Neural Networks: Different problems require different learning rate adaptive methods

Rémy Allard[*], Jocelyn Faubert

Visual perception and psychophysics laboratory and the NSERC-Essilor Industrial Research Chair, Université de Montréal, 3744 Jean-Brillant, Montréal, Québec, CANADA H3T 1P1

ABSTRACT

In a previous study, a new adaptive method (AM) was developed to adjust the learning rate in artificial neural networks: the generalized no-decrease adaptive method (GNDAM). The GNDAM is fundamentally different from other traditional AMs. Instead of using the derivative sign of a given weight to adjust its learning rate, this AM is based on a trial and error heuristic where global learning rates are adjusted according to the error rates produced by two identical networks using different learning rates. This AM was developed to solve a particular task: the orientation detection of an image defined by texture (the texture task). This new task is also fundamentally different from other traditional ones since its data set is infinite, each pattern is a template used to generate stimuli that the network learns to classify. In the previous study, the GNDAM showed its strength over standard backpropagation for this particular task. The present study compares this new AM to other traditional AMs on the texture task and other benchmark tasks. The results showed that some AMs work well for some tasks while others work better for other tasks. However, all of them failed to achieve a good performance on all tasks.

Keywords: neural network, backpropagation, learning rate, adaptive method, momentum, delta-bar-delta, super SAB, resilient propagation, GNDAM, texture-defined.

1. INTRODUCTION

In a previous study[1], the generalized no-decrease adaptive method (GNDAM) was developed to adjust the learning rate (LR) in artificial neural networks. The GNDAM is fundamentally different from other traditional adaptive methods (AMs). Instead of using the derivative sign of a given weight to adjust its LR, the GNDAM is based on a trial and error heuristic. This new AM was developed to solve a particular problem: the detection of an image orientation defined by texture (the texture task). This task is considered complex since it requires a large network and many iterations before learning emerges. This new task is also fundamentally different from other traditional tasks since each pattern of its data set is a template used to generate a stimulus that the network must learn to classify. Therefore the activation patterns vary from one epoch to another. In the previous study, the GNDAM showed its strength over backpropagation for this particular task. The goal of the present study was to compare this new AM to other traditional AMs with two new tasks and other standard benchmark tasks.

1.1. Algorithms

The backpropagation learning rule[2] is a powerful tool to adjust the weights in artificial neural networks. At time t, each weight (w) is updated by the following rule:

$$\Delta w(t) = -\eta \frac{\partial E}{\partial w}(t). \tag{1}$$

* remy.allard@umontreal.ca; phone (514) 343-6111 ext.8937; http://vision.opto.umontreal.ca

where the constant η is the LR and $E(t)$ the error at the epoch t. The difficulty in using this algorithm resides in finding a good LR given that there is no mathematical rule that determines it and its optimal value is task depend. The typical way to adjust the LR is by trial and error. Usually, the value of 0.1 is first tried. If the network saturates, the LR is decreased. If convergence is too slow, the LR is increased.

A variety of AMs have been developed to self adjust the LR. The weakness of these AMs is that they require other arbitrary parameters that need tuning. In the present study, five AMs were compared: momentum[2], delta-bar-delta[3], super SAB[4], resilient propagation[5] and the generalized no-decrease adaptive method[1]. Bellow, each of them will be presented briefly. A short description of their basic properties will be outlined. Given that the description of each procedure is beyond the scope of this article, certain details about each technique will be omitted.

1.1.1. Momentum (MOM)

MOM does not directly adjust the LR. Instead, this AM adds a term to the weight modification equation. This extra term is a fraction of the previous weight update, which smoothes the changes applied to the weight. Therefore, the learning rule for each weight becomes:

$$\Delta w(t) = -\eta \frac{\partial E}{\partial w}(t) + \alpha \Delta w(t-1). \tag{2}$$

where α is the momentum factor and $0 < \alpha < 1$.

1.1.2. Delta-Bar-Delta (DBD)

DBD directly adjusts the LR at each epoch. Using this AM, each weight has its own LR that varies over time ($\eta(t)$). If the weight correction follows the same direction as the previous ones, the LR is increased by a constant (η^+). If it is not in the same direction, a fraction (η^-) of the LR is subtracted from the LR. Mathematically:

$$\bar{\delta}(t) = (1-\theta)\frac{\partial E}{\partial w}(t) + \theta \bar{\delta}(t-1). \tag{3}$$

$$\Delta \eta(t) = \begin{cases} \eta^+ & \text{if } \bar{\delta}(t-1)\frac{\partial E}{\partial w}(t) > 0 \\ -\eta^- \eta(t) & \text{if } \bar{\delta}(t-1)\frac{\partial E}{\partial w}(t) < 0. \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$\Delta w(t) = -\eta(t)\frac{\partial E}{\partial w}(t). \tag{5}$$

Therefore, when the weight update is constantly in the same direction, the LR increases linearly. However, when the weight update oscillates the LR decreases exponentially.

