Fast large scale Gaussian process regression using approximate matrix-vector products

Vikas C. Raykar and Ramani Duraiswami
Department of Computer Science and Institute for advanced computer studies
University of Maryland, College Park, MD, USA
{vikas,ramani}@umiacs.umd.edu
www.umiacs.umd.edu/~vikas

Topic: learning algorithms

Preference: oral

Gaussian processes (GP) allow the treatment of non-linear non-parametric regression problems in a Bayesian framework [6]. Unfortunately its non-parametric nature causes computational problems for large data sets, due to an unfavorable $\mathcal{O}(N^3)$ time and $\mathcal{O}(N^2)$ memory scaling for training.

The key computational task involves inversion of an $N \times N$ covariance matrix $\mathbf{K} + \sigma^2\mathbf{I}$, where $[\mathbf{K}]_{ij} = K(x_i, x_j)$, K is the covariance function of the GP, and σ^2 is the noise variance. Direct computation of the inverse requires $\mathcal{O}(N^3)$ operations and $\mathcal{O}(N^2)$ storage, which is impractical even for problems of moderate size (typically a few thousands).

An important subfield of work in GP has attempted to bring this scaling down to $\mathcal{O}\left(m^2N\right)$ by making sparse approximations of size m to the full GP where $m \ll N$ (see [5] for a survey). Most of these methods are based on using a representative subset of the training examples of size m. Different schemes specify different strategies to effectively choose the subset. While these methods often work well, there is no guarantee on the quality of the GP that results from the sparse approximation. These methods can be considered to provide *exact inference in an approximate model.*

An alternate class of methods is to use iterative methods like conjugate-gradient (CG) for the solution of linear system [3], instead of directly computing the inverse. The use of iterative methods, as first suggested by [4], can reduce the cost to $\mathcal{O}(kN^2)$ where k are the number of iterations. The iterative method generates a sequence of approximate solutions at each step, which converge to the true solution. While iterating till $k = N$ gives the exact solution a practical CG scheme iterates till a specified tolerance. In contrast to the subset of the dataset methods these methods can be considered to provide *approximate inference in an exact model.*

The core computational step contributing to the quadratic complexity in each CG iteration involves the multiplication of the matrix $\mathbf{K}$ with some vector. We consider the use of ϵ-exact matrix-vector product algorithms to reduce the computational complexity to $\mathcal{O}(N)$. There are at least three methods proposed to accelerate the matrix-vector product using approximation ideas: the dual-tree method [1], the fast Gauss transform (FGT) [2], and the improved fast Gauss transform (IFGT) [7], which have their own areas of applicability and performance characteristics. These methods claim to provide the matrix-vector product with a guaranteed accuracy ϵ, and achieve $\mathcal{O}(N \log N)$ or $\mathcal{O}(N)$ performance at fixed ϵ in both time and memory.

An important question when using these methods is the influence of the approximate matrix-vector product on the convergence of the iterative method. Obviously these methods converge at machine precision. However, the accuracy necessary to guarantee convergence must be studied. Generally previous papers [10, 8] choose ϵ to a convenient small value such as 10^{-3} or 10^{-6} based on the application. We use a more theoretical approach and base our results on the theory of inexact Krylov subspace methods [9]. We show that *matrix-vector product may be performed in an increasingly inexact manner* as the iteration progresses and still allow convergence (see Figure 1).

We test our ideas using the IFGT and the dual-tree methods. We demonstrate the speedup achieved on large data sets using the negative squared exponential (Gaussian) covariance function. Our experiments indicated that for low dimensional data ($d \leq 8$) the IFGT gives substantial speedups. For the hyperparameters chosen the dual-tree methods were unable to give good speedups.

Unlike methods which rely on choosing a subset of the dataset we use all the available points and still achieve $\mathcal{O}(N)$ complexity. As an added advantage, for prediction of the mean the computational complexity

Table 1: *The dominant computational and space complexities.* We have N training points and M test points in d dimensions. k is the number of iterations required by the conjugate gradient procedure to converge to a specified tolerance. The memory requirements are for the case where the Gram matrix is constructed on the fly at each iteration. For the fast matrix-vector product algorithms, the constant $\mathcal{D}$ grows with d depending on the type of the algorithm used.

