Lecture Notes in Computer Science Vol. 5768: 789-798, 2009

Almost Random Projection Machine

Włodzisław Duch and Tomasz Maszczyk

Department of Informatics, Nicolaus Copernicus University
Grudziądzka 5, 87-100 Toruń, Poland
{wduch,tmaszczyk}@is.umk.pl
http://www.is.umk.pl

Abstract. Backpropagation of errors is not only hard to justify from biological perspective but also it fails to solve problems requiring complex logic. A simpler algorithm based on generation and filtering of useful random projections has better biological justification, is faster, easier to train and may in practice solve non-separable problems of higher complexity than typical feedforward neural networks. Estimation of confidence in network decisions is done by visualization of the number of nodes that agree with the final decision.

Key words: Neural networks, learning, random projections

1 Introduction

The discovery of backpropagation of errors (BP) algorithm [1] for training of the multilayer perceptrons (MLPs) broke the deadlock of training non-linear systems to solve non-separable problems. However, the degree of non-separability that can be handled, measured by the k-separability index [2], is rather low. Although various version of backpropagation algorithm can deal with the XOR problem finding optimal solution for higher than the 4-bit parity problems without assuming special architecture and initialization of the network is quite hard. The original BP article was entitled "Learning internal representations by error propagation", however these internal representations have rarely been analyzed, because they are not too informative. Neural networks that have clear neurobiological motivation create sparse, simple representation in their hidden layers [3]. Popular MLP neural networks are much simpler, they do not use internal inhibition and their only bias towards simple solutions is based on regularization [4], smoothing the mapping implemented by the network. This is not an appropriate bias for problems with complex logical structure, therefore poor generalization should be expected. Analysis of other useful biases and realistic learning targets is quite fruitful [5].

Biological neural networks solve complex learning problems inherent in optimization of behavior, creation of internal models, understanding of linguistic patterns. Creating algorithms capable of solving problems of similar complexity is an important challenge and is needed to open the doors for a new generation of ambitious machine learning applications. Backpropagation of errors is hard to justify from the neurobiological perspective. Algorithms that are biologically plausible and should be able to learn complex functions are therefore of great interest. Deep belief networks are one interesting candidate [6]. Maass et al. [7] have stressed that high-dimensional dynamics allows for real-time computing without stable states. Instead of attractor neural networks that require extensive training his Liquid State Machines can use much simpler high-dimensional dynamics that corresponds to very complex microcircuit resonances. Perceptrons may work then as readout neurons, extracting in real-time stable information from transient internal states formed by such high dimensional system. Kernel machines [8] work on similar principle, implicitly projecting data into high-dimensional spaces, where decision borders become flat and separation by linear hyperplanes is relatively easy.

The almost Random Projection Machine (aRPM) algorithm presented here is based on the following inspirations. Learning to read, learning multiplication table or similar tasks takes weeks, although brain plasticity of children is higher than adults. Synaptic learning is usually rather slow and it takes a long time before new connections will develop. Yet even old people may quickly learn and remember many things after a single exposure. The amount of synaptic learning must thus be rather limited. Neurons in association cortex form strongly connected microcircuits found in cortical minicolumns, resonating with different frequencies when an incoming signal $X(t)$ appears. A perceptron neuron “observing” the activity of thousands of microcircuits in these minicolumns learns to react to specific signals around particular frequency. However, resonators do not get excited when overall activity (weighted combination of inputs $\mathbf{W} \cdot \mathbf{X}$) is high, but rather react when specific levels of activity are reached, selecting only signals from some soft interval $G(\mathbf{W} \cdot \mathbf{X})$. If these signals are correlated with important activity its contribution is taken into account, otherwise the signal is not used.

The feature space created in this way is based on those combinations of inputs that have been found interesting for some task, and thus have some meaning and interpretation. These features are not learned but selected from random projections, with new features added if they show interesting correlations with some aspect of the problem being solved. In classification this would mean a subset of vectors from a single class, some of which have not yet been captured by too many other features and thus carry interesting information. In fact this model is not too far from the original Selfridge Pandemonium architecture [9], where demons, representing interesting observations, shout to influence decisions of demons that are higher in the hierarchy.

