High-Performance Bankruptcy Prediction Model using Graphics Processing Units

Bernardete Ribeiro, *Member, IEEE* and Noel Lopes and Catarina Silva, *Member, IEEE*

***Abstract*— In recent years the the potential and programmability of Graphics Processing Units (GPU) has raised a noteworthy interest in the research community for applications that demand high-computational power. In particular, in financial applications containing thousands of high-dimensional samples, machine learning techniques such as neural networks are often used. One of their main limitations is that the learning phase can be extremely consuming due to the long training times required which constitute a hard bottleneck for their use in practice. Thus their implementation in graphics hardware is highly desirable as a way to speed up the training process. In this paper we present a bankruptcy prediction model based on the parallel implementation of the Multiple BackPropagation (MBP) algorithm which is tested on a real data set of French companies (healthy and bankrupt). Results by running the MBP algorithm in a sequential processing CPU version and in a parallel GPU implementation show reduced computational costs with respect to the latter while yielding very competitive performance.**

I. Introduction

The hit rate of firms insolvency has increased exponentially during last year due to the global economic crisis. As a consequence, there is an ever-increasing need for fast automated recognition systems for bankruptcy prediction.

Enterprise bankruptcy forecasting is very important to all stakeholders (banks, insurance firms, creditors, and investors) to manage credit risk associated with counterparts. Although it is a widely studied topic, it is becoming harder to estimate as companies become more complex and develop more sophisticated schemes to hide their real situation. On the other hand as the inability to discharge all debts as they come due (insolvency) increases, the need for substantially more accurate predicting models and, at the same time, for faster decision-maker systems becomes crucial.

Due to the recent financial crisis and regulatory concerns, credit risk assessment is a very active area both from academic and business community. The ability to discriminate between trustworthy customers from potential bad ones is thus crucial for commercial banks and retailers.

The problem is stated as follows: given a set of parameters (mainly of financial nature) that describe the situation of a company over a given period, predict the probability that the company may become bankrupted during the following year. Neural Networks (NNs) are particularly suited for predicting the bankrupt probability, thus they are a strategic choice among other methods. Likewise, their properties make them often used in financial applications because of their excellent performances of treating non-linear data with self-learning capability [8]. A review of the topic of bankruptcy prediction with emphasis on NN models is given in [4]. More recently, in [14] there is a broad coverage of a wide range of other intelligent techniques such as fuzzy set theory, decision trees, rough sets, case-based reasoning, support vector machines, data envelopment analysis and soft computing. Although these models have been widely used in the last decades, still the pioneer statistical techniques are worth mentioning in the modeling of corporate bankruptcy prediction such as univariate and multivariate discriminant analysis [1], [2]. All these techniques aim at finding an optimal linear combination of explanatory input variables such as solvency and liquidity ratios, in order to analyze, model and predict corporate default risk.

Most of the prediction models use financial ratios as predictor variables, by employing the selection of only a few financial ratios according to a choice based criteria. Model selection of corporate distress prediction is advisable for reducing problem complexity saving computational costs. In [13] a linear pre-processing stage using principal component analysis (PCA) for dimensionality reduction purposes is tested. However, nonlinear projection methods have been successfully used [16] making them more suitable for this problem. With the same goal, non-negative matrix factorization (NMF) is used in [15] for extracting the most discriminative features.

Despite the numerous papers dealing with the problem it is often difficult to compare the techniques due to different data sets, algorithms and approaches. Moreover, all have been deployed as desktop applications running in the available hardware (PC or Laptop) at the time, possibly equipped with the most modern CPU. Our work compares (in the procedures and methods) to the previous published work on this topic, however our focus is to implement a nonparametric and hybrid design architecture in cutting-edge parallel GPU hardware. The rationale behind is twofold: to develop very effective bankruptcy recognition system and both to reduce complexity in model selection (no need to perform unnecessary pruning of predictor variables) and to reduce computational learning time.

For this study we used a large database of French companies. This database is very detailed containing information on a wide set of financial ratios spawning over a period of

Bernardete Ribeiro is with Department of Informatics Engineering and CISUC, Center of Informatics and Systems of University of Coimbra, Portugal (bribeiro@dei.uc.pt).

Noel Lopes is with School of Technology and Management, Polytechnic Institute of Guarda (IPG/UDI) and CISUC, Portugal (noel@ipg.pt).

