Noname manuscript No.
(will be inserted by the editor)

Optimized Parameter Search for Large Datasets of the Regularization Parameter and Feature Selection for Ridge Regression

Pieter Buteneers · Ken Caluwaerts · Joni Dambre · David Verstraeten · Benjamin Schrauwen

Received: date / Accepted: date

Abstract In this paper we propose mathematical optimizations to select the optimal regularization parameter for ridge regression using cross-validation. The resulting algorithm is suited for large datasets and the computational cost does not depend on the size of the training set. We extend this algorithm to forward or backward feature selection in which the optimal regularization parameter is selected for each possible feature set. These feature selection algorithms yield solutions with a sparse weight matrix using a quadratic cost on the norm of the weights.

A naive approach to optimizing the ridge regression parameter has a computational complexity of the order $O(RKN^2M)$ with R the number of applied regularization parameters, K the number of folds in the validation set, N the number of input features and M the number of data samples in the training set. Our implementation has a computational complexity of the order $O(KN^3)$. This computational cost is smaller than that of regression without regularization $O(N^2M)$ for large datasets and is independent of the number of applied regularization parameters and the size of the training set. Combined with a feature selection algorithm the algorithm is of complexity $O(RKNN_s^3)$ and $O(RKN^3N_r)$ for forward and backward feature selection respectively, with N_s the number of selected features and N_r the number of removed features. This is an order M faster than $O(RKNN_s^3M)$ and $O(RKN^3N_rM)$ for the naive implementation, with $N \ll M$ for large datasets.

To show the performance and reduction in computational cost, we apply this technique to train recurrent neural networks using the reservoir computing approach, windowed ridge regression, least-squares support vector machines (LS-SVMs) in primal space using the fixed-size LS-SVM approximation and extreme learning machines.

P. Buteneers
Electronics and Information Systems, Ghent University,
Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium
Tel.: +32 9 264 33 68
Fax: +32 9 264 35 94
E-mail: pieter.buteneers@ugent.be

Keywords Cross-validation · Feature Selection · Ridge Regression · Regularization Parameter Optimization · Computationally Efficient · Model Selection

1 Introduction

Linear regression is often used in machine learning to find an approximation of a desired output using a linear combination of its input features. It minimizes the sum of squared errors of the approximation. Ridge regression (RR) or L2 regularized linear regression [1] adds an additional term to the loss-function proportionate to the L2-norm of the output weights. The regularization parameter defines the relative cost of the weight magnitude and the sum-squared-error on the output. Finding a good regularization parameter is important to avoid overfitting on the training data. As the total cost is a non-convex function of the L2-norm, cross-validation is usually applied to optimize this parameter, which is often time consuming. For small datasets many algorithms have been proposed to speed-up the optimization process [2,3]. However the computational complexity and memory use of these techniques scale quadratically with the size of the dataset.

Feature selection [4] is a regularization technique which avoids overfitting by removing redundant features. Validating each possible combination of features is for most common tasks intractable. Therefore a forward or backward approach is mostly used to find a near to optimal set of input features. In forward feature selection (FFS), features are progressively added onto larger and larger subsets until no further performance increase is achieved. In contrast, backward feature selection (BFS) starts with a set containing all features. In every iteration, the removal of each feature is evaluated and the removal that cased the largest decrease in validation error is eliminated from the subset. The algorithm is repeated until there is no further improvement. Forward algorithms are generally faster and result in fewer features but backward algorithms often achieve a better performance because they tend to better preserve constructive relationships between seemingly irrelevant features [4].

RR does not automatically yield sparse weights as opposed to Lasso [5] or L1 regularized regression. However, many publications show the advantages of RR [6–8] or even linear regression [9] with sparse input features. In [7] and [9] a fast algorithm for feature selection with small datasets was proposed. The algorithm by Pahikkala et al. in [8] also combines feature selection with the optimization of the regularization parameter. However, none of these algorithms are suited for large datasets since their computational complexity and memory use still scale quadratically with the size of the dataset.

In this work, we propose mathematical optimizations for cross-validation on large datasets that allow for the optimization of the regularization parameter in combination with FFS and BFS on the regression input features. We also extend these algorithms to make them compatible with class reweighted RR (CRRR), which is optimized for classification tasks with unbalanced datasets. We apply our technique to reservoir computing (RC) [11], windowed RR, fixed-size least-squares support vector machines (FS-LS-SVMs) [12] and extreme learning machines (ELMs) [13]. For RC and windowed RR we apply it to an epileptic seizure detection system presented in [14]. For FS-LS-SVM we apply it to the UCI Adult dataset as described in [15] and the Heat Exchanger task as explained in [16], and for ELMs it is tested on several regression datasets used in [9]. A Python implementation is freely available as part of the Oger Toolbox[1] [17].

2 Ridge Regression

Ridge Regression minimizes the following loss function:

$$f_{loss} = ||\mathbf{X}_t\mathbf{W}_t - \mathbf{Y}_t||^2 + \lambda||\mathbf{W}_t||^2, \tag{1}$$

where $\mathbf{X}_t$ represents the input data from the training set, $\mathbf{Y}_t$ the desired output on the training data, $\mathbf{W}_t$ the output weights and λ the regularization parameter that adds an extra cost to the squared norm of the output weights. For mathematical convenience and since each desired output requires a different regularization parameter, we assume only 1 desired output throughout the rest of this work. Minimizing Equation (1) results in the following equation for the output weights:

$$\mathbf{W}_{t,opt} = (\mathbf{X}_t^T\mathbf{X}_t + \lambda\mathbf{I})^{-1}\mathbf{X}_t^T\mathbf{Y}_t. \tag{2}$$

If the optimal regularization parameter is known, calculating the optimal weights is of the order $O(N^2M + N^3)$ [18], with N the number of input features and M the number of data samples in the training set.

