armano and Prof. Anna Capietto for their support to this work.

We would like to thanks the anonymous referees whose suggestions have improved the paper.

References

[1] S. P. Adam, D. A. Karras, M. N. Vrahatis, *Revisiting the Problem of Weight Initialization for Multi–Layer Perceptrons Trained with Back Propagation*, Advances in Neuro–Information Processing, Lecture Notes in Computer Science, Vol. **5507**, 308–331, 2009.

[2] S. P. Adam, D. A. Karras, G. D. Magoulas, M. N. Vrahatis, *Solving the linear interval tolerance problem for weight initialization of neural networks*, Neural Networks, Vol. **54**, 17–37, 2014.

[3] R. Asadi, N. Mustapha, N. Sulaiman, *Training Process Reduction Based on Potential Weights Linear Analysis to Accelerate Back Propagation Network*, International Journal of Computer Science and Information Security, Vol. **3**, No. **1**, 229–239, 2009.

[4] R. Battiti, *First– and Second–Order Methods for Learning: Between Steepest Descente and Newton's Method*, Neural Computation, Vol. **4**, 141–166, 1992.

[5] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, *Greedy layer–wise training of deep networks*, Advances in Neural Information Processing Systems, Vol. **19**, 153–160, 2007.

[6] BIPM, IEC, IFCC, ISO, IUPAC, IUPAP, and OIML Evaluation of measurement data–guide to the expression of uncertainty in measurement (GUM 1995 with minor corrections) JCGM 100: 2008.

[7] G. E. D'Errico, N. Murru, *An Algorithm for Concurrent Estimation of Time–Varying Quantities*, Meas. Sci. Technol., Vol. **23**, Article ID **045008**, 9 pages, 2012.

[8] G. P. Drago, S. Ridella, *Statistically Controlled Activation Weight Initialization (SCAWI)*, IEEE Transactions. on Neural Networks, Vol. **3**, No. **4**, 627–631, 1992.

[9] W. Duch, R. Adamczak, N. Jankowski, *Initialization and Optimization of Multilayered Perceptrons*, Proceedings of the 3rd Conference on Neural Networks, Kule, Poland, 105–110, October 1997.

[10] D. Erdogmus, O. F. Romero, J. C. Principe, *Linear–Least–Squares Initialization of Multilayer Perceptrons through Backpropagation of the Desired Response*, IEEE Transactions on Neural Networks, Vol. **16**, No. **2**, 325–336, 2005.

[11] D. Erhan, Y. Bengio, A. Courville, P. A. Manzagol, P. Vincent, S. Bengio, *Why does unsupervised pre–training help deep learning?*, The Journal of Machine Learning Research, Vol. **11**, 625–660, 2010.

[12] F. Heimes, *Extended Kalman filter neural network training: experimental results and algorithm improvements*, IEEE International Conference on Systems, Man, and Cybernetics, Vol. **2**, 1639–1644, 1998.

[13] S. Haykin, *Kalman filtering and neural networks*, John Wiley and Sons, Inc., 2001.

[14] G. E. Hinton, R. R. Salakhutdinov, *Reducing the dimensionality of data with neural networks*, Science **313.5786**, 504–507, 2006.

[15] T. C. Hsiao, C. W. Lin, H. K. Chiang, *Partial Least Squares Algorithm for Weight Initialization of Backpropagation Network*, Neurocomputing, Vol. **50**, 237–247, 2003.

[16] S. J. Julier, J. K. Uhlmann, *Unscented filtering and nonlinear estimation*, Proceedings of the IEEE, Vol. **92**, No. **3**, 401–422, 2004.

[17] R. E. Kalman, *A new approach to linear filtering and prediction problems*, Trans. ASME D, J. Basic Eng., Vol. **82**, 35–45, 1960.

[18] T. Kathirvalavakumar, S. J. Subavathi, *A new Weight Initialization Method Using Cauchy's Inequality Based on Sensistivity Analysis*, Journal of Intelligent Learning Systems and Applications, Vol. **3**, 242–248, 2011.

[19] Y. K. Kim, J. B. Ra, *Weight Value Initialization for Improving Training Speed in the Backpropagation Network* Proc. of Int. Joint Conf. on Neural Networks, Vol. **3**, 2396–2401, 1991.