1.1.3. Super-SAB (SSAB)

There are two major differences between DBD and SSAB. First, SSAB increases and decreases the LR exponentially. Second, the steps that cause the weight derivative to change sign are undone. SSAB iterative steps are the following:
Do a MOM step:

$$\Delta w(t) = -\eta(t)\frac{\partial E}{\partial w}(t) + \alpha \Delta w(t-1). \tag{6}$$

If the derivative sign of a given weight (w) remained unchanged ($\frac{\partial E}{\partial w}(t)\frac{\partial E}{\partial w}(t-1) > 0$), the LR is increased ($\eta(t) = \eta^+ \eta(t-1)$) and the weight is updated ($w(t) = w(t-1) + \Delta w(t)$). If the derivative changed sign, this implies that the last step was too large and a minimum has been missed. In such cases, the LR is decreased

($\eta(t) = \eta^- \eta(t-1)$), the previous step is undone ($w(t) = w(t-1) - \Delta w(t-1)$) and $\Delta w(t)$ is set to 0 to eliminate the momentum influence on further weight updates.

1.1.4. Resilient propagation (RPROP)

RPROP is similar to SSAB since the LR increases and decreases exponentially and the steps that cause the weight derivative to change sign are undone. RPROP differs, however, from most AMs on two points. First, only the derivative sign, and not the magnitude of the derivative, influences the weight correction. Second, the LRs are bound between $\eta_{\min}$ and $\eta_{\max}$.

The first step in the iterative loop consists in updating the LR according to whether or not the derivative sign changed:

$$\eta(t) = \begin{cases} \min(\eta^+ \eta(t-1), \eta_{\max}) & \text{if } \dfrac{\partial E(t)}{\partial w}\dfrac{\partial E(t-1)}{\partial w} > 0 \\ \max(\eta^- \eta(t-1), \eta_{\min}) & \text{if } \dfrac{\partial E(t)}{\partial w}\dfrac{\partial E(t-1)}{\partial w} < 0. \\ \eta(t-1) & \text{otherwise} \end{cases} \tag{7}$$

If the derivative sign changes ($\frac{\partial E}{\partial w}(t)\frac{\partial E}{\partial w}(t-1) < 0$), the previous step is retracted ($\Delta w(t) = -\Delta w(t-1)$) and $\frac{\partial E(t)}{\partial w}$ is set to 0 to prevent a second consecutive LR decrease. Otherwise ($\frac{\partial E}{\partial w}(t)\frac{\partial E}{\partial w}(t-1) \geq 0$), the weight is updated according to the derivative sign:

$$\Delta w(t) = \begin{cases} -\eta(t) & \text{if } \dfrac{\partial E(t)}{\partial w} > 0 \\ +\eta(t) & \text{if } \dfrac{\partial E(t)}{\partial w} < 0. \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$w(t) = w(t-1) + \Delta w(t). \tag{9}$$

1.1.5. Generalized no-decrease adaptive method (GNDAM)

The GNDAM does not provide each weight its own LR. Instead, the weights are separated in different groups and a LR is associated with each one ($\eta_g(t)$ is the LR of the gth group). The LRs are updated one at the time. Therefore, the iterative algorithm loops on each group (g).

The first step consists in creating an identical network where only the LR of the group g is altered:

$$w_i'(t) = w_i(t) \quad \forall i \tag{10}$$

$$\eta_i'(t) = \eta_i(t) \quad \forall i \neq g \tag{11}$$

$$\eta_g'(t) = \eta_g(t) \cdot m_g(t-1) \tag{12}$$

The next step consists in applying backpropagation for $N_g(t)$ iterations (see bellow on how to adjust $N_g(t)$). The last step selects which network ($w(t)$ and $\eta(t)$, or $w'(t)$ and $\eta'(t)$) is going to be kept ($w(t+1)$ and $\eta(t+1)$) and which one is going to be discarded. If the highest LR produced the lowest error rate or if the difference between the error rates of the two networks was not significant, the network with the highest LR is kept ($w(t+1)$ and $\eta(t+1)$), and $m_g(t) = K$. Otherwise, the network with the lowest LR is kept and $m_g(t) = 1/K$.