To find the optimal regularization parameter λ, a list of possible λs is created and cross-validation is applied to select the optimal λ. If the training and validation sets are representative for the test set, one can assume that when the validation error is minimized the test error gets reduced. The loss function on the validation set is defined as follows:

$$f_{loss,v,\lambda} = ||\mathbf{X}_v\mathbf{W}_t - \mathbf{Y}_v||^2, \tag{3}$$

where $\mathbf{X}_v$ represents the validation data and $\mathbf{Y}_v$ the desired output on the validation data. Typically an output weight matrix is trained for each training set and each value of the regularization parameter and then the validation error is calculated using Equation (3). If R represents the number of regularization parameters, K the number of validation sets and M the number of data points in the training set, the computational complexity of this procedure is of the order $O(RKN^2M)$.

2.1 Covariance method

For large datasets, with $N \ll M$, the computational cost of Equation (2) is of the order $O(N^2M)$. To speed up the optimization of the regularization parameter we can rewrite Equation (3) as follows:

$$\begin{aligned} f_{loss,v,\lambda} &= \mathbf{W}_t^T\mathbf{X}_v^T\mathbf{X}_v\mathbf{W}_t - 2\mathbf{W}_t^T\mathbf{X}_v^T\mathbf{Y}_v + \mathbf{Y}_v^T\mathbf{Y}_v \\ &= \mathbf{W}_t^T(\mathbf{X}_v^T\mathbf{X}_v\mathbf{W}_t - 2\mathbf{X}_v^T\mathbf{Y}_v) + \mathbf{Y}_v^T\mathbf{Y}_v \\ &= \mathbf{W}_t^T(\mathbf{A}_v\mathbf{W}_t - 2\mathbf{B}_v) + c_v, \end{aligned} \tag{4}$$

[1] The Oger Toolbox can be downloaded from http://www.reservoir-computing.org

where $\mathbf{A}_v$ denotes the covariance matrix of the validation data, $\mathbf{B}_v$ the cross-covariance vector of the validation data and the desired output and c_v the covariance number of the desired output. Since each output is considered independently, these covariance matrices have dimensions NxN, Nx1 and 1x1 respectively.

Following the technique presented in [19] to optimize the regularization parameter for FS-LS-SVMs, the covariance matrix $\mathbf{A}$ on the training and validation set combined can be calculated as follows:

$$\begin{aligned}\mathbf{A} &= \mathbf{X}^T\mathbf{X}\\ &= \begin{pmatrix}\mathbf{X}_t\\ \mathbf{X}_v\end{pmatrix}^T \begin{pmatrix}\mathbf{X}_t\\ \mathbf{X}_v\end{pmatrix}\\ &= \mathbf{X}_t^T\mathbf{X}_t + \mathbf{X}_v^T\mathbf{X}_v\\ &= \mathbf{A}_t + \mathbf{A}_v.\end{aligned}$$

Following a similar analogy for $\mathbf{B}$, $\mathbf{W}_t$ can be rewritten as follows:

$$\begin{aligned}\mathbf{W}_t &= (\mathbf{A}_t + \lambda\mathbf{I})^{-1}\mathbf{B}_t\\ &= (\mathbf{A} - \mathbf{A}_v + \lambda\mathbf{I})^{-1}(\mathbf{B} - \mathbf{B}_v).\end{aligned} \tag{5}$$

The covariance matrices, which need to be calculated to find the optimal weights using Equation 2, can be calculated for each validation set before starting the validation procedure. One matrix inversion is needed per possible regularization parameter to compute the $\mathbf{W}_t$ matrix. This is of order $O(N^3)$ in a typical implementation [18]. Calculating the covariance matrices and optimizing the regularization parameter now becomes of the order $O(N^2M + RKN^3)$, with R the number of regularization parameters. We will refer to this technique as the Covariance Method.

2.2 Eigen method

We can reduce the computational cost of determining $\mathbf{W}_t$ using the eigendecomposition of the real and symmetric covariance matrix $\mathbf{A}_t = \mathbf{A} - \mathbf{A}_v = \mathbf{C}_t\mathbf{D}_t\mathbf{C}_t^T$:

$$\begin{aligned}\mathbf{W}_t &= (\mathbf{C}_t\mathbf{D}_t\mathbf{C}_t^T + \lambda\mathbf{I})^{-1}(\mathbf{B} - \mathbf{B}_v)\\ &= (\mathbf{C}_t\mathbf{D}_t\mathbf{C}_t^T + \mathbf{C}_t(\lambda\mathbf{I})\mathbf{C}_t^T)^{-1}(\mathbf{B} - \mathbf{B}_v)\\ &= \mathbf{C}_t(\mathbf{D}_t + \lambda\mathbf{I})^{-1}\mathbf{C}_t^T(\mathbf{B} - \mathbf{B}_v).\end{aligned} \tag{6}$$

Although applied differently, the eigendecomposition has been used to reduce the computational cost in [20] and [21]. If for each validation set the following matrices are calculated beforehand, the computational requirements for each λ are further reduced:

$$\begin{aligned}\mathbf{B}_{Ct} &= \mathbf{C}_t^T(\mathbf{B} - \mathbf{B}_v)\\ \mathbf{A}_{Cv} &= \mathbf{C}_t^T\mathbf{A}_v\mathbf{C}_t\\ \mathbf{B}_{Cv} &= \mathbf{C}_t^T\mathbf{B}_v.\end{aligned}$$

For each λ we then calculate:

$$\mathbf{W}_{Ct} = (\mathbf{D}_t + \lambda\mathbf{I})^{-1}\mathbf{B}_{Ct}. \tag{7}$$

