Nonmonotone Methods for Backpropagation Training with Adaptive Learning Rate

V.P. Plagianakos
University of Patras,
Department of Mathematics,
U.P. Artificial Intelligence
Research Center–UPAIRC,
GR-26500 Patras, Greece.

M.N. Vrahatis
University of Patras,
Department of Mathematics,
U.P. Artificial Intelligence
Research Center–UPAIRC,
GR-26500 Patras, Greece.

G.D. Magoulas
University of Athens,
Department of Informatics,
U.P. Artificial Intelligence
Research Center–UPAIRC,
GR-15771 Athens, Greece.

Abstract

In this paper, we present nonmonotone methods for feedforward neural network training, i.e. training methods in which error function values are allowed to increase at some iterations. More specifically, at each epoch we impose that the current error function value must satisfy an Armijo-type criterion, with respect to the maximum error function value of M previous epochs. A strategy to dynamically adapt M is suggested and two training algorithms with adaptive learning rates that successfully employ the above mentioned acceptability criterion are proposed. Experimental results show that the nonmonotone learning strategy improves the convergence speed and the success rate of the methods considered.

Introduction

The batch training of a Feedforward Neural Network (FNN) is consistent with the theory of unconstrained optimization and can be viewed as the minimization of the function E; that is to find a minimizer $w^* = (w_1^*, w_2^*, \ldots, w_n^*) \in \mathbb{R}^n$, such that:

$$w^* = \min_{w \in \mathbb{R}^n} E(w), \tag{1}$$

where E is the batch error measure defined as the sum–of–squared–differences error function over the entire training set.

The widely used batch Back–Propagation (BP) [28] is a first-order neural network training algorithm, which minimizes the error function using the steepest descent method [8]:

$$w^{k+1} = w^k - \eta \nabla E(w^k), \tag{2}$$

where k indicates iterations ($k = 0, 1, \ldots$), the gradient vector is usually computed by the back–propagation of the error through the layers of the FNN (see [28]) and η is a constant heuristically chosen learning rate. Appropriate learning rates help to avoid convergence to a saddle point or a maximum. In practice, a small constant learning rate is chosen ($0 < \eta < 1$) in order to secure the convergence of the BP training algorithm and to avoid oscillations in a direction where the error function is steep. It is well known that this approach tends to be inefficient [12, 28]. For difficulties in obtaining convergence of BP training algorithms utilizing a constant learning rate see [13, 16]. On the other hand, there are theoretical results that guarantee the convergence when the learning rate is constant. In this case the learning rate is proportional to the inverse of the Lipschitz constant which, in practice, is not easily available [1, 17].

The paper is organized as follows: in the next section the class of backpropagation algorithms with adaptive learning rate is presented and the advantages as well as the disadvantages of these algorithms are discussed. Then, the idea of nonmonotone training is introduced. Afterwards, two methods with adaptive learning rate, which incorporate the proposed nonmonotone learning strategy are described. Finally, simulation results are presented and the paper ends with some concluding remarks.

Adaptive Learning Rate Algorithms

Several adaptive learning rate algorithms have been proposed to accelerate the training procedure. The following strategies are usually suggested: (i) start with a small learning rate and increase it exponentially, if successive epochs reduce the error, or rapidly decrease it, if a significant error increase occurs [3, 31], (ii) start with a small learning rate and increase it, if successive epochs keep gradient direction fairly constant, or rapidly decrease it, if the direction of the gradient varies greatly at each epoch [6] and (iii) for each weight an individual learning rate is given, which increases if the successive changes in the weights are in the same direction and decreases otherwise [12, 22, 26, 29]. Note that all the above mentioned strategies employ heuristic parameters in an attempt to enforce the *monotone* decrease of the learning error and to secure the converge of the training algorithm.