Although the general concept of this approach is relatively simple, the implementation is more complex given that the number of iterations ($N_g(t)$) used to compare two LRs is critical. If it is too low, the difference will rarely be significant and the learning rate will become too large. If it is too high, the difference will often be significant and simulations showed that the learning rate would become too small. To keep $N_g(t)$ to the critical value that prevents the LR to constantly increase or decrease, the GNDAM uses an FQUEST[1]. This procedure is a modification of the QUEST adaptive method[6] used in psychophysics to evaluate a posterior probability density function (pdf) of a threshold. The FQUEST tries to approximate $N_g(t)$ so that the LR's chance of increasing is 50 %. To perform this, it iteratively updates a pdf ($PDF_g(t)$). To update the pdf, the FQUEST needs to know the probability that the highest LR causes the lowest error rate. This probability is approximated by:

$$p_g(t) = \alpha \cdot p_g(t-1) + (1-\alpha) \cdot I(t). \tag{13}$$

where $0 < \alpha < 1$ and $I(t)$ equals to 1 if the highest LR caused the lowest error rate at time *t* and 0 otherwise. $p_g(t)$ is used to define the desired probability of having a significant difference between the error rates produced by the two networks, which directly depends on $N_g(t)$. The goal of the FQUEST is to set $N_g(t)$ so it fixes the chance that the LR increases to 50 %. Since the probability of the LR to increase (0.5) equals the probability that the highest LR causes the smallest error rates ($p_g(t)$) multiplied by the probability of having a significant difference, the desired probability of having a significant difference equals to $0.5/p_g(t)$. Therefore, finding the proper $N_g(t)$ should permit the LRs to remain relatively stable. The function used to approximate the posterior probability density function can be summarized as a weighted product of psychometric functions:

$$PDF_g(t) = PDF_g(t-1)^{1/\sqrt[\lambda]{2}} \cdot \Psi(N_g(t), 0.5/p_g(t), S_g(t)). \tag{14}$$

where λ is the half-life (the number of trials that decays the weight of a given psychometric function by half) and Ψ is a psychometric function that modifies the pdf according to the a new trial (*t*). Ψ depends on the number of iterations ($N_g(t)$), the desired threshold probability of having a significant difference ($0.5/p_g(t)$) and whether or not a significant difference was noted at time *t* ($S_g(t)$). Given that it is beyond the scope of this paper to explain Ψ, Ψ is not explicitly defined in the present paper. The important factor in the preceding function is λ, which weights the different trials. If λ is high, many previous trials will be used to approximate the posterior pdf, which may result in a better pdf approximation if the threshold (the desired number of iterations) is stable. However, if the desired number of iterations changes, the pdf will require more time to readjust itself to the new threshold. With a relatively low λ, the opposite behavior occurs; the posterior pdf easily readjusts itself to a new threshold and its approximation is less precise.

2. METHODOLOGY

2.1. General parameter settings

The network architecture used was a standard feedforward neural network. For simplicity, a maximum of parameters was constant for all tasks. Among these, all input unit activations were scaled between –1 and +1, and the target values were either –1 or +1. All the non-input units were connected to a bias unit. The weights were initialized between –3/√f and 3/√f, where 'f' is the fan-in to the postsynaptic unit. The activation function for all non-input units was the standard hyperbolic tangent[7]:

$$f(net) = a \cdot \tanh(b \cdot net). \tag{15}$$

where $a = 1.716$ and $b = 2/3$.

The network was judged to produce a correct answer if all the output units produced a correct answer. The correctness of an output unit was based on the 40-20-40 rule. For each task and each AM, seven networks were computed on each combination of parameter setting. The total number of iterations was fixed to twenty million for MOM, DBD, SSAB and RPROP, and ten million for the GNDAM since this last AM requires twice the computation time. If, on average, a network caused an error rate greater than twice the optimal a priori error rate[1], the network was stopped before the expected time and was judge to saturate and hence was unable to find a solution. There were two advantages in stopping the training when saturation occurred. First, great computation time was saved and second, given that saturating networks often produce extreme output activations, omitting saturating networks permitted to discard networks that found a correct solution by chance.

2.2. Task dependent parameter settings

Each of the five algorithms was tested on four different tasks: parity-bit, encoder, texture detection and luminance detection. The present section summarizes these tasks and their parameter settings.

2.2.1. Parity-bit

The parity-bit task consisted of detecting if the number of input units activated was either even or odd. In our study, the input layer was composed of nine units with activation of either –1 (not active) or +1 (active). Therefore, the number of different activation patterns was 512 (2^9). To perform this task, only one output unit was required. The target value of this unit was +1 or –1, depending on the number of input unit activated, which was either even or odd. Only one hidden layer of nine units was used.