Catarina Silva is with School of Technology and Management, Polytechnic Institute of Leiria (IPL) and CISUC, Portugal (catarina@dei.uc.pt).

978-1-4244-8126-2/10/$26.00

several years. It contains up to three thousands distressed companies and about sixty thousand healthy ones.

The rest of the paper is organized as follows. In Section II we introduce the Graphical Processing Units (GPUs) and the CUDA programming model. In Section III we address the implementation of Multiple BackPropagation (MBP) algorithm for a neural network approach to the bankruptcy prediction model in a case study of the French Market. Moreover, we give an overview of the kernels used in the GPU implementation and explain a detailed kernel. In Section IV we describe the solvent (and default) firm data collected from French companies provider, detail historical organization of data as well as the experimental setup for prediction model design. Section V presents and discusses the results. In the last section conclusions will be drawn and further lines of work will be addressed.

II. Graphics Processing Units (GPUs)

Graphics Processing Units (GPUs) have been used as graphic processor engines in gaming industry due to their powerful capability and parallel characteristics. The graphics pipeline is well-suited for parallelism attaining high performances in matrix and vector operations as for instance when dealing with 2D and 3D graphics. Their enormous computational potential has led to an explosion of research to leverage GPUs for general-purpose computation on GPUs (GPGPU).

The increase of their programmability featuring floating-point arithmetic make them suitable for data high-demanding real time applications. This is especially important with machine learning algorithms such as neural networks which are often complex, placing high demands on memory and computing resources and CPUs are simply not powerful enough to solve them quickly for use in interactive applications.

Example of applications of GPU computing spawn a broad range of areas such as Bioinformatics [3], Medical Imaging [17], Linear Algebra [18], [6] and many other.

Unlike CPUs which use the paradigm SISD (Single Instruction Single Data), GPUs are optimized to perform floating-point operations (on large data sets) using the paradigm Single Instruction Multiple Data (SIMD). Subsequently, this architectural difference between both platforms leads to more complex programming tasks. To cope with this complexity NVIDIA [11], [7] developed a parallel technology namely CUDA (Compute United Device Architecture) which provides a programming model for its GPUs with an adequate API for non-graphics applications using standard ANSI C, extended with keywords that designate data-parallel functions.

The CUDA programming model is supported by an architecture built around a scalable array of multi-threaded Streaming Multiprocessors (SMs), as shown in Figure 1.

Over the past few years not only GPU evolved but also the programming model and programming tools able to allow the development of applications of this nature. The trend is to develop high parallel computers and to concentrate on adding cores rather than increase the capacity of single thread performance ones.

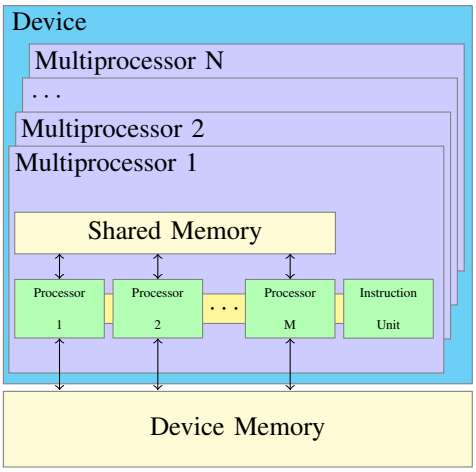

Fig. 1. CUDA hardware model

III. Parallel Implementation of Neural Networks

The promise in late eighties that neural networks (NNs) due to their eminent parallel capabilities could be implemented in hardware has never been met. Researchers with very high data demanding applications had expectations that NNs as non-linear powerful approximation devices could be used as a powerful solution, if implemented in parallel, for their problems but this never happened.

Most neural networks recognition systems have two major components: training and classification. The system is trained using a large number of labeled patterns with relevant features. The training is often iterative and proceeds until the error on a small set of testing patterns is sufficiently low.

However, the training process is computational intensive and time consuming especially when a large number of training patterns are involved. The classification accuracy of unknown patterns often depends on the effort spent on training and most applications settle down for a suitable trade-off. Using new training data to improve the performance is uncommon due to the large computational effort. On the contrary, the parallelism of a GPU is fully utilized by accumulating a lot of input feature vectors and weight vectors, then converting the many inner-product operations into one matrix operation [12], [5].