Since both $\mathbf{D}$ and $\mathbf{I}$ are diagonal matrices and $\mathbf{B}_{Ct}$ is a vector, the previous calculation can be done element-wise and is thus of the order N. If we integrate this in Equation (4) we get:

$$f_{loss,v,\lambda} = \mathbf{W}_{Ct}^T(\mathbf{A}_{Cv}\mathbf{W}_{Ct} - 2\mathbf{B}_{Cv}) + c_v.$$

$\mathbf{A}_{Cv}$ is a square matrix of dimension NxN and the other matrices are vectors of size N, so this is a vector-matrix multiplication of the order $O(N^2)$. It eliminates the matrix inversion of Equation (5) for each value of the regularization parameter which is of the order $O(N^3)$. However, one eigendecomposition ($O(N^3)$ [22]) needs to be computed for each validation set. Using this approach, optimizing the regularization parameter is of the order $O(KN^3 + RKN^2)$. If $R < N$ the order of the algorithm is $O(KN^3)$ and thus independent of the number of values for the regularization parameter R that are tested. If $KN < M$, which is mostly the case for large datasets, this is in fact lower than the computational cost of Equation 2. Finding the optimal regularization parameter and calculating the optimal output weights is then of the order $O(N^2M)$. This means that optimizing the regularization parameter adds little to no computational cost to RR for large datasets.

Applying the eigenvalue decomposition to an ill-conditioned matrix results in numerical errors [22]. This can be avoided by adding $\lambda_{max}\mathbf{I}$, with λ_{max} the largest regularization parameter, to the covariance matrices [22] and subtracting it afterwards from the eigenvalues $\mathbf{D}_t$.

3 Feature selection

Since RR does not in general yield sparse input features, it is often combined with feature selection [6–8]. If there are N input features to the RR algorithm and if N_s and N_r represent the number of selected and removed features in a forward or backward algorithm, respectively, naive implementations have a complexity of the order $O(RKNN_s^3M)$ and $O(RKN^3N_rM)$.

FFS and BFS can also be achieved by performing the operation described in the previous section on sub-matrices of the covariance matrices. Simply removing the row and column n for the NxN matrices and the row n for the Nx1 matrices, where n is the feature to be removed, will already be more efficient than the naive implementation. Selecting the best set of features and regularization parameters in this way, results in a complexity of the order $O(N^2M + KNN_s^4)$ and $O(N^2M + KN^4N_r)$ for the FFS and BFS respectively. In the following sections we will show how this can be further reduced. We start with BFS because it is conceptually easier.

3.1 Backward feature selection

Because the n-th element on the diagonal of $\mathbf{A}_t$ is inversely proportional to the sensitivity of the output on the n-th input [23, 24], one can remove a feature by setting the n-th diagonal element to ∞. If u is a vector containing zeros except for the n-th element which is equal to 1, we can calculate the reduced matrix inverse using the Sherman-Morrison formula [25] as follows:

$$\begin{aligned}
\mathbf{A}_r^{-1} &= \lim_{\gamma\to\infty} (\mathbf{A}_t + \gamma u u^T)^{-1} \\
&= \lim_{\gamma\to\infty} \left(\mathbf{A}_t^{-1} - \frac{\mathbf{A}_t^{-1}\gamma u u^T \mathbf{A}_t^{-1}}{1+\gamma u^T \mathbf{A}_t^{-1} u} \right) \\
&= \mathbf{A}_t^{-1} - \lim_{\gamma\to\infty} \left(\frac{\gamma \mathbf{A}_t^{-1} u u^T \mathbf{A}_t^{-1}}{1+\gamma u^T \mathbf{A}_t^{-1} u} \right) \\
&= \mathbf{A}_t^{-1} - \frac{\mathbf{A}_t^{-1} u u^T \mathbf{A}_t^{-1}}{u^T \mathbf{A}_t^{-1} u}.
\end{aligned}$$

If we replace $\mathbf{A}$ by its eigen-decomposition in the previous equation we get:

$$\begin{aligned}
\mathbf{A}_r^{-1} &= \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^T - \frac{\mathbf{C}\mathbf{D}^{-1}\mathbf{C}^T u u^T \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^T}{u^T \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^T u} \\
&= \mathbf{C}\left(\mathbf{D}^{-1} - \frac{\mathbf{D}^{-1}\mathbf{C}^T_{(n,:)}\mathbf{C}_{(n,:)}\mathbf{D}^{-1}}{\mathbf{C}_{(n,:)}\mathbf{D}^{-1}\mathbf{C}^T_{(n,:)}} \right)\mathbf{C}^T,
\end{aligned}$$

with $\mathbf{C}_{(n,:)}$ the n-th row of $\mathbf{C}$. If we introduce this in equations (6) and (7) we find that the output weights with a removed feature become:

$$\mathbf{W}_{Ctr} = \mathbf{W}_{Ct} - (\mathbf{D}+\lambda\mathbf{I})^{-1}\mathbf{C}^T_{(n,:)} \frac{\mathbf{C}_{(n,:)}\mathbf{W}_{Ct}}{\mathbf{C}_{(n,:)}(\mathbf{D}+\lambda\mathbf{I})^{-1}\mathbf{C}^T_{(n,:)}}.$$

Because $\mathbf{D}$ is a diagonal matrix and $\mathbf{W}_{Ct}$ and ${\mathbf{C}_{(\mathbf{i},:)}}^T$ have dimension Nx1, this equation is again of the order $O(N)$. Testing the exclusion of 1 feature is thus of the same order of complexity as testing one ridge-regression parameter, $O(N^2)$. If this is tested for each of the K validation sets, each of the R regularization parameters and, in the worst case, each of the N features, this becomes $O(RKN^3)$, which is of higher than that of the eigenvalue decomposition. We need to repeat this process N_r+1 times, with N_r the number of removed features, to find the optimal set of features. The combined complexity of the BFS and regularization parameter optimization is thus $O(RKN^3N_r)$. This results in a training algorithm with overall computational complexity $O(N^2M + RKN^3N_r)$.