2.2.2. N-M-N Encoder

The encoder task consisted of reproducing the same output activation pattern as the input one. The activation patterns were simple, one unit was active (+1) and all the others were not (-1). The complexity of this task resided in the fact that the number of hidden units was less than the number of input and output units ($M<N$). To solve this task, the network had to learn to encode the input activation pattern produced by the N input units with the M hidden units and then decode the hidden layer pattern to reproduce the same pattern as the input pattern. Seven hidden units (M=7) were used to encode the patterns produced by the sixteen input units (N=16).

2.2.3. Texture

The task consisted in detecting the orientation, either horizontal or vertical, of stripes defined by texture in an image. The stripes were defined by a square wave of two different textures (a low and a high contrast textures). Since eight-by-eight pixel images were used, the square wave of each image could have one of three different periods: 2, 4 or 8 pixels. The square wave was not required to start at the beginning of a cycle; it was shifted in any possible phase. Consequently, the data set consisted of 48 patterns (3 possible frequencies, 8 phases and 2 orientations). These patterns used random values, and therefore, from one epoch to another, generated different images. Each pixel of the image corresponded to an input layer unit. The intensity of the pixels varied from –1 (black) to +1 (white). Each texture was defined by an amplitude of luminance variation (T_1 and T_2). Both textures had the same luminance average (L). The intensity of a pixel belonging to texture i (i=1 or 2) was randomly chosen between L- T_i and L+ T_i. In the present study, for each image the two texture amplitudes (T_1 and T_2) were 0.1 and 1 and the luminance average (L) was 0. Figure 1 shows texture-defined stimuli examples.

One output unit was used with the target value of either –1 or +1 depending on the orientation of the stimulus presented. Two hidden layers were used; the first one had 128 units and the other had 64 units.

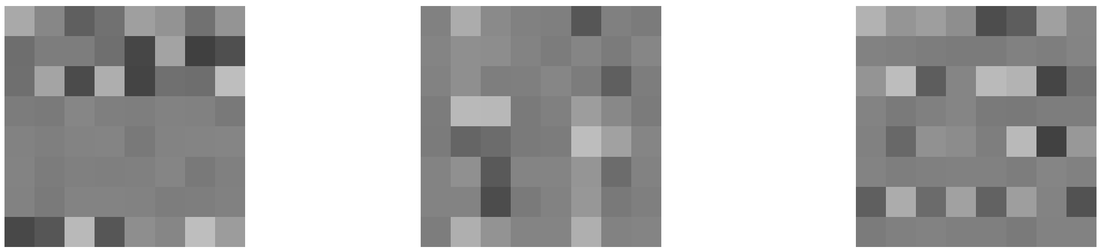
(a) (b) (c)

Figure 1. (a) is horizontally oriented and the cycle has a period of 8. (b) is vertically oriented and the cycle has a period of 4. (c) is horizontally oriented and the cycle has a period of 2.

2.2.4. Luminance

The luminance task was very similar to the texture task. The only difference was that the stripes were defined by a luminance modulation instead of the texture modulation. An image had two luminance values (L_1 and L_2) and one constant texture amplitude (T). The texture amplitude (T) was set to 0.5 and the luminance values (L_1 and L_2) were –0.2 and 0.2. The intensity of a pixel belonging to luminance *i* (*i*=1 or 2) was randomly chosen between L_i–T and L_i+T. Figure 2 shows luminance-defined stimuli examples. One hidden layer of 16 units was used.

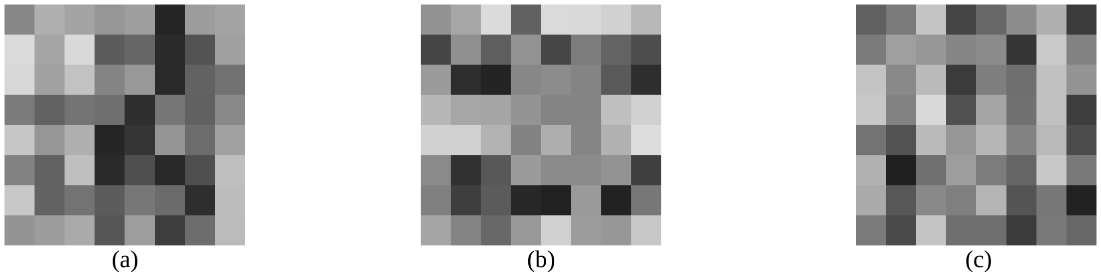
Figure 2. (a) is vertically oriented and the cycle has a period of 8. (b) is horizontally oriented and the cycle has a period of 4. (c) is vertically oriented and the cycle has a period of 2.