[20] M. Kusy, D. Szczepanski, *Influence of graphical weights interpretation and filtration algorithms on generalization ability of neural networks applied to digit recognition*, Neural Comput and Applic, Vol. **21**, 1783–1790, 2012.

[21] Y. Liu, J. Yang, L. Li, W. Wu, *Negative effects of sufficiently small initial-weights on back–propagation neural networks*, J Zhejiang Univ–Sci C (Comput and Electron), Vol. **13**, No. **8**, 585–592, 2012.

[22] Y. Liu, C. F. Zhou, Y. W. Chen, *Weight Initialization of Feedforward Neural Networks by means of Partial Least Squares*, International Conference on Maching Learning and Cybernetics, Dalian, 3119–3122, 13–16 August 2006.

[23] McCulloch, W. S. and Pitts, W. H. (1943). *A logical calculus of the ideas immanent in nervous activity.* Bulletin of Mathematical Biophysics, 5:115-133.

[24] L. A. Muhammed, *Impact of Mutation Weights on Training Backpropagation Neural Networks*, International Journal of Engineering Research and Technology, Vol. **2**, No. **12**, 3574–3577, 2013.

[25] G. Palubinskas, *Data–driven Weight Initialization of Back–propagation for Pattern Recognition*, Proc. of the Int. Conf. on Artificial Neural Networks, Vol. **2**, 851–854, 1994.

[26] M. Petrini, *Improvements to the backpropagation algorithm*, Annals of the University of Petrosani, Economics, Vol. **12**, No. **4**, 185–192, 2012.

[27] M. F. Redondo, C. H. Espinoza, *Weight Initialization Methods for Multilayer Feedforward*, ESANN 2001 Proocedings – European Symposium on Artificial Neural Networks, Bruges (Belgium), 119–124, April 2001.

[28] I. Rivals, L. Personnaz, *A recursive algorithm based on the extended Kalman filter for the training of feedforward neural models*, Neurocomputing, Vol. **20**, 279–294, 1998.

[29] R. Rojas, *Neural networks. A Systematic Introduction.*, Springer. Berlin Heidelberg NewYork, 1996.

[30] N. N. Schrusolph, *Fast Curvature Matrix–Vector Products for Second Order Gradient Descent*, Neural Computing, Vol. **14**, No. **7**, 1723–1738, 2002.

[31] H. Shimodaira, *A Weight Value Initialization Method for Improved Learning Performance of the Back Propagation Algorithm in Neural Networks*, Proc. of the 6th International Conference on Tools with Artificial Intelligence, 672–675, 1994.

[32] S. Singhal, L. Wu, *Training multilayer perceptrons with the extended Kalman algorithm*, Advances in neural information processing systems **1**, Morgan Kaufmann Publishers Inc., San Francisco, CA, 133–140, 1989.

[33] S. S. Sodhi, P. Chandra, *Interval Based Weight Initialization Method for Sigmoidal Feedforward Artificial Neural Networs*, AASRI Procedia, Vol. **6**, 19–25, 2014.

[34] G. Thimm, E. Fiesler, *High Order and Multilayer Perceptron Initialization*, IEEE Transactions on Neural Networks, Vol. **8**, No. **2**, 349–359, 1997.

[35] T. M. Varnava, A. Meade, *An initialization method for feedforward artificial neural networks using polynomial bases*, Advances in Adaptive Data Analysis, Vol. **3**, No. **3**, 385–400, 2011.

[36] K. Watanabe, S. G. Tzafestas, *Learning algorithms for neural networks with the Kalman filters*, Journal of Intelligent and Robotic Systems, Vol. **3**, Issue **4**, 305–319, 1990.

[37] Y. F. Yam, T. W. S. Chow, C. T. Leung, *A New Method in Determining Initial Weights of Feedforward Neural Networks for Training Enhancement*, Nuerocomputing, Vol. **16**, 23–32, 1997.

[38] H. Yoon, C. Bae, B. Min, *Neural networks using modified initial connection strengths by the importance of feature elements*, Int. Joint Conf. on Systems, Man and Cybernetics, Vol. **1**, 458–461, 1995.