A different approach is based on Goldstein's and Armijo's work on steepest–descent and gradient methods. The method of Goldstein [9] requires the assumption that E is twice continuously differentiable on $\mathcal{S}(w^0)$, where $\mathcal{S}(w^0) = \{w : E(w) \leq E(w^0)\}$ is bounded, for some initial vector w^0. It also requires that η is chosen to satisfy the relation $\sup \|H(w)\| \leq \eta^{-1} < \infty$ in some bounded region where the relation $E(w) \leq E(w^0)$ holds, where $H(w)$ denotes the Hessian of E at w. However, the manipulation of the full Hessian is too expensive in computation and storage for FNNs with several hundred weights [4]. Le Cun [14] proposed a technique, based on appropriate perturbations of the weights, for estimating on–line the extreme eigenvalues and eigenvectors of the Hessian without calculating the full matrix H. According to experiments reported in [14] the largest eigenvalue of the Hessian is mainly determined by the FNN architecture, the initial weights and by short–term low–order statistics of the training data. This technique could be used to determine η requiring additional presentations of the training set in the early training.

An alternative approach is based on the work of Armijo [1]. Following this approach, the value of the learning rate η is related to the value of the Lipschitz constant L, which depends on the morphology of the error surface. In this case, the BP algorithm takes the form:

$$w^{k+1} = w^k - \frac{1}{2L}\nabla E(w^k), \tag{3}$$

and converges to the point w^* which minimizes E (see [1] for conditions under which convergence occurs and a convergence proof).

Nonmonotone Learning and Global Convergence

A training algorithm can be made globally convergent by determining the learning rate in such a way that the error is exactly minimized along the current search direction at each epoch, i.e. $E(w^{k+1}) < E(w^k)$. To this end, an iterative search, which is often expensive in terms of error function evaluations, is required. It must be noted that the above simple condition does not guarantee global convergence for general functions, i.e. converges to a local minimizer from any initial condition (see [7] for a general discussion on globally convergent methods).

The use of adaptive learning rate algorithms which enforce monotone error reduction using inappropriate values for the critical heuristic learning parameters can considerably slow the rate of training, or even lead to divergence and to premature saturation [15, 27]. Moreover, using heuristics it is not possible to develop globally convergent training algorithms.

To alleviate this situation we propose the following approach that exploits the accumulated information regarding the M most recent values of the error function, to accelerate convergence. The following condition is used to formulate the proposed approach and to define a criterion of acceptance of any weight iterate [10]:

$$E\big(w^k - \eta^k \nabla E(w^k)\big) - \max_{0\leq j\leq M} E(w^{k-j}) \leq -\sigma\, \eta^k \left\|\nabla E(w^k)\right\|^2, \tag{4}$$

where M is a nonnegative integer, $0 < \sigma < 1$, and η^k indicate the learning rate in the kth epoch. The above condition allows an increase in the function values without affecting the global convergence properties as has been proved theoretically in [10, 25] and experimentally in [24]. Experiments indicate that the choice of the parameter M is critical for the implementation and depends on the problem. The following procedure provides an elegant way to dynamically adapt the value of M at each epoch:

$$M^k = \begin{cases} M^{k-1} + 1, & \Lambda^k < \Lambda^{k-1} < \Lambda^{k-2}, \\ M^{k-1} - 1, & \Lambda^k > \Lambda^{k-1} > \Lambda^{k-2}, \\ M^{k-1} \quad , & \text{otherwise}, \end{cases} \tag{5}$$

where Λ^k is the local estimation of the Lipschitz constant [17] at the kth iteration, defined as:

$$\Lambda^k = \frac{\left\|\nabla E(w^k) - \nabla E(w^{k-1})\right\|}{\|w^k - w^{k-1}\|}. \tag{6}$$

Note that the local approximation of the Lipschitz constant is easily obtained, without any additional error

function and gradient evaluations. Obviously, M has to be positive. Thus, if Relation (5) gives a non positive number, the value of M is set to $M = 1$. Moreover, in practice, we have to enforce an upper limit to the values of M, as well. Experimental results indicate that $M = 10$, is a good choice for the maximum value of M.

If Λ^k is increased for two consecutive epochs, the sequence of the weight vectors approaches a steep region, so we decrease the value of M, to avoid overshooting a possible minimum point. Reversely, when Λ^k is decreased for two consecutive epochs, the method possibly enters a valley in the weight space, so we increase the value of M. This allows the method to accept larger learning rates and move faster out of the flat region. Finally, when the value of Λ^k has a rather random behavior (increasing and decreasing for consecutive epochs), we leave the value of M unchanged.

