Manuscript Number: 3262

Fast Generalized Cross Validation Algorithm For Sparse Model Learning

S Sundararajan
Philips Electronics India Ltd
1 Murphy Road, Ulsoor
Bangalore, India

Shirish Shevade
Computer Science and Automation
Indian Institute of Science
Bangalore, India

S. Sathiya Keerthi
Yahoo! Research Labs
210 S.DeLacey Avenue
Pasadena, CA-91105, USA

Abstract

In this paper we propose a fast incremental algorithm for designing linear regression models. The proposed algorithm generates a sparse model by optimizing multiple smoothing parameters using the generalized cross validation approach. The performances on synthetic and real-world datasets are compared with other incremental algorithms such as Tipping and Faul's fast Relevance Vector Machine, Chen et al.'s Orthogonal Least Squares and Orr's Regularized Forward Selection. The results demonstrate that the proposed algorithm is competitive.

1 Introduction

In recent years, there has been a lot of focus on designing sparse models in machine learning. For example, the support vector machine (SVM) (Cortes and Vapnik, 1995) and the relevance vector machine (RVM) (Tipping, 2001; Tipping and Faul, 2003) have been proven to provide sparse solutions to both regression and classification problems. Some of the earlier successful approaches for regression problems include Chen et al.'s orthogonal least squares and Orr's Regularized forward selection algorithms (Chen, Cowan, and Grant, 1991; Orr, 1995b).

In this paper, we consider only the regression problem. Often, the target function model takes the form:

$$\mathbf{y}(\mathbf{x}) = \sum_{m=1}^{M} w_m \phi_m(\mathbf{x}) \tag{1}$$

For notational convenience, we do not indicate the dependency of $\mathbf{y}$ on $\mathbf{w}$, the weight vector. The available choices in selecting the set of 'basis vectors' $\{\boldsymbol{\phi}_m(\mathbf{x}), m = 1, ..., M\}$ make the model flexible. Given a dataset $\{\mathbf{x}_n, t_n\}_{n=1}^{N}, t_n \in R \; \forall \; n$, we can write the target vector $\mathbf{t} = (t_1, ..., t_N)^T$ as the sum of the approximation vector $\mathbf{y}(\mathbf{x}) = (y(\mathbf{x}_1), ..., y(\mathbf{x}_N))^T$ and an error vector $\boldsymbol{\eta}$:

$$\mathbf{t} = \boldsymbol{\Phi}\mathbf{w} + \boldsymbol{\eta} \tag{2}$$

where $\boldsymbol{\Phi} = (\, \boldsymbol{\phi}_1 \boldsymbol{\phi}_2 \cdots \boldsymbol{\phi}_M \,)$ is the design matrix and $\boldsymbol{\phi}_i$ is the i^{th} column vector of $\boldsymbol{\Phi}$ of size $N \times 1$ representing the response of the i^{th} basis vector for all the input samples $\mathbf{x}_j, \; j = 1, \ldots, N$. One possible way to obtain this design matrix is to use a Gaussian kernel with $\boldsymbol{\Phi}_{ij} = \exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2z^2})$. The error vector $\boldsymbol{\eta}$ can be modeled as an independent zero mean Gaussian vector with variance σ^2.

In this context, controlling the model complexity in avoiding over-fitting is an important task. This problem has been addressed in the past by regularization approaches (Bishop, 1995). In the classical approach, the sum squared error with weight decay regularization having a single regularization parameter α controls the trade-off between fitting the training data and smoothing the output function. In the local smoothing approach, each weight in $\mathbf{w}$ is associated with a regularization or ridge parameter (Hoerl and Kennard, 1970; Orr, 1995a; Denison and George, 2000). An interested reader can refer to (Denison and George, 2000) for a nice discussion on the generalized ridge regression and Bayesian approach.

For our discussion, we consider the weight decay regularizer with multiple regularization parameters. In this case, the optimal weight vector is obtained by minimizing the following cost function (for a given set of hyperparameter values, $\boldsymbol{\alpha}, \sigma^2$):

$$C(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2) \;=\; \frac{1}{2\sigma^2}||\mathbf{t} - \boldsymbol{\Phi}\mathbf{w}||^2 \;+\; \frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w} \tag{3}$$

where $\mathbf{A}$ is a diagonal matrix with elements $\boldsymbol{\alpha} \;=\; (\alpha_1, ..., \alpha_M)^T$, and the optimal weight vector is given by

$$\mathbf{w} \;=\; \frac{1}{\sigma^2}\mathbf{S}^{-1}\boldsymbol{\Phi}^T\mathbf{y} \tag{4}$$

where $\mathbf{S} = \mathbf{R} + \mathbf{A}$ and $\mathbf{R} = \frac{\boldsymbol{\Phi}^T\boldsymbol{\Phi}}{\sigma^2}$. Note that this solution depends on the product $\boldsymbol{\alpha}\sigma^2$. This solution is same as the maximum aposteriori (MAP) solution obtainable from defining the Gaussian likelihood and Gaussian prior for the weights in a Bayesian framework (Tipping, 2001).

The hyperparameters are typically selected using iterative approaches like marginal likelihood maximization (Tipping, 2001). In practice, many of the α_i approach ∞. This results in the removal of associated basis vectors, thereby making the model sparse. The final model consists of a small number of basis vectors L ($L \ll M$), called *relevance vectors* and hence, is known by the name Relevance Vector Machine (RVM). This procedure is computationally intensive and needs $O(M^3)$ effort at least for the initial iterations while starting with the full model ($M = N$ or $M = N + 1$ if the bias term is included). Hence, it is not suitable for large datasets. This limitation was addressed in (Tipping and Faul, 2003), where a computationally efficient algorithm was proposed. In this algorithm, basis vectors are added sequentially starting from an empty model. It also allows to delete the basis vectors which may subsequently become redundant.

There are various other basis vector selection heuristics which can be used to design sparse models (Chen et al., 1991; Orr, 1995b). Chen et al. (1991) proposed an orthogonal least squares algorithm as a forward regression procedure to select the basis vectors. At each step of the regression, the increment to the explained variance of the desired output is maximized. Orr (1995b) proposed an algorithm which combines regularization and cross-validated selection of basis vectors. Some other promising incremental approaches are the algorithms of Csato and Opper (2002), Lawrence, Seeger, and Herbrich (2003), Seeger, Williams, and Lawrence (2003) and Smola and Bartlett (2000). But they apply to Gaussian processes and are not directly related to the problem formulation addressed in this paper.

