Reservoir Size, Spectral Radius and Connectivity in Static Classification Problems

Luís A. Alexandre[1] and Mark J. Embrechts[2]

[1] Department of Informatics, University of Beira Interior and
IT - Instituto de Telecomunicações, Covilhã, Portugal
[2] Rensselaer Polytechnic Institute - Decision Sciences and Engineering Systems
CII 5219, Troy, NY 12180, USA

Abstract. Reservoir computing is a recent paradigm that has proved to be quite effective given the classical difficulty in training recurrent neural networks. An approach to using reservoir recurrent neural networks has been recently proposed for static problems and in this paper we look at the influence of the reservoir size, spectral radius and connectivity on the classification error in these problems. The main conclusion derived from the performed experiments is that only the size of the reservoir is relevant with the spectral radius and the connectivity of the reservoir not affecting the classification performance.

1 Introduction

Recently a new paradigm for Recurrent Neural Networks (RNNs) has been proposed with the Echo State Networks (ESN) [1] and the Liquid State Machines (LSM) [2].It is called reservoir computation [3].

These types of networks have been used traditionally for time domain tasks. But a proposal for its use in static classification problems has been done in [4,5]. These papers showed how the reservoir approach could be used in static classification problems and made a benchmark study showing that this approach is quite competitive. One of the key issues is the specification of the reservoir: its size, connectivity and spectral radius. These values depend, of course, on the influence they have on the performance of the learning machine. A study on how they influence the performance is needed.

In this paper we perform such study: we made controlled experiments where only one of these parameters is varied at a time such that its effect on the performance can be observed. This is not straightforward since changing either the size, connectivity or the random initialization of the reservoir weights changes the spectral radius in a way that is not analytically foreseeable.

The presented experiments showed the following results: first, the size of the reservoir in terms of the number of neurons it possesses is directly related to the classification performance: as the size increases so does the error. Second, the spectral radius does not seem to have any influence on the performance, as long as it is under 1.0 (otherwise the stabilization process would not converge). This is in contrast to what seems to be the case of time dependent problems [6]. The third conclusion is that the connectivity does not seem to affect the performance either.

C. Alippi et al. (Eds.): ICANN 2009, Part I, LNCS 5768, pp. 1015–1024, 2009.

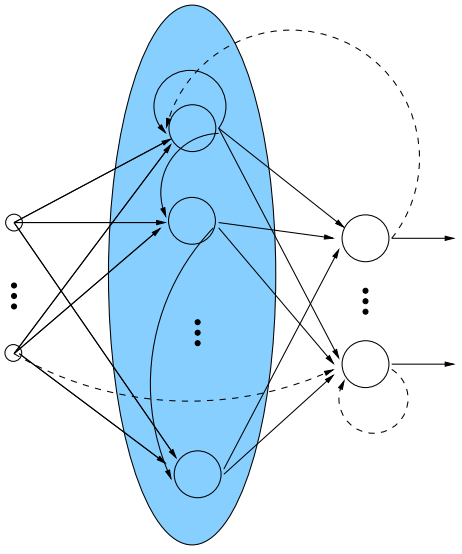

Fig. 1. Representation of a reservoir network. Large circles are the neurons and the small ones represent the input data. The middle (blue) layer is the reservoir. Dashed connections are optional (and not used in this paper). Only a small subset of all possible connections is represented.

The remaining of the paper is organized as follows: the next section presents a brief introduction to reservoir computing; section 3 explains how reservoir computing can be adapted to static classification problems. In section 4 we present the experiments and the last section contains the conclusions.

2 Reservoir Computing

The idea of reservoir computing revolves around using a large recurrent hidden layer (called the reservoir) with fixed weights and only adjusting the output layer weights (see figure 1). Since the output layer neurons use linear activation functions (usually the identity function), learning can be done with a linear system solver. This solves a big problem in traditional RNNs training which had to deal with the minimization of a usually complex function.

For the most simple case (the only recurrent connections are in the reservoir), we can write the state $x(t+1)$ of a N neurons reservoir at time $t+1$, of an ESN with K inputs and L outputs, as:

$$x(t+1) = f(Wx(t) + W^{in}u(t+1)) \tag{1}$$

where $f(\cdot)$ is the reservoir nonlinear activation function, W is the $N \times N$ reservoir weight matrix, W^{in} is the $N \times K$ input weight matrix and $u(t+1)$ is the input data.

