Sparse Kernel Ridge Regression Using Backward Deletion

Ling Wang, Liefeng Bo, and Licheng Jiao

Institute of Intelligent Information Processing
710071, Xidian University, Xi'an, China
{wliiip, blf0218}@163.com

Abstract. Based on the feature map principle, Sparse Kernel Ridge Regression (SKRR) model is proposed. SKRR obtains the sparseness by backward deletion feature selection procedure that recursively removes the feature with the smallest leave-one-out score until the stop criterion is satisfied. Besides good generalization performance, the most compelling property of SKRR is rather sparse, and moreover, the kernel function needs not to be positive definite. Experiments on synthetic and benchmark data sets validate the feasibility and validity of SKRR.

1 Introduction

Regression problem is one of the fundamental problems in the field of supervised learning. It can be thought of as estimating the real valued function from a samples set of noise observation. A very successful approach for regression is Support Vector Regression (SVR) [1-2] that attempts to simultaneously minimize empirical risk and confidence interval, leading to good generalization. Due to ε-insensitive loss function, SVR obtains a sparse model (prediction for a new input only needs the subset of training samples). Though very successful, SVR also has some disadvantages [3]:

- The solution is usually not very sparse and the prediction speed for a new input is significantly slower than that of some other learning machines such as Neural Networks [4-5].
- Kernel function must satisfy Mercer's condition. It is well known that different kernel functions will induce different algorithms, achieving different performance. However, kernel functions in SVR must satisfy Mercer's positive definite condition, which limits the usable kernels.

In order to deal with the problems mentioned above, Relevance Vector Machines (RVM) [3] is proposed, which is very elegant and obtains a high sparse solution. However, RVM needs to solve linear equations, whose cost is very expensive, and therefore, it is not applicable to large scale problems.

In this paper, we propose a new learning model, Sparse Kernel Ridge Regression (SKRR) to overcome the above problems. In SKRR, samples are mapped into the feature space whose dimension is equal to the sample size, and then Ridge Regression (RR) [6] is implemented in the feature space. When the training process is completed, a backward deletion feature selection procedure is applied to obtain a sparse model.

Q. Yang and G. Webb (Eds.): PRICAI 2006, LNAI 4099, pp. 365 – 374, 2006.

Besides good generalization performance, the most compelling property of SKRR is its sparseness that is comparable with that of RVM. Another advantage of SKRR is that the kernel function needs not to be positive definite. Experiments on synthetic and benchmark data sets assess the feasibility and validity of SKRR.

2 Kernel Ridge Regression

Let $\{(\mathbf{x}_1, y_1),\cdots,(\mathbf{x}_l, y_l)\}$ be empirical samples set drawn from

$$y_i = f(\mathbf{x}_i)+\varepsilon_i,\quad i=1,2,\cdots,l, \tag{1}$$

where y_i is corrupted by additive noise ε_i, whose distributions are usually unknown. Learning aims to infer the function $f(\mathbf{x})$ from the finite data set $\{(\mathbf{x}_1, y_1),\cdots,(\mathbf{x}_l, y_l)\}$. A classical method for the problem is Ridge Regression (RR), which is an extension of linear regression by adding a quadratic penalizing term:

$$\hat{\mathbf{w}} = \arg\min\left(\sum_{i=1}^{l}\left(\mathbf{w}^T\mathbf{x}_i + w_{l+1} - y_i\right)^2 + \lambda\|\mathbf{w}\|^2\right), \tag{2}$$

where λ is a fixed positive constant, called regularization coefficient. Equation (**2**) can be rewritten as

$$\hat{\mathbf{w}} = \arg\min\left(\mathbf{w}^T\left(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I}\right)\mathbf{w} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{y}\right), \tag{3}$$

where $\mathbf{X} = \begin{bmatrix}\mathbf{x}_1^{'},1\\ \vdots\\ \mathbf{x}_l^{'},1\end{bmatrix}$, $\mathbf{w} = \begin{bmatrix}w_1\\ \vdots\\ w_{l+1}\end{bmatrix}$.

