Chapter 8

Approximation Methods for Large Datasets

As we have seen in the preceding chapters a significant problem with Gaussian process prediction is that it typically scales as $\mathcal{O}(n^3)$. For large problems (e.g. $n > 10,000$) both storing the Gram matrix and solving the associated linear systems are prohibitive on modern workstations (although this boundary can be pushed further by using high-performance computers).

An extensive range of proposals have been suggested to deal with this problem. Below we divide these into five parts: in section 8.1 we consider reduced-rank approximations to the Gram matrix; in section 8.2 a general strategy for greedy approximations is described; in section 8.3 we discuss various methods for approximating the GP regression problem for fixed hyperparameters; in section 8.4 we describe various methods for approximating the GP classification problem for fixed hyperparameters; and in section 8.5 we describe methods to approximate the marginal likelihood and its derivatives. Many (although not all) of these methods use a subset of size $m < n$ of the training examples.

8.1 Reduced-rank Approximations of the Gram Matrix

In the GP regression problem we need to invert the matrix $K + \sigma_n^2 I$ (or at least to solve a linear system $(K + \sigma_n^2 I)\mathbf{v} = \mathbf{y}$ for $\mathbf{v}$). If the matrix K has rank q (so that it can be represented in the form $K = QQ^\top$ where Q is an $n \times q$ matrix) then this matrix inversion can be speeded up using the matrix inversion lemma eq. (A.9) as $(QQ^\top + \sigma_n^2 I_n)^{-1} = \sigma_n^{-2} I_n - \sigma_n^{-2} Q(\sigma_n^2 I_q + Q^\top Q)^{-1} Q^\top$. Notice that the inversion of an $n \times n$ matrix has now been transformed to the inversion of a $q \times q$ matrix.[1]

[1] For numerical reasons this is not the best way to solve such a linear system but it does illustrate the savings that can be obtained with reduced-rank representations.

In the case that the kernel is derived from an explicit feature expansion with N features, then the Gram matrix will have rank $\min(n, N)$ so that exploitation of this structure will be beneficial if $n > N$. Even if the kernel is non-degenerate it may happen that it has a fast-decaying eigenspectrum (see e.g. section 4.3.1) so that a reduced-rank approximation will be accurate.

If K is not of rank $< n$, we can still consider reduced-rank *approximations* to K. The optimal reduced-rank approximation of K w.r.t. the Frobenius norm (see eq. (A.16)) is $U_q \Lambda_q U_q^\top$, where Λ_q is the diagonal matrix of the leading q eigenvalues of K and U_q is the matrix of the corresponding orthonormal eigenvectors [Golub and Van Loan, 1989, Theorem 8.1.9]. Unfortunately, this is of limited interest in practice as computing the eigendecomposition is an $\mathcal{O}(n^3)$ operation. However, it does suggest that if we can more cheaply obtain an approximate eigendecomposition then this may give rise to a useful reduced-rank approximation to K.

We now consider selecting a subset I of the n datapoints; set I has size $m < n$. The remaining $n - m$ datapoints form the set R. (As a mnemonic, I is for the included datapoints and R is for the remaining points.) We sometimes call the included set the active set. Without loss of generality we assume that the datapoints are ordered so that set I comes first. Thus K can be partitioned as

$$K = \begin{pmatrix} K_{mm} & K_{m(n-m)} \\ K_{(n-m)m} & K_{(n-m)(n-m)} \end{pmatrix}. \tag{8.1}$$

The top $m \times n$ block will also be referred to as K_{mn} and its transpose as K_{nm}.

In section 4.3.2 we saw how to approximate the eigenfunctions of a kernel using the Nyström method. We can now apply the same idea to approximating the eigenvalues/vectors of K. We compute the eigenvectors and eigenvalues of K_{mm} and denote them $\{\lambda_i^{(m)}\}_{i=1}^m$ and $\{\mathbf{u}_i^{(m)}\}_{i=1}^m$. These are extended to all n points using eq. (4.44) to give

$$\tilde{\lambda}_i^{(n)} \triangleq \frac{n}{m} \lambda_i^{(m)}, \qquad i = 1, \ldots, m \tag{8.2}$$

$$\tilde{\mathbf{u}}_i^{(n)} \triangleq \sqrt{\frac{m}{n}} \frac{1}{\lambda_i^{(m)}} K_{nm} \mathbf{u}_i^{(m)}, \qquad i = 1, \ldots, m \tag{8.3}$$

where the scaling of $\tilde{\mathbf{u}}_i^{(n)}$ has been chosen so that $|\tilde{\mathbf{u}}_i^{(n)}| \simeq 1$. In general we have a choice of how many of the approximate eigenvalues/vectors to include in our approximation of K; choosing the first p we get $\tilde{K} = \sum_{i=1}^p \tilde{\lambda}_i^{(n)} \tilde{\mathbf{u}}_i^{(n)} (\tilde{\mathbf{u}}_i^{(n)})^\top$. Below we will set $p = m$ to obtain

$$\tilde{K} = K_{nm} K_{mm}^{-1} K_{mn} \tag{8.4}$$

Nyström approximation

using equations 8.2 and 8.3, which we call the Nyström approximation to K. Computation of $\tilde{K}$ takes time $\mathcal{O}(m^2 n)$ as the eigendecomposition of K_{mm} is $\mathcal{O}(m^3)$ and the computation of each $\tilde{\mathbf{u}}_i^{(n)}$ is $\mathcal{O}(mn)$. Fowlkes et al. [2001] have applied the Nyström method to approximate the top few eigenvectors in a computer vision problem where the matrices in question are larger than $10^6 \times 10^6$ in size.

The Nyström approximation has been applied above to approximate the elements of K. However, using the approximation for the ith eigenfunction $\tilde{\phi}_i(\mathbf{x}) = (\sqrt{m}/\lambda_i^{(m)})\mathbf{k}_m(\mathbf{x})^\top \mathbf{u}_i^{(m)}$, where $\mathbf{k}_m(\mathbf{x}) = (k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_m))^\top$ (a restatement of eq. (4.44) using the current notation) and $\lambda_i \simeq \lambda_i^{(m)}/m$ it is easy to see that in general we obtain an approximation for the kernel $k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{N} \lambda_i \phi_i(\mathbf{x})\phi_i(\mathbf{x}')$ as

$$\tilde{k}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{m} \frac{\lambda_i^{(m)}}{m} \tilde{\phi}_i(\mathbf{x})\tilde{\phi}_i(\mathbf{x}') \tag{8.5}$$

$$= \sum_{i=1}^{m} \frac{\lambda_i^{(m)}}{m} \frac{m}{(\lambda_i^{(m)})^2} \mathbf{k}_m(\mathbf{x})^\top \mathbf{u}_i^{(m)} (\mathbf{u}_i^{(m)})^\top \mathbf{k}_m(\mathbf{x}') \tag{8.6}$$

$$= \mathbf{k}_m(\mathbf{x})^\top K_{mm}^{-1} \mathbf{k}_m(\mathbf{x}'). \tag{8.7}$$

Clearly eq. (8.4) is obtained by evaluating eq. (8.7) for all pairs of datapoints in the training set.