A. Multiple Back-Propagation Algorithm

Multiple Back-Propagation (MBP) is a generalization of the Back-Propagation (BP) algorithm that can be used to train Multiple Feed-Forward (MFF) networks. A MFF network is obtained by integrating two feed-forward (FF) networks: a main network and a space network. Details of the algorithm can be found in [9]. Figure 2 shows the scheme interaction between the two networks that form the MFF architecture.

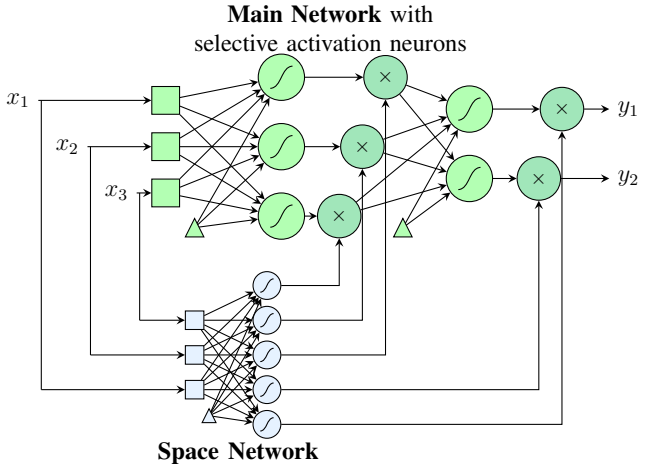

Fig. 2. Example of a Multiple Feed-Forward (MFF) network. Squares represent inputs, darker circles (with an $\times$) multipliers, lighter circles neurons, and triangles the bias.

Both the MBP and BP algorithms are implemented in the Multiple Back-Propagation software, which is a highly optimized software, developed in C++.[1]

B. GPU Implementation

A detailed description of the GPU parallel implementation of MBP for MFF networks can be found in [10]. The GPU version was benchmarked for classification and regression problems on two different devices: a NVIDIA GeForce 8600 GT with 4 SM (32 cores) and a NVIDIA GTX 280 with 30 SM (240 cores). The CPU version was benchmarked on a Intel Core 2 6600 CPU running at 2.4 GHz.

The current GPU implementation relies on six kernels (special functions that are to be executed in parallel, on a physically separate device (GPU)): *FireLayer* which calculates the outputs of all neurons in a given layer; *FireOutputLayer* which calculates the outputs of the network output layer and the local gradients of its neurons; *CalcLocalGradients* which calculates the local gradient of all neurons in a hidden layer; *CorrectWeights* which adjusts the weights of a given layer; *CalculateRMS* which calculates the Root Mean Square (RMS) error of the network; and *RobustLearning* which ensures the training RMS error does not rise, by restoring the network best weights (obtained so far) if needed. Figure 3 presents the CUDA code of the *FireLayer* kernel. The remainder of the code can be obtained together with the Multiple Back-Propagation software source code (`http:dit.ipg.pt/MBP`) or by downloading GPUMLib – an open source Graphic Processing Unit Machine Learning Library (`http://gpumlib.sourceforge.net/`).

IV. Experimental Setup and Discussion

A. Data Description

We used Diane database which contains financial statements of French companies. The initial sample consisted of financial ratios of about 60 000 industrial French companies (for the years of 2002 to 2006) with at least 10 employees. From these companies, about 3000 were declared bankrupted in 2007 or presented a restructuring plan to the court for approval by the creditors. In Table I 30 variables are defined to be used further in the prediction model. The ultimate goal is class (healthy, bankrupt) prediction. These financial predictors allow to describe firms in terms of the financial strength, liquidity, solvability, productivity of labor and capital, margins, net profitability and return on investment. Upon appropriate treatment of the database to eliminate firms with missing values, a final set of 600 default examples was obtained. In order to obtain a balanced dataset we randomly selected 600 non-default examples resulting in a set of 1200 examples. To accommodate historical information yearly variations of important financial ratios reflecting the balance sheet were then evaluated. Thus information from the past 3 years preceding the default (bankrupt) were also included. Therefore, the number of inputs was increased from 30 to 90 ratios.

