An Exchanging-based Refinement to Sparse Gaussian Process Regression

Ping Sun and Xin Yao
School of Computer Science, University of Birmingham

May 18, 2006

Abstract

We propose a backward deletion procedure to Sparse Gaussian Process Regression (SGPR) model, which can be used to refine a number of sequential forward selection algorithms addressed recently. Some experimental results demonstrate the effectiveness of our approach.

1 Introduction

Recently a number of sparse approximations to sparse Gaussian Process Regression (GPR) have been proposed. Of those, forward selection algorithms [5, 4, 1, 3, 7, 6] have attracted much attention due to its simplicity and efficiency. This kind of method involves iteratively selecting a subset of training examples (known as *basis vectors*) based on various scoring criteria. It should be noted that all of these approaches follow the path of forward stagewise fitting. A natural raised question is: can we have a corresponding backward elimination procedure, which is used to to remove some redundant basis vectors included in previous iterations [1]? The simple answer is yes and we will detail how to develop this procedure in this paper. Moreover, a simple deletion criterion based on least increasing regularised training error was also addressed. Combining forward selection and our backward approach together, we finally present an exchanging-based strategy to refine previous forward selection algorithms.

2 Sparse Gaussian Process Regression

Given a training data $D = \{(x_1, y_1), ..., (x_n, y_n)\}$ composed of n examples, the MAP estimate of GPR [5, 3] can be equivalently viewed as the following optimisation problem

$$\min_{\alpha} \pi(\alpha) := \frac{1}{2}\|y - K\alpha\|^2 + \frac{\sigma^2}{2}\alpha^\top K\alpha, \tag{1}$$

[1]Lehel and Manfred [2] proposed a backward step for the on-line version of Gaussian Process but it cannot be applied to batch version of GPR we considered here.

where $y = [y_1, ..., y_n]^\top$ is the target vector, K is the covariance matrix generated by evaluating paired inputs on a kernel function $k(x_i, x_j)$, σ^2 is the noise variance and $\alpha = [\alpha_1, ..., \alpha_n]^\top$ denotes the weight vector. After solving α from (1), we can get the corresponding regression model $f(x) = \sum_{i=1}^n \alpha_i k(x_i, x)$.

In order to deal with large-scale datasets, sometimes we are more interested in a sparse solution to (1), which means some redundant entries of α are exactly zeros. Let α_p be all the non-zero entries of α indexed by $I_p = [i_1, ..., i_p]$ which corresponds to the set of selected basis vectors, the MAP estimate of sparse GPR is equivalent to minimizing

$$\min_{\alpha_p} \pi_p(\alpha_p) := \frac{1}{2}\|y - K_p\alpha_p\|^2 + \frac{\sigma^2}{2}\alpha_p^\top Q_p \alpha_p, \tag{2}$$

where K_p is a $n \times p$ matrix of the covariance function between all the training examples and selected basis vectors and Q_p is the $p \times p$ covariance matrix of evaluating basis vectors on the kernel function $k(\cdot, \cdot)$.

A number of forward selection algorithms have previously been developed to select the set of basis vectors. The central idea is to include the next basis vector which results in the largest reduction in the cost function (2) at each iteration, i.e., the $(p+1)$-th basis vector is chosen as

$$i_{p+1} = \arg\max_{i \notin I_p} \{|\pi_p - \pi_{p+1}^i|\} \tag{3}$$

where π_{p+1}^i is the minimum of (2) after the training instance x_i is added to the set of basis vectors. To overcome the computational cost of computing (3) for all remaining basis candidates, some authors have recently suggested a number of efficient implementation strategies [5, 1, 3, 7]. Our work in this paper is concerned with a procedure of backward elimination, which is based on optimising the cost (2) as well.