2.3. Algorithm dependent parameter settings

The goal of an AM is to self-adjust the learning rate. However, the main problem with AMs is that they require many parameters that are, like the LR, problem dependent. Given that it is impossible to test all possible parameter combinations, comparing AMs may be difficult. Usually, to compare AMs for a given task, authors affirm that they tried to find "good" parameter settings. Using this methodology, it is unlikely that these authors find the optimal parameter setting for each AM. They defend this methodology by arguing that the easiness of an AM to find good parameter values is part of the strength and weakness of the AM.

An important factor that may introduce a bias using this methodology, is the time spent to find good parameter values. If the authors spent more time to adjust the parameters of a given AM, chances are that those parameters may be better tuned. Consciously or not, authors using this methodology can easily introduce a bias in their study. Given that the number of parameters that need tuning makes it virtually impossible to perform an extensive search on all the parameters, in the current study we chose to limit the extensive search to two parameters. An extensive search was performed on the LR for MOM and on the initial LR for the other AMs. This parameter was chosen since it is the only one that is shared by all the AMs. The extensive search for the initial LR was done with $\log_4$ steps. The other parameter (further called the free parameter), for which an extensive search was applied, varied from an AM to another. For each AM, the most critical parameter was selected. The extensive search for a good free parameter value was done using $\log_2$ steps, with the exception of MOM, for which its free parameter was set to values often used in the literature.

For MOM, the only remaining parameter is the momentum factor. By necessity, this parameter was selected as the free parameter. Its tested values were 0, 0.5, 0.7, 0.9 and 0.99, which are often used in the literature[2,7-8]. Note that when the momentum factor is null, MOM is equivalent to backpropagation.

For DBD, SSAB and RPROP, the increase factor (η^+) was the second parameter for which an extensive search was performed. For all three AMs, the decrease factor (η^-) was set to its default value: 0.5. The truncation parameters ($\eta_{\min}$ and $\eta_{\max}$) for RPROP were set to their default values of 0.000001 and 50. SSAB momentum factor was also set to its default value of 0.7.

For the GNDAM, the FQUEST half lifetime pdf parameter (λ) was set as the free parameter. The factor (α) used to approximate the posterior probability that the highest LR causes a lower error rate was set to 0.9. The constant K that separates two compared LRs was set to 2. And all the weights corresponding to a layer were defined as a group.

3. RESULTS

3.1. Parity-bit

No parameter combination using the GNDAM was found that enabled the network to solve the task at least once. All the other AMs showed many parameter combinations that could solve the task several times although none solved the task all seven times. As shown in table 1, MOM and DBD gave the better results achieving a performance of 6/7 followed by RPROP (4/7) and then SSAB (2/7).

MOM					
	0	0.5	0.7	0.9	0.99
4^{-4}	2	2	3	3	0
4^{-5}	0	1	3	6	2
4^{-6}	0	0	0	3	5
4^{-7}	0	0	0	0	6
4^{-8}	0	0	0	0	1

DBD					
	2^{-14}	2^{-13}	2^{-12}	2^{-11}	2^{-10}
4^{-4}	2	3	4	1	2
4^{-5}	3	1	3	4	2
4^{-6}	2	1	1	4	2
4^{-7}	2	1	3	2	2
4^{-8}	3	2	2	3	6

SSAB					
	1.0125	1.025	1.05	1.1	2
4^{-5}	0	2	0	0	0
4^{-6}	2	0	0	0	0
4^{-7}	2	0	0	0	0
4^{-8}	2	0	1	1	0
4^{-9}	0	0	0	0	0

RPROP					
	1.05	1.1	1.2	1.4	1.8
4^{-4}	2	3	3	1	1
4^{-5}	2	2	2	1	0
4^{-6}	3	2	1	2	1
4^{-7}	3	1	2	2	0
4^{-8}	2	2	3	3	4

Table 1. Performances on the parity-bit task. Each element represents the number of times the AM solve the task (max=7) for a given parameter combination. The lines correspond to different LRs and the columns to different free parameter values.

3.2. Encoder

The GNDAM, RPROP and SSAB all solved the task seven times out of seven with at least one parameter combination. For DBD and MOM, although many parameter combinations solved the task, none resulted in a one hundred percent efficacy.