The L-dimensional network output is given by

$$y(t) = g(W^{out}x(t)) \tag{2}$$

where $g(\cdot)$ is the output linear activation function and W^{out} is a $L \times N$ output weight matrix. We consider the activation function $g(\cdot)$ to be the identity function.

Regarding the solution of the linear problem that yields the output layer weights, any linear solver can be used. Consider m to be the number of training patterns. The states are collected in a $m \times N$ matrix, S, and the desired values go into a $m \times L$ matrix, D. The system to be solved is

$$S(W^{out})' = D \tag{3}$$

The problem with this formulation is that S is not squared and depends on m. With a simple operation it is possible to deal with both these issues: just multiply both sides by S'

$$(S'S)(W^{out})' = (S'D) \tag{4}$$

This transformation does not change the solution of the system but makes it independent of m and with the new matrix $R = S'S$, square.

Moreover, more evolved ways than the simple solver can be used, such as, ridge regression or partial least squares [7]. The problem with these approaches is that another parameter must be found: in the first case, the regularization parameter and in the second, the number of latent variables. In this paper we used the LAPACK's [8] linear solver to solve directly the system in (4) thus requiring no further parameters but not benefiting from regularization.

3 Reservoir Approach to Static Pattern Classification

The way to use a reservoir network for static pattern classification was introduced, to our best knowledge, in [4].

The issue was that the reservoir had a dependency of its state, $x(t+1)$, on the previous network state, $x(t)$, as can be seen in equation (1).

To break this dependency, that is meaningless in static classification tasks, a stabilization phase was proposed in [4]: keep each input signal present in the network input until the outputs of the reservoir neurons have almost no change. During stabilization the output layer is ignored. This way the reservoir can now process data that is not ordered in time.

The index t in the equations is only used to distinguish between different patterns and does not represent time.

The state equation is iterated (upper index) until $x(t+1)$ does not change significantly. Notice that we do not use the activation function here.

$$x(t+1)^{(i)} = Wx(t+1)^{(i-1)} + W^{in}u(t+1) \tag{5}$$

where $x(t+1)^{(0)} = 0$. Equation (5) replaces equation (1) in the case of static classification tasks.

4 Experiments

In this section we present experiments to evaluate the importance of the reservoir size, its spectral radius and connectivity.

4.1 Datasets

We chose 5 datasets freely available to perform the tests.

All datasets are from the UCI repository [9] with the exception of `Olive` which is from [10]. Table 1 contains their main features.

Table 1. Dataset list with its properties

Dataset	N. classes	N. features	N. points
`Diabetes`	2	8	768
`Iris`	3	4	150
`Olive`	9	8	572
`Sonar`	2	60	208
`Wine`	3	13	178

4.2 Parameters

Consider N to be the number of neurons in the reservoir, s the connectivity of the neurons in the reservoir (e.g., $s = 0.1$ means that only 10% of the possible connections are not null) and ρ to be the spectral radius (the size of the largest eigenvector) of the reservoir weight matrix.

The issue in these experiments was: how to vary only one of the parameters without varying the remaining ? It is not possible to change the spectral radius, ρ, directly. It depends on N, s, the distribution of the random weights in the reservoir and also on the range of possible values for these weights: $[-init, init]$. Changes in either N, s or $init$ will all change ρ in a way that is not possible to analytically predict since the matrices are randomly generated.

To be able to vary only one of the parameters we generated 10 random reservoirs for each triplet $(N, s, init)$ and ploted the average ρ. This was done keeping the value of N fixed. The left plot in figure 2 shows the results when N was kept at 200. The right plot presents the contours of left plot: this information was used to follow a contour and find the values of $init$ that made ρ constant for different values of s, for this particular value of N. The whole process was repeated for all values of N listed above.

A similar analysis was done for the case where s was fixed and N and $init$ varied.

Using the information gathered on the contours lines, we produced 10 reservoirs with the desired ρ up to an error of 0.005, by trial and error, for each combination on N, s and ρ used in the experiments: each reservoir was used in one of the repetitions of a 2-fold cross validation.

In the graphics below, the values of the parameters varied according to:

$N = \{20, 35, 50, 65, 80, 100, 150, 200, 250, 300, 350\}$,
$s = \{0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35\}$ and
$\rho = \{0.20, 0.40, 0.60, 0.80, 0.99\}$.