Generalized Cross Validation (GCV) is another important approach for the selection of hyperparameters and has been shown to exhibit good generalization performance (Sundararajan and Keerthi, 2001; Orr, 1995a). Orr (1995a) used the GCV approach to estimate the multiple smoothing parameters of the full model. This approach, however, is not suitable for large datasets due to its computational complexity. Therefore, there is a need to devise a computationally efficient algorithm based on GCV approach to handle large datasets.

In this paper, we propose a new fast incremental GCV algorithm which can be used to design sparse models exhibiting good generalization performance. In the algorithm, we start with an empty model and sequentially add the basis functions to reduce the GCV error. The GCV error can also be reduced by deleting those basis functions which subsequently become redundant. This important feature offsets the inherent greediness exhibited by other sequential algorithms. This algorithm has the same computational complexity as that of the algorithm given in (Tipping and Faul, 2003) and is suitable for large datasets as well. Preliminary results on synthetic and real-world benchmark datasets indicate that the new approach gains on generalization but at the expense of a moderate increase in the number of basis vectors.

The paper is organized as follows. In Section 2, we describe the GCV approach and compare the GCV error function with marginal likelihood function. Section 3 describes the fast incremental algorithm, computational complexity and the numerical issues involved;

the update expressions mentioned in this section are detailed in the Appendices I to VI. In Section 4, we present the simulation results. Section 5 concludes the paper.

2 Generalized Cross Validation

The standard techniques that estimate the prediction error of a given model are the leave-one-out (LOO) cross-validation (Stone, 1974) and the closely related GCV described in (Golub, Heath, and Wahba, 1979; Orr, 1995b). The generalization performance of the GCV approach is quite good, like that of the LOO cross-validation approach. The advantage in using the GCV error is that it takes a much simpler form compared to the LOO error and is given by

$$V(\boldsymbol{\alpha}, \sigma^2) = N \frac{\sum_{i=1}^{N} (t_i - y(\mathbf{x}_i))^2}{(tr(\mathbf{P}))^2} \tag{5}$$

where

$$\mathbf{P} = \mathbf{I} - \frac{\mathbf{\Phi} \mathbf{S}^{-1} \mathbf{\Phi}^T}{\sigma^2} \tag{6}$$

When this GCV error is minimized with respect to the hyperparameters, many of the α_i approach ∞, making the model sparse.

The equation (5) can be written as

$$V(\boldsymbol{\alpha}, \sigma^2) = N \frac{\mathbf{t}^T \mathbf{P}^2 \mathbf{t}}{(tr(\mathbf{P}))^2} \tag{7}$$

Note that $\mathbf{P}$ is dependent only on $\boldsymbol{\zeta} = \boldsymbol{\alpha}\sigma^2$. This means that we can get rid of σ^2 from (4) and (5). Then, it is sufficient to work directly with $\boldsymbol{\zeta}$. However, using the optimal $\boldsymbol{\zeta}$

obtained from minimizing the GCV error, the noise level can be computed from:

$$\hat{\sigma}^2 = \frac{\mathbf{t}^T \mathbf{P}^2 \mathbf{t}}{tr(\mathbf{P})} \tag{8}$$

We now discuss the algorithm, proposed by Orr, to determine the optimal set of hyperparameters.

2.1 Orr's Algorithm

Starting with the full model, Orr (1995a) proposed an iterative scheme to minimize the GCV error (7) with respect to the hyperparameters. Although this algorithm was originally described in terms of the variable ζ_j, we describe it here using the variables α_j and σ^2 for convenience. Each α_j is optimized in turn while the others are held fixed. The minimization thus proceeds by a series of one-dimensional minimizations. This can be achieved by rewriting (7) using

$$\mathbf{t}^T \mathbf{P}^2 \mathbf{t} = \frac{a_j \Delta_j^2 - 2 b_j \Delta_j + c_j}{\Delta_j^2}$$

$$tr(\mathbf{P}) = \frac{\delta_j \Delta_j - \epsilon_j}{\Delta_j}$$

where

$$a_j = \mathbf{t}^T \mathbf{P}_j^2 \mathbf{t} \tag{9}$$

$$b_j = (\mathbf{t}^T \mathbf{P}_j^2 \boldsymbol{\phi}_j)(\mathbf{t}^T \mathbf{P}_j \boldsymbol{\phi}_j) \tag{10}$$

$$c_j = (\boldsymbol{\phi}_j^T \mathbf{P}_j^2 \boldsymbol{\phi}_j)(\mathbf{t}^T \mathbf{P}_j \boldsymbol{\phi}_j)^2 \tag{11}$$

$$\delta_j = tr(\mathbf{P}_j) \tag{12}$$

$$\epsilon_j = (\boldsymbol{\phi}_j^T \mathbf{P}_j^2 \boldsymbol{\phi}_j) \tag{13}$$

The above relationships follow from the rank-one update relationship between $\mathbf{P}$ and $\mathbf{P}_j$.

$$\mathbf{P} = \mathbf{P}_j - \frac{1}{\Delta_j}\mathbf{P_j}\boldsymbol{\phi}_j\boldsymbol{\phi}_j^T\mathbf{P_j} \tag{14}$$

where $\mathbf{P}_j = \mathbf{I} - \frac{\boldsymbol{\Phi}_j \mathbf{S}_j^{-1} \boldsymbol{\Phi}_j{}^T}{\sigma^2}$ and

$$\Delta_j = \boldsymbol{\phi}_j^T \mathbf{P}_j \boldsymbol{\phi}_j + \alpha_j \sigma^2 \tag{15}$$

Here, $\boldsymbol{\Phi}_j$ denotes the matrix $\boldsymbol{\Phi}$ with the column $\boldsymbol{\phi}_j$ removed. Therefore, $\mathbf{S}_j = \mathbf{R}_j + \mathbf{A}_j$ with the subscript j having the same interpretation as in $\boldsymbol{\Phi}_j$. Note that $\mathbf{S}_j$ does not contain α_j and hence, $\mathbf{P}_j$ also does not contain α_j.