By multiplying out eq. (8.4) using $K_{mn} = [K_{mm} K_{m(n-m)}]$ it is easy to show that $K_{mm} = \tilde{K}_{mm}$, $K_{m(n-m)} = \tilde{K}_{m(n-m)}$, $K_{(n-m)m} = \tilde{K}_{(n-m)m}$, but that $\tilde{K}_{(n-m)(n-m)} = K_{(n-m)m} K_{mm}^{-1} K_{m(n-m)}$. The difference $K_{(n-m)(n-m)} - \tilde{K}_{(n-m)(n-m)}$ is in fact the *Schur complement* of K_{mm} [Golub and Van Loan, 1989, p. 103]. It is easy to see that $K_{(n-m)(n-m)} - \tilde{K}_{(n-m)(n-m)}$ is positive semi-definite; if a vector $\mathbf{f}$ is partitioned as $\mathbf{f}^\top = (\mathbf{f}_m^\top, \mathbf{f}_{n-m}^\top)$ and $\mathbf{f}$ has a Gaussian distribution with zero mean and covariance K then $\mathbf{f}_{n-m}|\mathbf{f}_m$ has the Schur complement as its covariance matrix, see eq. (A.6).

The Nyström approximation was derived in the above fashion by Williams and Seeger [2001] for application to kernel machines. An alternative view which gives rise to the same approximation is due to Smola and Schölkopf [2000] (and also Schölkopf and Smola [2002, sec. 10.2]). Here the starting point is that we wish to approximate the kernel centered on point $\mathbf{x}_i$ as a linear combination of kernels from the active set, so that

$$k(\mathbf{x}_i, \mathbf{x}) \simeq \sum_{j \in I} c_{ij} k(\mathbf{x}_j, \mathbf{x}) \triangleq \hat{k}(\mathbf{x}_i, \mathbf{x}) \tag{8.8}$$

for some coefficients $\{c_{ij}\}$ that are to be determined so as to optimize the approximation. A reasonable criterion to minimize is

$$E(C) = \sum_{i=1}^{n} \|k(\mathbf{x}_i, \mathbf{x}) - \hat{k}(\mathbf{x}_i, \mathbf{x})\|_{\mathcal{H}}^2 \tag{8.9}$$

$$= \operatorname{tr} K - 2 \operatorname{tr}(C K_{mn}) + \operatorname{tr}(C K_{mm} C^\top), \tag{8.10}$$

where the coefficients are arranged into a $n \times m$ matrix C. Minimizing $E(C)$ w.r.t. C gives $C_{\text{opt}} = K_{nm} K_{mm}^{-1}$; thus we obtain the approximation $\hat{K} = K_{nm} K_{mm}^{-1} K_{mn}$ in agreement with eq. (8.4). Also, it can be shown that $E(C_{\text{opt}}) = \operatorname{tr}(K - \hat{K})$.

Smola and Schölkopf [2000] suggest a greedy algorithm to choose points to include into the active set so as to minimize the error criterion. As it takes

$\mathcal{O}(mn)$ operations to evaluate the change in E due to including one new datapoint (see exercise 8.7.2) it is infeasible to consider all members of set R for inclusion on each iteration; instead Smola and Schölkopf [2000] suggest finding the best point to include from a randomly chosen subset of set R on each iteration.

Recent work by Drineas and Mahoney [2005] analyzes a similar algorithm to the Nyström approximation, except that they use biased sampling with replacement (choosing column i of K with probability $\propto k_{ii}^2$) and a pseudoinverse of the inner $m \times m$ matrix. For this algorithm they are able to provide probabilistic bounds on the quality of the approximation. Earlier work by Frieze et al. [1998] had developed an approximation to the singular value decomposition (SVD) of a rectangular matrix using a weighted random subsampling of its rows and columns, and probabilistic error bounds. However, this is rather different from the Nyström approximation; see Drineas and Mahoney [2005, sec. 5.2] for details.

Fine and Scheinberg [2002] suggest an alternative low-rank approximation to K using the incomplete Cholesky factorization (see Golub and Van Loan [1989, sec. 10.3.2]). The idea here is that when computing the Cholesky decomposition of K pivots below a certain threshold are skipped.[2] If the number of pivots greater than the threshold is k the incomplete Cholesky factorization takes time $\mathcal{O}(nk^2)$.

8.2 Greedy Approximation

Many of the methods described below use an active set of training points of size m selected from the training set of size $n > m$. We assume that it is impossible to search for the optimal subset of size m due to combinatorics. The points in the active set could be selected randomly, but in general we might expect better performance if the points are selected greedily w.r.t. some criterion. In the statistics literature greedy approaches are also known as forward selection strategies.

A general recipe for greedy approximation is given in Algorithm 8.1. The algorithm starts with the active set I being empty, and the set R containing the indices of all training examples. On each iteration one index is selected from R and added to I. This is achieved by evaluating some criterion Δ and selecting the data point that optimizes this criterion. For some algorithms it can be too expensive to evaluate Δ on all points in R, so some working set $J \subset R$ can be chosen instead, usually at random from R.

Greedy selection methods have been used with the subset of regressors (SR), subset of datapoints (SD) and the projected process (PP) methods described below.

[2] As a technical detail, symmetric permutations of the rows and columns are required to stabilize the computations.

input: m, desired size of active set
Initialization $I = \emptyset$, $R = \{1, \ldots, n\}$
for $j := 1 \ldots m$ **do**
  Create working set $J \subseteq R$
  Compute Δ_j for all $j \in J$
  $i = \text{argmax}_{j \in J} \Delta_j$
  Update model to include data from example i
  $I \leftarrow I \cup \{i\}$, $R \leftarrow R \backslash \{i\}$
end for
return: I

Algorithm 8.1: General framework for greedy subset selection. Δ_j is the criterion function evaluated on data point j.

8.3 Approximations for GPR with Fixed Hyperparameters

We present six approximation schemes for GPR below, namely the subset of regressors (SR), the Nyström method, the subset of datapoints (SD), the projected process (PP) approximation, the Bayesian committee machine (BCM) and the iterative solution of linear systems. Section 8.3.7 provides a summary of these methods and a comparison of their performance on the SARCOS data which was introduced in section 2.5.

8.3.1 Subset of Regressors

Silverman [1985, sec. 6.1] showed that the *mean* GP predictor can be obtained from a finite-dimensional generalized linear regression model $f(\mathbf{x}_*) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_*, \mathbf{x}_i)$ with a prior $\boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, K^{-1})$. To see this we use the mean prediction for linear regression model in feature space given by eq. (2.11), i.e. $\bar{f}(\mathbf{x}_*) = \sigma_n^{-2} \boldsymbol{\phi}(\mathbf{x}_*)^\top A^{-1} \Phi \mathbf{y}$ with $A = \Sigma_p^{-1} + \sigma_n^{-2} \Phi \Phi^\top$. Setting $\boldsymbol{\phi}(\mathbf{x}_*) = \mathbf{k}(\mathbf{x}_*)$, $\Phi = \Phi^\top = K$ and $\Sigma_p^{-1} = K$ we obtain

$$\bar{f}(\mathbf{x}_*) = \sigma_n^{-2} \mathbf{k}^\top(\mathbf{x}_*)[\sigma_n^{-2} K(K + \sigma_n^2 I)]^{-1} K \mathbf{y} \tag{8.11}$$

$$= \mathbf{k}^\top(\mathbf{x}_*)(K + \sigma_n^2 I)^{-1} \mathbf{y}, \tag{8.12}$$

in agreement with eq. (2.25). Note, however, that the predictive (co)variance of this model is different from full GPR.