4.3 Results

Consider figures 3 to 7. Regarding the left plot, each color ball represents the average error of 10 repetitions of a 2-fold cross validation, on the respective dataset. The color

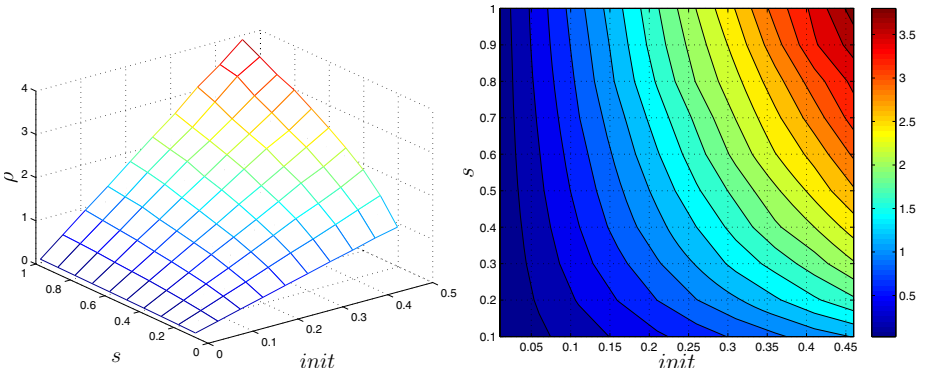

Fig. 2. The left plot shows the spectral radius as a function of connectivity and the half-size of the random reservoir weights $init$. The right figure contains the contour plot of left one. The values of the pairs (s,$init$) for the first 5 contours from the left were used to keep the spectral radius fixed for varying s.

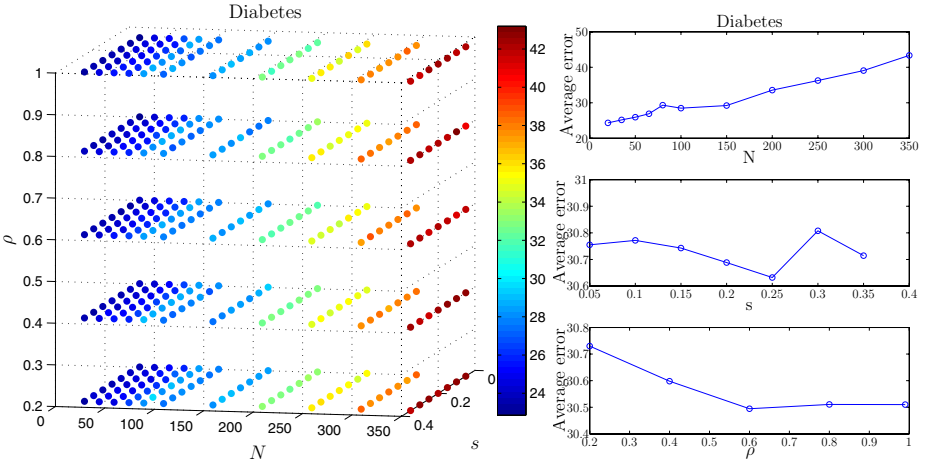

Fig. 3. Results for the `Diabetes` dataset

encodes the average error according to the scales by the main graphics. The right plots contain the average error for each of N, s and ρ found averaging the error over all the remaining parameters, i.e., for $N = 20$, the figures show the average error when $N = 20$ for all the possible values of s and ρ.

4.4 Condition Number

To evaluate if the problem with increasing the size of the reservoir is due to numeric instability in the linear system solver, we found the condition number of the matrix R for each dataset.