Next, we propose a high-level description of an algorithm that employs the above acceptability criterion. The kth iteration of the algorithm consists of the following steps:

1: Compute the new learning rate η^k using any learning rate adaptation strategy.

2: Update the weights $w^{k+1} = w^k - \eta^k \nabla E(w^k)$.

3: If the acceptability condition (4) is fulfilled store w^{k+1} and terminate; otherwise go to the next step.

4: Use a tuning technique for η^k and return to Step 3.

Remark 1.: A simple technique to tune η^k at Step 4 is to decrease the learning rate by a reduction factor $1/q$, where $q > 1$ [21]. We remark here that the selection of q is not critical for successful learning, however it has an influence on the number of error function evaluations required to satisfy the condition (4). The value $q = 2$ is usually suggested in the literature [1] and indeed it was found to work without problems in the experiments reported in the paper.

Remark 2.: The above model constitutes an efficient method to determine an appropriate learning rate without additional gradient evaluations. As a consequence, the number of gradient evaluations (GE) is, in general, less than the number of error function evaluations (FE).

Nonmonotone Training Algorithms with Adaptive Learning Rate

The nonmonotone learning strategy can be incorporated in any training algorithm. In this section, we briefly describe two recently proposed training algorithms with adaptive learning rate, which can be successfully used for nonmonotone backpropagation training.

1) Adapting the rate of learning by locally estimating the Lipschitz constant. The BPVS algorithm [17] exploits the local shape of the error surface by estimating the Lipschitz constant at each epoch and setting the learning rate $\eta^k = 0.5/\Lambda^k$, where Λ^k is the local estimation of the Lipschitz constant L at the kth epoch (see Relation 6). Thus, when the error surface has steep regions, Λ^k is large and a small value for the learning rate is appropriate in order to guarantee convergence. On the other hand when the error surface has flat regions, Λ^k is small and a large learning rate is appropriate to accelerate the convergence speed. If $\eta^k = 0.5/\Lambda^k$ does not satisfy condition (4), the learning rate tuning technique of Remark 1 is applied. This version of the BPVS that provides nonmonotone training is named NMBPVS.

2) Adapting the rate of learning by using the Barzilai and Borwein formula. In a previous work [23] we have proposed a neural network training algorithm called BBP, which is based on the Barzilai and Borwein [2] learning rate update formula:

$$\eta^k = \frac{\langle \delta^{k-1}, \delta^{k-1} \rangle}{\langle \delta^{k-1}, \psi^{k-1} \rangle}, \tag{7}$$

where $\delta^{k-1} = w^k - w^{k-1}$, $\psi^{k-1} = \nabla E(w^k) - \nabla E(w^{k-1})$ and $\langle \cdot, \cdot \rangle$ denotes the standard inner product. The key features of this method are the low storage requirements and the inexpensive computations. Moreover, it does *not* guarantee descent in the error function E. Our experiments in [23, 24], show that this property is valuable in neural network training, because very often the method escapes from local minima and flat valleys where other methods are trapped. To secure that condition (4) is fulfilled, the learning rate tuning technique of Remark 1 is used to reduce relatively large learning rates. This modified training algorithm is named NMBBP.

Experimental Results

In this section, we evaluate the performance of the proposed algorithms. The algorithms have been tested using the same random initial weights, and received the same sequence of input patterns. The weights of the network are updated only after the entire set of patterns to be learned has been presented.

We have fixed the value of $\sigma = 10^{-5}$. Furthermore, the maximum value of M allowed was $M = 10$.

For each of the test problem, a figure summarizing the performance of the algorithms for simulations that reached solution is presented. A total number of 100 simulations for each algorithm has been made. In figures 1–6, the x-axis denotes the value of $M = 1, \ldots, 10$. For $M = 1$, we have the original BPVS and BBP methods using the Armijo's criterion, while for $M = V$, we have the proposed methods with adaptive M.