	Direct Method		*Conjugate gradient*		*Conjugate gradient+Fast MVM*	
	Time	Space	Time	Space	Time	Space
Training	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$	$\mathcal{O}(kdN^2)$	$\mathcal{O}(dN)$	$\mathcal{O}(k\mathcal{D}N)$	$\mathcal{O}(dN+\mathcal{D})$
Prediction	$\mathcal{O}(dMN)$	$\mathcal{O}(dM+dN)$			$\mathcal{O}(\mathcal{D}M+\mathcal{D}N)$	$\mathcal{O}(dM+dN+\mathcal{D})$
Uncertainty	$\mathcal{O}(MN^2)$		$\mathcal{O}(kdMN^2)$	$\mathcal{O}(dM+dN)$	$\mathcal{O}(k\mathcal{D}MN)$	$\mathcal{O}(dM+dN+\mathcal{D})$

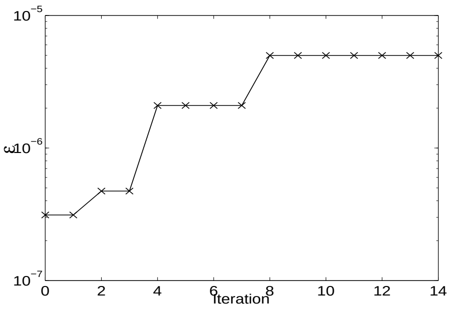

Figure 1: The accuracy required for fast matrix-vector product at each iteration for a sample 1D regression problem.

is reduced from $\mathcal{O}(N)$ to $\mathcal{O}(1)$.

We have also used Gaussian process regression for fast implicit surface fitting from the point cloud data (see Figure 2). This is an application where it is important to use all the available data to get a good surface representation.

While the scope of this paper is to speed up the original GPR it should be noted that methods which use a subset of the data can also be further speeded up using these algorithms. This is because even these methods require matrix-vector products to be taken with a smaller subset of the data.

We were unable to get good speedups for high dimensional datasets like SARCOS (a 21 dimensional robot arm dataset) using the IFGT. However the subset of data methods can be used with a higher dimensional data set such as SARCOS.

Table 1 compares the computational and space complexities for different stages of Gaussian process regression using different methods.

References

[1] A. G. Gray and A. W. Moore. Nonparametric density estimation: Toward computational tractability. In *SIAM Int. conference on Data Mining*, 2003.

[2] L. Greengard and J. Strain. The fast Gauss transform. *SIAM J. of Sci. and Stat. Computing*, 12(1):79–94, 1991.

[3] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations.* SIAM, 1995.

[4] D. MacKay and M. N. Gibbs. Efficient implementation of Gaussian processes. unpublished, 1997.

[5] J. Quiñonero Candela and C. E. Rasmussen. A unifying view of sparse approximate gaussian process regression. *J. of Machine Learning Research*, 6:1935–1959, 2005.

[6] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning.* The MIT Press, 2006.

[7] V. C. Raykar, C. Yang, R. Duraiswami, and N. Gumerov. Fast computation of sums of Gaussians in high dimensions. Technical Report CS-TR-4767, Dep. of Comp. Science, Univ. of Maryland, CollegePark, 2005.

[8] Y. Shen, A. Ng, and M. Seeger. Fast Gaussian process regression using KD-trees. In NIPS 2006.

[9] V. Simoncini and D. B. Szyld. Theory of inexact Krylov subspace methods and applications to scientific computing. *SIAM J. Sci. Comput.*, 25(2):454–477, 2004.

[10] C. Yang, R. Duraiswami, and L. Davis. Efficient kernel machines using the improved fast Gauss transform. In NIPS 2005

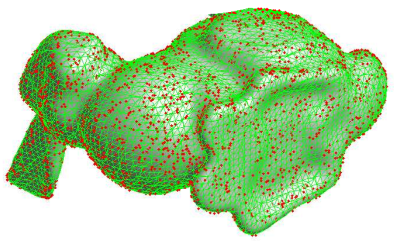

Figure 2: The isosurface extracted using the function learnt by Gaussian Process regression. The point cloud data is also shown. It took 6 minutes to learn this implicit surface form 10,000 surface points.