Thus, equation (7) can be seen as a rational polynomial in α_j alone, with a single minimum in the range $[0, \infty]$. The details of minimizing this polynomial with respect to α_j are given in appendix I. After one complete cycle in which each parameter is optimized once, the GCV score is calculated and compared with the score at the end of the previous cycle. If significant decrease has occurred a new cycle is begun, else the algorithm terminates. As detailed in (Orr, 1995a), the computational complexity of one cycle of the above algorithm is $O(N^3)$, at least during the first few iterations. Although, this will consequently reduce to $O(LN^2)$ as the basis vectors are pruned, this algorithm is not suitable for large datasets.

2.2 Comparison of GCV error with marginal likelihood

We now compare the GCV error and marginal likelihood by studying their dependence on α_j. In the marginal likelihood method, the optimal α_j's are determined by maximizing the marginal likelihood with respect to α_j. On the other hand, the GCV error is minimized to get the optimal α_j's.

First, we study the behaviour of the GCV error with reference to α_j. The term in the denominator of the GCV error has $tr(\mathbf{P}) = \delta_j - \frac{\epsilon_j}{\Delta_j}$ where δ_j and ϵ_j are independent of α_j and $\mathbf{P}$ is a positive semi-definite matrix. Further, $tr(\mathbf{P})$ increases monotonically as a function of α_j. Therefore, maximizing the $tr(\mathbf{P})$ (in order to minimize the GCV error) will prefer the optimal value of α_j to be ∞. Thus, the denominator term in equation (7) prefers a simple model. The term, $\mathbf{t}^T\mathbf{P}^2\mathbf{t}$, in the numerator of equation (7) is the squared error at the optimal weight vector in (4). Let $g(\boldsymbol{\alpha}) = \mathbf{t}^T\mathbf{P}^2\mathbf{t}$. Differentiating $g(\boldsymbol{\alpha})$ with respect to α_j, we get

$$\frac{\partial g(\boldsymbol{\alpha})}{\partial \alpha_j} = \frac{2\sigma^2}{\Delta_j^2}(b_j - \frac{c_j}{\Delta_j})$$

where b_j and c_j are independent of α_j and $c_j, \Delta_j \geq 0$. If b_j is non-positive then $g(\alpha_j)$ is a monotonically decreasing function of α_j. Minimization of $g(\alpha_j)$ with respect to α_j would thus prefer α_j to be ∞. On the other hand, if b_j is positive, then the minimum of $g(\alpha_j)$ would depend upon the sign of $\sigma^2 b_j \bar{s}_j - c_j$ where $\bar{s}_j \overset{\text{def}}{=} \phi_j^T \mathbf{\Sigma}_{-j}^{-1}\phi_j$, $\mathbf{\Sigma}_{-j}$ is $\mathbf{\Sigma}$ with the contribution of basis vector j removed and $\mathbf{\Sigma} = \sigma^2 I + \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}$. Note that $\mathbf{P} = \sigma^2\mathbf{\Sigma}^{-1}$. Therefore, the optimal choice of α_j using the GCV error method depends on the trade-off between the data dependent term in the numerator and the term in the denominator that prefers α_j to be ∞.

The logarithm of marginal likelihood function is

$$\mathcal{L}(\alpha) = -\frac{1}{2}\left[N\log(2\pi) + \log|\mathbf{\Sigma}| + \mathbf{t}^T\mathbf{\Sigma}^{-1}\mathbf{t}\right].$$

Considering the dependence of $\mathcal{L}(\alpha)$ on a single hyperparameter α_j, the above equation can be rewritten (Tipping and Faul, 2003) as,

$$\mathcal{L}(\alpha) = \mathcal{L}(\alpha_{-j}) + l(\alpha_j)$$

where $l(\alpha_j) = \frac{1}{2}\left[\log(\frac{\alpha_j}{\alpha_j + \bar{s}_j}) + \frac{\bar{q}_j^2}{\alpha_j + \bar{s}_j}\right]$. $\mathcal{L}(\alpha_{-j})$ is the marginal likelihood with ϕ_j excluded and is thus independent of α_j. Here, we have defined $\bar{q}_j \overset{\text{def}}{=} \phi_j^T \mathbf{\Sigma}_{-j}^{-1}\mathbf{t}$. Note that $\bar{q}_j$ and $\bar{s}_j$ are independent of α_j. The second term in $l(\alpha_j)$ comes from the data dependent term, $\mathbf{t}^T\mathbf{\Sigma}^{-1}\mathbf{t}$, in the logarithm of the marginal likelihood function and maximization of this term prefers α_j to be zero. On the other hand, the first term in $l(\alpha_j)$ comes from the "Ockham factor" (Tipping, 2001) and maximization of this term chooses α_j to be ∞. So the optimal value of α_j is a trade-off between the data dependent term and the Ockham factor.

Note that $\mathbf{t}^T\mathbf{\Sigma}^{-1}\mathbf{t} = \frac{1}{\sigma^2}\mathbf{t}^T\mathbf{P}^2\mathbf{t} + \mathbf{w}^T\mathbf{A}\mathbf{w}$. The term on the left hand side of this equation appears in the negative logarithm of marginal likelihood function while the first term on the right hand side of this equation appears in the numerator of the GCV error. The key difference in the choice of the basis function is because of the additional term that is present in the data dependent term of the marginal likelihood. For the marginal likelihood method, it has been shown that for a given basis function j, if $\bar{q}_j^2 \leq \bar{s}_j$ then the optimal value of α_j is ∞; otherwise, the optimal value is $\frac{\bar{s}_j^2}{\bar{q}_j^2 - \bar{s}_j}$ (Tipping and Faul, 2003). For the GCV method, the optimal value of α_j depends on $\bar{q}_j, \bar{s}_j$ and some other quantities as detailed in Appendix I. Further, the sufficient condition for a basis function to be not relevant for the marginal likelihood method is $\bar{q}_j^2 \leq \bar{s}_j$ and that for the GCV error method is $b_j \leq 0$. Note that b_j is independent of $\bar{s}_j$ but is dependent on $\bar{q}_j$. In general, a relevant vector obtained using marginal likelihood (or GCV error) method need not be relevant in the GCV error (or marginal likelihood) method. This fact was also observed in our experiments.

We now discuss the effect of scaling on the GCV error. First note that the GCV error is a function of $\mathbf{P}$. With the scaling of the output $\mathbf{t}$, there will be an associated scaling of basis functions. However, $\mathbf{P}$ is invariant to such scaling (see equation (6)). This will make the GCV error invariant to scaling. Also, a similar result holds for the log marginal likelihood function.