B. Evaluation metrics

Performance metrics were defined based on the classification contingency matrix[2] represented in Table II. Measures such as Recall (R=$\frac{tp}{tp+fn}$) and Precision (P=$\frac{tp}{tp+fp}$) are calculated for final model evaluation. Also important is a "Type I error" (or false positive rate, i.e. $\frac{fp}{fp+tn}$) which indicates the misclassification of a healthy firm as distressed. Conversely, a "Type II error" (or false negative rate,i.e. $\frac{fn}{fn+tp}$) indicates that a distressed firm is misclassified by the predictor as viable. An "overall hit" refers to the total correct classifications for the set ($\frac{tp+tn}{tp+fp+fn+tn}$) regardless of type. Another measure combining recall and precision into a single utility criterion by taking their weighted harmonic mean is F1-score ($F1 = 2*P*R/(P+R)$). It quantifies the tradeoff between Recall and Precision and is fairly indicative of the performance of the overall algorithm. The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. All the illustrated results represent mean values obtained in test financial data. For completeness standard deviations are also indicated.

TABLE II
Contingency matrix.

real class	predicted class		
	Bankrupt	Healthy	total
Bankrupt	tp	fn	pos
Healthy	fp	tn	neg
total	pos_p	neg_p	T

C. Results Analysis

The bankruptcy prediction model is designed with 30 financial ratios for the historical 3 years considered, therefore

[1]The latest version of Multiple Back-Propagation software can be freely obtained at `http://dit.ipg.pt/MBP`.

[2]tp, fp, tn, fn represent the usual notation for the confusion matrix in terms of true (or false) and positive (or negative) results from the classifier.

```
#define INPUT threadIdx.x
#define NUM_INPUTS_INCLUDING_BIAS blockDim.x
#define NUM_INPUTS (NUM_INPUTS_INCLUDING_BIAS - 1)
#define BIAS 0
#define NEURON threadIdx.y
#define NUM_NEURONS blockDim.y
#define PATTERN blockIdx.x
#define THREAD_ID connection

__device__ void SumInputWeight(int connection, float * inputs, float * weights) {
  extern __shared__ float iw[];

  iw[connection] = weights[connection];
  if (INPUT > BIAS) iw[connection] *= inputs[PATTERN * NUM_INPUTS + (INPUT - 1)];
  __syncthreads();

  int numberElemSum = NUM_INPUTS_INCLUDING_BIAS;
  for(int sumUpTo = (numberElemSum >> 1); numberElemSum > 1; sumUpTo = (numberElemSum >> 1)) {
    int nextNumberElemSum = sumUpTo;
    if (numberElemSum & 1) nextNumberElemSum++;
    if (INPUT < sumUpTo) iw[connection] += iw[connection + nextNumberElemSum];
    numberElemSum = nextNumberElemSum;
    __syncthreads();
  }
}

__global__ void FireLayer(float * inputs, float * weights, float * m, int mOffset, int totalNeuronsWithSelectiveActivation, float * outputs) {
  extern __shared__ float iw[];

  int connection = NEURON * NUM_INPUTS_INCLUDING_BIAS + INPUT;
  SumInputWeight(connection, inputs, weights);

  if (INPUT == 0) {
    int n = PATTERN * NUM_NEURONS + NEURON;
    float output = CUDA_SIGMOID(iw[THREAD_ID]);
    if (m != NULL) output *= m[PATTERN * totalNeuronsWithSelectiveActivation + NEURON + mOffset];
    outputs[n] = output;
  }
}
```

Fig. 3. FireLayer CUDA kernel.

TABLE I
DIANE DATA BASE FINANCIAL RATIOS

	Variable Description			
DIANA DATA BASE	x_1 -	Number of Employees Previous year	x_{16} -	Cashflow / Turnover
	x_2 -	Capital Employed / Fixed Assets	x_{17} -	Working Capital / Turnover days
	x_3 -	Financial Debt / Capital Employed	x_{18} -	Net Current Assets/Turnover days
	x_4 -	Depreciation of Tangible Assets	x_{19} -	Working Capital Needs / Turnover
	x_5 -	Working Capital / Current Assets	x_{20} -	Export
	x_6 -	Current ratio	x_{21} -	Added Value per Employee in k Euros
	x_7 -	Liquidity Ratio	x_{22} -	Total Assets Turnover
	x_8 -	Stock Turnover days	x_{23} -	Operating Profit Margin
	x_9 -	Collection Period days	x_{24} -	Net Profit Margin
	x_{10} -	Credit Period days	x_{25} -	Added Value Margin
	x_{11} -	Turnover per Employee k Euros	x_{26} -	Part of Employees
	x_{12} -	Interest / Turnover	x_{27} -	Return on Capital Employed
	x_{13} -	Debt Period days	x_{28} -	Return on Total Assets
	x_{14} -	Financial Debt / Equity	x_{29} -	EBIT Margin
	x_{15} -	Financial Debt / Cashflow	x_{30} -	EBITDA Margin