3.2 Forward feature selection

When a feature is removed, the n-th row and column of the inverse of the reduced covariance matrix $\mathbf{A}_r^{-1}$, are all zero, with n the index of the removed feature. When features are added we can thus start from the matrix $\mathbf{A}_r$. If we want to add a feature to this matrix we need to compute $\mathbf{A}_e = \mathbf{A}_r + \mathbf{A}_a$, where $\mathbf{A}_a$ is a rank 2 matrix that contains all zeros except for the elements missing in $\mathbf{A}_r$ to create the covariance matrix with the added feature $\mathbf{A}_e$. Because $\mathbf{A}_a$ is symmetric and has rank 2 it can be decomposed in $\mathbf{A}_a = \mathbf{U}\mathbf{R}\mathbf{U}^T$, with $\mathbf{U}$ of dimension (N_s+1)x2 and $\mathbf{R}$ of dimension 2x2, with N_s the number of already selected features. If the last row and column of $\mathbf{A}_r$ corresponds to the missing features, the elements of $\mathbf{U}$ and $\mathbf{R}$ can be easily determined as follows:

$$\mathbf{R} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$
$$\mathbf{U}^T = \begin{pmatrix} \mathbf{0} & 1 \\ \mathbf{A}_{(n,1:N_s)} & \frac{1}{2}\mathbf{A}_{(n,n)} \end{pmatrix},$$

with $\mathbf{A}_{(n,n)}$ the n'th diagonal element of $\mathbf{A}$ and $\mathbf{A}_{(n,1:N_s)}$ the elements corresponding to the already selected features on the n'th row of $\mathbf{A}$. Using the Sherman-Morrison-Woodbury formula [22] we can now determine the inverse of $\mathbf{A}_e$ as follows:

$$\begin{aligned}\mathbf{A}_e^{-1} &= (\mathbf{A}_r + \mathbf{U}\mathbf{R}\mathbf{U}^T)^{-1} \\ &= \mathbf{A}_r^{-1} - \mathbf{A}_r^{-1}\mathbf{U}(\mathbf{R}^{-1} + \mathbf{U}^T\mathbf{A}_r^{-1}\mathbf{U})^{-1}\mathbf{U}^T\mathbf{A}_r^{-1} \\ &= \mathbf{C}(\mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{C}^T\mathbf{U}(\mathbf{R}^{-1} + \mathbf{U}^T\mathbf{C}\mathbf{D}^{-1}\mathbf{C}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{C}\mathbf{D}^{-1})\mathbf{C}^T \\ &= \mathbf{C}(\mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{C}_U^T(\mathbf{R}^{-1} + \mathbf{C}_U\mathbf{D}^{-1}\mathbf{C}_U^T)^{-1}\mathbf{C}_U\mathbf{D}^{-1})\mathbf{C}^T,\end{aligned}$$

with the eigenvalue decomposition of $\mathbf{A}_r = \mathbf{C}\mathbf{D}\mathbf{C}^T$ and $\mathbf{C}_U = \mathbf{U}^T\mathbf{C}$. If we introduce this in equations (6) and (7) we find that the output weights with an added feature become:

$$\mathbf{W}_{Cte} = \mathbf{W}_{Ct} - (\mathbf{D} + \lambda\mathbf{I})^{-1}\mathbf{C}_U^T(\mathbf{R}^{-1} + \mathbf{C}_U(\mathbf{D} + \lambda\mathbf{I})^{-1}\mathbf{C}_U^T)^{-1}\mathbf{C}_U\mathbf{W}_{Ct}.$$

$\mathbf{D}$ is a diagonal matrix, $\mathbf{W}_{Ct}$ has dimension Nx1, $\mathbf{C}_U$ has dimension Nx2 and $\mathbf{R}$ has dimension 2x2. Hence, this expression, when evaluated in the right order, has complexity $O(2N_s) = O(N_s)$, with N_s the number of selected features. Thus testing the addition of 1 feature or testing 1 regularization parameter has complexity $O(N_s^2)$. The eigenvalue decomposition for each validation set has complexity $O(KN_s^3)$. After that, one needs to test the addition of, at most, N features and R regularization parameters for each validation set, which is of the order $O(RKNN_s^2)$. To find the optimal set of features, this process needs to be repeated $N_s + 1$ times. Finding the optimal features and regularization parameter using FFS is thus of the order $O(N^2M + KN_s^4 + RKNN_s^3) = O(N^2M + RKNN_s^3)$.

4 Class reweighted ridge regression

CRRR was introduced by Toh in [10] to achieve better classification results using a reweighted form of regression. As opposed to equation 1, CRRR minimizes the following loss function:

$$f_{loss} = ||\mathbf{R}_t(\mathbf{X}_t\mathbf{W}_t - \mathbf{Y}_t)||^2 + \lambda||\mathbf{W}_t||^2,$$

with $\mathbf{R}_t$ a diagonal matrix that reweights all positive examples by a factor $\frac{1}{\sqrt{n_{t,pos}}}$ and all negative samples to $\frac{1}{\sqrt{n_{t,neg}}}$ with $n_{t,pos}$ and $n_{t,neg}$ the number of samples in the training set in the positive and negative class, respectively.