A simple approximation to this model is to consider only a subset of regressors, so that

$$f_{\text{SR}}(\mathbf{x}_*) = \sum_{i=1}^{m} \alpha_i k(\mathbf{x}_*, \mathbf{x}_i), \qquad \text{with} \qquad \boldsymbol{\alpha}_m \sim \mathcal{N}(\mathbf{0}, K_{mm}^{-1}). \tag{8.13}$$

Again using eq. (2.11) we obtain

$$\bar{f}_{\mathrm{SR}}(\mathbf{x}_*) \;=\; \mathbf{k}_m(\mathbf{x}_*)^\top (K_{mn}K_{nm} + \sigma_n^2 K_{mm})^{-1} K_{mn}\mathbf{y}, \tag{8.14}$$

$$\mathbb{V}[f_{\mathrm{SR}}(\mathbf{x}_*)] \;=\; \sigma_n^2 \mathbf{k}_m(\mathbf{x}_*)^\top (K_{mn}K_{nm} + \sigma_n^2 K_{mm})^{-1} \mathbf{k}_m(\mathbf{x}_*). \tag{8.15}$$

Thus the posterior mean for $\boldsymbol{\alpha}_m$ is given by

$$\bar{\boldsymbol{\alpha}}_m \;=\; (K_{mn}K_{nm} + \sigma_n^2 K_{mm})^{-1} K_{mn}\mathbf{y}. \tag{8.16}$$

This method has been proposed, for example, in Wahba [1990, chapter 7], and in Poggio and Girosi [1990, eq. 25] via the regularization framework. The name "subset of regressors" (SR) was suggested to us by G. Wahba. The computations for equations 8.14 and 8.15 take time $\mathcal{O}(m^2 n)$ to carry out the necessary matrix computations. After this the prediction of the mean for a new test point takes time $\mathcal{O}(m)$, and the predictive variance takes $\mathcal{O}(m^2)$.

SR marginal likelihood

Under the subset of regressors model we have $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \tilde{K})$ where $\tilde{K}$ is defined as in eq. (8.4). Thus the log marginal likelihood under this model is

$$\log p_{\mathrm{SR}}(\mathbf{y}|X) = -\frac{1}{2}\log|\tilde{K} + \sigma_n^2 I_n| - \frac{1}{2}\mathbf{y}^\top(\tilde{K} + \sigma_n^2 I_n)^{-1}\mathbf{y} - \frac{n}{2}\log(2\pi). \tag{8.17}$$

Notice that the covariance function defined by the SR model has the form $\tilde{k}(\mathbf{x}, \mathbf{x}') = \mathbf{k}(\mathbf{x})^\top K_{mm}^{-1}\mathbf{k}(\mathbf{x}')$, which is exactly the same as that from the Nyström approximation for the covariance function eq. (8.7). In fact if the covariance function $k(\mathbf{x}, \mathbf{x}')$ in the predictive mean and variance equations 2.25 and 2.26 is replaced systematically with $\tilde{k}(\mathbf{x}, \mathbf{x}')$ we obtain equations 8.14 and 8.15, as shown in Appendix 8.6.

If the kernel function decays to zero for $|\mathbf{x}| \to \infty$ for fixed $\mathbf{x}'$, then $\tilde{k}(\mathbf{x}, \mathbf{x})$ will be near zero when $\mathbf{x}$ is distant from points in the set I. This will be the case even when the kernel is stationary so that $k(\mathbf{x}, \mathbf{x})$ is independent of $\mathbf{x}$. Thus we might expect that using the approximate kernel will give poor predictions, especially underestimates of the predictive variance, when $\mathbf{x}$ is far from points in the set I.

An interesting idea suggested by Rasmussen and Quiñonero-Candela [2005] to mitigate this problem is to define the SR model with $m + 1$ basis functions, where the extra basis function is centered on the test point $\mathbf{x}_*$, so that $y_{\mathrm{SR}^*}(\mathbf{x}_*) = \sum_{i=1}^m \alpha_i k(\mathbf{x}_*, \mathbf{x}_i) + \alpha_* k(\mathbf{x}_*, \mathbf{x}_*)$. This model can then be used to make predictions, and it can be implemented efficiently using the partitioned matrix inverse equations A.11 and A.12. The effect of the extra basis function centered on $\mathbf{x}_*$ is to maintain predictive variance at the test point.

So far we have not said how the subset I should be chosen. One simple method is to choose it randomly from X, another is to run clustering on $\{\mathbf{x}_i\}_{i=1}^n$ to obtain centres. Alternatively, a number of greedy forward selection algorithms for I have been proposed. Luo and Wahba [1997] choose the next kernel so as to minimize the residual sum of squares (RSS) $|\mathbf{y} - K_{nm}\boldsymbol{\alpha}_m|^2$ after optimizing $\boldsymbol{\alpha}_m$. Smola and Bartlett [2001] take a similar approach, but choose as their criterion the quadratic form

$$\frac{1}{\sigma_n^2}|\mathbf{y} - K_{nm}\bar{\boldsymbol{\alpha}}_m|^2 + \bar{\boldsymbol{\alpha}}_m^\top K_{mm}\bar{\boldsymbol{\alpha}}_m \;=\; \mathbf{y}^\top(\tilde{K} + \sigma_n^2 I_n)^{-1}\mathbf{y}, \tag{8.18}$$

where the right hand side follows using eq. (8.16) and the matrix inversion lemma. Alternatively, Quiñonero-Candela [2004] suggests using the approximate log marginal likelihood $\log p_{\mathrm{SR}}(\mathbf{y}|X)$ (see eq. (8.17)) as the selection criterion. In fact the quadratic term from eq. (8.18) is one of the terms comprising $\log p_{\mathrm{SR}}(\mathbf{y}|X)$.

For all these suggestions the complexity of evaluating the criterion on a new example is $\mathcal{O}(mn)$, by making use of partitioned matrix equations. Thus it is likely to be too expensive to consider all points in R on each iteration, and we are likely to want to consider a smaller working set, as described in Algorithm 8.1.

Note that the SR model is obtained by selecting some subset of the datapoints of size m in a random or greedy manner. The relevance vector machine (RVM) described in section 6.6 has a similar flavour in that it automatically selects (in a greedy fashion) which datapoints to use in its expansion. However, note one important difference which is that the RVM uses a *diagonal* prior on the α's, while for the SR method we have $\boldsymbol{\alpha}_m \sim \mathcal{N}(\mathbf{0}, K_{mm}^{-1})$.

comparison with RVM

8.3.2 The Nyström Method

Williams and Seeger [2001] suggested approximating the GPR equations by replacing the matrix K by $\tilde{K}$ in the mean and variance prediction equations 2.25 and 2.26, and called this the *Nyström method* for approximate GPR. Notice that in this proposal the covariance function k is not systematically replaced by $\tilde{k}$, it is only occurrences of the matrix K that are replaced. As for the SR model the time complexity is $\mathcal{O}(m^2 n)$ to carry out the necessary matrix computations, and then $\mathcal{O}(n)$ for the predictive mean of a test point and $\mathcal{O}(mn)$ for the predictive variance.

Experimental evidence in Williams et al. [2002] suggests that for large m the SR and Nyström methods have similar performance, but for small m the Nyström method can be quite poor. Also the fact that k is not systematically replaced by $\tilde{k}$ means that embarrassments can occur like the approximated predictive variance being negative. For these reasons we do not recommend the Nyström method over the SR method. However, the Nyström method can be effective when λ_{m+1}, the $(m+1)$th eigenvalue of K, is much smaller than σ_n^2.

