Fast Kernel Matrix-Vector Multiplication with Application to Gaussian Process Learning

Alexander Gray

February 2004
CMU-CS-04-110

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

A number of core computational problems in machine learning, both old and new, can be cast as a matrix-vector multiplication between a kernel matrix or class-probability matrix and a vector of weights. This arises prominently, for example, in the kernel estimation methods of nonparametric statistics, many common probabilistic graphical models, and the more recent kernel machines. After highlighting the existence of this computational problem in several well-known machine learning methods, we focus on a solution for one specific example for clarity, Gaussian process (GP) prediction – one whose applicability has been particularly hindered by this computational barrier. We demonstrate the application of a recent N-body approach developed specifically for statistical problems, employing adaptive computational geometry and finite-difference approximation. This core algorithm reduces the $O(N^2)$ matrix-vector multiplications within GP learning to $O(N)$, making the resulting overall learning algorithm $O(N)$. GP learning for N = 1 million points is demonstrated.

The author was supported by the NASA Graduate Research Fellowship.

Keywords: Gaussian processes, kernel matrix, matrix-vector multiplication, fast algorithms

1 Kernel Matrix-Vector Multiplications in Learning

A *kernel matrix* Φ contains the kernel interaction of each point $\underline{x}_q$ in a query (test) dataset $\underline{X}_Q$ (having size N_Q) with each point from a reference (training) dataset $\underline{X}_{\mathcal{R}}$ (having size $N_{\mathcal{R}}$), where the *kernel function* $K()$ often has some scale parameter σ (the 'bandwidth'). Often the 'kernel function' is actually a probability density function, such as the Gaussian. In such cases Φ is typically a *class probability matrix*. The query and reference set can be the same set. Often the core computational cost of a statistical method boils down to a multiplcation of this matrix Φ with some vector of weights w. For example, in the weighted form of *kernel density estimation*, the density estimate at the q^{th} test point $\underline{x}_q$ is

$$\hat{p}(\underline{x}_q) = \frac{1}{N_{\mathcal{R}}} \sum_{r=1}^{N_{\mathcal{R}}} w_r \frac{1}{V_{Dh}} K\left(\frac{\|\underline{x}_q - \underline{x}_r\|}{h}\right) \tag{1}$$

where D is the dimensionality of the data and $V_{Dh} = \int_{-\infty}^{\infty} K_h(z)dz$, a normalizing constant depending on D and h. This can be written in matrix notation as

$$\hat{p} = \frac{1}{N_{\mathcal{R}} V_{Dh}} \Phi w \tag{2}$$

where w is the vector of $N_{\mathcal{R}}$ weights and Φ is the N_Q by $N_{\mathcal{R}}$ kernel matrix.

Many fundamental statistical methods contain such matrix-vector multiplications, including Nadaraya-Watson regression, EM iterations for mixture models, radial-basis function networks, Gaussian processes, and a number of the more recent *kernel methods* [1], including support vector machines. More discussion of some important special cases appears at the end of this paper.

Generalized N-body problems. The main idea here is to observe that, for certain common kinds of kernel functions, such kernel matrix-vector multiplications contain geometric structure which is not exploited when the operation is viewed linear-algebraically. We instead approach such operations as N-body problems, generalizing from the computational physics problem of efficiently computing all pairwise potentials of a set of points to a much larger class of problems which can be defined abstractly in group-theoretic terms [6], unifying a number of important previously unrelated problems. The kind of summation discussed here is one sub-class of these problems.

2 Gaussian Process Learning and Prediction

In this paper we will consider Gaussian processes, because the appearance of such N-body problems occurs in one of the least straightforward ways in this case, unlike many of the other statistical hosts of this kind of problem. We hope that demonstrating a solution for this difficult case will provide inspiration and insight for other problems.

Gaussian processes represent both an old and a new statistical technique. They are classical nonparametric techniques from geostatistics, originally called *kriging* [13], which have attracted recent interest in machine learning in their Bayesian form [12].