Note that the number of error function evaluations (FE) is, in general, larger than the number of gradient evaluations (GE), due to the learning rate acceptability criterion we use. As a consequence, even when our algorithms fail to converge within the predetermined limit of *function evaluations*, their number of gradient evaluations is smaller than the corresponding number of other methods. Keeping in mind that a gradient evaluation is more costly than an error function evaluation (see for example [18], where Møller suggests to count a gradient evaluation three times more than an error function evaluation), one can understand that our methods require fewer floating point operations and are actually much faster.

The font learning problem

For this problem, 26 matrices with the capital letters of the English alphabet are presented to a 35-30-26 FNN (1830 weights, 56 biases). Each letter has been defined in terms of binary values on a grid of size 5×7. The network is based on hidden neurons of logistic activations and on linear output neurons. The performance of the methods is exhibited in Figure 1 and 2.

For this problem the weights have been initialized following the Nguyen–Widrow method [19]. The termination condition is a sum-of-squared-differences error less than 0.1 and the error function evaluations limit was 2000. In this problem the nonmonotone learning strategy accelerates the convergence of both BPVS and BBP methods. Note that the performance of the methods with adaptive M is at least as good as the average performance of the methods with a predefined M. All the algorithms exhibited very good performance, as they had complete success percentage (see Figure 3).

Texture classification problem

A total of 12 Brodatz texture images [5]: 3, 5, 9, 12, 15, 20, 51, 68, 77, 78, 79, 93 (see Figure 1 in [17]) of size 512×512 is acquired by a scanner at 150dpi. From each texture image 10 subimages of size 128×128 are randomly selected, and the co-occurrence method, introduced by Haralick [11] is applied. In the co-occurrence method, the relative frequencies of gray–level pairs of pixels at certain relative displacements are computed and stored in a matrix. As suggested by other researchers [20, 30], the combination of the nearest neighbor pairs at orientations 0^o, 45^o, 90^o and 135^o are used in the experiment. A set of 10 sixteenth-dimensional training patterns are created from each image. The patterns are presented in a finite sequence $C = (c_1, c_2, \ldots, c_p)$ of input–output pairs $c_p = (u_p, t_p)$ where u_p are the real valued input vectors in $\mathbb{R}^{16}$ and t_p are binary output vectors in $\mathbb{R}^{12}$, for $p = 1, \ldots, 120$, determining the corresponding training pattern. A 16-8-12 FNN (224 weights, 20 biases) is trained to classify the patterns to the 12 texture types. The network is based on neurons of logistic activations with biases, and the weights and biases were initialized with random numbers from the interval $(-1, 1)$.

Detailed results regarding the training performance of the algorithms are presented in Figure 4 and 5. The termination condition is a classification error $CE < 3\%$ [17]; that is the network classifies correctly 117 out of the 120 patterns.

The successfully trained FNNs are tested for their generalization capability, using test patterns from 20 subimages of the same size randomly selected from each image. To evaluate the generalization performance of the FNN the *max* rule is used, i.e. a test pattern is considered to be correctly classified if the corresponding output neuron has the greatest value among the output neurons. The methods with variable M exhibited slightly better generalization capabilities than the other methods.

Concluding Remarks

In this paper we have presented a new approach for generating nonmonotone learning rules based on an acceptability criterion. We incorporate information given by the local estimation of the Lipschitz constant to dynamically adapt this criterion, according to the local shape of the error function. The simulation results suggest that the use of this acceptability criterion, can significantly accelerate the convergence speed of the training algorithms. The behavior of the nonmonotone algorithms, in the considered experiments, proved to be robust against phenomena such as oscillations due to large learning rates.

References

[1] L. Armijo, Minimization of functions having Lipschitz-continuous first partial derivatives, Pacific J. Math., 16, 1–3, (1966).

[2] J. Barzilai and J.M. Borwein, Two point step size gradient methods, IMA J. Numer. Anal., 8, 141–148, (1988).

[3] R. Battiti, Accelerated backpropagation learning: two optimization methods, Complex Systems, 3, 331–342, (1989).

[4] S. Becker and Y. Le Cun, Improving the convergence of the back–propagation learning with second order methods, in Proc. of the 1988 Connectionist Models Summer School, D.S. Touretzky, G.E. Hinton, and T.J. Sejnowski (eds.), 29–37, Morgan Koufmann, San Mateo, CA, (1988).