MOM					
	0	0.5	0.7	0.9	0.99
4^{-5}	3	4	2	2	0
4^{-6}	4	2	4	3	0
4^{-7}	4	3	4	3	3
4^{-8}	4	2	4	4	4
4^{-9}	0	1	3	4	3

DBD					
	2^{-7}	2^{-6}	2^{-5}	2^{-4}	2^{-3}
4^{-5}	3	3	3	4	4
4^{-6}	4	4	3	4	5
4^{-7}	3	3	3	3	4
4^{-8}	5	4	3	4	2
4^{-9}	3	4	4	3	3

SSAB					
	1.0031	1.0063	1.0125	1.025	1.05
4^{-5}	6	6	6	5	6
4^{-6}	5	6	7	5	7
4^{-7}	4	6	6	6	4
4^{-8}	4	5	6	5	4
4^{-9}	4	6	6	4	7

RPROP					
	1.05	1.1	1.2	1.4	1.8
4^{-4}	6	6	5	5	5
4^{-5}	7	6	4	6	7
4^{-6}	4	6	5	6	7
4^{-7}	6	6	1	6	6
4^{-8}	4	7	6	5	5

GNDAM					
	2^{0}	2^{1}	2^{2}	2^{3}	2^{4}
4^{-4}	2	6	7	6	4
4^{-5}	4	5	7	4	3
4^{-6}	4	5	5	7	4
4^{-7}	3	4	5	6	5
4^{-8}	3	4	5	4	4

Table 2. Performances on the encoder task. Each element represents the number of times the AM solve the task (max=7) for a given parameter combination. The lines correspond to different LRs and the columns to different free parameter values.

3.3. Texture

No parameter combination was found that could solve the texture task using MOM, SSAB or RPROP. DBD solved the task only if the initial LR was 4^{-2}. With the proper initial LR the other DBD parameter (incremental constant) showed greater flexibility, permitting a perfect performance on a width value range (2^{-8} to 2^{-12}). Table 3 shows the GNDAM performance for the same task. As shown in this table, both parameters demonstrated great flexibility, even though the performance was not always perfect depending on the parameter values.

DBD					
	2^{-14}	2^{-13}	2^{-12}	2^{-11}	2^{-10}
4^{1}	0	0	0	0	0
4^{0}	0	0	0	0	0
4^{-1}	7	7	7	7	7
4^{-2}	0	0	0	0	0
4^{-3}	0	0	0	0	0

GNDAM					
	2^{3}	2^{4}	2^{5}	2^{6}	2^{7}
4^{-11}	3	6	5	2	0
4^{-12}	0	6	6	4	0
4^{-13}	3	7	5	5	0
4^{-14}	3	7	6	7	5
4^{-15}	0	6	6	6	0

Table 3. Performances on the texture task. Each element represents the number of times the AM solve the task (max=7) for a given parameter combination. The lines correspond to different LRs and the columns to different free parameter values.

3.4. Luminance

Using SSAB, only two parameter combinations resolved the task, and this with 1/7 performance. As shown in table 4, although none of them achieved a perfect score, many parameter combinations using RPROP were found to solve the luminance task. For the other three AMs, a width variety of parameter combinations were found, which in turn produced a perfect performance. Tables 4-MOM, 4-DBD and 4-GNDAM show these parameter combinations and the average number of iterations they required to solve the task. As shown, MOM solved the task slightly faster than DBD and both solved it faster than the GNDAM.

MOM					
	0	0.5	0.7	0.9	0.99
4^{-4}	60	30	32	17	179
4^{-5}	199	70	37	40	29
4^{-6}	634	225	82	52	45
4^{-7}	2424	826	309	134	79
4^{-8}	9377	2921	1028	489	163

DBD					
	2^{-11}	2^{-10}	2^{-9}	2^{-8}	2^{-7}
4^{-4}	152	74	45	41	2872
4^{-5}	91	66	47	54	333
4^{-6}	57	44	41	54	118
4^{-7}	55	44	34	61	147
4^{-8}	55	44	32	41	246

SSAB					
	1.05	1.1	1.2	1.4	1.8
4^{-5}	0	0	0	0	0
4^{-6}	0	0	0	0	1
4^{-7}	0	0	0	0	0
4^{-8}	0	0	0	0	1
4^{-9}	0	0	0	0	0

RPROP					
	1.1	1.2	1.4	1.8	2.6
4^{-4}	0	0	4	2	0
4^{-5}	0	0	6	6	0
4^{-6}	0	0	5	5	0
4^{-7}	0	0	3	5	0
4^{-8}	0	0	6	6	0

GNDAM					
	2^{-1}	2^{0}	2^{1}	2^{2}	2^{3}
4^{-9}	120	208	253	533	3141
4^{-10}	122	255	469	301	1347
4^{-11}	542	114	308	605	461
4^{-12}	300	246	325	307	922
4^{-13}	205	210	167	275	1194

Table 4. Performances on the luminance task. For tables 4-SSAB and 4-RPROP, each element represents the number of times the AM solve the task (max=7) for a given parameter combination. Tables 4-MOM, 4-DBD and 4-GNDAM give the average number of iteration (x1000) required to solve the luminance task. The lines correspond to different LRs and the columns to different free parameter values.