In the next section aRPM algorithm is formally introduced, including relations to the research on random projections. Section three presents empirical tests and comparisons with standard machine learning methods, and the last section some conclusions.

2 Almost random projections

Presentation of new input activates large number of microcircuits in the cortex, but competition and local inhibition will finally leave only a small number of the most active circuits that provide relevant information. They provide several views on the same data, in each case discovering a particular angle and projecting a group of similar (from this particular angle) cases, while cutting off the remaining cases from the projection. A simple threshold neuron may then read out the level of activation of specific circuits, estimating familiarity of the presented item by activation proportional to the number of clusters from each category that this item excites. Similar idea has been used in the liquid state machines [7] designed to analyze spatio-temporal patterns. Many random oscillators are postulated, projecting the signal into highly dimensional space, and a threshold neuron is used to read out the activity of the column and discriminate between different categories. In this paper random filters are used and those that find something interesting are selected to contribute to the output.

Multi-layer perceptron (MLP), the most successful neural network model, is based on a perceptron model, or a neuron that performs soft threshold logic operation using weighted sum of input signals [10]. This is a rough but useful abstraction of activity of a single biological neuron. Logical threshold neurons, for various noisy input signal distributions concentrated around some average values, estimate conditional probabilities that change in a sigmoidal way, depending on the strength of the signal [11]. Perceptrons may thus be seen as logical devices operating on noisy data. Many random perceptrons form a hidden layer that projects the data into high-dimensional space.

Two important ideas come from such neurocognitive inspirations. First, many views of the same item should be considered, generating interesting transformations $\mathcal{T}_i(X)$ that involve non-local projections $\mathbf{W}_i \cdot \mathbf{X}$. Such projections are filtered through localized functions $\mathcal{T}_i(X) = G_i(\mathbf{W}_i \cdot \mathbf{X})$ discovering useful features specific to a given category. The number of features should not be fixed, as they are dynamically generated until there is sufficient information to make decision. The interplay between local and global analysis has been missing in neural networks and other types of machine learning algorithms. Transformations $\mathcal{T}_i(X)$ map input cases to one-dimensional clusters that should be either relatively pure or at least partially discriminative, excluding some categories [12]. A single large projected cluster is sufficient for categorization if there is no strong competition, but some redundancy should be preferred. The winner-takes-most mechanism of biological networks should be approximated to make final decision based on memberships in projected clusters [3]. It is surprising that so far neural networks took only the simplest inspirations from biology.

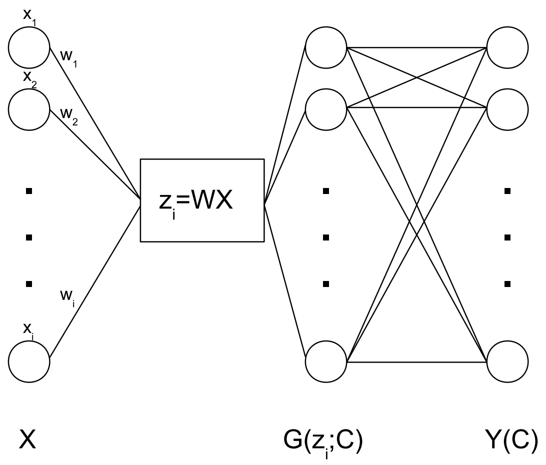

Fig. 1. Network structure of the aRPM algorithm.

Algorithm 1 aRPM

```
Require: Flag all vectors $\mathbf{X}$ as "new".
 1: for $i = 0$ to $N_{rep}$ do
 2:     Randomly assign weights $\underline{w}_i$, $w_i \in [-1, 1]$.
 3:     Generate new projection $z_i = \underline{w}_i \mathbf{X}$.
 4:     Analyze $p(z_i|C)$ distributions to determine "intersting" clusters.
 5:     Add them as new features $G(z_i; C)$, or class-labeled hidden network nodes.
 6:     Sum the activity of hidden node subsets for each class to calculate network outputs
        $y(C|\mathbf{X}) = \sum_i G(z_i; C)$.
 7:     Remove flag "new" from all vectors that reach $y(C|\mathbf{X}) \geq \beta$.
 8: end for
    Validate the network.
 9: if Accuracy does not increase then
10:     return network.
11: else
12:     goto 1
13: end if
```