Two matrix-vector multiplications. Despite their appeal, their relevance has been limited to small datasets by the fact that prediction entails inversion of an N by N matrix, where N is the number of training data. When performed directly, matrix inversion has $O(N^3)$ cost. However, Gibbs and MacKay [5] showed how an iterative linear-algebraic method proposed by Skilling [17], related to the Lanczos algorithm and other conjugate gradient methods [15], can be applied to this problem, reducing the cost of the inversion to $O(N^2 I)$, where I is the number of iterations (which ranges up to N, for exact inversion), and the $O(N^2)$ factor arises due to the cost of multiplying an N by N matrix by a vector of length N, which occurs in each iteration.

In addition, there is a second major kernel matrix-vector multiplication occurring in GP learning. If prediction is to be performed for M query (test) points, a second kernel matrix Φ' must be computed, consisting of all pairwise kernel interactions between each query point and each training point. This M by N matrix multiplies the vector $\Phi^{-1}t$. The main focus of this paper is to show how these matrix-vector multiplications, which control the complexity of the learning algorithm as a whole, can be executed in $O(N)$ time, thus reducing the cost of GP learning to $O(NI)$.

2.1 Skilling's Method

We now briefly review the Skilling method for GP prediction, following the description given in [5]. The main computation of interest is the calculation of $\Phi^{-1}u$, where Φ is the kernel matrix and w is a vector. Skilling's recurrence consists of intermediate vectors g_i and h_i and scalars λ_i and γ_i at each iteration i, resulting in a new approximation y_i to $\Phi^{-1}u$. It is initialized by $g_0 = w$, $h_0 = g_0$, $y_0 = 0$, and proceeds with the following updates:

$$\lambda_i = \frac{g_i^T g_i}{g_i^T \Phi h_i} \tag{3}$$

$$g_{i+1} = g_i - \lambda_i \Phi h_i \tag{4}$$

$$\gamma_i = \frac{g_{i+1} g_{i+1}}{g_i g_i} \tag{5}$$

$$h_{i+1} = g_{i+1} + \gamma_i h_i \tag{6}$$

$$y_{i+1} = y_i + \lambda_i h_i \tag{7}$$

The multiplications Φh_i are our main concern within this procedure.

While we have not shown the recurrence for the lower and upper bounds on the approximation that are available in Skilling's method (which closely mirrors the above recurrence), it is the case that all of the computations needed in training/testing (prediction estimate and lower/upper bounds) are governed by the cost of these two instances of matrix-vector multiplication.

3 N-Body Approach

Despite the existence of well-known N-body solvers for problems in computational physics, these approaches are not generally applicable to N-body problems in statistical inference.[1] In a series of papers [8, 9, 6] we developed an N-body approach specifically for the more general setting posed by statistical N-body problems (higher dimension, highly non-uniform data), taking a different perspective. State-of-the-art N-body solutions have two major components – a space-partitioning scheme and a function approximation scheme. We reverse the emphasis of standard methods, placing more sophistication in the data structure by drawing from computational geometry, and using a lightweight finite-difference approximation scheme which accomodates a wide variety of kernel functions in arbitrary dimension.

3.1 Adaptive space-partitioning trees.

kd-trees [4] are simple yet effective space-partitioning trees, where each node in the tree defines a distinct region in the data space using a bounding hyperrectangle (which is maximally tight in each coordinate) and each split is made along a single coordinate, the one having largest variance [4]. The tree is often built all the way down to some small predefined number of points ρ at the leaves.

We can in fact improve the data-adaptivity of *kd*-trees by removing their axis-parallel restriction. If we instead partition by selecting centroid *points* to define the left and right sets of the partitioning (according to the Voronoi diagram induced by the two points), we can escape this representation problem as well, effectively replacing axis-parallel hyperplane separators with arbitrarily-aligned hyperplanes. The result is called a *ball-tree* [14] or 'metric tree' [3]. As with *kd*-trees we augment the structure with problem-dependent cached sufficient statistics.