4. DISCUSSION

The first and obvious conclusion that can be drawn from these results is that no AM could attain a better performance than all the others on all tasks. However, some AMs had similar behaviors. Among these, MOM and DBD had a similar behavior when they were used on the luminance, encoder and parity task. The only task of which they clearly differed was the texture task where MOM never solved the task as opposed to DBD.

SSAB and RPROP showed similar patterns given that neither solved the texture task and both had a similar performance on the encoder task that was significantly greater than all the other AMs. However, RPROP showed a net advantage over SSAB for the parity-bit and luminance tasks.

These results seem to suggest a dichotomy between the MOM-DBD group and the SSAB-RPROP group. For the parity-bit task, SSAB was clearly less efficient than the three other AMs. Since MOM and DBD had at least one parameter combination that produced a 6/7 efficacy and that RPROP's optimal performance was only 4/7, it seems that the MOM-DBD group performed better than the SSAB-RPROP group for the parity-bit task.

The texture task did not separate the two groups since only DBD was able to solve the task. The other two tasks, encoder and luminance, made a clear distinction among these groups. For the luminance task, MOM and DBD gave perfect results while SSAB and RPROP had no parameter combination that produce a 7/7 performance. Opposite results were found with the encoder task; MOM and DBD were not able to achieve a 7/7 performance while SSAB and RPROP had some parameter combinations that did.

It was not possible to classify the GNDAM in either of the groups presented above. For the luminance task, the GNDAM had a performance similar to MOM-DBD group where a width parameter range had a perfect performance. For the parity-bit task, the GNDAM was the only AM that was never able to solve the task. And for the encoder task, the GNDAM had a similar behavior to the SSAB-RPROP group. Consequently, the five AMs could be classified in three distinct groups, each of them having their pros and cons.

4.1. Why exponentially SSAB and RPROP cannot solve the texture task?

SSAB and RPROP are similar AMs. The major characteristic that they share is that both use the derivative sign to increase or decrease the LR exponentially. In the present section, an explanation to these AMs failure on the texture task is proposed.

For batch learning, the derivative of the error (*E*) produced according to a given weight (w_{ij} , where *i* is the presynaptic unit and *j* the postsynaptic unit) can be expressed as:

$$\frac{\partial E}{\partial w_{ij}} = \sum_{p=1}^{N} \delta_{jp} a_{ip} . \qquad (16)$$

where N is the number of patterns, δ_{jp} is the sensitivity of unit *j* after the presentation of the pattern *p* and a_{ip} is the activation output of unit *i* when the pattern *p* is presented. Suppose that *i* is an input unit and *j* a unit of the first hidden layer. A particularity of the texture and luminance tasks is that the activation input of a given pattern (a_{ip}) varies from one epoch to another. Another particularity of the texture and luminance task is their large input layers. In the present study, the input layer for both tasks had 64 units. Given that there is many input units, each one of them has a limited impact on the network's behavior and, specifically, on the sensitivity of the *j*th unit (δ_j).

For the texture task, given that each a_{ip} may either be positive or negative (equally distributed) regardless of the pattern presented and that the impact of a_{ip} on δ_j is not significant, the chance of having a negative or a positive weight derivative ($\frac{\partial E}{\partial w_{ij}} = \sum_{p=1}^{N} \delta_{jp} a_{ip}$) is 50 %. Consequently, half of the time the derivative sign will be in the same direction as the previous one and half of the time it will be in the opposite direction. Since exponentially increasing and decreasing AMs sets $\eta^{-}\eta^{+} < 1$ to prevent saturation, many epochs will cause the LRs of the first layer to constantly decrease.

4.2. Why MOM cannot solve the texture task?

The GNDAM is using standard backpropagation, which is identical to MOM when the momentum factor is set to 0. There are two fundamental differences between the GNDAM and MOM. First, the LR varies over time and second, different weight groups (layers) may have different LRs. Therefore, one of these two differences must prevent MOM to solve the texture task. Simulations showed[1] that the GNDAM selected largely different LRs for different layers. If the optimal LR for each layer is largely different from one layer to another, a single global LR (MOM) would either be too large for some layers causing saturation or be too small for others resulting in very slow learning. This could explain why MOM did not solve the texture task.