8.3.3 Subset of Datapoints

The subset of regressors method described above approximated the form of the predictive distribution, and particularly the predictive mean. Another simple approximation to the full-sample GP predictor is to keep the GP predictor, but only on a smaller subset of size m of the data. Although this is clearly wasteful of data, it can make sense if the predictions obtained with m points are sufficiently accurate for our needs.

Clearly it can make sense to select which points are taken into the active set I, and typically this is achieved by greedy algorithms. However, one has to be

wary of the amount of computation that is needed, especially if one considers each member of R at each iteration.

Lawrence et al. [2003] suggest choosing as the next point (or site) for inclusion into the active set the one that maximizes the *differential entropy score* $\Delta_j \triangleq H[p(f_j)] - H[p^{\text{new}}(f_j)]$, where $H[p(f_j)]$ is the entropy of the Gaussian at site $j \in R$ (which is a function of the variance at site j as the posterior is Gaussian, see eq. (A.20)), and $H[p^{\text{new}}(f_j)]$ is the entropy at this site once the observation at site j has been included. Let the posterior variance of f_j before inclusion be v_j. As $p(f_j|\mathbf{y}_I, y_j) \propto p(f_j|\mathbf{y}_I)\mathcal{N}(y_j|f_j, \sigma^2)$ we have $(v_j^{\text{new}})^{-1} = v_j^{-1} + \sigma^{-2}$. Using the fact that the entropy of a Gaussian with variance v is $\log(2\pi ev)/2$ we obtain

$$\Delta_j \;=\; \tfrac{1}{2}\log(1 + v_j/\sigma^2). \tag{8.19}$$

Δ_j is a monotonic function of v_j so that it is maximized by choosing the site with the largest variance. Lawrence et al. [2003] call their method the *informative vector machine* (IVM)

IVM

If coded naïvely the complexity of computing the variance at all sites in R on a single iteration is $\mathcal{O}(m^3 + (n-m)m^2)$ as we need to evaluate eq. (2.26) at each site (and the matrix inversion of $K_{mm} + \sigma_n^2 I$ can be done once in $\mathcal{O}(m^3)$ then stored). However, as we are incrementally growing the matrices K_{mm} and $K_{m(n-m)}$ in fact the cost is $\mathcal{O}(mn)$ per inclusion, leading to an overall complexity of $\mathcal{O}(m^2 n)$ when using a subset of size m. For example, once a site has been chosen for inclusion the matrix $K_{mm} + \sigma_n^2 I$ is grown by including an extra row and column. The inverse of this expanded matrix can be found using eq. (A.12) although it would be better practice numerically to use a Cholesky decomposition approach as described in Lawrence et al. [2003]. The scheme evaluates Δ_j over all $j \in R$ at each step to choose the inclusion site. This makes sense when m is small, but as it gets larger it can make sense to select candidate inclusion sites from a subset of R. Lawrence et al. [2003] call this the randomized greedy selection method and give further ideas on how to choose the subset.

The differential entropy score Δ_j is not the only criterion that can be used for site selection. For example the *information gain* criterion $\text{KL}(p^{\text{new}}(f_j)||p(f_j))$ can also be used (see Seeger et al., 2003). The use of greedy selection heuristics here is similar to the problem of *active learning*, see e.g. MacKay [1992c].

8.3.4 Projected Process Approximation

The SR method has the unattractive feature that it is based on a degenerate GP, the finite-dimensional model given in eq. (8.13). The SD method is a non-degenerate process model but it only makes use of m datapoints. The projected process (PP) approximation is also a non-degenerate process model but it can make use of all n datapoints. We call it a *projected* process approximation as it represents only $m < n$ latent function values, but computes a likelihood involving all n datapoints by projecting up the m latent points to n dimensions.

One problem with the basic GPR algorithm is the fact that the likelihood term requires us to have f-values for the n training points. However, say we only represent m of these values explicitly, and denote these as $\mathbf{f}_m$. Then the remaining f-values in R denoted $\mathbf{f}_{n-m}$ have a conditional distribution $p(\mathbf{f}_{n-m}|\mathbf{f}_m)$, the mean of which is given by $\mathbb{E}[\mathbf{f}_{n-m}|\mathbf{f}_m] = K_{(n-m)m}K_{mm}^{-1}\mathbf{f}_m$.[3] Say we replace the true likelihood term for the points in R by $\mathcal{N}(\mathbf{y}_{n-m}|\mathbb{E}[\mathbf{f}_{n-m}|\mathbf{f}_m], \sigma_n^2 I)$. Including also the likelihood contribution of the points in set I we have

$$q(\mathbf{y}|\mathbf{f}_m) \;=\; \mathcal{N}(\mathbf{y}|K_{nm}K_{mm}^{-1}\mathbf{f}_m, \sigma_n^2 I), \tag{8.20}$$

which can also be written as $q(\mathbf{y}|\mathbf{f}_m) = \mathcal{N}(\mathbf{y}|\mathbb{E}[\mathbf{f}|\mathbf{f}_m], \sigma_n^2 I)$. The key feature here is that we have absorbed the information in all n points of $\mathcal{D}$ into the m points in I.

The form of $q(\mathbf{y}|\mathbf{f}_m)$ in eq. (8.20) might seem rather arbitrary, but in fact it can be shown that if we consider minimizing $\mathrm{KL}(q(\mathbf{f}|\mathbf{y})||p(\mathbf{f}|\mathbf{y}))$, the KL-divergence between the approximating distribution $q(\mathbf{f}|\mathbf{y})$ and the true posterior $p(\mathbf{f}|\mathbf{y})$ over all q distributions of the form $q(\mathbf{f}|\mathbf{y}) \propto p(\mathbf{f})R(\mathbf{f}_m)$ where R is positive and depends on $\mathbf{f}_m$ only, this is the form we obtain. See Seeger [2003, Lemma 4.1 and sec. C.2.1] for detailed derivations, and also Csató [2002, sec. 3.3].

To make predictions we first have to compute the posterior distribution $q(\mathbf{f}_m|\mathbf{y})$. Define the shorthand $P = K_{mm}^{-1}K_{mn}$ so that $\mathbb{E}[\mathbf{f}|\mathbf{f}_m] = P^\top \mathbf{f}_m$. Then we have

$$q(\mathbf{y}|\mathbf{f}_m) \;\propto\; \exp\big(-\frac{1}{2\sigma_n^2}(\mathbf{y} - P^\top\mathbf{f}_m)^\top(\mathbf{y} - P^\top\mathbf{f}_m)\big). \tag{8.21}$$

Combining this with the prior $p(\mathbf{f}_m) \propto \exp(-\mathbf{f}_m^\top K_{mm}^{-1}\mathbf{f}_m/2)$ we obtain

$$q(\mathbf{f}_m|\mathbf{y}) \;\propto\; \exp\big(-\frac{1}{2}\mathbf{f}_m^\top(K_{mm}^{-1} + \frac{1}{\sigma_n^2}PP^\top)\mathbf{f}_m + \frac{1}{\sigma_n^2}\mathbf{y}^\top P^\top \mathbf{f}_m\big), \tag{8.22}$$

which can be recognized as a Gaussian $\mathcal{N}(\boldsymbol{\mu}, A)$ with

$$A^{-1} = \sigma_n^{-2}(\sigma_n^2 K_{mm}^{-1} + PP^\top) = \sigma_n^{-2}K_{mm}^{-1}(\sigma_n^2 K_{mm} + K_{mn}K_{nm})K_{mm}^{-1}, \tag{8.23}$$

$$\boldsymbol{\mu} = \sigma_n^{-2}AP\mathbf{y} = K_{mm}(\sigma_n^2 K_{mm} + K_{mn}K_{nm})^{-1}K_{mn}\mathbf{y}. \tag{8.24}$$