3 Backward Elimination

In contrast to forward selection, the philosophy of backward elimination is to sequentially remove from the index set of basis vectors I_p, one at a time, those basis vectors that lead to the smallest increase in the cost function. The increase in the cost caused by excluding a particular basis vector, i_j, can be computed efficiently as described in the following. Since the cost function (2) arrives at its optimum at $\alpha_p = \Sigma_p K_p^\top y$ where $\Sigma_p = (K_p^\top K_p + \sigma^2 Q_p)^{-1}$, we can rewrite it as $\pi_p = \frac{1}{2} y^\top (Id_n - K_p \Sigma_p K_p^\top) y$, where Id_n is the identity matrix of size $n \times n$. Let $\pi_{p-1}^{\backslash j}$ denote the cost after the removal of i_j from I_p, we can have the following recursive equation:

$$\pi_{p-1}^{\backslash j} = \pi_p + \frac{1}{2}\alpha_p^2(j)/\Sigma_p(j,j), \tag{4}$$

where $\alpha_p(j)$ is the j-th entry of α_p and $\Sigma_p(j,j)$ is the j-th diagonal of Σ_p. So the resulting criteria of choosing the basis to be excluded is simply

$$i_j = \arg\min_{j=1,\ldots,p} \frac{1}{2}\alpha_p^2(j)/\Sigma_p(j,j). \tag{5}$$

Assuming that α_p and the diagonal of Σ_p are implicitly computed at each iteration, this criteria would incur negligible computational cost. Once a basis vector is selected, we need to update the weight vector α_p and other necessary quantities which include $K_p, Q_p, L_p, G_p, M_p, \mu_p, D_p$ and the index set I_p, where L_p is the factor of Cholesky decomposition: $L_p L_p^\top = Q_p$, $G_p = K_p L_p^{-\top}$, M_p is the Cholesky factor: $M_p M_p^\top = G_p^\top G_p + \sigma^2 Id_p$, $\mu_p = y - K_p\alpha_p$ is the residual and D_p is the diagonal of Σ_p. The updating steps of backward eliminiation are summarized below (the corresponding forward steps are detailed in [4, 6]):

```
1: Given: $K_p, Q_p, L_p, G_p, M_p, \alpha_p, \mu_p, D_p, I_p$ and the index $j$ to be removed;
2: Downdate $L_p, M_p$ at the $j$-th row and $G_p$ at the $j$-th column:
       $L_p \leftarrow \Pi L_p R_i^1 \dots R_{p-1}^1$, $G_p \leftarrow G_p R_i^1 \dots R_{p-1}^1$ and $M_p \leftarrow M_p R_i^2 \dots R_{p-1}^2$, where
   $\Pi$ is an appropriate permutation matrix and $R_i^1, R_i^2$ denote appropriate rotations
   in planes $i, i+1$ for $i = j, \dots, p-1$.
3: $K_p \leftarrow K_p \Pi^\top, Q_p \leftarrow \Pi Q_p \Pi^\top, \alpha_p \leftarrow \Pi\alpha_p, D_p \leftarrow \Pi D_p$ and $I_p \leftarrow \Pi I_p$;
4: Let $L_p = \begin{bmatrix} L_{p-1} & l_j \\ l_j^\top & l_j^* \end{bmatrix}$, $G_p = [G_{p-1}\, g_j]$,$M_p = \begin{bmatrix} M_{p-1} & m_j \\ m_j^\top & m_j^* \end{bmatrix}$, $\alpha_p =$
   $\begin{bmatrix} \alpha_{p-1} \\ a_j \end{bmatrix}$,we have $b = a_j l_j^* (m_j^*)^2, \eta = M_{p-1}^{-\top} m_j$ and $d = g_j - G_{p-1}\eta$.
5: Define $\kappa = L_{p-1}^{-\top}(l_j + l_j^*\eta)$, we have $\alpha_{p-1} = \alpha_p + a_j\kappa$, $\mu_{p-1} = \mu_p - bd/(m_j^*)^2$ and
   $D_{p-1} = D_p - \kappa^2/(l_j^* m_j^*)^2$
6: Outputs: $K_{p-1}, Q_{p-1}, L_{p-1}, G_{p-1}, M_{p-1}, \alpha_{p-1}, \mu_{p-1}, D_{p-1}$ and $I_{p-1}$. Note that
   the appropriate row and\or column is removed from some of variables.
```