To use CRRR in combination with the proposed algorithm it suffices to calculate covariance matrices, $\mathbf{X}^T\mathbf{X}$, $\mathbf{X_v}^T\mathbf{X_v}$, $\mathbf{X}^T\mathbf{Y}$, $\mathbf{X_v}^T\mathbf{Y_v}$ and $\mathbf{Y_v}^T\mathbf{Y_v}$, independently for each class. The positive and negative covariance matrices can be combined using the following formula:

$$\mathbf{A} = \frac{1}{n_{pos}}\mathbf{A}^+ + \frac{1}{n_{neg}}\mathbf{A}^-,$$

with $\mathbf{A}^+$ and $\mathbf{A}^-$ respectively the positive and negative covariance matrix. After calculating the covariance matrices the rest of the previously described algorithms can be executed.

5 Experiments

To show the gain in computational efficiency, we test the previously described algorithms on a 2.6 GHz Intel Core 2 Quad with 8 GB of RAM. The performance is evaluated on four different machine learning techniques and several tasks from literature.

For RR without feature selection we compare the naive implementation, the implementation that makes use of the covariance matrices and the implementation that uses the eigen-decomposition. To evaluate the combination of RR and feature selection we tested the covariance method and the eigen-decomposition combined with the Sherman-Morisson technique for both FFS and BFS. The naive implementation was not tested for feature selection since its computation time was more than 100 days for the classification tasks. Since L1 regularized regression is the default technique for sparse regression in literature, we compare our technique with the least angle regression (LARS) algorithm [5], a fast implementation of the LASSO paradigm[2]. We also show the performance of optimally-pruned linear regression (OP-LR) [9], a sparse regression technique that uses LARS for feature ranking, typically used in OP-ELMs. For completeness we also implemented an L2-regularised version of OP-LR using our algorithms, which we call optimally-pruned RR (OP-RR). For each of the tasks and algorithms the fixed computational cost of the algorithm, the added time to optimize the parameters and feature set, the condition number of the covariance matrix, the percentage of removed input features and the test error are given.

5.1 Reservoir computing applied to epileptic seizure detection

Feed-forward neural networks are known for their generalization properties and ability to learn complex relationships between input features and outputs with limited training. Recurrent neural networks add recurrent connections and thus time information to these networks, resulting in a dynamical system that can find relationships between the desired output and the input history. Traditionally all interconnection weights between the neurons in an recurrent neural network are trained. In contrast, RC makes use of a randomly created neural network, called a reservoir, from which only a single linear output is trained, often using RR.

[2] We used the scikits.learn toolbox available from http://scikit-learn.sourceforge.net

Table 1 Results using RC applied to epileptic seizure detection. The fixed computational cost, optimization time, percentage of removed features, error and variance on the error are given using leave-one-time-series-out or in this case 23-fold cross-validation and 60 values for the regularization parameters. The training set consisted of 345 000 samples and the reservoir consisted of 200 neurons. The condition number of the covariance matrix is $8.3 \cdot 10^{11}$.

RC	RR			FFS		BFS		LARS	OP	
	naive	cov.	eig.	cov.	eig.	cov.	eig.		LR	RR
t_{fixed}		**20s**		**20s**		**20s**		2m	**20s**	
t_{opt}	10h	13s	**1.0s**	5m	4m	1.2d	6m	5m	3m	4m
% rm	0	0	0	**98**	**98**	36	36	**98**	3	45
e (%)	4.8	4.8	4.8	4.7	4.7	**4.2**	**4.2**	4.9	4.3	5.1
std	3.3	3.3	3.3	3.4	3.4	**2.9**	**2.9**	3.9	3.0	3.8

That way the long training time and stability issues of regular recurrent neural networks are avoided without losing the desired generalization abilities. For more information on RC we refer to literature [11].

To show the time gain achieved with the proposed training technique and the performance of feature selection in combination with RC, we tested it on an epileptic seizure detection task using the detection system presented in [14]. It uses one wavelet feature extracted from EEG as input for a 200 neuron reservoir. The training set consists of 23 time-series of 5 minutes or 15 000 samples from 23 different subjects. To improve classification results we use CRRR to train the reservoir. Table 1 shows the test error given as balanced error rate (BER)[3] with the standard deviation over the 23 subjects in the test set. The optimization times were achieved using leave-one-time-series-out or in this case 23-fold cross-validation for 60 values of the regularization parameter. Because of the high correlation between subsequent data points, leave-one-time-series-out is the most suited way to perform leave-one-out cross-validation for time-series [3].

5.2 Windowed ridge regression applied to epileptic seizure detection

To apply RR on time-series windowed RR is often used. The input vector, which typically contains the current input samples, is then extended with a windowed representation of the past input samples. To test windowed RR in combination with the presented algorithms we apply it to the same epileptic seizure detection dataset as used in the previous section. Instead of 1 wavelet feature we now extract 4 different wavelet features from the EEG. Each of these features resulted in 3000 samples per validation set. As input we use the last 5 seconds, or 50 samples, for each of these 4 features. The results for leave-one-time-series-out cross-validation and a set of 60 regularization parameter values are given in Table 2. Since 4 features are used instead of 1, the test error is not comparable with the previous experiment.