Ridge regression [6] is a well-known approach for the solution of regression problems, which has a good generalization performance as well as SVR, and the model does not need the kernel function satisfying Mercer's condition, moreover, there is an efficient leave-one-out cross-validation model selection method. In order to make RR applicable to the nonlinear problems, we generalize it by a feature map. Define a vector made up of a set of real-valued functions $\{k(\mathbf{x},\mathbf{x}_i)\,|\,i=1,2,\cdots,l\}$, as shown

$$\mathbf{z} = \left[k(\mathbf{x},\mathbf{x}_1),\cdots,k(\mathbf{x},\mathbf{x}_l)\right]^T, \tag{4}$$

where $k(\mathbf{x},\mathbf{x}^{'})$ is kernel function that needs not to be positive definite.

We call

$$F = \left\{\mathbf{z}\,|\,\mathbf{z} = \left[k(\mathbf{x},\mathbf{x}_1),\cdots,k(\mathbf{x},\mathbf{x}_l)\right]^T, \mathbf{x}\in R^n\right\} \tag{5}$$

feature space. In particular, $\left\{(\mathbf{z}_1, y_1),\cdots,(\mathbf{z}_l, y_l)\,|\,\mathbf{z}_i = \left[k(\mathbf{x}_i,\mathbf{x}_1),\cdots,k(\mathbf{x}_i,\mathbf{x}_l)\right]^T\right\}$ are l empirical samples in the feature space. Substituting $\mathbf{z}_i$ for $\mathbf{x}_i$, we have the Kernel Ridge Regression (KRR) [7] in the feature space F

$$\hat{\mathbf{w}} = \arg\min\left(\mathbf{w}^T\left(\mathbf{Z}^T\mathbf{Z}+\lambda\mathbf{I}\right)\mathbf{w} - 2\mathbf{w}^T\mathbf{Z}^T\mathbf{y}\right), \tag{6}$$

where $\mathbf{Z} = \begin{bmatrix} \mathbf{z}_1^{'},1 \\ \vdots \\ \mathbf{z}_l^{'},1 \end{bmatrix}$, $\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_{l+1} \end{bmatrix}$. The corresponding decision function is

$$f(\mathbf{x}) = \sum_{i=1}^{l} w_i k(\mathbf{x},\mathbf{x}_i) + w_{l+1}. \tag{7}$$

From equation (**6**) and (**7**), we can obtain

$$\hat{\mathbf{w}} = [\mathbf{Z}^T\mathbf{Z}+\lambda\mathbf{I}]^{-1}\mathbf{Z}^T\mathbf{Y}. \tag{8}$$

3 Sparse Kernel Ridge Regression

For a suitable λ, KRR has a good generalization performance. Its key disadvantage is no sparseness, which seems to prohibit its application in some fields. In this paper, we consider how to delete redundant features and simultaneously keep good generalization performance at an acceptable computational cost.

3.1 Feature Selection by Backward Deletion

In order to obtain the sparseness, a backward deletion procedure is implemented after the training process. Backward deletion procedure recursively removes the feature with the smallest leave-one-out score until the stop criterion is satisfied.

Let $\mathbf{H} = (\mathbf{Z}^T\mathbf{Z}+\lambda\mathbf{I})$ and $\mathbf{b} = \mathbf{Z}^T\mathbf{y}$, then equation (**6**) can be rewritten as

$$\hat{\mathbf{w}} = \arg\min\left(L = \left(\mathbf{w}^T\mathbf{H}\mathbf{w} - 2\mathbf{w}^T\mathbf{b}\right)\right). \tag{9}$$

When the k^{th} feature is deleted at the t^{th} iteration, let $\mathbf{H}^{(t,k)}$ represent the sub-matrix formed by omitting the k^{th} row and column of $\mathbf{H}$. Let $\mathbf{R}^{(t,k)}$ represent the inverse of $\mathbf{H}^{(t,k)}$, $\mathbf{w}^{(t,k)}$ the weights and $f^{(t,k)}$ the optimal value of L. According to equation (**8**), we have

$$f^{(t,k)} = -\sum_{i,j\in P-\{k\}} b_i\left(\mathbf{R}^{(t,k)}\right)_{ij} b_j. \tag{10}$$

where P is a set of remaining features (variables) at the t^{th} iteration.