3 Fast GCV Algorithm

In this section we describe the fast GCV (FGCV) algorithm which constructs the model sequentially starting from an empty model. The basis vectors are added sequentially and their weightings are modified to get the maximum reduction in the GCV error. The GCV error can also be decreased by deleting those basis vectors which subsequently become redundant.

By maintaining the following set of variables for every basis vector, m, we can find the optimal value of α_m for every basis vector and the corresponding GCV error efficiently.

$$r_m = \mathbf{t}^T \mathbf{P} \boldsymbol{\phi}_m \tag{16}$$

$$\gamma_m = \boldsymbol{\phi}_m^T \mathbf{P} \boldsymbol{\phi}_m \tag{17}$$

$$\xi_m = \mathbf{t}^T \mathbf{P}^2 \boldsymbol{\phi}_m \tag{18}$$

$$u_m = \boldsymbol{\phi}_m^T \mathbf{P}^2 \boldsymbol{\phi}_m \tag{19}$$

In addition, we need

$$v = tr(\mathbf{P}) \tag{20}$$

$$q = \mathbf{t}^T \mathbf{P}^2 \mathbf{t} \tag{21}$$

Further, updation of these variables, after every minimization process, can be done efficiently using rank-one update given in (14).

We now give the algorithm and discuss the relevant implementation details and storage and computational requirements.

3.1 Algorithm

1. Initialize σ^2 to some reasonable value (e.g. $var(t) \times .1$).

2. Select the first basis vector ϕ_k (which forms the initial relevance vector set) and set the corresponding α_k to its optimal value. The remaining α's are notionally set to ∞.

3. Initialize $\mathbf{S}^{-1}$, $\mathbf{w}$ (which are scalars initially) and the variables given in equations (16)-(21).

4. ${\alpha_k}^{\text{old}} := \alpha_k$.

5. For all j, find the optimal solution α_j keeping the remaining $\alpha_i,\ i \neq j$ fixed and the corresponding GCV error. Select the basis vector k for which reduction in the GCV error is maximum.

6. If ${\alpha_k}^{\text{old}} < \infty$ and $\alpha_k < \infty$, the relevance vector set remains unchanged.

7. If ${\alpha_k}^{\text{old}} = \infty$ and $\alpha_k < \infty$, **add** ϕ_k to the relevance vector set.

8. If ${\alpha_k}^{\text{old}} < \infty$ and $\alpha_k = \infty$, then **delete** ϕ_k from the relevance vector set.

9. Update $\mathbf{S}^{-1}$, $\mathbf{w}$ and the variables given in equations (16)-(21).

10. Estimate the noise level using equation (8). This step may be repeated once in, for example, five iterations.

11. If there is no significant change in the values of α and σ^2, then stop. Otherwise, go to step 4.

3.2 Implementation Details

1. Since we start from the empty model, the basis vector which gives the minimum GCV error is selected as the first basis vector (step 2). The details of this procedure are given in appendix II. The first basis vector can also be selected based on the largest normalized projection onto the target vector as suggested in (Tipping and Faul, 2003).

2. The initialization of relevant variables in step 3 is described in appendix III.

3. Appendices IV and I describe the method to estimate the optimal α_i and the corresponding GCV error (step 5).

4. Updation of variables in step 9 is done using the details given in appendix V for the case in step 6 (**re-estimation**) or step 8 (**deletion**) of the algorithm. The details corresponding to step 7 (**addition**) are given in appendix VI.

5. In practice, numerical accuracies may be affected as the iterations progress. More specifically, the quantities γ_m and u_m that are expected to remain non-negative may become negative while updating the variables. When any of these two quantities becomes negative, it is a good idea to compute the quantities in equations (16)-(21) afresh using direct computations. If the problem still persists (this typically happens when the matrix $\mathbf{P}$ becomes ill-conditioned, for example, when the width parameter z used in a Gaussian kernel is large), we terminate the algorithm.

6. When the noise level is also to be estimated (step 10), all the relevant variables are calculated afresh. This computation is simplified by expanding the matrix $\mathbf{P}$ using (6).

7. In an experiment, some of the α_j may reach 0 as can be seen in appendix I. This may affect the solution. Therefore, it is useful to set such α_j to a relatively small value, α_{min}. Setting this value to $\frac{1}{N}$ was found to work well in practice.

3.3 Storage and Computational Complexity

The storage requirements of the FGCV algorithm are more than that of the fast RVM (FRVM) algorithm in (Tipping and Faul, 2003) and arise from the additional variables ξ_m and u_m to be maintained. However, they are still linear in N.

The computational requirements of the proposed algorithm are similar to those of the FRVM algorithm. Step 5 of the algorithm has computational complexity of $O(N)$. This is possible using the expressions given in appendix IV. The computational complexity of the re-estimation or deletion of a basis vector is $O(LN)$ while that of the addition of a basis vector is $O(N^2)$. Step 10 of the algorithm, however, has a computational complexity of $O(LN^2)$ as it requires the re-estimation of the relevant variables.

We observed that the FGCV algorithm was 1.25 to 1.5 times slower than the FRVM algorithm in our simulations. The main reasons for this are

1. The error function is shallow near the solution which results in more iterations.

2. Higher number of relevance vectors at the solution.

3. Updation of additional variables, ξ_m and u_m.

4 Simulations

The proposed Fast GCV algorithm is evaluated on four popular benchmark datasets and is compared with the algorithms described in (Tipping and Faul, 2003; Chen et al., 1991;

Orr, 1995b) and referred to as Fast RVM (FRVM), Orthogonal Least Squares (OLS) and Regularized Forward Selection (RFS) respectively. Two of these datasets were generated, as described by Friedman (1991) and are referred to as Friedman2 and Friedman3. The input dimension for each of these datasets was four. For these datasets, the training set consisted of 200 randomly generated examples while the test set had 1000 noise-free examples and the experiment was repeated 100 times for different training set examples. We report the normalized mean squared error (NMSE) (normalized with respect to the output variance) on the test set. The third dataset used was the Boston housing dataset obtained from the *StatLib* archive[1]. This dataset comprised of 506 examples with 13 variables. The data was split into 481/25 training/testing splits randomly and the partitioning was repeated 100 times independently. The fourth dataset used was the Abalone dataset[2]. After mapping the gender encoding (male/female/infant) into {(1,0,0),(0,1,0),(0,0,1)}, the ten-dimensional data was split into 3000/1177 training/testing splits randomly. The partitioning was repeated ten times independently. The exact partitions for the last two datasets were obtained from *http://www.gatsby.ucl.ac.uk/˜chuwei/regression.html*. For all the datasets, Gaussian kernel was used and the width parameter was chosen by using five-fold cross-validation. For the OLS and RFS algorithms, the readily available Matlab functions[3] were used with the default settings. We adhered to the guidelines provided in (Tipping and Faul, 2003) for FRVM.