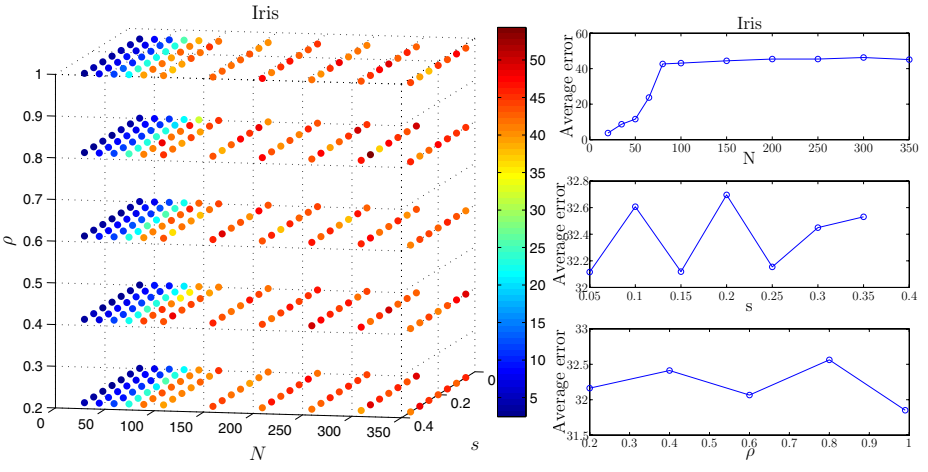

Fig. 4. Results for the `Iris` dataset

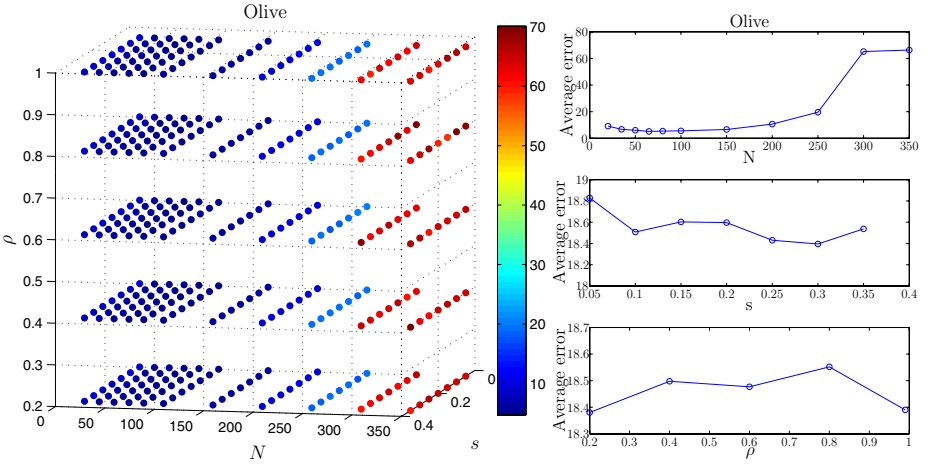

Fig. 5. Results for the `Olive` dataset

The condition number of a matrix A is $\kappa(A) = ||A||.||A^{-1}||$. If we choose the L2 norm we find that $\kappa(A) = \sigma_{max}/\sigma_{min}$ where σ_{max} is the maximum and σ_{min} the minimum singular values of A. The condition number should be small to ensure the stability of the numeric calculations.

Since the previous experiments showed that the values of the spectral radius and the connectivity did not influence the performance, we repeated the experiments just taking N into account (fixing the other two parameters).

Figure 8 shows the value of the condition number for each of the datasets as N varies.

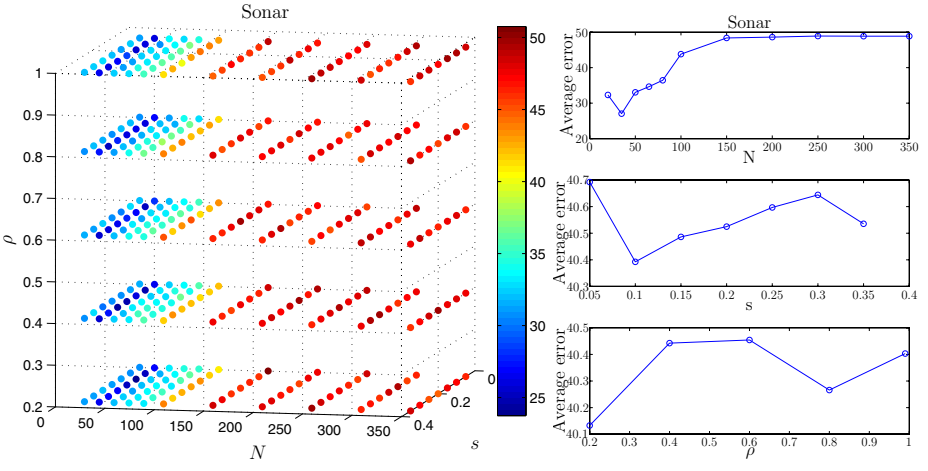

Fig. 6. Results for the `Sonar` dataset

Table 2. Best results and respective parameters for all datasets

Dataset	Error	Std. dev.	N	s	ρ
`Diabetes`	22.85	1.07	20	0.05	0.99
`Iris`	2.47	1.04	20	0.20	0.80
`Olive`	4.63	0.52	65	0.25	0.80
`Sonar`	23.75	2.9	35	0.15	0.20
`Wine`	3.99	0.93	20	0.10	0.20