4.3. How did DBD solve the texture task?

Although DBD uses the derivative sign to adjust the LRs, it has a major different characteristic from SSAB and RPROP: the LR increases linearly and decreases exponentially. The linearly increasing factor with the exponentially decreasing factor limits the possible LR range. The probability of having a small LR ($\eta(t) << \eta^+ / \eta^-$) is little since many consecutive alternating derivative signs would be required to achieve a small LR. The probability of having a large LR ($\eta(t) >> \eta^+ / \eta^-$) is also small since many consecutive derivatives having the same signs would be required. Therefore, DBD permits the LR to vary but this variation is limited to remain relatively close to η^+ / η^- .

During the simulations DBD found a solution to the texture task only if the initial LR was set to $2^{-2} = 0.25$ and that the incremental constant was set around $2^{-10} \approx 0.001$. Following our previous reasoning, it is highly probable that the LR decreased relatively rapidly from 0.25 to ≈ 0.002. If this was the case, why would an initial LR of 2^{-4}=0.00625 not be able to produce learning? A logical explanation would be that a relatively high initial LR would permit the learning of a layer that required a large LR and that this high LR caused saturation on another layer. Once the LR dropped bellow the saturation level of this other layer, learning emerged since both layers had, at one point, their proper LR. Consequently, it is reasonable to assume that DBD found a solution by “chance” and that DBD may not be a robust AM to solve the texture task.

4.4. The strength and weakness of the GNDAM

The first strength of the GNDAM is that it does not use the derivative sign to adjust the LR. As discussed above, for certain tasks such as the texture task, the derivative sign might not be a good indicator to adjust the LR.

Instead of adjusting a LR for each weight, the GNDAM adjusts one LR per weight group. A weight group usually corresponds to all the weights of a given layer. Therefore, the GNDAM may be a good choice for certain types of tasks that require LRs that largely varies between layers. It is important to note, however, that although our stimulation demonstrates that this strategy worked well for certain tasks (texture and luminance tasks) it did not work well for other tasks such as the parity-bit.

The present paper raises the doubt that AMs based on the sign of the derivative may not be appropriate for certain types of task such as the texture task.

5. CONCLUSIONS

A new AM (the GNDAM) and two new tasks (texture and luminance) were introduced. The GNDAM and other traditional AMs were tested on the new tasks and traditional benchmark tasks. For each AM, an extensive search was made for its two most critical parameters.

The GNDAM failed to solve the parity-bit task, which was solved by all the other traditional AMs. For the texture task, only one traditional AM, DBD, solved it and the three others, MOM, SSAB and RPROP, failed. Some evidence suggests that DBD may not generalize well to texture task variations. Therefore, the GNDAM seems to be the most appropriate choice for the texture task.

Currently, there is no optimal AM for all tasks. Instead of simply searching for such optimal AM, a more modest goal could be to categorize each AM according to their expertise domain. In other words, determine for each AM type, which tasks it can and cannot solve.

ACKNOWLEDGMENTS

This work was supported by the NSERC-Essilor Industrial Research Chair on Presbyopia and Visual Perception and by a Canadian Foundation for Innovation infrastructure grant.

REFERENCES

1. R. Allard, J. Faubert, “The generalized no-decrease adaptive method for large-slow-learning neural networks”, Neural Networks (submitted)
2. D.E. Rumelhart, J.L. McClelland, PDP Research Group, *Parallelel distributed processing, vol.1 Foundations*, 318-364, The MIT Press, Cambridge, MA, 1987.
3. R.A. Jacobs, “Increased rates of convergence through learning rate adaptation”, Neural Networks, **1**, 295-307, 1988.
4. T. Tollenaere, “SuperSAB: Fast adaptive back propagation with good scaling properties”, Neural Networks, **3**, 561-573, 1990.
5. M. Riedmiller, “Advanced supervised learning in multi-layer perceptrons – From backpropagation to adaptive learning algorithms”, Computer Standards & Interfaces, **16**, 265-278, 1994.
6. A.B. Watson, D.G. Pelli, “QUEST: A Bayesian adaptive psychometric method”, Perception & Psychophysics, **33 (2)**, 113-120, 1983.
7. R.O. Duda, P.E. Hart, D.G. Stork, *Pattern classification*, 308-309, John Wiley & Sons, New York, 2001.
8. R.D. Reed, R.J. Marks II, *Neural Smithing: Supervised learning in feedforward artificial neural networks*, 62-63, The MIT Press, Cambridge, MA, 1999.