The aRPM algorithm inspired by the ideas mentioned above has only a few parameters (see Algorithm 1). First, a relevance index [13] is applied to determine if the projected cluster is interesting, taking into account only "new" vectors, that is those that have not been already covered more than β times by other clusters (if $\beta = 1$ new nodes should cover training vectors only once). Boosting-like variants may be considered [14], but here only the simplest version is tested. Any filter based on information indices, purity or other criterion is suitable. Second, to justify adding new features (attaching hidden node to the output) new clusters that give rise to features should not be too small, covering at least α fraction of all vectors, and at least one new vector. Intervals $G(z_i; C)$ that extract clusters from projections may for example be modeled by a difference of two logistic functions, providing a soft trapezoidal function [15], but below only a simple [min,max] intervals have been used. The number of repetitions N_{rep} has been set here to 10.

The weights in this algorithm are randomly selected, and their values do not change – no additional learning of weights is needed, in contrast to all other methods for neural network training. In some feedforward network models learning is restricted only to the linear output layer [16], but here it is replaced by a simple addition of appropriate inputs. The aRPM simply generates sufficient number of random weights and selects new useful features $G(z_i; C)$. Each node corresponds then to a perceptron capturing clusters that may be surrounded by samples from other classes. To make final decision aRPM uses winner-takes-most mechanism.

Many variants of basic aRPM algorithm are possible and will be presented in a longer paper. New features may be used in a Naive Bayes type of estimation, but then one should avoid redundant nodes, while in the additive model the more nodes are excited by a given vector the better. There is no reason why all clusters should be pure, although projections that overlap with existing ones may be discouraged to generate more interesting views on the data. Additional parameter γ may determine the threshold for relevance of the new feature, for example the purity level of each new cluster.

Another possibility is to introduce weights proportional to the size of the cluster (number of vectors that pass through a particular filter), or use linear discrimination on the activity of the hidden nodes (this is more costly). $G(z_i; C)$ may reflect class-dependent probability density.

3 Illustrative examples

The usefulness of aRPM algorithm has been evaluated on two artificial and four real datasets downloaded from the UCI Machine Learning Repository [17] and from [18] (Leukemia). A summary of these datasets is presented in Tab. 1. Artificial 8 and 10-bit parity datasets have been selected because they are very difficult to analyze correctly by standard MLPs, Support Vector Machines or other machine learning algorithms. The four other datasets are standard examples of benchmark type and are used here to enable typical comparison of different learning methods. Leukemia has 7129 dimensions and it would be quite easy to get perfect results with such a large space, therefore 100 best features from a simple Fischer Discriminant Analysis (FDA) ranking index have been used [13]. Vectors with missing values have been removed (6 vectors from Heart disease dataset, and 16 from Wisconsin cancer), although it is quite easy to use projections based only on features that have been defined (formally undefined features should turn off nodes that use it, but it is enough to give it sufficiently low value).

Title	#Features	#Samples	#Samples per class		Source
Parity8	8	256	128 even	128 odd	artificial
Parity10	10	1024	512 even	512 odd	artificial
Leukemia	100	72	47 ALL	25 AML	[18]
Heart	13	297	160 absence	137 presence	[17]
Wisconsin	10	683	444 benign	239 malignant	[19]
Liver	6	345	145 C_1	200 C_2	[17]

Table 1. Summary of datasets used for comparison of aRPM algorithm with other methods.

To compare aRPM with 4 popular classification methods 10-fold crossvalidation tests have been repeated 10 times and average results collected in Table 2, with accuracies and standard deviations for each dataset. Additionally in each column the complexity of the generated models have also been noted: for C4.5 size of the tree, for kNN the number of nearest neighbours, for SVM the number of support vectors, and for MLP and aRPM the number of hidden nodes. Only linear SVM has been used as for these dataset results obtained with Gaussian kernel are not better. Parameters of all classifiers have been optimized. In case of aRPM pure clusters were enforced, with the minimum number of vectors in each cluster set as 1% of all vectors for the Heart and Wisconsin datasets, for the Leukemia at least 10 vectors have been required, and for the parity very large clusters were enforced, for the 10-bit parity over 200 elements.