4.5 Discussion

Table 2 presents the best results for each dataset and the respective parameters. The values do not point to a "best" value of s or ρ. But for N, we see that smaller values are preferred (3 out of 5 of the values are the smallests tested, 20; another is the next in terms of size, 35; the last one is 65 which is still much smaller than the largest tested value of 350).

From the figures 3 to 7 and table 2 we conclude that as N increases the error increases for all datasets, with the exception of `Sonar` in which case the error decreases from $N = 20$ to $N = 35$ and then continues increasing. The error for the `Olive` dataset also shows a small decrease for the first values of N but it rapidly increases as N grows. This is surprising as one would suppose no harm would come from an oversized reservoir, other than computational cost. What appears to be happening is a problem with the liner system solving. As can be seen from the values of the condition number of the matrix R in figure 8, the problems become ill-conditioned very fast with the increase in the size N.

The spectral radius and the connectivity do not seem to influence the performance at all. Notice the behavior of error as a function of these two parameters in the two lower right plots of figures 3 to 7 and also the figure's scale: parameter changes only make

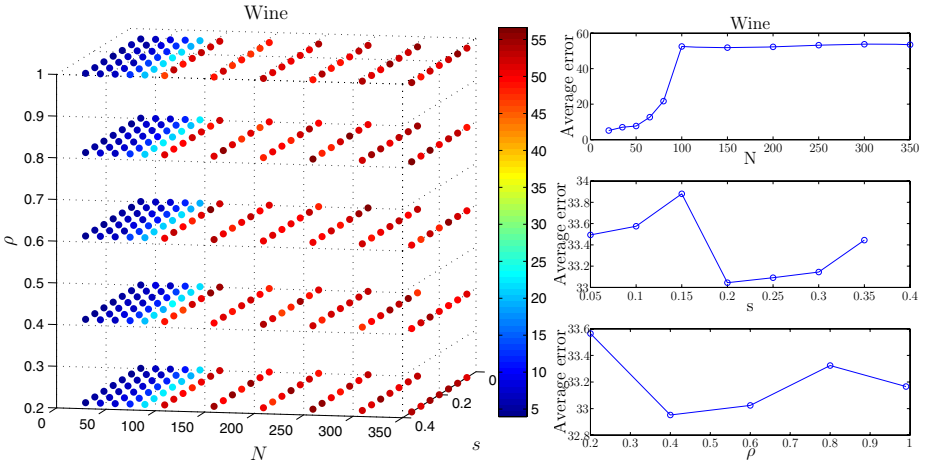

Fig. 7. Results for the `Wine` dataset

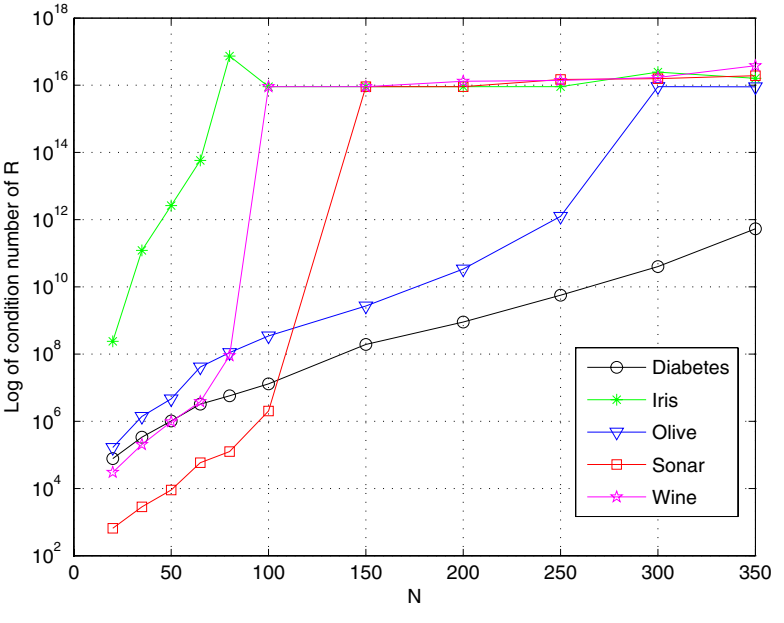

Fig. 8. Logarithm of the condition number of R for each of the datasets as N varies