In terms of a rank-1 update [8-9], $\mathbf{R}^{(t,k)}$ and $\mathbf{w}^{(t,k)}$ can be formulated in equation (**11**) and (**12**) (see Appendix for details), where $\mathbf{R}^t$ represents the inverse of $\mathbf{H}^t$.

$$\left(\mathbf{R}^{(t,k)}\right)_{ij} = \left(\mathbf{R}^t\right)_{ij} - \frac{(\mathbf{R}^t)_{ik}(\mathbf{R}^t)_{kj}}{(\mathbf{R}^t)_{kk}}, \quad i,j\in P-\{k\}. \tag{11}$$

$$\left(\mathbf{w}^{(t,k)}\right)_i = \sum_{j\in P-\{k\}}^{n}\left(\left(\mathbf{R}^t\right)_{ij} - \frac{(\mathbf{R}^t)_{ik}(\mathbf{R}^t)_{kj}}{(\mathbf{R}^t)_{kk}}\right)b_j, \quad i\in P-\{k\}. \tag{12}$$

Together with $\mathbf{w}_P^t = \mathbf{R}^t\mathbf{b}_P$, equation (**12**) is simplified as

$$\left(\mathbf{w}^{(t,k)}\right)_i = \mathbf{w}_i^t - \mathbf{w}_k^t\frac{(\mathbf{R}^t)_{ik}}{(\mathbf{R}^t)_{kk}}, \quad i\in P-\{k\}. \tag{13}$$

Substituting equation (**11**) into (**10**), we obtain

$$\begin{aligned} f^{(t,k)} &= -\sum_{i,j\in P} b_i\left(\mathbf{R}^t\right)_{ij}b_j + \sum_{i,j\in P} b_i\frac{(\mathbf{R}^t)_{ik}(\mathbf{R}^t)_{kj}}{(\mathbf{R}^t)_{kk}}b_j \\ &= f^t + \frac{\left(\sum_{j\in P}(\mathbf{R}^t)_{kj}b_j\right)^2}{(\mathbf{R}^t)_{kk}} \end{aligned}. \tag{14}$$

By the virtue of $\sum_{j\in P}(\mathbf{R}^t)_{kj}b_j = w_k^t$, equation (**14**) is translated into

$$\Delta f^{(t,k)} = f^{(t,k)} - f^t = \frac{\left(w_k^t\right)^2}{(\mathbf{R}^t)_{kk}}. \tag{15}$$

We call $\Delta f^{(t,k)}$ leave-one-out score. At each iteration, we remove the feature with the smallest leave-one-out score. The feature to be deleted can be obtained by the following expression.

$$s = \arg\min_{k\in P-\{l+1\}}\left(\Delta f^{(t,k)}\right). \tag{16}$$

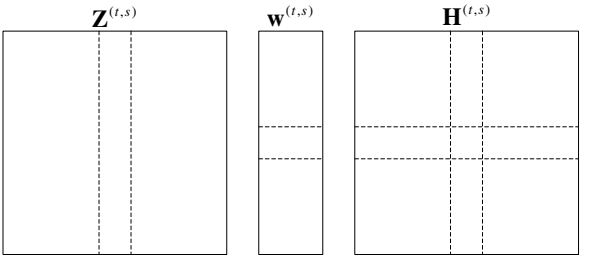

Fig. 1. Distribution of parameters after the s^{th} feature was deleted

Figure 1 shows the distribution of parameters after the s^{th} feature was deleted. Note that the $(l+1)^{th}$ variable is bias that is reserved during the feature selection process.