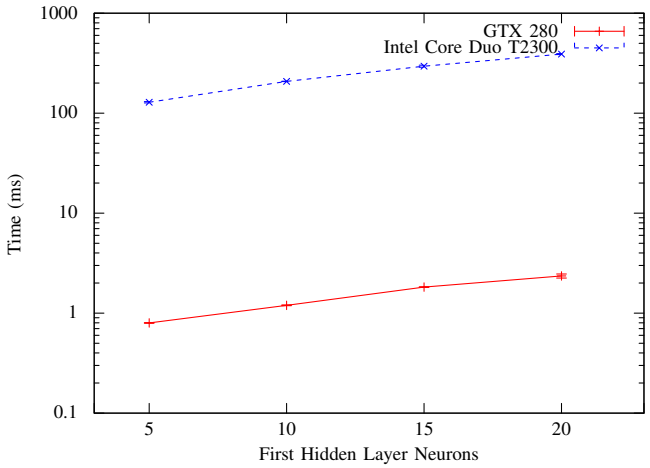

Fig. 4. Number of milliseconds required to train one epoch using the MBP algorithm in the French Financial Data.

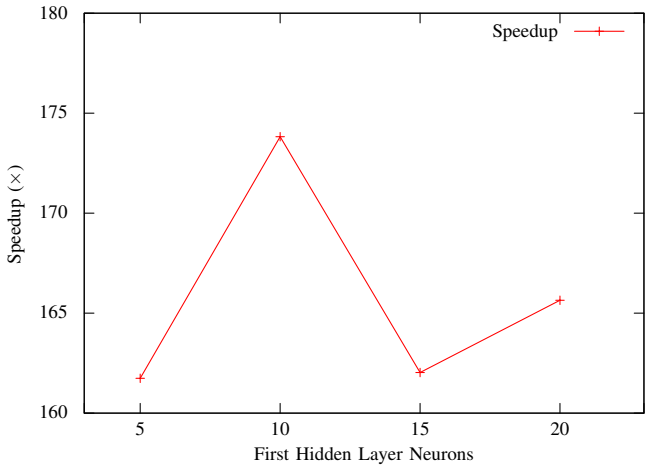

Fig. 5. Average speedup (×) delivered by the GPU, relatively to the CPU, for the MBP algorithm using the French Financial Data.

with 90 inputs. The experiments were performed with 10 networks per configuration which allowed to obtain mean values and the corresponding standard deviations. The configuration with two hidden layers was found to yield better results. The GPU implementation was tested in a NVIDIA GTX 280 with 30 SM (240 cores). The sequential processing standalone version run on a Intel Core 2 T2300 CPU running at 1.66 GHz. Both algorithms BP and MBP were parallelized. In particular, the sequential BP used for comparison includes an improved implementation with both adaptive learning rate and adaptive *momentum*[3]. We run the experiments for both algorithms and both hardware implementations under the same conditions, i.e the training was stopped considering a RMS (root mean square) error less than 0.01. Figure 4 depicts the number of milliseconds necessary to train one epoch versus the number of hidden neurons in the first layer for MBP in both hardware types considered: GPU and a dual core CPU processor chip. Figure 5 shows the the corresponding average speedups obtained by GPU. The largest speedup (173.82) is obtained when using 10 hidden neurons in the first hidden layer.

Table III presents performances measures obtained on the test data set for both NNs implementations. The meanings of the second row titles (from 4th column onwards) are as follows: R is Recall, P is Precision, F1 is F1 measure, E1 is Type I Error, and E2 is Type II Error. In general the results are quite good, with high F1 (tradeoff between Recall and Precision) and low Type II error. In the case of the bankruptcy prediction the Type II error is of greater importance, since it corresponds to the inability of the learning machine to detect that a company is at risk making it difficult for creditors to take the correct decision. Despite the importance of type II errors, most bankruptcy prediction methods take into account only the global classification error. Furthermore, the characteristics of the hybrid architecture running MBP (see [9]), which consists of two networks (the main network and the space network) make this design well-suited for this problem. The selective actuation neurons of the space network tune nicely the training scheme by casting the input patterns into their appropriate range, thus yielding an accurate model prediction capability.