It can be easily seen that the computational cost of one backward step just leads to $O(np)$ time which is linear in the number of training examples. The combination of forward and backward steps can be used to refine the results of any forward selection approach. The idea is certainly straightforward: at the end of forward procedure, we apply the criteria (5) to decide on the basis vector to be disregarded and then use forward procedure to select a new basis vector from all the remaining vectors for replacing the former. This process would initially increase the cost which is then more than compensated for by an additional basis vector.

4 Numerical experiments

We demonstrate the proposed exchanging strategy on the KIN40K dataset [2] which is composed of 40000 examples with 8 inputs. We randomly split the mother data into 10000 training and 30000 testing examples and produce 20 repetitions, respectively. All the hyperparameters involved (σ^2 and kernel parameters) are estimated via a full GPR model with commonly used *squared-exponential* kernel on a subset of 1000 examples randomly selected from the original dataset.

[2] See `http://ida.first.fraunhofer.de/~anton/data.html`

We employ the DMAX approach [3] as the choice of forward selection procedure in our experiment. The number of selected basis vectors is fixed on 500 and the exchanging steps are from 100 to 300. Table 1 summarizes test NMSE (normalized mean squared error) and NLPD (negative logarithm of predictive distribution) results of applying the DMAX method with our proposed exchanging strategy, where '$500 \pm \Delta$' denotes 500 forward selection steps followed by Δ exchanging steps.

Table 1: Results of the DMAX [3] approach with our proposed exchanging strategy on KIN40K dataset. The p-value is for the paired *t*-test on test error.

Steps	test NMSE	p-value	test NLPD	p-value
500± 0	0.0355± 0.0009	-	-1.3673±0.0059	-
500± 100	**0.0318±0.0008**	9.9×1e-16	**-1.3736±0.0057**	1.2×1e-3
500± 200	**0.0300±0.0007**	4.9×1e-22	**-1.3790±0.0050**	4.8×1e-8
500± 300	**0.0290±0.0007**	1.9×1e-24	**-1.3825±0.0055**	2.6×1e-10

From Table 1, it can be noted that our exchanging strategy could significantly improve the result of the forward selection approach DMAX, which is decided by a p-value threshold of 0.01 in the paired *t*-test. It has also been observed that the more exchanging steps we run, the more benefits we can achieve in the KIN40K dataset.

5 Future work

Most of existing forward selection algorithms and our proposed work are all based on optimising the cost function (2). Actually, the forward selection step like [7] and the backward step developed here can be extended to optimizing marginal likelihood cost function which is commonly used for model selection. It would produce a more reliable way of learning kernel hyperparameters for sparse GPR model through a combination of backward and forward steps.

References

[1] F. R. Bach and M. I. Jordan. Predictive low-rank decomposition for kernel methods. In *ICML'05*, pages 33–40, 2005.

[2] L. Csato and M. Opper. Sparse on-line Gaussian Processes. *Neural Computation*, 14(3):641–668, 2002.

[3] S. S. Keerthi and W. Chu. A matching pursuit approach to sparse Gaussian process regression. In *NIPS 18*. MIT Press, 2005.

[4] M. Seeger, C. K. I. Williams, and N. D. Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *AISTATS'03*, Key West, Florida, 2003.

[5] A. J. Smola and P. Bartlett. Sparse greedy gaussian process regression. In *NIPS 14*, pages 619–625. MIT Press, 2001.

[6] P. Sun and X. Yao. A Gradient-based Forward Greedy Algorithm for Sparse Gaussian Process Regression. In *Trends in Neural Computation*. Springer, 2006. to appear.

[7] P. Sun and X. Yao. Greedy forward selection algorithms to sparse Gaussian Process Regression. In *IJCNN'06*, 2006. to appear.