At the t^{th} iteration, the total increase of loss function L is

$$\Delta f^t = -\sum_{i \in P} b_i w_i^t - L^{opt}. \tag{17}$$

where L^{opt} is the minimum of equation (**9**). We terminate the algorithm if

$$\Delta f^t \geq \varepsilon \left| L^{opt} \right|. \tag{18}$$

where ε is a small positive number.

According to the derivation above, Backward Deletion Feature Selection (BDFS) can be described as the following Algorithm 1:

Algorithm 1. BDFS

1. **Set** $P = \{1, 2, \cdots, l\}$, $\mathbf{R}^1 = \mathbf{H}^{-1}$, $\mathbf{w}^1 = \mathbf{R}^1 \mathbf{b}$;
2. **For** $k = 1$ to l, **do**:
3. (a) $s = \arg \min_{k \in P - \{l+1\}} \left(\frac{\left(w_k^t\right)^2}{(\mathbf{R}^t)_{kk}} \right)$;
4. (b) $\left(\mathbf{R}^{(t,s)}\right)_{ij} = \left(\mathbf{R}^t\right)_{ij} - \frac{(\mathbf{R}^t)_{is} (\mathbf{R}^t)_{sj}}{(\mathbf{R}^t)_{ss}}, \quad i, j \in P - \{s\}$;
5. (c) $\left(\mathbf{w}^{(t,s)}\right)_i = \mathbf{w}_i^t - \mathbf{w}_s^t \frac{(\mathbf{R}^t)_{is}}{(\mathbf{R}^t)_{ss}}, \quad i \in P - \{s\}$;
6. (d) $P = P - \{s\}$, $\mathbf{R}^{t+1} = \mathbf{R}^{(t,s)}$, $\mathbf{w}^{t+1} = \mathbf{w}^{(t,s)}$.
7. (e) **IF** $\left(\mathbf{b}_P^T \mathbf{w}^{t+1} - \mathbf{b}^T \mathbf{w}^1\right) \geq \varepsilon \mathbf{b}^T \mathbf{w}^1$, **Stop.**
8. **End For**
9. **End Algorithm**

3.2 Model Selection

There exist free parameters including kernel parameter and regularization parameter in SKRR. In order to obtain good generalization, it is needed to choose the suitable parameters. Cross-validation is a model selection method often used in estimating the generalization performance of statistical classifiers. 10-fold cross-validation is often used in some kernel-based learning algorithms such as SVMs. Leave-one-out cross validation is the most extreme form of cross-validation.

Leave-one-out cross validation error is an attractive model selection criterion since it provides an almost unbiased estimator of generalization performance [2]. However, this method is rarely adopted in the kernel machines because it is computationally expensive.

Fortunately for SKRR, there is an efficient implementation for leave-one-out cross validation that only incurs a computational cost of $O(l^3)$ [10]. Let $E(\Gamma)$ be leave-one-out cross validation error, where Γ denotes free parameters.

Lemma 1. [11-12]. $E(\Gamma)=\frac{1}{l}\left\|\mathbf{B}(\Gamma)(\mathbf{I}-\mathbf{A})\mathbf{y}\right\|_2^2$, where $\mathbf{B}(\Gamma)$ is a diagonal matrix with the jjth entry $1/(1-\alpha_{jj}(\Gamma))$, $\alpha_{jj}(\Gamma)$ being the jjth entry of $\mathbf{A}(\Gamma)=\mathbf{Z}\left(\mathbf{Z}^T\mathbf{Z}+\lambda\mathbf{I}\right)^{-1}\mathbf{Z}^T$.

Let the singular value decomposition of $\mathbf{Z}$ be

$$\mathbf{Z}=\mathbf{U}\mathbf{D}\mathbf{V}^T, \tag{19}$$

where $\mathbf{U},\mathbf{V}$ are orthogonal matrices, $\mathbf{D}$ is a diagonal matrix. Substituting equation (**19**) into $\mathbf{A}(\Gamma)$, it can be simplified as

$$\mathbf{A}(\Gamma)=\mathbf{U}\mathbf{D}\left(\mathbf{D}^T\mathbf{D}+\lambda\mathbf{I}\right)^{-1}\mathbf{D}^T\mathbf{U}^T, \tag{20}$$

where $\mathbf{D}^T\mathbf{D}+\lambda\mathbf{I}$ is a diagonal matrix.