The results obtained using the four algorithms (FGCV - Algorithm 1, FRVM - Algorithm 2, RFS - Algorithm 3 and OLS - Algorithm 4) on these datasets are presented in Figures 1-4. From these box plots, it is clear that the FGCV algorithm generalizes well compared to the other algorithms. However, this happens at the expense of a moderate increase in the number of basis vectors as compared to the FRVM algorithm. It is worth noting that the sparseness of the solution obtained from the FGCV algorithm is still very

[1] *http://lib.stat.cmu.edu/datasets/boston*

[2] *ftp://ftp.ics.uci.edu/pub/machine-learning-databases/abalone/*

[3] *http://www.anc.ed.ac.uk/˜mjo/software/rbf2.zip*

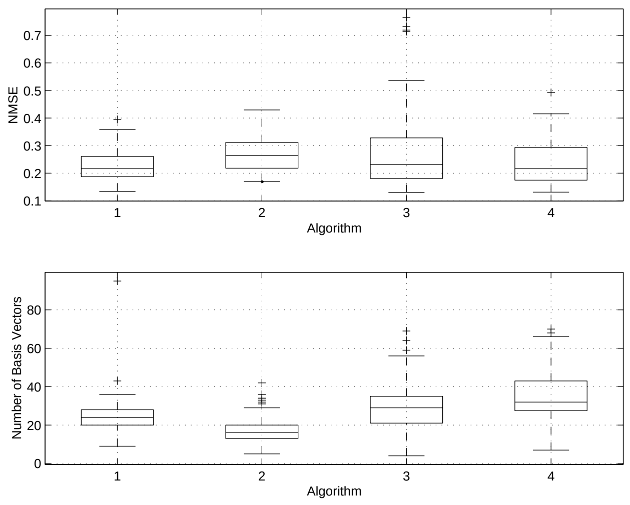

Figure 1: Results on the Friedman2 dataset

Table 1: Statistical significance (Wilcoxon signed rank) test results on the Friedman2 dataset

	FGCV	FRVM	RFS
OLS	.47	9.4e-5	.119
RFS	.013	.097	
FRVM	1.3e-12		

good. On the Abalone dataset, the FRVM algorithm is slightly better than the FGCV algorithm on the average.

Box plots in Figures 1-4 show that the distribution of MSE is non-symmetric. Therefore, we used Wilcoxon matched-pair signed rank tests to compare the four algorithms and the results are given in Tables 1-4. Each box in the table compares an algorithm in the column to an algorithm in the row. The null hypothesis is that the two medians of the test error are same, while the alternate hypothesis is that they are different. The p-value of this hypothesis test is given in the box.

We remark that the following comparisons are made with respect to the significance level of 0.05. If a p-value is smaller than 0.05, then the algorithm in the column (row)

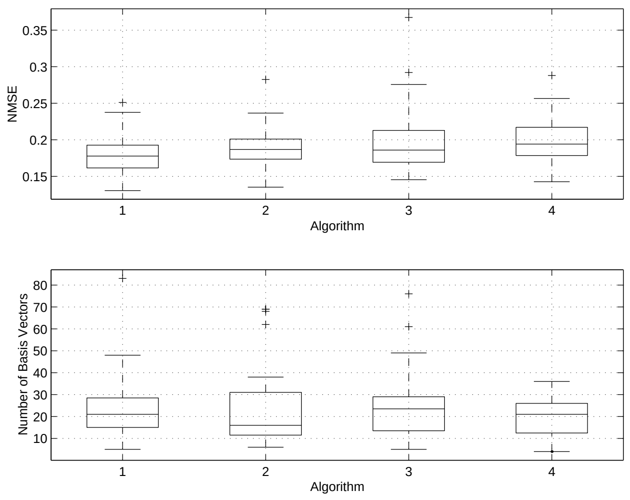

Figure 2: Results on the Friedman3 dataset

Table 2: Statistical significance (Wilcoxon signed rank) test results on the Friedman3 dataset

	FGCV	FRVM	RFS
OLS	4.6e-9	.002	.054
RFS	2.1e-6	.139	
FRVM	3.0e-6		

is statistically superior to the algorithm in the row (column). Table 1 suggests that for the Friedman2 dataset, the FGCV algorithm is statistically superior to the FRVM and RFS algorithms. On the other hand, it is not significantly different from the OLS algorithm. The FGCV algorithm is statistically superior to all the other algorithms on the Friedman3 and the Boston housing datasets as is evident from Tables 2 and 3. For the Abalone dataset, the results in Table 4 show that the performances of the FGCV and the FRVM algorithms are not significantly different. However, these algorithms are statistically superior to the OLS and RFS algorithms.

We also compared the FGCV and FRVM algorithms with the Gaussian process regression (GPR) algorithm on the Boston housing and Abalone datasets (results reported

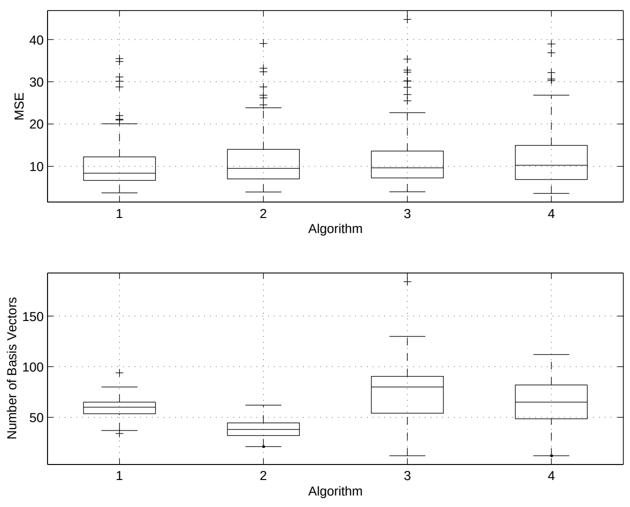

Figure 3: Results on the Boston housing dataset

Table 3: Statistical significance (Wilcoxon signed rank) test results on the Boston housing dataset