Dataset	Method				
	C4.5	kNN	MLP	SVM	aRPM
Parity8	31.6 ± 1.3 (1)	100 ± 0 (17)	94.1 ± 2.1 (17)	32.4 ± 4.4 (230)	99.2 ± 1.6 (12)
Parity10	40.4 ± 1.6 (1)	100 ± 0 (21)	89.2 ± 12.3 (21)	39.1 ± 6.5 (920)	99.5 ± 0.9 (12)
Leukemia	82.6 ± 8.3 (5)	97.2 ± 1.6 (2)	95.8 ± 3.6 (52)	98.7 ± 3.9 (15)	96.1 ± 8.6 (19)
Heart	77.8 ± 2.1 (33)	81.8 ± 6.6 (45)	79.5 ± 1.3 (8)	81.5 ± 1.3 (94)	78.3 ± 4.2 (43)
Wisconsin	94.7 ± 2.0 (21)	97.0 ± 1.7 (5)	94.2 ± 0.2 (6)	96.3 ± 2.1 (49)	97.9 ± 1.6 (30)
Liver	65.8 ± 2.2 (51)	62.0 ± 1.1 (44)	67.5 ± 3.1 (5)	69.2 ± 10.3 (236)	61.1 ± 5.1 (47)

Table 2. Accuracy results

High-dimensional parity problem is very difficult for most classification methods. Many papers have been published on special neural models for parity functions, and the reason is quite obvious. Linear separation cannot be easily achieved because this is a k-separable problem that should be separated into $n+1$ intervals for n bits [2, 20]. This is a very interesting example showing that aRPM solves quite easily difficult problems in almost perfect way even when most standard classifiers fails. Although kNN may also work perfectly well it requires $k > 2n$ for n-bit parity to overcome the influence of the nearest neighbors.

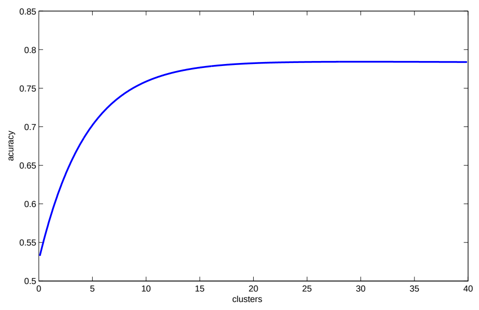

Fig. 2. Typical convergence curve, showing errors as a function of the number of nodes (Heart dataset); dashed line - training errors, solid line - test errors.

Reduced Leukemia dataset is classified by aRPM at 96%, significantly better than C4.5 result and at the similar level as all other systems. For Cleveland Heart data the new algorithm gives about $78 \pm 4\%$ accuracy, with the base rate of 54%. This is not significantly different from other classifiers because variance is rather high, although SVM has some advantage here, with over 100 support vectors creating rather complicated model. Wisconsin breast cancer dataset is classified by aRPM with higher accuracy then other classifiers, and relatively small variance $98 \pm 2\%$. For Liver dataset variance of most methods is quite high, but SVM may have some advantage, other classifiers do not give significantly different results.

These results are by no means the limits of aRPM algorithm as α and β parameters have not been optimized and $\gamma = 0$ was taken. How many hidden nodes should be created? If α is small more nodes are created, giving higher training accuracy, but clusters covered by these nodes have to be smaller, or less pure, so some learning of optimal values is needed. The simplest version used here is very fast as it does not perform any learning, except for setting the interval in one-dimensional projections. Typical convergence with respect to the number of nodes (Fig. 2) is quite fast and monotonic, quickly saturating. For example for the Heart data accuracy saturates at 78% for about $\approx$43 nodes.

aRPM classifier allows for easy estimation of confidence in the results. This is seen in Fig. 3 scatterplots. The network has two linear outputs and their value, for binary activations of hidden nodes, is simply an integer number between 0 and the number of nodes for each class. Each plot presents output of aRPM model trained inside crossvalidation and then applied to all dataset, thus showing test and training errors. Most vectors activate only nodes from the correct class, some of them as many as 8. Large pure clusters show high-confidence predictions. Some vectors are rejected and fall into (0,0) cluster; they may be assigned to a default majority class. In case of Heart one test vector excites as many as 5 nodes from the wrong class, showing that it may be an outlier. For each cluster confidence factor is equal to the purity of this node estimated in crossvalidation.