Hence for different λ, we only need to perform once matrix decomposition. For clarity, here we give the detail steps of leave-one-out model selection algorithm in which σ represents kernel parameter:

Algorithm 2. LOO Model Selection

For σ_i, $i=1,2,\cdots,q$

$\mathbf{Z}=\mathbf{U}\mathbf{D}\mathbf{V}^T$;

For λ_j, $j=1,2,\cdots,p$

$\alpha_{kk}=\mathbf{U}_k\mathbf{D}_k(\mathbf{D}^T\mathbf{D}+\lambda_j\mathbf{I})^{-1}\mathbf{D}_k^T\mathbf{U}_k^T$;

$\mathbf{B}_{kk}=1/(1-\alpha_{kk})$;

$r_{(k)}=\mathbf{B}_{kk}(y_k-\mathbf{A}_k y_k)$

$E_{ij}=\frac{1}{l}\sum_{k=1}^{l}\{r_{(k)}\}_k^2$

End for

End for

$(\lambda^{opt},\sigma^{opt})=\arg\min_{i,j}(E_{ij})$.

Thus, according to the analysis in section 3.1 and 3.2, SKRR can be described in the following Algorithm 3:

Algorithm 3. SKRR

1. Let λ be noise variance, and choose the kernel parameters using LOO model selection algorithm;
2. Train KRR using the selected parameters;
3. Implement BDFS for KRR;
4. Re-estimate λ using LOO model selection algorithm;
5. Re-train KRR on the simplified features.

4 Simulation

In order to evaluate the performance of the proposed algorithm, we performed SKRR on three data sets: Sinc, Boston Housing and Abalone data sets, and compared its performance with that of KRR, SVR, and RVM. For the sake of comparison, different algorithms use the same input sequence. The elements of Gram Matrix are constructed using Gaussian kernel of the form $k(\mathbf{x},\mathbf{y}) = \exp\left(\frac{-\|\mathbf{x}-\mathbf{y}\|_2^2}{2\sigma^2}\right)$. For SVR and RVM, we utilize 10-fold cross validation procedure to choose the free parameters. For all datasets, parameter ε in BDFS is set 0.01. Large amounts of experiments show that it is a good selection.

4.1 Toy Experiment

The Sinc function $y = \sin(x)/x + \sigma N(0,1)$ is a popular choice to illustrate support vector machines regression. We generate training samples from Sinc function at 100 equally-spaced x-value in $[-10,10]$ with added Gaussian noise of standard deviation 0.1. Results are averaged over 100 random instantiations of the noise, with the error being measured over 1000 noise-free test samples in $[-10,10]$. The decision functions and support vectors obtained by SKRR and SVR with $\varepsilon = 0.1$ are shown in Figure 2. Support vectors number and mean square error are summarized in Table1. For Sinc data set, SKRR outperforms other three algorithms. SKRR and RVM obtain similar support vectors number that is significantly less than that of SVR.

Table 1. Generalization error obtained by the four algorithms on Sinc data set. MSE denotes mean square error and NSV denotes support vectors number