Thus the predictive mean is given by

$$\mathbb{E}_q[f(\mathbf{x}_*)] = \mathbf{k}_m(\mathbf{x}_*)^\top K_{mm}^{-1}\boldsymbol{\mu} \tag{8.25}$$

$$= \mathbf{k}_m(\mathbf{x}_*)^\top(\sigma_n^2 K_{mm} + K_{mn}K_{nm})^{-1}K_{mn}\mathbf{y}, \tag{8.26}$$

which turns out to be just the same as the predictive mean under the SR model, as given in eq. (8.14). However, the predictive variance is different. The argument is the same as in eq. (3.23) and yields

$$\begin{aligned}\mathbb{V}_q[f(\mathbf{x}_*)] \;&=\; k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_m(\mathbf{x}_*)^\top K_{mm}^{-1}\mathbf{k}_m(\mathbf{x}_*) \\ &\quad + \mathbf{k}_m(\mathbf{x}_*)^\top K_{mm}^{-1}\mathrm{cov}(\mathbf{f}_m|\mathbf{y})K_{mm}^{-1}\mathbf{k}_m(\mathbf{x}_*) \\ &= k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_m(\mathbf{x}_*)^\top K_{mm}^{-1}\mathbf{k}_m(\mathbf{x}_*) \\ &\quad + \sigma_n^2\mathbf{k}_m(\mathbf{x}_*)^\top(\sigma_n^2 K_{mm} + K_{mn}K_{nm})^{-1}\mathbf{k}_m(\mathbf{x}_*).\end{aligned} \tag{8.27}$$

[3] There is no a priori reason why the m points chosen have to be a subset of the n points in $\mathcal{D}$—they could be disjoint from the training set. However, for our derivations below we will consider them to be a subset.

Notice that predictive variance is the sum of the predictive variance under the SR model (last term in eq. (8.27)) plus $k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_m(\mathbf{x}_*)^\top K_{mm}^{-1} \mathbf{k}_m(\mathbf{x}_*)$ which is the predictive variance at $\mathbf{x}_*$ given $\mathbf{f}_m$. Thus eq. (8.27) is never smaller than the SR predictive variance and will become close to $k(\mathbf{x}_*, \mathbf{x}_*)$ when $\mathbf{x}_*$ is far away from the points in set I.

As for the SR model it takes time $\mathcal{O}(m^2 n)$ to carry out the necessary matrix computations. After this the prediction of the mean for a new test point takes time $\mathcal{O}(m)$, and the predictive variance takes $\mathcal{O}(m^2)$.

We have $q(\mathbf{y}|\mathbf{f}_m) = \mathcal{N}(\mathbf{y}|P^\top \mathbf{f}_m, \sigma_n^2 I)$ and $p(\mathbf{f}_m) = \mathcal{N}(\mathbf{0}, K_{mm})$. By integrating out $\mathbf{f}_m$ we find that $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, K + \sigma_n^2 I_n)$. Thus the marginal likelihood for the projected process approximation is the same as that for the SR model eq. (8.17).

Again the question of how to choose which points go into the set I arises. Csató and Opper [2002] present a method in which the training examples are presented sequentially (in an "on-line" fashion). Given the current active set I one can compute the novelty of a new input point; if this is large, then this point is added to I, otherwise the point is added to R. To be precise, the novelty of an input $\mathbf{x}$ is computed as $k(\mathbf{x}, \mathbf{x}) - \mathbf{k}_m(\mathbf{x})^\top K_{mm}^{-1} \mathbf{k}(\mathbf{x})$, which can be recognized as the predictive variance at $\mathbf{x}$ given non-noisy observations at the points in I. If the active set gets larger than some preset maximum size, then points can be deleted from I, as specified in section 3.3 of Csató and Opper [2002]. Later work by Csató et al. [2002] replaced the dependence of the algorithm described above on the input sequence by an expectation-propagation type algorithm (see section 3.6).

As an alternative method for selecting the active set, Seeger et al. [2003] suggest using a greedy subset selection method as per Algorithm 8.1. Computation of the information gain criterion after incorporating a new site takes $\mathcal{O}(mn)$ and is thus too expensive to use as a selection criterion. However, an approximation to the information gain can be computed cheaply (see Seeger et al. [2003, eq. 3] and Seeger [2003, sec. C.4.2] for further details) and this allows the greedy subset algorithm to be run on all points in R on each iteration.

8.3.5 Bayesian Committee Machine

Tresp [2000] introduced the Bayesian committee machine (BCM) as a way of speeding up Gaussian process regression. Let $\mathbf{f}_*$ be the vector of function values at the test locations. Under GPR we obtain a predictive Gaussian distribution for $p(\mathbf{f}_*|\mathcal{D})$. For the BCM we split the dataset into p parts $\mathcal{D}_1, \ldots, \mathcal{D}_p$ where $\mathcal{D}_i = (X_i, \mathbf{y}_i)$ and make the approximation that $p(\mathbf{y}_1, \ldots, \mathbf{y}_p|\mathbf{f}_*, X) \simeq \prod_{i=1}^p p(\mathbf{y}_i|\mathbf{f}_*, X_i)$. Under this approximation we have

$$q(\mathbf{f}_*|\mathcal{D}_1, \ldots, \mathcal{D}_p) \;\propto\; p(\mathbf{f}_*) \prod_{i=1}^p p(\mathbf{y}_i|\mathbf{f}_*, X_i) \;=\; c\, \frac{\prod_{i=1}^p p(\mathbf{f}_*|\mathcal{D}_i)}{p^{p-1}(\mathbf{f}_*)}, \tag{8.28}$$

where c is a normalization constant. Using the fact that the terms in the numerator and denomination are all Gaussian distributions over $\mathbf{f}_*$ it is easy

to show (see exercise 8.7.1) that the predictive mean and covariance for $\mathbf{f}_*$ are given by

$$\mathbb{E}_q[\mathbf{f}_*|\mathcal{D}] = [\mathrm{cov}_q(\mathbf{f}_*|\mathcal{D})] \sum_{i=1}^{p} [\mathrm{cov}(\mathbf{f}_*|\mathcal{D}_i)]^{-1} \mathbb{E}[\mathbf{f}_*|\mathcal{D}_i], \tag{8.29}$$

$$[\mathrm{cov}_q(\mathbf{f}_*|\mathcal{D})]^{-1} = -(p-1)K_{**}^{-1} + \sum_{i=1}^{p} [\mathrm{cov}(\mathbf{f}_*|\mathcal{D}_i)]^{-1}, \tag{8.30}$$

where K_{**} is the covariance matrix evaluated at the test points. Here $\mathbb{E}[\mathbf{f}_*|\mathcal{D}_i]$ and $\mathrm{cov}(\mathbf{f}_*|\mathcal{D}_i)$ are the mean and covariance of the predictions for $\mathbf{f}_*$ given $\mathcal{D}_i$, as given in eqs. (2.23) and (2.24). Note that eq. (8.29) has an interesting form in that the predictions from each part of the dataset are weighted by the inverse predictive covariance.