[3] The average between the error on the sensitivity and the specificity

Table 2 Results using windowed RR applied to epileptic seizure detection. The fixed computational cost, optimization time, percentage of removed features, error and variance on the error are given using leave-one-time-series-out or in this 23-fold cross-validation and 60 values for the regularization parameters. The training set consisted of 69 000 samples and contained 200 feature vectors. The condition number of the covariance matrix is $6.9 \cdot 10^4$.

windowed	RR			FFS		BFS		LARS	OP	
RR	naive	cov.	eig.	cov.	eig.	cov.	eig.		LR	RR
t_{fixed}		**3.9s**		**3.9s**		**3.9s**		6s	**3.9s**	
t_{opt}	2h	13s	**1.0s**	1.4h	41m	1.8d	6m	4s	1.5m	1.8m
% rm	0	0	0	67	67	56	56	74	**80**	**80**
e (%)	10.1	10.1	10.1	8.3	8.3	8.2	8.2	**5.1**	8.8	7.7
std	7.1	7.1	7.1	4.2	4.2	3.3	3.3	**2.3**	5.8	4.7

5.3 Fixed-size least-squares support vector machines applied to the Adult dataset

In contrast to the more common SVM, the LS-SVM uses a quadratic loss-function instead of the hinge loss and is conceptually easier, as training is reduced to solving a set of linear equations [12]. A disadvantage of this approach is that LS-SVMs do not result in sparse solutions. Especially with large datasets the computational demands of training LS-SVMs can be high. To cope with large datasets FS-LS-SVMs were introduced in [15]. Instead of choosing all the data points as support vectors, only the N data points that result in the largest Renyi entropy are selected. Using these data points a Gramm-matrix $\tilde{\mathbf{K}}$ is constructed and a Nyström approximation is made using the eigendecomposition of $\tilde{\mathbf{K}}$:

$$\tilde{\mathbf{K}} = \mathbf{C}\mathbf{D}\mathbf{C}^T$$

This allows the creation of a feature vector $\mathbf{x}$ in primal space for a given data point $\mathbf{u}$ using the following formula:

$$\mathbf{x}_{(i)} = \sqrt{N} \sum_{j=1}^{N} \frac{\mathbf{C}_{(i,j)}}{\mathbf{D}_{(j,j)}} k(\mathbf{u}, \tilde{\mathbf{U}}_{(j,:)}),$$

where $x_{(I)}$ is feature i of data point $\mathbf{x}$ and $k(\mathbf{u}, \tilde{\mathbf{U}}_{(j,:)})$ is the kernel function of the input vector $\mathbf{u}$ and the j-th selected data point. This formula can be used on the entire training set $\mathbf{U}$ to generate the input data $\mathbf{X}$. For more details on this process we refer to [12] and [19].

We apply the FS-LS-SVM approach to the UCI Adult dataset as described in [15] and the heat exchanger task as described in [16]. The first task is a classification task and the second is a regression task. The LS-SVM-LAB Matlab toolbox[4] was used to determine the optimal subset and to calculate the features in primal space, our algorithms were used for training. The results of this experiment using 10-fold cross-validation and a set of 50 possible values for the regularization parameter are given in Tables 3 and 4, respectively.

[4] Available at http://www.esat.kuleuven.be/sista/lssvmlab/toolbox.html

Table 3 Results FS-LS-SVM's applied to the UCI Adult dataset. The fixed computational cost, optimization time, percentage of removed features and misclassification error are given using 10-fold cross-validation and 50 values for the regularization parameters. The training set consisted of 16 281 samples and contained 400 feature vectors. The condition number of the covariance matrix is $9.1 \cdot 10^4$.

FS-LS	RR			FFS		BFS		LARS	OP	
SVM	naive	cov.	eig.	cov.	eig.	cov.	eig.		LR	RR
t_{fixed}		**1.4s**		**1.4s**		**1.4s**		1.6s	**1.4s**	
t_{opt}	11m	15s	**2s**	23h	1.3h	4.5d	16m	18s	4m	4m
% rm	0	0	0	44	44	47	47	**87**	30	11
e (%)	**15.0**	**15.0**	**15.0**	**15.0**	**15.0**	**15.0**	**15.0**	16.3	28.5	**15.0**

Table 4 Results FS-LS-SVM's applied on the heat exchanger task. The fixed computational cost, optimization time, percentage of removed features and normalized root mean squared error multiplied by 10 are given using 10-fold cross-validation and 50 values for the regularization parameters. The training set consisted of 3800 samples and contained 100 feature vectors. The condition number of the covariance matrix is $1.7 \cdot 10^8$.

FS-LS	RR			FFS		BFS		LARS	OP	
SVM	naive	cov.	eig.	cov.	eig.	cov.	eig.		LR	RR
t_{fixed}		**0.008s**		**0.008s**		**0.008s**		0.009s	**0.008s**	
t_{opt}	2m	0.49s	**0.046s**	5m	29s	28m	8.6s	0.9s	1.1s	2s
% rm	0	0	0	96	96	53	53	**97**	29	15
e×10	**6.20**	**6.20**	**6.20**	8.15	8.15	**6.20**	**6.20**	12.7	9.32	6.87

5.4 Extreme learning machine applied to regression datasets

Guang-Bin Huang et al. proposed in [13] a fast algorithm to train the weights of the hidden neurons in feed-forward neural networks called ELMs. The weights from the input to the hidden layer are randomly initialised, and similarly to RC, only the weights from the hidden layer to the output are trained using regression. In practise an ELM performs a non-linear mapping ψ of the input vector $\mathbf{u}$ to result in the following formulation for the feature vector:

$$\mathbf{x} = \psi(\mathbf{u})$$

In [9] an extension to this algorithm was proposed called OP-ELMs which added pruning of the hidden neurons to the ELM approach. The pruning algorithm, in this work referred to as OP-LR, first ranks all the hidden neurons using LARS. The best combination of features is then selected using cross-validation. For more details on ELMs and OP-ELMs we refer to literature [13] and [9].