	SKRR	KRR	SVR	RVM
MSE	0.85 ± 0.42	0.88 ± 0.41	1.45 ± 0.64	0.98 ± 0.47
NSV	7.20 ± 0.49	100 ± 0.00	35.41 ± 4.71	6.97 ± 0.38

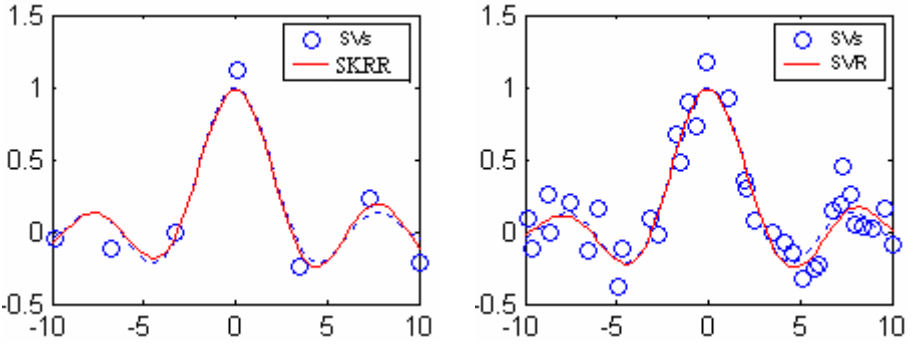

Fig. 2. Support vectors and decision boundary obtained by SKRR and SVR. Real red lines denote the decision boundary and dot lines denote the Sinc function; circles denote the support vectors.

4.2 Benchmark Comparison

Boston Housing and Abalone data sets that come from STATLOG COLLECTION [13] are popular choices to evaluate the performance of algorithms. Boston Housing data set includes 506 examples, 13 attributes of each example and Abalone data set includes 4177 examples, 7 attributes of each example. For the Boston Housing data set, we average our results over 100 random splits of the full dataset into 481 training samples and 25 testing samples. For the Abalone data set, we average our results over 10 random splits of the mother dataset into 1000 training samples and 3177 testing samples. Before experiments, we scale all training samples into [-1, 1] and then adjust testing samples using the same linear transformation. The results are summarized in Table 2 and 3.

Table 2. Generalization error obtained by the four algorithms on Boston Housing data set. MSE denotes mean square error and NSV denotes support vectors number.

	SKRR	KRR	SVR	RVM
MSE	10.05 ± 6.72	9.75 ± 6.90	10.51 ± 7.99	10.08 ± 6.82
NSV	52.23 ± 1.76	481.00 ± 0.00	165.98 ± 4.90	48.80 ± 2.49

From the experimental results (Table 1, 2 and 3), we observe the following

- SKRR, KRR and RVM obtain similar performance and are slightly superior to SVR.
- Both SKRR and RVM are rather sparse. Support (relevance) vectors number of SKRR and RVM is much less than that of SVR.

Table 3. Generalization error obtained by the four algorithms on Abalone data set. MSE denotes mean square error and NSV denotes support vectors number.

	SKRR	KRR	SVR	RVM
MSE	4.64 ± 0.12	4.61 ± 0.10	4.66 ± 0.11	4.59 ± 0.12
NSV	18.90 ± 5.38	1000 ± 0.00	468.90 ± 58.77	11.30 ± 2.45

5 Conclusion

Backward deletion feature selection provides a state-of-the-art tool that can delete redundant features and simultaneously keep good generalization performance at an acceptable computational cost. Based on BDFS, we propose SKRR that obtains high sparseness. Further application of BDFS includes condensing the two-stage RBF Networks. Application to linear regression is in progress.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under grant No. 60372050 and the National Grand Fundamental Research 973 Program of China under Grant No. 2001CB309403.