We are free to choose how to partition the dataset $\mathcal{D}$. This has two aspects, the number of partitions and the assignment of data points to the partitions. If we wish each partition to have size m, then $p = n/m$. Tresp [2000] used a random assignment of data points to partitions but Schwaighofer and Tresp [2003] recommend that clustering the data (e.g. with p-means clustering) can lead to improved performance. However, note that compared to the greedy schemes used above clustering does not make use of the target y values, only the inputs $\mathbf{x}$.

Although it is possible to make predictions for any number of test points n_*, this slows the method down as it involves the inversion of $n_* \times n_*$ matrices. Schwaighofer and Tresp [2003] recommend making test predictions on blocks of size m so that all matrices are of the same size. In this case the computational complexity of BCM is $\mathcal{O}(pm^3) = \mathcal{O}(m^2 n)$ for predicting m test points, or $\mathcal{O}(mn)$ per test point.

The BCM approach is *transductive* [Vapnik, 1995] rather than inductive, in the sense that the method computes a test-set dependent model making use of the test set input locations. Note also that if we wish to make a prediction at just one test point, it would be necessary to "hallucinate" some extra test points as eq. (8.28) generally becomes a better approximation as the number of test points increases.

8.3.6 Iterative Solution of Linear Systems

One straightforward method to speed up GP regression is to note that the linear system $(K + \sigma_n^2 I)\mathbf{v} = \mathbf{y}$ can be solved by an iterative method, for example conjugate gradients (CG). (See Golub and Van Loan [1989, sec. 10.2] for further details on the CG method.) Conjugate gradients gives the exact solution (ignoring round-off errors) if run for n iterations, but it will give an approximate solution if terminated earlier, say after k iterations, with time complexity $\mathcal{O}(kn^2)$. This method has been suggested by Wahba et al. [1995] (in the context of numerical weather prediction) and by Gibbs and MacKay [1997] (in the context of general GP regression). CG methods have also been used in the context

Method	m	SMSE	MSLL	mean runtime (s)
SD	256	0.0813 ± 0.0198	-1.4291 ± 0.0558	0.8
	512	0.0532 ± 0.0046	-1.5834 ± 0.0319	2.1
	1024	0.0398 ± 0.0036	-1.7149 ± 0.0293	6.5
	2048	0.0290 ± 0.0013	-1.8611 ± 0.0204	25.0
	4096	0.0200 ± 0.0008	-2.0241 ± 0.0151	100.7
SR	256	0.0351 ± 0.0036	-1.6088 ± 0.0984	11.0
	512	0.0259 ± 0.0014	-1.8185 ± 0.0357	27.0
	1024	0.0193 ± 0.0008	-1.9728 ± 0.0207	79.5
	2048	0.0150 ± 0.0005	-2.1126 ± 0.0185	284.8
	4096	0.0110 ± 0.0004	-2.2474 ± 0.0204	927.6
PP	256	0.0351 ± 0.0036	-1.6580 ± 0.0632	17.3
	512	0.0259 ± 0.0014	-1.7508 ± 0.0410	41.4
	1024	0.0193 ± 0.0008	-1.8625 ± 0.0417	95.1
	2048	0.0150 ± 0.0005	-1.9713 ± 0.0306	354.2
	4096	0.0110 ± 0.0004	-2.0940 ± 0.0226	964.5
BCM	256	0.0314 ± 0.0046	-1.7066 ± 0.0550	506.4
	512	0.0281 ± 0.0055	-1.7807 ± 0.0820	660.5
	1024	0.0180 ± 0.0010	-2.0081 ± 0.0321	1043.2
	2048	0.0136 ± 0.0007	-2.1364 ± 0.0266	1920.7

Table 8.1: Test results on the inverse dynamics problem for a number of different methods. Ten repetitions were used, the mean loss is shown ± one standard deviation.

of Laplace GPC, where linear systems are solved repeatedly to obtain the MAP solution $\tilde{\mathbf{f}}$ (see sections 3.4 and 3.5 for details).

One way that the CG method can be speeded up is by using an approximate rather than exact matrix-vector multiplication. For example, recent work by Yang et al. [2005] uses the improved fast Gauss transform for this purpose.

8.3.7 Comparison of Approximate GPR Methods

Above we have presented six approximation methods for GPR. Of these, we retain only those methods which scale linearly with n, so the iterative solution of linear systems must be discounted. Also we discount the Nyström approximation in preference to the SR method, leaving four alternatives: subset of regressors (SR), subset of data (SD), projected process (PP) and Bayesian committee machine (BCM).

Table 8.1 shows results of the four methods on the robot arm inverse dynamics problem described in section 2.5 which has $D = 21$ input variables, 44,484 training examples and 4,449 test examples. As in section 2.5 we used the squared exponential covariance function with a separate length-scale parameter for each of the 21 input dimensions.

Method	Storage	Initialization	Mean	Variance
SD	$\mathcal{O}(m^2)$	$\mathcal{O}(m^3)$	$\mathcal{O}(m)$	$\mathcal{O}(m^2)$
SR	$\mathcal{O}(mn)$	$\mathcal{O}(m^2n)$	$\mathcal{O}(m)$	$\mathcal{O}(m^2)$
PP	$\mathcal{O}(mn)$	$\mathcal{O}(m^2n)$	$\mathcal{O}(m)$	$\mathcal{O}(m^2)$
BCM	$\mathcal{O}(mn)$		$\mathcal{O}(mn)$	$\mathcal{O}(mn)$

Table 8.2: A comparison of the space and time complexity of the four methods using random selection of subsets. Initialization gives the time needed to carry out preliminary matrix computations before the test point $\mathbf{x}_*$ is known. Mean (resp. variance) refers to the time needed to compute the predictive mean (variance) at $\mathbf{x}_*$.

For the SD method a subset of the training data of size m was selected at random, and the hyperparameters were set by optimizing the marginal likelihood on this subset. As ARD was used, this involved the optimization of $D+2$ hyperparameters. This process was repeated 10 times, giving rise to the mean and standard deviation recorded in Table 8.1. For the SR, PP and BCM methods, the same subsets of the data and hyperparameter vectors were used as had been obtained from the SD experiments.[4] Note that the $m = 4096$ result is not available for BCM as this gave an out-of-memory error.

These experiments were conducted on a 2.0 GHz twin processor machine with 3.74 GB of RAM. The code for all four methods was written in MATLAB.[5]

A summary of the time complexities for the four methods are given in Table 8.2. Thus for a test set of size n_* and using full (mean and variance) predictions we find that the SD method has time complexity $\mathcal{O}(m^3) + \mathcal{O}(m^2n_*)$, for the SR and PP methods it is $\mathcal{O}(m^2n) + \mathcal{O}(m^2n_*)$, and for the BCM method it is $\mathcal{O}(mnn_*)$. Assuming that $n_* \geq m$ these reduce to $\mathcal{O}(m^2n_*)$, $\mathcal{O}(m^2n)$ and $\mathcal{O}(mnn_*)$ respectively. These complexities are in broad agreement with the timings in Table 8.1.

The results from Table 8.1 are plotted in Figure 8.1. As we would expect, the general trend is that as m increases the SMSE and MSLL scores decrease. Notice that it is well worth doing runs with small m so as to obtain a learning curve with respect to m; this helps in getting a feeling of how useful runs at large m will be. In terms of SMSE we see that (not surprisingly) SD is inferior to the other methods, which all have similar performance. For MSLL again SD is inferior to the other methods, although here the PP method is inferior to SR and BCM for larger m.