	FGCV	FRVM	RFS
OLS	2.6e-5	.096	.156
RFS	.003	.647	
FRVM	2.0e-5		

in *http://www.gatsby.ucl.ac.uk/˜chuwei/regression.html*). These comparisons were also done using Wilcoxon matched-pair signed-rank test with significance level of .05. For the Boston housing dataset, the performance comparison gave p-values of $3.4e-10$ (FGCV) and $1.22e-12$ (FRVM). This shows that the GPR algorithm (with all basis vectors) is statistically superior to the FRVM algorithm. A similar comparison on the Abalone dataset resulted in the p-values of .012 (FGCV) and .095 (FRVM). On this dataset, the GPR algorithm is statistically superior to the FGCV algorithm while it is not statistically superior to the FRVM algorithm.

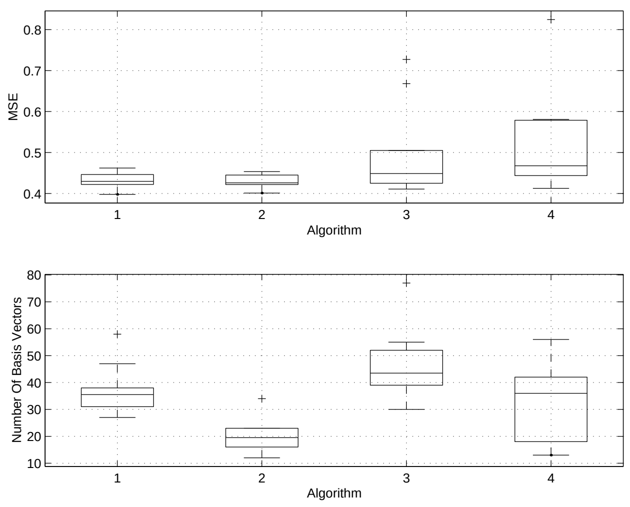

Figure 4: Results on the Abalone dataset

Table 4: Statistical significance (Wilcoxon signed rank) test results on the Abalone dataset

	FGCV	FRVM	RFS
OLS	9.8e-4	9.8e-4	.403
RFS	.023	.005	
FRVM	.462		

5 Conclusion

In this paper we proposed a fast incremental GCV algorithm for designing sparse regression models. This algorithm is very efficient and constructs the model sequentially starting from an empty model. In each iteration, it adds or deletes or re-estimates the basis vectors depending on the maximum reduction in the GCV error. The experimental results suggest that, considering the requirements of sparseness, good generalization performance and computational complexity, the FGCV algorithm is competitive. Clearly, this algorithm is an excellent alternative to the FRVM algorithm of Tipping and Faul (2003).

We mainly compared our approach against OLS, RFS and FRVM since they were quite directly related to our problem formulation. We also compared against GPR. We did not compare against the other sparse incremental GP algorithms mentioned in (Csato and Opper, 2002; Lawrence et al., 2003; Seeger et al., 2003) and (Smola and Bartlett, 2000) since we felt that those methods will be slightly inferior to GPR. But, those comparisons could be interesting, especially if we compare at the same levels of sparsity. This will be taken up in future work. It will also be interesting to extend the proposed algorithm to classification problems.

References

Bishop, C. (1995). *Neural Networks for Pattern Recognition.* Clarenden Press, Oxford.

Chen, S., Cowan, C. F. N., and Grant, P. M. (1991). Orthogonal least squares learning for radial basis function networks. *IEEE Trans on Neural Networks*, *2*, 302–309.

Cortes, C., and Vapnik, V. N. (1995). Support vector networks. *Machine Learning*, *20*, 273–297.

Csato, L., and Opper, M. (2002). Sparse on-line Gaussian processes. *Neural Computation*, *14*(3), 641–668.

Denison, D., and George, E. (2000). Bayesian prediction using adaptive ridge estimators. Tech. Report. See *http://stats.ma.ic.ac.uk/dgtd/public_html/Papers/grr.ps.*

Friedman, J. H. (1991). Multivariate adaptive regression splines. *Annals of Statistics*, *19*, 1–141.

Golub, G. H., Heath, M., and Wahba, G. (1979). Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, *21*, 215–223.

Hoerl, A. E., and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*, 55–67.

Lawrence, N., Seeger, M., and Herbrich, R. (2003). Fast sparse gaussian process methods: The informative vector machine. In *NIPS*, pp. 609–616.

Orr, M. J. L. (1995a). Local smoothing of radial basis function networks. In *Proceedings of International Symposium On Neural Networks* Hsinchu, Taiwan.

Orr, M. J. L. (1995b). Regularization in the selection of radial basis function centres. *Neural Computation*, *7*(3), 606–623.

Seeger, M., Williams, C., and Lawrence, N. D. (2003). Fast forward selection to speed up sparse Gaussian process regression. In Bishop, C. M., and Frey, B. J. (Eds.), *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics* San Francisco. Morgan Kaufmann.

Smola, A. J., and Bartlett, P. L. (2000). Sparse greedy Gaussian process regression. In *NIPS*, pp. 619–625.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions (with discussion). *Journal of Royal Statistical Society (series-B)*, *36*, 111–147.

Sundararajan, S., and Keerthi, S. S. (2001). Predictive approaches for choosing hyperparameters in Gaussian processes. *Neural Computation*, *13*(5), 1103–1118.

Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, *1*, 211–244.

Tipping, M. E., and Faul, A. (2003). Fast marginal likelihood maximisation for sparse Bayesian models. In Bishop, C. M., and Frey, B. J. (Eds.), *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics* San Francisco. Morgan Kaufmann.