References

1. Vapnik, V: The Nature of Statistical Learning Theory. New York Springer-Verlag (1995)
2. Vapnik, V: Statistical Learning Theory. New York Wiley-Interscience Publication (1998)
3. Tipping, M.E: The Relevance Vector Machine. Proc. Advances in Neural Information Processing Systems 12, Solla,S., Leen,T. and Müller K.-R. eds. Cambridge, Mass MIT Press (2001) 652-658.
4. Neal, R: Bayesian Learning for Neural Networks. New York Springer Verlag (1996)
5. Ripley R: Pattern Recognition and Neural Networks. Cambridge, U.K. Cambridge Univ. Press (1996)
6. Hoerl, A. and Kennard, R: Ridge Regression: Biased Estimation for Nonorthogonal Problems. Technometrics 12 (1970) 55-67
7. Saunder C. and Gammerman A: Ridge regression learning algorithm in dual variables. In: Shavlik J. editor. Machine learning Proceedings of the 15th International Conference, Morgan Kaufmann (1998)
8. Stoer, J. and Bulirsch, R: Introduction to Numerical Analysis. Springer Verlag, New York, Second Edition (1993)
9. Bo, L.F., Wang, L., and Jiao, L.C.: Sparse Gaussian processes using backward elimination. Lecture Notes in Computer Science 3971 (2006) 1083-1088
10. Bo, L.F., Wang, L., and Jiao, L.C.: Feature scaling for kernel Fisher discriminant analysis using leave-one-out cross validation. Neural Computation 18(4) (2006) 961-978
11. Allen, D.M: The relationship between variable selection and prediction. Technometrics 16 (1974) 125-127
12. Gavin C.C.: Efficient leave-one-out cross-validation of kernel fisher discriminant classifiers. Pattern Recognition 36(11) (2003) 2585-2592
13. Michie, D, Spiegelhalter, D. J., and Taylor, C.C.: Machine Learning, Neural and Statistical Classification. Prentice Hall, Englewood Cliffs, N.J., 1994, Data available at anonymous ftp: ftp.ncc.up.pt/pub/statlog/.

Appendix

The matrix inversion formula for a symmetric matrix with block sub-matrices is given in the following Lemma 2.

Lemma 2 [8]: Given invertible matrices $\mathbf{A}$ and $\mathbf{C}$, and matrix $\mathbf{B}$, the following equality holds:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}\mathbf{E}^{-1}\mathbf{B}^T\mathbf{A}^{-1} & -\mathbf{E}^{-1}\mathbf{B}\mathbf{C}^{-1} \\ -\mathbf{C}^{-1}\mathbf{B}^T\mathbf{E}^{-1} & \mathbf{C}^{-1} + \mathbf{C}^{-1}\mathbf{B}^T\mathbf{D}^{-1}\mathbf{B}\mathbf{C}^{-1} \end{bmatrix}, \tag{A.1}$$

where $\begin{matrix} \mathbf{D} = \mathbf{A} - \mathbf{B}\mathbf{C}^{-1}\mathbf{B}^T \\ \mathbf{E} = \mathbf{C} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B} \end{matrix}$.

Equation (**11**) can be derived in terms of Lemma 2. We assume here that $\begin{bmatrix} \mathbf{B} \\ \mathbf{C} \end{bmatrix}$ is the k^{th} feature to be deleted. Thus in our problem, $\mathbf{R}^{(t,k)}$ corresponds to $\mathbf{A}^{-1}$, and

$(\mathbf{R}^t)_{kk}$ corresponds to $\mathbf{E}^{-1}$, and $(\mathbf{R}^t)_{ik}$ corresponds to $-\mathbf{A}^{-1}\mathbf{B}\mathbf{E}^{-1}$, $i \in P - \{k\}$. Observing the top left block of the last part in equation (**A.1**), we have

$$\mathbf{A}^{-1} + \frac{\mathbf{A}^{-1}\mathbf{B}\mathbf{E}^{-1}\mathbf{E}^{-1}\mathbf{B}^T\mathbf{A}^{-1}}{\mathbf{E}^{-1}}, \tag{A.2}$$

which corresponds to

$$\left(\mathbf{R}^{(t,k)}\right)_{ij} + \frac{(\mathbf{R}^t)_{ik}(\mathbf{R}^t)_{kj}}{(\mathbf{R}^t)_{kk}}, \quad i, j \in P - \{k\}. \tag{A.3}$$

Hence, we have

$$\left(\mathbf{R}^{(t,k)}\right)_{ij} = \left(\mathbf{R}^t\right)_{ij} - \frac{(\mathbf{R}^t)_{ik}(\mathbf{R}^t)_{kj}}{(\mathbf{R}^t)_{kk}}, \quad i, j \in P - \{k\}. \tag{A.4}$$