Three main aspects are worth mentioning. First, we observe (see Figure 4) the trend of the depicted timing curves as the number of hidden neurons increase for both platforms. In the sequential processing implementation task there is an increase of complexity whereas in the parallel version there is a clear benefit in terms of both time and performance. Second, we notice competitive performance results from the parallel version (both algorithms) as compared with the sequential version (both algorithms). For instance, the results for the MBP (20 neurons in the first hidden layer) in the GPU hardware are impressive. Moreover, despite the very low reduced time in the training phase, the best result is attained for F1 while a very low Type II error is also achieved. In the latter it means that the machine commits only a few errors to detect that a company is at risk to bankrupt. Finally, the prediction is very fast. To sum it up, Type II errors for the MBP in GPU (presented in bold in Table III) are lower than the sequential implementation of the algorithm (the standard deviations are also very small) which adds to our analysis that despite of being fast the decision is subjected to less errors. In summary, above properties are highly desirable for financial decisors.

V. Conclusion and Future Work

In the midst of a global economic crisis, the availability of accurate instruments for fast and reliable decision making by financial regulators is strongly desirable. A step forward in this direction is the design of automated recognition systems for bankruptcy detection using GPU. The graphics processing units have enjoyed a great deal of interest recently for being programmed to solve high-dimensional problems with a large amount of sampled data. Our contribution is the parallel

[3]MBP Software Toolbox (see Subsection III-A) has been used for the experiments.

TABLE III
PERFORMANCE MEASURES (%) (MEAN) AND (STDEV): GPU VERSUS CPU

Neural Network		Hardware	GPU GTX 280					CPU - Core 2 T2300				
Algorithm	Configuration	Statistics	R	P	F1	E1	E2	R	P	F1	E1	E2
BP	(5-2-1)	mean	90.23	90.89	90.55	9.67	9.77	91.17	91.64	91.40	8.91	9.37
		stdev	2.02	0.70	1.03	0.88	2.02	1.38	1.34	1.05	1.51	1.36
BP	(10-2-1)	mean	90.68	91.90	91.28	8.56	9.32	91.97	91.20	91.55	9.55	8.61
		stdev	1.45	0.98	0.81	1.16	1.45	1.91	1.84	0.69	2.35	1.73
BP	(15-2-1)	mean	91.49	92.73	92.10	7.66	8.51	92.78	91.05	91.89	9.77	7.85
		stdev	1.12	0.76	0.86	0.81	1.12	1.60	0.79	0.62	1.06	1.57
BP	(20-2-1)	mean	91.17	92.25	91.70	8.20	8.83	91.56	92.31	91.90	8.22	8.89
		stdev	1.15	1.14	0.69	1.36	1.15	1.39	1.13	0.058	1.41	1.31
MBP	(5-2-1)	mean	91.28	91.44	91.35	9.15	8.72	92.93	90.20	91.53	10.84	7.77
		stdev	0.89	1.16	0.71	1.36	0.89	1.60	1.57	0.62	2.09	1.50
MBP	(10-2-1)	mean	91.21	91.44	91.31	9.15	**8.79**	91.56	92.41	91.98	8.05	8.93
		stdev	0.96	1.35	0.70	1.63	**0.96**	0.82	0.89	0.67	1.02	0.81
MBP	(15-2-1)	mean	90.86	91.79	91.31	8.72	**9.14**	91.24	91.49	91.36	9.08	9.34
		stdev	1.14	1.50	0.73	1.77	**0.14**	0.15	0.54	0.27	0.63	1.14
MBP	(20-2-1)	mean	91.31	92.09	**91.69**	8.41	**8.69**	91.61	90.91	91.24	9.83	9.01
		stdev	0.88	1.20	**0.56**	1.43	**0.88**	1.80	1.40	0.90	1.75	1.72

implementation of the Multiple backpropagation algorithm (MBP) for training a Multiple Feed Forward neural network (MFF) bringing to the edge the design of a nonparametric prediction model for bankruptcy prediction in a case study of the French market. Its hybrid nonparametric nature enabled modeling irregularities in the system overall function over the feature space yielding improved accuracy as well as decreasing Type II errors. From the experimental results, we conclude that an implementation running in GPU presents high speedups as compared to the running time of a CPU version. Besides being accurate, our model is able to predict much faster. Future work will study how to increase the system bias-variance tradeoff possibly including consensus among chosen classifiers.