minor random changes in the performance. This point is made clear in table 3 where the difference between the largest and the smallest values of the error when both the spectral radius and the connectivity are varied, is presented for all datasets, in absolute value and as a percentage of the mean error. The differences in the error that come from varying both the spectral radius and the connectivity are always under 0.9% in absolute terms

Table 3. Percentage changes in the error induced by changes in the spectral radius and the connectivity both in absolute and relative terms

	Spectral radius		Connectivity	
	Absolute [%]	Relative [%]	Absolute [%]	Relative [%]
Diabetes	0.24	0.77	0.18	0.57
Iris	0.71	2.21	0.58	1.79
Olive	0.17	0.93	0.43	2.33
Sonar	0.32	0.80	0.30	0.74
Wine	0.61	1.85	0.84	2.50

and under 2.5% in relative terms. With the exception of the change in the spectral radius for the Diabetes dataset that decreases the error as ρ increases (see bottom right plot in figure 3) there is no other linear relation between the changes in these parameters and the error variations. Even in this case, we believe that this apparent inverse correlation is casual since only 5 different values of ρ are considered.

Regarding the values of N at which the values of the error stop increasing (top right plot in figures 3 to 7), they are the same as the ones where the condition number also stops increasing. These values of N are the ones closer to the size of the training sets. Remember, the evaluation is done using 2-fold cross validation. For instance, in the case of Iris, the training set will have 75 points. The error shows its peak around this value of N (see the right plot in figure 4) and also the condition number stops increasing after this value (see figure 8). So one concludes that, when using a simple linear solver, the value of N should always be smaller than the size of training set.

5 Conclusions

This paper presented a study on the influence of several parameters on the classification performance of reservoir networks for static pattern classification.

The study showed that: 1) The spectral radius of the reservoir matrix does not influence the classification performance given that it remains in the interval $(0, 1)$ such that the stabilization process converges; 2) An increase in the size of the reservoir increases the error apparently as a result of the ill-conditioning of the linear system solver (improvements might be obtained using, for instance, ridge regression); 3) The connectivity does not change the results; 4) The value of N should not exceed the size of training set when using a simple linear solver.

References

1. Jaeger, H.: 'The echo state' approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, Fraunhofer Institute for Autonomous Intelligent Systems (2001)
2. Maas, W., Natschläger, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Computation 14(11), 2531–2560 (2002)

3. Verstraeten, D., Schrauwen, B., D'Haene, M., Stroobandt, D.: An experimental unification of reservoir computing methods. Neural Networks 20, 391–403 (2007)
4. Embrechts, M., Alexandre, L., Linton, J.: Reservoir computing for static pattern recognition. In: 17th European Symposium on Artificial Neural Networks – ESANN 2009, Bruges, Belgium (2009)
5. Alexandre, L., Embrechts, M., Linton, J.: Benchmarking reservoir computing on time-independent classification tasks. In: International Joint Conference on Neural Networks – IJCNN 2009, Atlanta, Georgia, USA, pp. 89–93 (2009)
6. Legenstein, R., Maass, W.: Edge of chaos and prediction of computational performance for neural circuit models. Neural Networks 20, 323–334 (2007)
7. Wold, S., Sjöström, M., Eriksson, L.: PLS-regression: a basic tool of chemometrics. Chemometrics and Intelligent Laboratory Systems 58, 109–130 (2001)
8. Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.: LAPACK Users' Guide, 3rd edn. Society for Industrial and Applied Mathematics, Philadelphia (1999)
9. Blake, C., Keogh, E., Merz, C.: UCI repository of machine learning databases (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
10. Forina, M., Armanino, C.: Eigenvector projection and simplified nonlinear mapping of fatty acid content of italian olive oils. Ann. Chem. 72, 125–127 (1981)