OP-ELMs have been shown to render good results on several regression datasets[5] in [9]. We used the OP-ELM toolbox[6] to generate an ELM with 100 hidden neurons. Our implementations were used to train the linear readout of the network and evaluate the error. To improve performance direct connections from the input to the output were added to the processed data. The results of these experiments using 20-fold cross-validation and a set of 80 possible regularization parameters are shown in Tables 5 and 6. Since the regression datasets are rather small compared

[5] Available at http://www.liaad.up.pt/∼ltorgo/Regression/DataSets.html
[6] Available at http://research.ics.tkk.fi/eiml/software.shtml

Table 5 The normalized root mean squared error multiplied by 10, the percentage or removed input features and the average optimization time for the ELM regression tasks.

ELM		RR			FFS		BFS		LARS	OP	
		naive	cov.	eig.	cov.	eig.	cov.	eig.		LR	RR
Abal	e×10	6.77	6.77	6.77	6.72	6.72	**6.65**	**6.65**	7.80	6.74	6.76
	% rm	0	0	0	80	80	73	73	**92**	39	52
Aile	e×10	4.32	4.32	4.32	4.30	4.30	**4.16**	**4.16**	8.42	4.24	4.37
	% rm	0	0	0	92	92	30	30	**95**	37	2
Bank	e×10	6.74	6.74	6.74	6.79	6.79	**6.73**	**6.73**	6.76	6.76	6.76
	% rm	0	0	0	**93**	**93**	66	66	72	79	79
Canc	e×10	5.11	5.11	5.11	**4.09**	**4.09**	4.76	4.76	5.25	5.76	5.76
	% rm	0	0	0	**94**	**94**	81	81	92	**94**	**94**
Elev	e×10	**3.89**	**3.89**	**3.89**	4.18	4.18	**3.89**	**3.89**	4.59	3.97	**3.89**
	% rm	0	0	0	71	71	38	38	**72**	18	1
Pric	e×10	4.42	4.42	4.42	**2.98**	**2.98**	3.86	3.86	189	4.60	4.60
	% rm	0	0	0	**98**	**98**	77	77	4	76	76
Serv	e×10	5.13	5.13	5.13	**4.87**	**4.87**	4.91	4.91	9.91	5.65	5.13
	% rm	0	0	0	92	92	78	78	**94**	88	2
Average t_{opt}		7m	0.3s	**0.1s**	5m	4m	4h	1.3m	0.7s	5s	17s

Table 6 The fixed computational cost, condition number of the covariance matrix, number of features and the number of samples in the trainings set for the ELM regression tasks.

ELM	t_{fixed}		Cond.	# feats	# samp
	RR	LARS	nr.	(N)	(M)
Abal	**0.016s**	0.14s	9.0e11	108	2785
Aile	**0.055s**	0.52s	6.5e12	131	7154
Bank	**0.028s**	0.29s	2.3e11	132	4500
Canc	**0.0062s**	0.060s	1.4e13	117	129
Elev	**0.038s**	0.47s	1.4e11	115	8752
Pric	**0.0060s**	0.054s	1.4e18	105	106
Serv	**0.0061s**	0.035s	1.9e19	103	111

to the classification datasets and because of the high similarity in optimization time, only average times are given in Table 5.

6 Discussion

Because the big-O notation does not always provide a good estimation for the real computational cost of an algorithm, we did several experiments on datasets from literature. From Tables 1 to 5, we can conclude that the algorithms presented in this paper are significantly faster than the naive implementation and the covariance method. For the regularization parameter optimization, FFS and BFS, computation times were improved by one order of magnitude for the small datasets and up to 4 orders of magnitude for the large datasets. Because of this reduction in computational cost, feature selection can be executed within a reasonable time frame for these datasets. Even though for the FS-LS-SVM classification task, the feature selection procedure required more than 1 hour, this is relatively short compared to the 14 hours needed to find the subset of 400 data points with the highest Renyi entropy. From the regularization parameter optimization in the RC and windowed RR tasks we can also deduct that in this case the calculation time is independent of the size of the dataset but scales proportionately to the number of features.

The results also show that there is no difference between the task performances achieved by the ‘naive’, ‘covariance’ and ‘eigen’ implementations even for covariance matrices with a high condition number. This is because the same regularization parameter is chosen by all methodologies and the same features are removed. However from Tables 5 and 6 we can conclude that the LARS algorithm underperforms when the covariance matrix has a large condition number.

As shown in Tables 1 to 5, the combination of feature selection and RR can lead to better test results. The error is at least equal or better than that of LARS, OP-LR and OP-RR on most tasks. Usually BFS performs better than FFS. This is due to the fact that FFS algorithms have difficulty seeing the constructive correlation between separate features [4]. As a side effect, FFS often yields sparser solutions. For the FS-LS-SVMs tasks and the Elevator task there is no performance increase nor decrease using BFS. Tables 3 and 5 show, however, that more than 40% of the features are removed, which results in a much faster execution time on the test data. For the seizure detection task using windowed RR we noticed that the presented algorithms were outperformed by the LARS algorithm, even though they both removed all time windows of 2 irrelevant wavelet filters. This shows that L1-regularization is better suited for this task.

Most cross-validation algorithms discussed in literature [2,3,7,8] require that the full Gram matrix[7] fits in the system memory. This matrix has a dimension MxM which rapidly becomes too large for large datasets. The presented algorithms on the other hand limit the memory use to $O(N^2)$ variables, which is independent of the size of the dataset.

7 Conclusion

Calculating the output weights using RR has a computational complexity of $O(N^2M)$ with N the number of input features and M the number of data samples in the training set. To optimize the regularization parameter using cross-validation, the algorithm presented in this paper adds a computational cost of complexity $O(KN^3)$, with K equal to the number of validation sets. The order is independent of the number of regularization parameter values R if $R < N$. For large datasets, such that $KN < M$, we can conclude that the additional cost of our algorithm is smaller than the cost of the original RR algorithm. This is in strong contrast with the, to our knowledge, most commonly used naive implementation of optimizing the regularization parameter which has a computational cost of $O(RK^2N^2M)$.