APPENDIX

ACKNOWLEDGMENT

Financial support from "Fundação da Ciência e Tecnologia" under the project PTDC/GES/70168/2006 is gratefully acknowledged.

REFERENCES

[1] E. I. Altman. Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *Journal of Finance*, 23(4):589 – 609, September 1968.

[2] E. I. Altman. *Corporate Financial Distress and Bankruptcy: A Complete Guide to Predicting and Avoiding Distress and Profiting from Bankruptcy*. John Wiley & Sons, 2nd edition, 1993.

[3] J.A. Anderson, C.D. Lorenz, and A. Travesset. General purpose molecular dynamics simulations fully implemented on graphics processing units. *J. Computational Physics*, 227(10):5342–5359, 2008.

[4] A.F. Atiya. Bankruptcy prediction for credit risk using neural networks: A survey and new results. *Neural Networks, IEEE Transactions on*, 12(4):929 – 935, Jul 2001.

[5] A. Brandstetter and A. Artusi. Radial basis function networks GPU based implementation. *IEEE Transactions on Neural Networks*, 19(12):2150–2154, 2008.

[6] T. Brandvik and G. Pullan. Acceleration of a 3D euler solver using commodity graphics hardware. In *48th AIAA Aerospace Sciences Meeting and Exhibit*, page 607. AIAA Press, 2008.

[7] S. Che, M. Boyer, J. Meng, D. Tarjan, J.W. Sheaffer, and K. Skadron. A performance study of general-purpose applications on graphics processors using CUDA. *Journal of Parallel and Distributed Computing*, 68(10):1370–1380, 2008.

[8] Huang Fu-yuan. A genetic fuzzy neural network for bankruptcy prediction in chinese corporations. In *International Conference on Risk Management & Engineering Management (ICRMEM '08)*, pages 542 – 546, Nov. 2008.

[9] Noel Lopes and Bernardete Ribeiro. An efficient gradient-based learning algorithm applied to neural networks with selective actuation neurons. *Neural, Parallel and Scientific Computations*, 11:253–272, 2003.

[10] Noel Lopes and Bernardete Ribeiro. GPU implementation of the multiple back-propagation algorithm. In *Proceedings of Intelligent Data Engineering and Automated Learning*, pages 449–456. LNCS, Springer-Verlag, 2009.

[11] NVIDIA. Graphics hardware specifications, available [on line]: `http://www.nvidia.com`. 2006.

[12] K.-S. Oh and K. Jung. GPU implementation of neural networks. *Pattern Recognition*, 37(6):1311–1314, 2004.

[13] G.A. Rekba Pai, R. Annapoorani, and G.A. Vijayalakshmi Pai. Performance analysis of a statistical and an evolutionary neural network based classifier for the prediction of industrial bankruptcy. In *IEEE Conference on Cybernetics and Intelligent Systems*, volume 2, pages 1033 – 1038, 2004.

[14] P. Ravi Kumar and V. Ravi. Bankruptcy prediction in banks and firms via statistical and intelligent techniques - a review. *European Journal of Operational Research*, 180(1):1 – 28, July 2007.

[15] B Ribeiro, C Silva, A Vieira, and J Carvalho das Neves. Extracting discriminative features using non-negative matrix factorization in financial distress data. In M. Kolehmainen et al. (Eds.), editor, *Int Conf on Adaptive and Natural Computing Algorithms*, volume 4432, pages 537–547, Berlin Heidelberg, April 2009. Lecture Notes in Computer Science (LNCS), Springer-Verlag.

[16] B Ribeiro, A Vieira, J Duarte, C Silva, J C das Neves, Q Liu, and A H. Sung. Learning manifolds for bankruptcy analysis. In M. Köppen et al. (Eds.), editor, *Int. Conf. on Neural Information Processing*, volume 5506, pages 722–729, Berlin Heidelberg, 2009. Lecture Notes in Computer Science (LNCS), Springer-Verlag.

[17] Sam S. Stone, Haoran Yi, Justin P. Haldar, Wen-Mei W. Hwu, Bradley P. Sutton, and Zhi-Pei Liang. Accelerating advanced MRI

reconstructions on GPUs. In *5th International Conference on Computing Frontiers*, 2008.

[18] V. Volkov and J.W. Demmel. LU, QR and Cholesky factorizations using vector capabilities of GPUs. Technical Report EECS-2008-49, EECS Dept., Univ. of Calif., Berkeley, 2008.