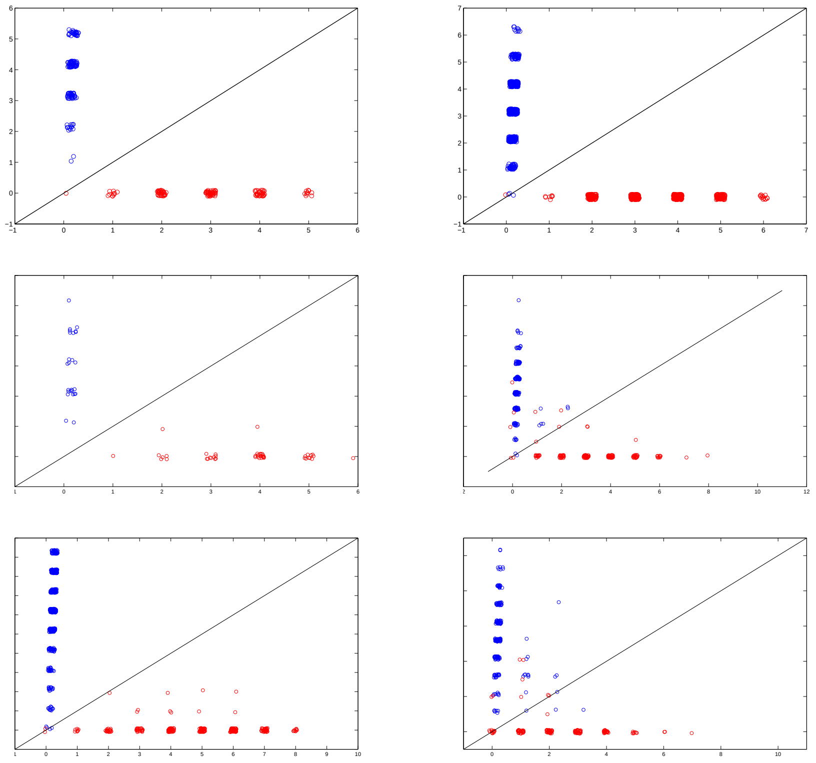

Fig. 3. Output of aRPM algorithm for training and test data, top row: Parity8 and Parity10, middle row: Leukemia and Heart, bottom row: Wisconsin and Liver.

4 Discussion and new directions

Neurocognitive informatics draws inspirations from neurobiological processes responsible for learning. So far only a few general inspirations have been used in computational intelligence: threshold neurons that perform parallel distributed processing, organized in networks. Even with our limited understanding of the brain many more inspirations may be drawn and used in practical learning and object recognition algorithms. Arguments for biological plausibility of random projections rather than slow learning mechanisms have been presented and simplest version of almost Random Projection Machine tested. Surprisingly, results of a model that does not perform any optimization are on benchmark problems at least as good as MLPs, and on parity problem which is quite difficult to learn for MLPs and SVMs are almost perfect.

Many variants of aRPM will be discussed in a longer paper: they improve results by optimizing minimal number of vector per cluster, adding impure clusters and enforcing minimal number of new vectors that have not been correctly classified, changing hard-limit intervals into more soft-window filters, and setting the threshold for similarity of new projections before new nodes are created. The brain does not use fixed number of features, as most pattern recognition algorithms do, but starting from a small number of features actively searches for new, most discriminative features that neural filters may provide. Objects are recognized using different features that characterize them. Thus feature selection and construction is not separable from the actual process of categorization and learning. This is easily incorporated into our algorithms by using subsets of all available original features to create new hidden nodes. If a very large (or small) number is inserted for the unknown value nodes that use this feature will be inactive while nodes that do not use it will provide normal activations. Visualization of outputs and the ability to estimate confidence in predictions made by such classifiers is a very useful feature of all variants of these algorithms.