[5] P. Brodatz, Textures - a Photographic Album for Artists and Designers, Dover, New York, (1966).

[6] L.W. Chan and F. Fallside, An adaptive training algorithm for back–propagation networks, Computers Speech and Language, 2, 205–218, (1987).

[7] J.E. Dennis and R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice-Hall, Englewood Cliffs, NJ, (1983).

[8] P.E. Gill, W. Murray, and M.H. Wright, Practical Optimization, Academic Press, NY, (1981).

[9] A.A. Goldstein, Cauchy's method of minimization, Numer. Math., 4, 146–150, (1962).

[10] L. Grippo, F. Lampariello, and S. Lucidi, A nonmonotone line search technique for Newton's method, SIAM J. Numer. Anal., 23, 707–716, (1986).

[11] R. Haralick, K. Shanmugan, and I. Dinstein, Textural features for image classification, IEEE Trans. System, Man and Cybernetics, 3, 610–621, (1973).

[12] R.A. Jacobs, Increased rates of convergence through learning rate adaptation, Neural Networks, 1, 295–307, (1988).

[13] C.M. Kuan and K. Hornik, Convergence of learning algorithms with constant learning rates, IEEE Trans. Neural Networks, 2, 484–488, (1991).

[14] Y. Le Cun, P.Y. Simard, and B.A. Pearlmutter, Automatic learning rate maximization by on–line estimation of the Hessian's eigenvectors, in Advances in Neural Information Processing Systems 5, S.J. Hanson, J.D. Cowan, and C.L. Giles (eds.), 156–163, Morgan Kaufmann, San Mateo, CA, (1993).

[15] Y. Lee, S.H. Oh and M.W. Kim, An analysis of premature saturation in backpropagation learning, Neural Networks, 6, 719–728, (1993).

[16] R. Liu, G. Dong, and X. Ling, A convergence analysis for neural networks with constant learning rates and non–stationary inputs, Proceedings of the 34th Conf. on Decision and Control, New Orleans, 1278-1283, (1995).

[17] G.D. Magoulas, M.N. Vrahatis, and G.S. Androulakis, Effective back–propagation with variable stepsize. Neural Networks, 10, 69–82, (1997).

[18] M.F. Møller, A scaled conjugate gradient algorithm, for fast supervised learning, Neural Networks, 6, 525–533, (1993).

[19] D. Nguyen and B. Widrow, Improving the learning speed of 2-layer neural network by choosing initial values of the adaptive weights, IEEE First International Joint Conference on Neural Networks, 3, 21–26, (1990).

[20] P.P. Ohanian and R.C. Dubes, Performance evaluation for four classes of textural features, Pattern Recognition, 25, 819–833, (1992).

[21] J.M. Ortega and W.C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, NY, (1970).

[22] Pfister M. and Rojas R., Speeding-up backpropagation - A comparison of orthogonal techniques, in Proc. of the Joint Conference on Neural Networks, Nagoya, Japan, 517–523, (1993).

[23] V.P. Plagianakos, D.G. Sotiropoulos, and M.N. Vrahatis, Automatic adaptation of learning rate for Backpropagation Neural Networks, Recent Advances in circuits and systems, Nikos E. Mastorakis, ed., World Scientific, (1998).

[24] V.P. Plagianakos, D.G. Sotiropoulos, and M.N. Vrahatis, A Nonmonotone Backpropagation Training Method for Neural Networks, Proceedings of the 4th Hellenic–European Conference on Computer Mathematics and its Applications, Athens, (1998).

[25] M. Raydan, The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem, SIAM J. Optim., 7, 26–33, (1997).

[26] M. Riedmiller and H. Braun, A direct adaptive method for faster backpropagation learning: the Rprop algorithm, Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA, 586–591, (1993).