These results were obtained using a random selection of the active set. Some experiments were also carried out using active selection for the SD method (IVM) and for the SR method but these did not lead to significant improvements in performance. For BCM we also experimented with the use of p-means clustering instead of random assignment to partitions; again this did not lead to significant improvements in performance. Overall on this dataset our con-

[4] In the BCM case it was only the hyperparameters that were re-used; the data was partitioned randomly into blocks of size m.

[5] We thank Anton Schwaighofer for making his BCM code available to us.

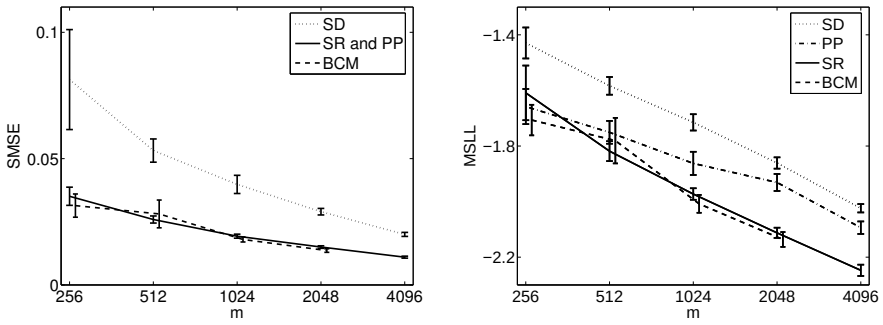

Figure 8.1: Panel(a): plot of SMSE against m. Panel(b) shows the MSLL for the four methods. The error bars denote one standard deviation. For clarity in both panels the BCM results are slightly displaced horizontally w.r.t. the SR results.

clusion is that for fixed m SR is the method of choice, as BCM has longer running times for similar performance. However, notice that if we compare on runtime, then SD for $m = 4096$ is competitive with the SR and BCM results for $m = 1024$ on both time and performance.

In the above experiments the hyperparameters for all methods were set by optimizing the marginal likelihood of the SD model of size m. This means that we get a direct comparison of the different methods using the same hyperparameters and subsets. However, one could alternatively optimize the (approximate) marginal likelihood for each method (see section 8.5) and then compare results. Notice that the hyperparameters which optimize the approximate marginal likelihood may depend on the method. For example Figure 5.3(b) shows that the maximum in the marginal likelihood occurs at shorter length-scales as the amount of data increases. This effect has also been observed by V. Tresp and A. Schwaighofer (pers. comm., 2004) when comparing the SD marginal likelihood eq. (8.31) with the full marginal likelihood computed on all n datapoints eq. (5.8).

Schwaighofer and Tresp [2003] report some experimental comparisons between the BCM method and some other approximation methods for a number of synthetic regression problems. In these experiments they optimized the kernel hyperparameters for each method separately. Their results are that for fixed m BCM performs as well as or better than the other methods. However, these results depend on factors such as the noise level in the data generating process; they report (pers. comm., 2005) that for relatively large noise levels BCM no longer displays an advantage. Based on the evidence currently available we are unable to provide firm recommendations for one approximation method over another; further research is required to understand the factors that affect performance.

8.4 Approximations for GPC with Fixed Hyperparameters

The approximation methods for GPC are similar to those for GPR, but need to deal with the non-Gaussian likelihood as well, either by using the Laplace approximation, see section 3.4, or expectation propagation (EP), see section 3.6. In this section we focus mainly on binary classification tasks, although some of the methods can also be extended to the multi-class case.

For the subset of regressors (SR) method we again use the model $f_{\mathrm{SR}}(\mathbf{x}_*) = \sum_{i=1}^{m} \alpha_i k(\mathbf{x}_*, \mathbf{x}_i)$ with $\boldsymbol{\alpha}_m \sim \mathcal{N}(\mathbf{0}, K_{mm}^{-1})$. The likelihood is non-Gaussian but the optimization problem to find the MAP value of $\boldsymbol{\alpha}_m$ is convex and can be obtained using a Newton iteration. Using the MAP value $\hat{\boldsymbol{\alpha}}_m$ and the Hessian at this point we obtain a predictive mean and variance for $f(\mathbf{x}_*)$ which can be fed through the sigmoid function to yield probabilistic predictions. As usual the question of how to choose a subset of points arises; Lin et al. [2000] select these using a clustering method, while Zhu and Hastie [2002] propose a forward selection strategy.

The subset of datapoints (SD) method for GPC was proposed in Lawrence et al. [2003], using an EP-style approximation of the posterior, and the differential entropy score (see section 8.3.3) to select new sites for inclusion. Note that the EP approximation lends itself very naturally to sparsification: a sparse model results when some *site precisions* (see eq. (3.51)) are zero, making the corresponding likelihood term vanish. A computational gain can thus be achieved by ignoring likelihood terms whose site precisions are very small.

The projected process (PP) approximation can also be used with non-Gaussian likelihoods. Csató and Opper [2002] present an "online" method where the examples are processed sequentially, while Csató et al. [2002] give an expectation-propagation type algorithm where multiple sweeps through the training data are permitted.

The Bayesian committee machine (BCM) has also been generalized to deal with non-Gaussian likelihoods in Tresp [2000]. As in the GPR case the dataset is broken up into blocks, but now approximate inference is carried out using the Laplace approximation in each block to yield an approximate predictive mean $\mathbb{E}_q[\mathbf{f}_*|\mathcal{D}_i]$ and approximate predictive covariance $\mathrm{cov}_q(\mathbf{f}_*|\mathcal{D}_i)$. These predictions are then combined as before using equations 8.29 and 8.30.

8.5 Approximating the Marginal Likelihood and its Derivatives *

We consider approximations first for GP regression, and then for GP classification. For GPR, both the SR and PP methods give rise to the same approximate marginal likelihood as given in eq. (8.17). For the SD method, a very simple

approximation (ignoring the datapoints not in the active set) is given by

$$\log p_{\mathrm{SD}}(\mathbf{y}_m|X_m) \;=\; -\tfrac{1}{2}\log|K_{mm}+\sigma^2 I| - \tfrac{1}{2}\mathbf{y}_m^\top (K_{mm}+\sigma^2 I)^{-1}\mathbf{y}_m - \tfrac{m}{2}\log(2\pi), \tag{8.31}$$

where $\mathbf{y}_m$ is the subvector of $\mathbf{y}$ corresponding to the active set; eq. (8.31) is simply the log marginal likelihood under the model $\mathbf{y}_m \sim \mathcal{N}(\mathbf{0}, K_{mm}+\sigma^2 I)$.

For the BCM, a simple approach would be to sum eq. (8.31) evaluated on each partition of the dataset. This ignores interactions between the partitions. Tresp and Schwaighofer (pers. comm., 2004) have suggested a more sophisticated BCM-based method which approximately takes these interactions into account.