RR can be combined with a FFS of BFS algorithm to result in a sparse solution. Optimizing the optimal set of features and selecting the optimal regularization parameter results, in a naive implementation, in an algorithm of complexity $O(RKNN_s^3M)$ and $O(RKN^3N_rM)$ for respectively FFS and BFS. The algorithms we propose in this paper have a computational complexity of $O(RKNN_s^3)$ and $O(RKN^3N_r)$, which is M times faster than the naive implementation and thus independent of the size of the training set.

[7] In our notation the Gram matrix is calculated as follows: $\mathbf{XX}^T$.

To show the computational cost of the proposed algorithms and the performance of ridge regression in combination with feature selection, we applied them to different tasks and machine learning techniques from literature. We applied RC and windowed RR to an epileptic seizure detection task, FS-LS-SVMs to the UCI Adult dataset and Heat Exchanger task, and ELMs to several regression datasets from literature. Results show that feature selection results a lower error or at least a reduction of the number of features needed to achieve the same performance. BFS often outperforms FFS and even Lasso on all but one task. However, FFS results in sparser output weights. The regularization parameter optimization algorithm and the feature selection algorithms reduced the computational cost by one order of magnitude for small datasets and over 4 orders of magnitude for larger datasets, compared to naive implementations.

Acknowledgement

The work presented in this paper is funded by a Ph.D. grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen), a Ph.D. fellowship of the Research Foundation - Flanders (FWO), the EC FP7 project ORGANIC (FP7-231267), the BOF-GOA project Home-MATE funded by the Ghent university Special Research Fund, the Interuniversity Attraction Poles Program (Belgian Science Policy) project Photonics@be IAP6/10 and the FWO project RECAP.

References

1. A.N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-Posed Problems.* Winston and Sons, 1977.
2. G. Cawley and N. Talbot. Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural Networks*, 17:1467–1475, 2004.
3. T. Pahikkala, J. Boberg, and T. Salakoski. Fast n-fold cross-validation for regularized least-squares. In *Proceedings of the Ninth Scandinavian Conference on Artificial Intelligence (SCAI)*, 2006.
4. I. Guyon and A. Elisseeff. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.
5. B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32:407–451, 2004.
6. X. Dutoit, B. Schrauwen, J. Van Campenhout, D. Stroobandt, H. Van Brussel, and M. Nuttin. Pruning and regularization in reservoir computing. *Neurocomputing*, 72:1534–1546, 2009.
7. F. Ojeda, J. Suykens, and B. De Moor. Low rank updated LS-SVM classifiers for fast variable selection. *Neural Networks*, 21:437–449, 2008.
8. T. Pahikkala, A. Airola, and T. Salakoski. Feature selection for regularized least-squares: new computational short-cuts and fast algorithmic implementations. In *IEEE International Workshop on Machine Learning for Signal Processing*, 2010.
9. Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse. Op-elm: Optimally pruned extreme learning machine. *IEEE Transactions on Neural Networks*, 21:158–162, 2010.
10. K. A. Toh. Deterministic Neural Classification. *Neural Computation*, 20:1565–1595, 2008.
11. M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.
12. J. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines.* World Scientific Publishing, 2002.

13. G. Huang, Q. Zhu, and C. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70:489–501, 2006.
14. P. Buteneers, D. Verstraeten, P. van Mierlo, T. Wyckhuys, D. Stroobandt, R. Raedt, H. Hallez, and B. Schrauwen. Automatic detection of epileptic seizures on the intracranial electroencephalogram of rats using reservoir computing. *Artificial Intelligence in Medicine*, 2011. (in press).
15. L. Hoegaerts, J. Suykens, J. Vandewalle, and B. De Moor. Primal space sparse kernel partial least squares regression for large scale problems. In *IEEE International Joint Conference on Neural Networks*, 2004.
16. M. Espinoza, J.A.K. Suykens, and B. De Moor. Least squares support vector machines and primal space estimation. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 4, pages 3451–3456. IEEE, 2003.
17. D. Verstraeten, B. Schrauwen, S. Dieleman, P. Brakel, P. Buteneers, and D. Pecevski. Oger: Modular learning architectures for large-scale sequential processing. 2011. (in press).
18. W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in FORTRAN: The Art of Scientific Computing, Second Edition.* Cambridge University Press, 1992.
19. K. De Brabanter, J. De Brabanter, J. Suykens, and B. De Moor. Optimized fixed-size kernel models for large data sets. *Computational Statistics & Data Analysis*, 54:1484–1504, 2010.
20. K. Pelckmans, J. De Brabanter, J.A.K. Suykens, and B. De Moor. The differogram: Non-parametric noise variance estimation and its use for model selection. *Neurocomputing*, 69(1):100–122, 2005.
21. K. Pelckmans, J.A.K. Suykens, and B. De Moor. Additive regularization trade-off: Fusion of training and validation levels in kernel methods. *Machine Learning*, 62(3):217–252, 2006.
22. G. Golub and C. Van Loan. *Matrix computations.* The Jonhs Hopkins University Press, 1989.
23. P. Holland. Weighted Ridge Regression: Combining Ridge and Robust Regression Methods. *Technical Report 0011, National Bureau of Economic Research, Inc.*, 1973.
24. D. Allen. The Relationship Between Variable Selection and Data Augmentation and Slow Feature Analysis. *Technometrics*, 16, 1974.
25. J. Sherman and W. J. Morisson. Adjustments of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix. *The Annals of Mathematical Statistics*, 21:124–127, 1950.