Appendices

Appendix I. α Estimation using GCV approach

Using equations (9)-(13), the numerator of the derivative of GCV error with respect to α_j can be shown to take the form, $g_j + h_j\alpha_j$ where $g_j = (\delta_j b_j - a_j\epsilon_j)\psi_j - (\delta_j c_j - b_j\epsilon_j)$, $h_j = (\delta_j b_j - a_j\epsilon_j)\sigma^2$ and

$$\psi_j = \boldsymbol{\phi}_j^T \mathbf{P}_j \boldsymbol{\phi}_j \tag{22}$$

It is easy to verify that the denominator of the derivative of GCV error with respect to α_j is non-negative and noting that $\alpha_j \geq 0$, the solution can be obtained directly using g_j and h_j or using the sign information of the gradient. More specifically, the optimal solution α_j (lying in the interval $[0, \infty)$) is obtained from one of the following possibilities:

(1) If $\{g_j, h_j\} < 0$, then $\alpha_j = \infty$.

(2) If $\{g_j, h_j\} > 0$, then, $\alpha_j = 0$.

(3) If $g_j < 0, h_j > 0$, then unique solution exists and is given by $\alpha_j = -\frac{g_j}{h_j}$.

(4) If $g_j > 0, h_j < 0$, then unique solution exists and is given by $\alpha_j = -\frac{g_j}{h_j}$. But, in this case the derivative changes from positive value to negative value while crossing zero. Therefore, it is possible to have solution either at 0 or at ∞. In this case, we can evaluate the function value at 0 and ∞ and choose the right one.

(5) When $h_j = 0$, the solution is dependent on the sign of g_j.

Appendix II. Selection of the First Basis Vector

Two quantities that are of interest in finding the GCV error for all the basis vectors are given by:

$$\mathbf{t}^T \mathbf{P}_m^2 \mathbf{t} = \frac{a\Delta_m^2 - 2b_m\Delta_m + c_m}{\Delta_m^2} \tag{23}$$

$$tr(\mathbf{P}_m) = N - \frac{\boldsymbol{\phi}_m^T \boldsymbol{\phi}_m}{\boldsymbol{\phi}_m^T \boldsymbol{\phi}_m \ + \ \alpha_m \sigma^2} \tag{24}$$

where $\mathbf{P}_m = \mathbf{I} - \frac{\boldsymbol{\phi}_m \mathbf{S}_m^{-1} \boldsymbol{\phi}_m^T}{\sigma^2}$, $\mathbf{S}_m = \frac{\boldsymbol{\phi}_m^T \boldsymbol{\phi}_m + \alpha_m \sigma^2}{\sigma^2}$ and $\Delta_m = \mathbf{S}_m$. Then, the optimal solution α_m can be obtained, as described in appendix I, from the coefficients (9)-(13) using $\mathbf{P}_j = \mathbf{I}$. After substituting the optimal solution α_m into (23) and (24), we can evaluate the GCV error for a given m. Finally, the basis vector j is selected as the index for which the GCV error is minimum.

Appendix III. Initialization

Once the first basis vector,$\boldsymbol{\phi}_j$, is selected based on the GCV error with the optimal α_j, initialization of all the relevant quantities are done as follows:

$$r_m \ = \ \mathbf{t}^T \boldsymbol{\phi}_m \ - \ \frac{(\mathbf{t}^T \boldsymbol{\phi}_j)(\boldsymbol{\phi}_j^T \boldsymbol{\phi}_m)}{\Delta_j} \tag{25}$$

$$\gamma_m \ = \ \boldsymbol{\phi}_m^T \boldsymbol{\phi}_m \ - \ \frac{(\boldsymbol{\phi}_j^T \boldsymbol{\phi}_m)^2}{\Delta_j} \tag{26}$$

$$u_m \ = \ \boldsymbol{\phi}_m^T \boldsymbol{\phi}_m - 2\frac{(\boldsymbol{\phi}_j^T \boldsymbol{\phi}_m)^2}{\Delta_j} + \frac{(\boldsymbol{\phi}_m^T \boldsymbol{\phi}_m)(\boldsymbol{\phi}_j^T \boldsymbol{\phi}_m)^2}{\Delta_j^2} \tag{27}$$

$$\xi_m \ = \ \mathbf{t}^T \boldsymbol{\phi}_m - 2\frac{(\mathbf{t}^T \boldsymbol{\phi}_j)(\boldsymbol{\phi}_j^T \boldsymbol{\phi}_m)}{\Delta_j} + \frac{(\mathbf{t}^T \boldsymbol{\phi}_j)(\boldsymbol{\phi}_j^T \boldsymbol{\phi}_m)(\boldsymbol{\phi}_j^T \boldsymbol{\phi}_j)}{\Delta_j^2} \tag{28}$$

$$v \ = \ N \ - \ \frac{\boldsymbol{\phi}_j^T \boldsymbol{\phi}_j}{\Delta_j} \tag{29}$$

$$q \ = \ \mathbf{t}^T \mathbf{t} \ - \ 2\frac{(\mathbf{t}^T \boldsymbol{\phi}_j)^2}{\Delta_j} \ + \ \frac{(\mathbf{t}^T \boldsymbol{\phi}_j)^2 (\boldsymbol{\phi}_j^T \boldsymbol{\phi}_j)}{\Delta_j^2} \tag{30}$$

where $\Delta_j = \boldsymbol{\phi}_j^T \boldsymbol{\phi}_j + \alpha_j \sigma^2$. Further, $\mathbf{w} = \frac{\mathbf{t}^T \boldsymbol{\phi}_j}{\Delta_j}$, $\mathbf{S}^{-1} = [\frac{\sigma^2}{\Delta_j}]$ and $\boldsymbol{\Phi} = [\boldsymbol{\phi}_j]$. Note that $\mathbf{S}^{-1}$ is a single element matrix when there is a single basis vector.