For GPC under the SR approximation, one can simply use the Laplace or EP approximations on the finite-dimensional model. For SD one can again ignore all datapoints not in the active set and compute an approximation to $\log p(\mathbf{y}_m|X_m)$ using either Laplace or EP. For the projected process (PP) method, Seeger [2003, p. 162] suggests the following lower bound

$$\begin{aligned}
\log p(\mathbf{y}|X) \;&=\; \log \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f})\,d\mathbf{f} \;=\; \log \int q(\mathbf{f})\frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f})}{q(\mathbf{f})}\,d\mathbf{f} \\
&\geq \int q(\mathbf{f})\log\Big(\frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f})}{q(\mathbf{f})}\Big)\,d\mathbf{f} \qquad (8.32)\\
&= \int q(\mathbf{f})\log q(\mathbf{y}|\mathbf{f})\,d\mathbf{f} - \mathrm{KL}(q(\mathbf{f})||p(\mathbf{f})) \\
&= \sum_{i=1}^{n}\int q(f_i)\log p(y_i|f_i)\,df_i - \mathrm{KL}(q(\mathbf{f}_m)||p(\mathbf{f}_m)),
\end{aligned}$$

where $q(\mathbf{f})$ is a shorthand for $q(\mathbf{f}|\mathbf{y})$ and eq. (8.32) follows from the equation on the previous line using Jensen's inequality. The KL divergence term can be readily evaluated using eq. (A.23), and the one-dimensional integrals can be tackled using numerical quadrature.

We are not aware of work on extending the BCM approximations to the marginal likelihood to GPC.

Given the various approximations to the marginal likelihood mentioned above, we may also want to compute derivatives in order to optimize it. Clearly it will make sense to keep the active set fixed during the optimization, although note that this clashes with the fact that methods that select the active set might choose a different set as the covariance function parameters $\boldsymbol{\theta}$ change. For the classification case the derivatives can be quite complex due to the fact that site parameters (such as the MAP values $\hat{\mathbf{f}}$, see section 3.4.1) change as $\boldsymbol{\theta}$ changes. (We have already seen an example of this in section 5.5 for the non-sparse Laplace approximation.) Seeger [2003, sec. 4.8] describes some experiments comparing SD and PP methods for the optimization of the marginal likelihood on both regression and classification problems.

8.6 Appendix: Equivalence of SR and GPR using the Nyström Approximate Kernel *

In section 8.3 we derived the subset of regressors predictors for the mean and variance, as given in equations 8.14 and 8.15. The aim of this appendix is to show that these are equivalent to the predictors that are obtained by replacing $k(\mathbf{x}, \mathbf{x}')$ systematically with $\tilde{k}(\mathbf{x}, \mathbf{x}')$ in the GPR prediction equations 2.25 and 2.26.

First for the mean. The GPR predictor is $\mathbb{E}[f(\mathbf{x}_*)] = \mathbf{k}(\mathbf{x}_*)^\top (K + \sigma_n^2 I)^{-1}\mathbf{y}$. Replacing all occurrences of $k(\mathbf{x}, \mathbf{x}')$ with $\tilde{k}(\mathbf{x}, \mathbf{x}')$ we obtain

$$\begin{aligned}
\mathbb{E}[\tilde{f}(\mathbf{x}_*)] &= \tilde{\mathbf{k}}(\mathbf{x}_*)^\top (\tilde{K} + \sigma_n^2 I)^{-1}\mathbf{y} && (8.33)\\
&= \mathbf{k}_m(\mathbf{x}_*)^\top K_{mm}^{-1} K_{mn} (K_{nm} K_{mm}^{-1} K_{mn} + \sigma_n^2 I)^{-1}\mathbf{y} && (8.34)\\
&= \sigma_n^{-2} \mathbf{k}_m(\mathbf{x}_*)^\top K_{mm}^{-1} K_{mn} \left[I_n - K_{nm} Q^{-1} K_{mn}\right] \mathbf{y} && (8.35)\\
&= \sigma_n^{-2} \mathbf{k}_m(\mathbf{x}_*)^\top K_{mm}^{-1} \left[I_m - K_{mn} K_{nm} Q^{-1}\right] K_{mn}\mathbf{y} && (8.36)\\
&= \sigma_n^{-2} \mathbf{k}_m(\mathbf{x}_*)^\top K_{mm}^{-1} \left[\sigma_n^2 K_{mm} Q^{-1}\right] K_{mn}\mathbf{y} && (8.37)\\
&= \mathbf{k}_m(\mathbf{x}_*)^\top Q^{-1} K_{mn}\mathbf{y}, && (8.38)
\end{aligned}$$

where $Q = \sigma_n^2 K_{mm} + K_{mn}K_{nm}$, which agrees with eq. (8.14). Equation (8.35) follows from eq. (8.34) by use of the matrix inversion lemma eq. (A.9) and eq. (8.38) follows from eq. (8.36) using $I_m = (\sigma_n^2 K_{mm} + K_{mn}K_{nm})Q^{-1}$. For the predictive variance we have

$$\begin{aligned}
\mathbb{V}[\tilde{f}_*] &= \tilde{k}(\mathbf{x}_*, \mathbf{x}_*) - \tilde{\mathbf{k}}(\mathbf{x}_*)^\top (\tilde{K} + \sigma_n^2 I)^{-1} \tilde{\mathbf{k}}(\mathbf{x}_*) && (8.39)\\
&= \mathbf{k}_m(\mathbf{x}_*)^\top K_{mm}^{-1} \mathbf{k}_m(\mathbf{x}_*) - && (8.40)\\
&\quad \mathbf{k}_m(\mathbf{x}_*)^\top K_{mm}^{-1} K_{mn} (K_{nm} K_{mm}^{-1} K_{mn} + \sigma_n^2 I)^{-1} K_{nm} K_{mm}^{-1} \mathbf{k}_m(\mathbf{x}_*)\\
&= \mathbf{k}_m(\mathbf{x}_*)^\top K_{mm}^{-1} \mathbf{k}_m(\mathbf{x}_*) - \mathbf{k}_m(\mathbf{x}_*)^\top Q^{-1} K_{mn} K_{nm} K_{mm}^{-1} \mathbf{k}_m(\mathbf{x}_*) && (8.41)\\
&= \mathbf{k}_m(\mathbf{x}_*)^\top \left[I_m - Q^{-1} K_{mn} K_{nm}\right] K_{mm}^{-1} \mathbf{k}_m(\mathbf{x}_*) && (8.42)\\
&= \mathbf{k}_m(\mathbf{x}_*)^\top Q^{-1} \sigma_n^2 K_{mm} K_{mm}^{-1} \mathbf{k}_m(\mathbf{x}_*) && (8.43)\\
&= \sigma_n^2 \mathbf{k}_m(\mathbf{x}_*)^\top Q^{-1} \mathbf{k}_m(\mathbf{x}_*), && (8.44)
\end{aligned}$$

in agreement with eq. (8.15). The step between eqs. (8.40) and (8.41) is obtained from eqs. (8.34) and (8.38) above, and eq. (8.43) follows from eq. (8.42) using $I_m = (\sigma_n^2 K_{mm} + K_{mn}K_{nm})Q^{-1}$.

8.7 Exercises

1. Verify that the mean and covariance of the BCM predictions (equations 8.29 and 8.30) are correct. If you are stuck, see Tresp [2000] for details.

2. Using eq. (8.10) and the fact that $C_{\text{opt}} = K_{nm}K_{mm}^{-1}$ show that $E(C_{\text{opt}}) = \text{tr}(K - \tilde{K})$, where $\tilde{K} = K_{nm}K_{mm}^{-1}K_{mn}$. Now consider adding one datapoint into set I, so that K_{mm} grows to $K_{(m+1)(m+1)}$. Using eq. (A.12)

show that the change in E due to adding the extra datapoint can be computed in time $\mathcal{O}(mn)$. If you need help, see Schölkopf and Smola [2002, sec. 10.2.2] for further details.