The final goal of learning is to categorize, but the intermediate representations are also important. Finding interesting views on the data, or constructing interesting information filters, is the most important thing. Each filter does its own feature selection or feature weighting. Instead of using networks with fixed number of inputs systems that actively sample data, trying to "see it" through their filters, are needed. Once they have sufficient information to categorize data structures they have done their job. This opens the way to new algorithms that may learn from objects that have diverse structures, including many missing values. It is much easier to achieve non-linear separability in the hidden layers of neural networks than linear separability [5]. If the structure of non-linear mapping that creates image of data is known it may be then analyzed and understood. The most important part for good generalization in learning systems is to create large clusters, as small clusters are not reliable and will be washed out by neural noise. The learning process is greatly simplified by changing the goal of learning to easier target and handling the remaining nonlinearities with well defined structure.

Random projections facilitate rapid learning, but in biology rapid learning is followed by slow learning that perfects the function. Learning to increase usefulness of individual nodes to increase purity/separation of hidden nodes may follow initial creation of a functional network. Projection pursuit with Quality of Projected Clusters index [21] may be used for discovery of interesting views, and it should be very interesting to use it as an algorithm for deep belief networks [6] that are trained using Restricted Boltzmann Machines. All these approaches create interesting hidden representation of the data.

References

1. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In D. E. Rumelhart, J.L.M., ed.: Parallel Distributed Processing: Explorations in Microstructure of Congnition. Volume 1: Foundations. MIT Press, Cambridge (1986) 318–362
2. Duch, W.: k-separability. Lecture Notes in Computer Science **4131** (2006) 188–197
3. O'Reilly, R., Munakata, Y.: Computational Explorations in Cognitive Neuroscience. MIT-Press (2000)
4. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press (1995)
5. Duch, W.: Towards comprehensive foundations of computational intelligence. In Duch, W., Mandziuk, J., eds.: Challenges for Computational Intelligence. Volume 63. Springer (2007) 261–316
6. Hinton, G., Osindero, S., Teh, Y.: A fast learning algorithm for deep belief nets. Neural Computation **18** (2006) 381–414
7. Maass, W., Natschläger, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Computation **14** (2002) 2531–2560
8. Schölkopf, B., Smola, A.: Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA (2001)
9. Selfridge, O.G.: Pandemonium: A paradigm for learning. In Blake, D.V., Uttley, A.M., eds.: Proceedings of the Symposium on Mechanisation of Thought Processes. HM Stationery Office, London (1959) 511–529
10. Haykin, S.: Neural Networks - A Comprehensive Foundation. Maxwell MacMillian Int., New York (1994)
11. Duch, W.: Uncertainty of data, fuzzy membership functions, and multi-layer perceptrons. IEEE Transactions on Neural Networks **16** (2005) 10–23
12. Duch, W., Adamczak, R., Hayashi, Y.: Eliminators and classifiers. In Lee, S.Y., ed.: 7th International Conference on Neural Information Processing (ICONIP), Dae-jong, Korea (2000) 1029–1034
13. Duch, W.: Filter methods. In Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L., eds.: Feature extraction, foundations and applications. Physica Verlag, Springer, Berlin, Heidelberg, New York (2006) 89–118
14. Schapire, R., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Machine Learning **37** (1999) 297?–336
15. Duch, W., Jankowski, N.: Survey of neural transfer functions. Neural Computing Surveys **2** (1999) 163–213
16. Huang, G., Chen, L., Siew, C.: Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Transactions on Neural Networks **17** (2006) 879–892
17. Asuncion, A., Newman, D.: UCI machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html (2009)
18. Golub, T.: Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. Science **286** (1999) 531–537

19. Wolberg, W.H., Mangasarian, O.: Multisurface method of pattern separation for medical diagnosis applied to breast cytology. In: Proceedings of the National Academy of Sciences. Volume 87., U.S.A. (1990) 9193–9196
20. Grochowski, M., Duch, W.: Learning highly non-separable Boolean functions using Constructive Feedforward Neural Network. Lecture Notes in Computer Science **4668** (2007) 180–189
21. Grochowski, M., Duch, W.: Projection Pursuit Constructive Neural Networks Based on Quality of Projected Clusters. Lecture Notes in Computer Science **5164** (2008) 754–762