Appendix IV. Computing α_j

Here, we find the set of coefficients required to compute the optimal α_j from the set of quantities defined in (16) - (21). All the results given below are obtained using the rank one update relationship between $\mathbf{P}$ and $\mathbf{P}_j$. With $o_j = \frac{\alpha_j \sigma^2 r_j}{\alpha_j \sigma^2 - t_j}$ and $\psi_j = \frac{\alpha_j \sigma^2 t_j}{\alpha_j \sigma^2 - t_j}$, we have

$$a_j = q + \frac{o_j}{\Delta_j}\Big(\frac{\epsilon_j o_j}{\Delta_j} + 2\xi_j \frac{\Delta_j}{\alpha_j \sigma^2}\Big) \tag{31}$$

$$b_j = o_j\Big(\xi_j \frac{\Delta_j}{\alpha_j \sigma^2} + \frac{\epsilon_j o_j}{\Delta_j}\Big) \tag{32}$$

$$c_j = \epsilon_j o_j^2 \tag{33}$$

where $\Delta_j = \psi_j + \alpha_j \sigma^2$ and $\epsilon_j = \frac{u_j}{(\alpha_j \sigma^2)^2} \Delta_j^2$. Further,

$$g_j = (\delta_j b_j - a_j \epsilon_j)\psi_j - (\delta_j c_j - b_j \epsilon_j) \tag{34}$$

$$h_j = (\delta_j b_j - a_j \epsilon_j)\sigma^2 \tag{35}$$

where $\delta_j = v + \frac{\epsilon_j}{\Delta_j}$. For j not in the relevance vector set, we do not have to find the above set of quantities with $\mathbf{P}_j$ as $\Delta_j = \infty$ and $\mathbf{P} = \mathbf{P}_j$. Therefore, the quantities in equations (9)-(13) required to compute the optimal α_j can be found in a much simpler way. Next, the optimal solution α_j is obtained using g_j and h_j as mentioned earlier. (See the discussion in the paragraph below equation (22).)

Appendix V. Re-estimation and deletion of a basis vector

Recall that the matrix $\mathbf{P} = \mathbf{I} - \frac{\mathbf{\Phi}\mathbf{S}^{-1}\mathbf{\Phi}^T}{\sigma^2}$. Then, with σ^2 fixed (at least for the iteration under consideration or for few iterations), change in α_j (which is essentially the re-estimation process itself) results in change in $\mathbf{S}^{-1}$. Let $\mathbf{s}_j$ denote the j^{th} column and s_{jj} denote the j^{th} diagonal element of $\mathbf{S}^{-1}$. Note that the computations are similar to the one used for re-estimation except for the coefficient K_j. In the case of deletion $K_j = \frac{1}{s_{jj}}$ and in the case of re-estimation $K_j = (s_{jj} + (\hat{\alpha}_j - \alpha_j)^{-1})^{-1}$. Here, $\hat{\alpha}_j$ denotes the new optimal solution. The final set of computations required for the re-estimation or deletion of basis vectors is given below. Defining $\rho_{jm} = \mathbf{s}_j^T \mathbf{\Phi}^T \boldsymbol{\phi}_m$, we have

$$r_m = r_m + K_j w_j \rho_{jm} \tag{36}$$

$$\gamma_m = \gamma_m + \frac{K_j}{\sigma^2} \rho_{jm}^2 \tag{37}$$

Defining $\chi_{jm} = \mathbf{s}_j^T \mathbf{A} \mathbf{S}^{-1} \mathbf{\Phi}^T \boldsymbol{\phi}_m$ we have

$$u_m = u_m + \frac{K_j}{\sigma^2} \rho_{jm} (2\chi_{jm} + \tau_j \rho_{jm}) \tag{38}$$

where $\tau_j = \frac{K_j}{\sigma^2} \mathbf{s}_j^T \mathbf{\Phi}^T \mathbf{\Phi} \mathbf{s}_j$. Defining $\kappa_j = \mathbf{w}^T \mathbf{A} \mathbf{s}_j$, we have

$$\xi_m = \xi_m + K_j \rho_{jm} \kappa_j + K_j w_j (\chi_{jm} + \rho_{jm} \tau_j) \tag{39}$$

$$v = v + \tau_j \tag{40}$$

$$q = q + K_j \sigma^2 w_j (2\kappa_j + \tau_j w_j) \tag{41}$$

Finally, $\mathbf{w} = \mathbf{w} - K_j w_j \mathbf{s}_j$ and $\mathbf{S}^{-1} = \mathbf{S}^{-1} - K_j \mathbf{s}_j \mathbf{s}_j^T$. Though the set of equations given above are common for re-estimation and deletion procedures, the j^{th} row and/or column is to be removed from $\mathbf{S}^{-1}$, $\mathbf{w}$ and $\mathbf{\Phi}$ after making the necessary updates in the deletion procedure.

Appendix VI. Adding a new basis vector

On adding the new basis vector j, the dimension of $\mathbf{\Phi}$ changes and a new finite α_j gets defined. Defining $\mathbf{l}_j = \frac{1}{\sigma^2}\mathbf{S}^{-1}\mathbf{\Phi}^T\boldsymbol{\phi}_j$ and $\mathbf{e}_j = \boldsymbol{\phi}_j - \mathbf{\Phi}\mathbf{l}_j$ and $\mu_{jm} = \mathbf{e}_j^T\boldsymbol{\phi}_m$ we get

$$r_m = r_m - w_j\mu_{jm} \tag{42}$$

$$\gamma_m = \gamma_m - \frac{s_{jj}}{\sigma^2}\mu_{jm}^2 \tag{43}$$

where $s_{jj} = \frac{1}{\frac{t_j}{\sigma^2} + \alpha_j}$ and $w_j = \frac{s_{jj}}{\sigma^2}r_j$. Next, defining $\nu_{jm} = \mu_{jm} - \mathbf{l}_j^T\mathbf{A}\mathbf{S}^{-1}\mathbf{\Phi}^T\boldsymbol{\phi}_m$, we have

$$\xi_m = \xi_m - \frac{s_{jj}}{\sigma^2}\mu_{jm}(\xi_j - w_ju_j) - w_j\nu_{jm} \tag{44}$$

$$u_m = u_m + \frac{s_{jj}}{\sigma^2}\mu_{jm}(\frac{s_{jj}}{\sigma^2}u_j\mu_{jm} - 2\nu_{jm}) \tag{45}$$

Next,

$$v = v - \frac{s_{jj}}{\sigma^2}u_j \tag{46}$$

$$q = q + w_j(w_ju_j - 2\xi_j) \tag{47}$$

Also,

$$\mathbf{S}^{-1} = \begin{pmatrix} \mathbf{S}^{-1} + s_{jj}\mathbf{l}_j\mathbf{l}_j^T & -s_{jj}\mathbf{l}_j \\ -s_{jj}\mathbf{l}_j^T & s_{jj} \end{pmatrix} \tag{48}$$

Finally, $\mathbf{w} = \begin{pmatrix} \mathbf{w} - w_j\mathbf{l}_j \\ w_j \end{pmatrix}$ and $\mathbf{\Phi} = [\mathbf{\Phi}\ \boldsymbol{\phi}_j]$.