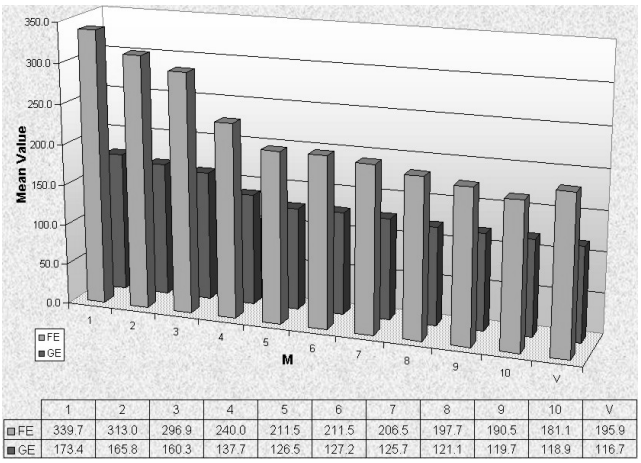
[27] A.K. Rigler, J.M. Irvine, and T.P. Vogl, Rescaling of variables in backpropagation learning, Neural Networks, 4, 225–229, (1991).

[28] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, Learning Internal Representations by Error Propagation. In D. E. Rumelhart, and J. L. McClelland, eds., Parallel Distributed Processing: Explorations in the Microstructure of Cognition, 1, pp. 318–362. MIT Press, Cambridge, Massachusetts.

[29] F. Silva and L. Almeida, Acceleration techniques for the back–propagation algorithm, Lecture Notes in Computer Science, 412, 110–119. Springer-Verlag, Berlin, (1990).

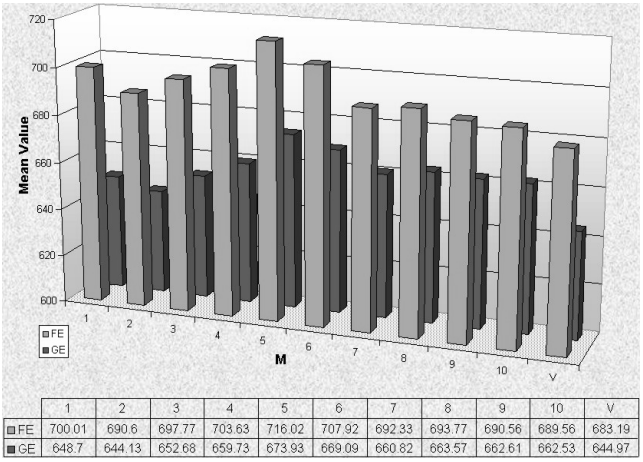
[30] J. Strang and T. Taxt, Local frequency features for texture classification, Pattern Recognition, 27, 1397–1406, (1994).

[31] T.P. Vogl, J.K. Mangis, J.K. Rigler, W.T. Zink, and D.L. Alkon, Accelerating the convergence of the backpropagation method, Biological Cybernetics, 59, 257–263,(1988).



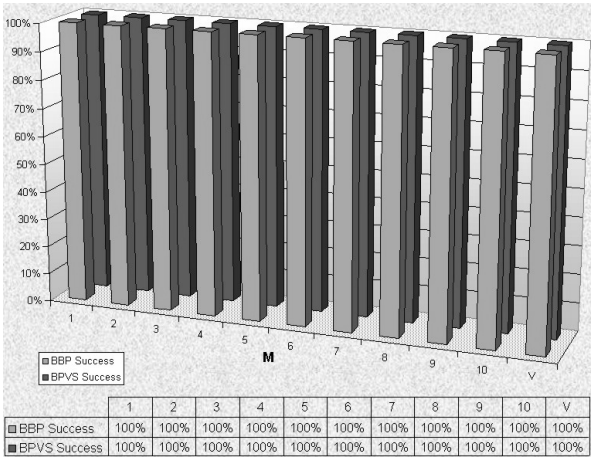
	1	2	3	4	5	6	7	8	9	10	V
FE	339.7	313.0	296.9	240.0	211.5	211.5	206.5	197.7	190.5	181.1	195.9
GE	173.4	165.8	160.3	137.7	126.5	127.2	125.7	121.1	119.7	118.9	116.7

Figure 1: Mean FE and GE numbers of BBP method for the font problem



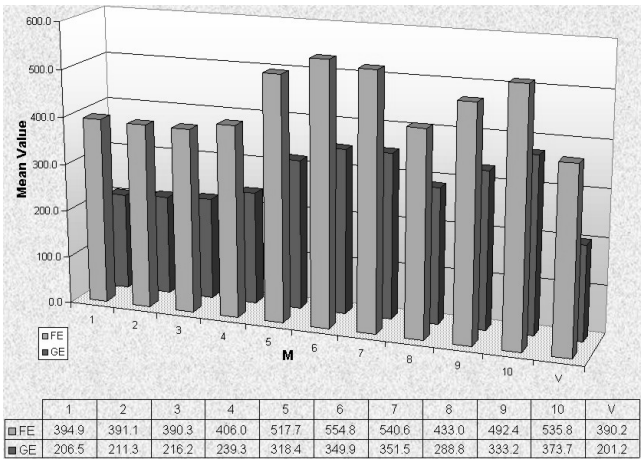
	1	2	3	4	5	6	7	8	9	10	V
FE	700.01	690.6	697.77	703.63	716.02	707.92	692.33	693.77	690.56	689.56	683.19
GE	648.7	644.13	652.68	659.73	673.93	669.09	660.82	663.57	662.61	662.53	644.97

Figure 2: Mean FE and GE numbers of BPVS method for the font problem



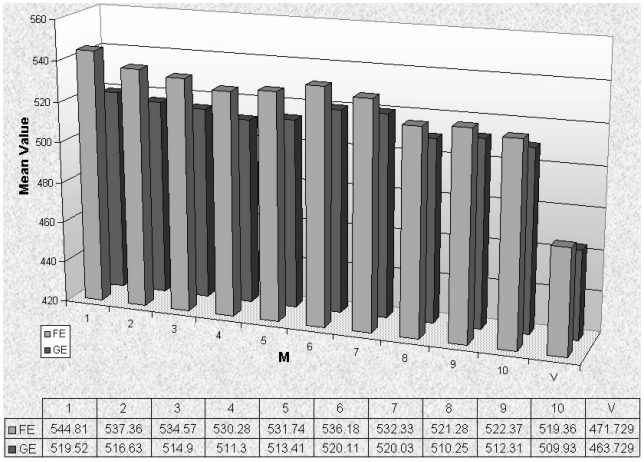
	1	2	3	4	5	6	7	8	9	10	V
BBP Success	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
BPVS Success	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Figure 3: Success percentage of the BBP and BPVS methods for the font problem



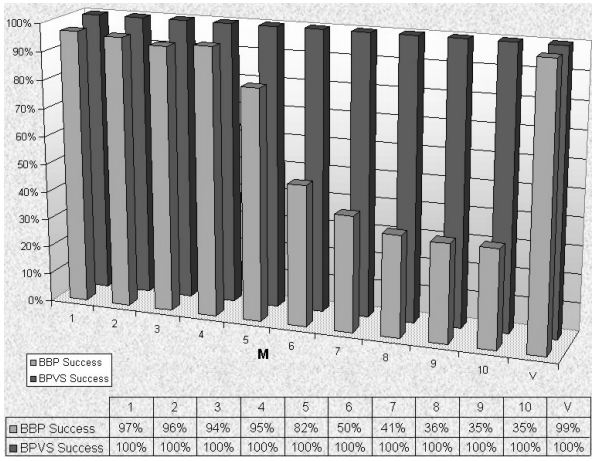
	1	2	3	4	5	6	7	8	9	10	V
FE	394.9	391.1	390.3	406.0	517.7	554.8	540.6	433.0	492.4	535.8	390.2
GE	206.5	211.3	216.2	239.3	318.4	349.9	351.5	288.8	333.2	373.7	201.2

Figure 4: Mean FE and GE numbers of BBP method for the texture problem



	1	2	3	4	5	6	7	8	9	10	V
FE	544.81	537.36	534.57	530.28	531.74	536.18	532.33	521.28	522.37	519.36	471.729
GE	519.52	516.63	514.9	511.3	513.41	520.11	520.03	510.25	512.31	509.93	463.729

Figure 5: Mean FE and GE numbers of BPVS method for the texture problem



	1	2	3	4	5	6	7	8	9	10	V
BBP Success	97%	96%	94%	95%	82%	50%	41%	36%	35%	35%	99%
BPVS Success	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Figure 6: Success percentage of the BBP and BPVS methods for the texture problem