Distance bounds. We have $O(D)$ bounds on the distance from a point $\underline{x}_q$ to any point within the node R due to the triangle inequality:

$$\min_r \|\underline{x}_q - \underline{x}_r\|^2 \geq \max\left\{(\|\underline{x}_q - \underline{c}_R\| - s_R)^2, 0\right\} \tag{8}$$

[1]The multipole methods [10] are known to suffer from the large constants inherent in the multipole expansion approximation strategy, which are exponential in the explicit dimension D of the data. The Barnes-Hut method [2] is more flexible but has poorer complexity. Both methods, designed for dynamical simulation problems, rely on simple hierarchical grids which are insensitive to the data distribution and are thus less effective computationally than the adaptive structures which are common in computational geometry.

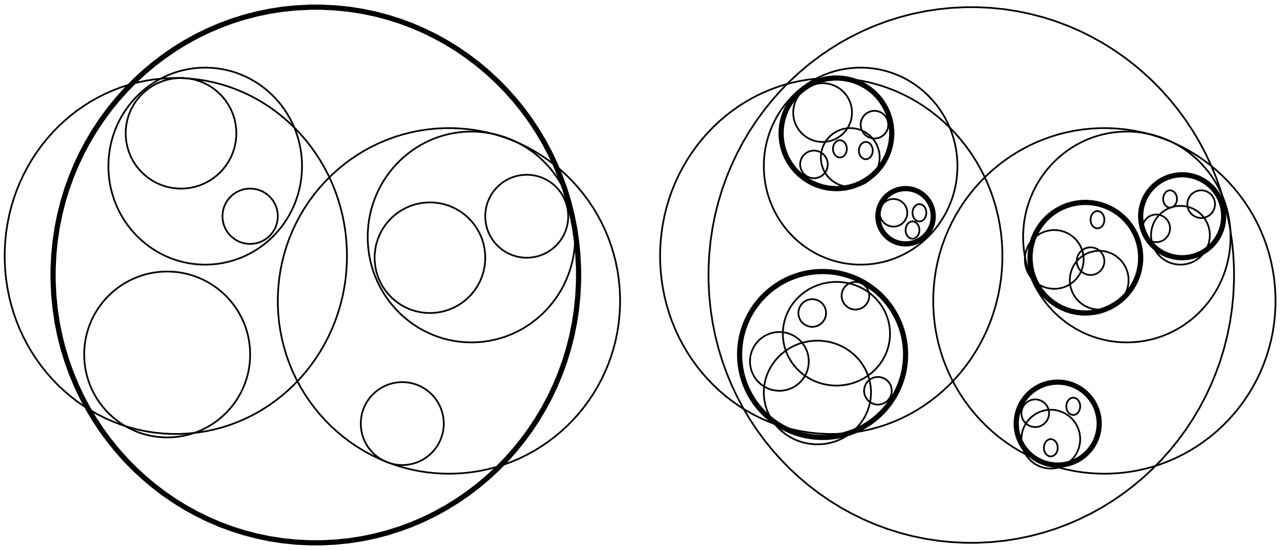

Figure 1: Levels of a ball-tree. The bold circles indicate the root (left figure) and the 'middle' layer of nodes occurring in the anchors construction algorithm (right figure).

$$\max_r \|\underline{x}_q - \underline{x}_r\|^2 \quad \leq \quad (\|\underline{x}_q - \underline{c}_R\| + s_R)^2$$

where $\underline{c}_R$ is the centroid of R and s_R is its radius, the distance of the farthest point in R to $\underline{c}_R$.

Dual-tree traversal. Our algorithm will proceed by comparing nodes to nodes, forming a traversal over two trees, one built on the query set $\underline{X}_\mathcal{Q}$ and one on the reference set $\underline{X}_\mathcal{R}$. At all times bounds are maintained on the sum to be computed for each query, denoted ϕ, beginning with maximally pessimistic estimates at the start of the traversal. Analogous distance bounds can be written for the closest and farthest distances between two nodes.

3.2 Quadrature, interpolation, and finite differences.

We can regard our problem as a kind of *quadrature*, or numerical integration problem, where the problem is to find $\int_{\delta_{QR}^{\max}}^{\delta_{QR}^{\min}} K_h(\delta)d\delta$, though we actually want the value at a finite number of evaluation points given by our data, $\sum_{qr} K_h(\delta_{qr} d\delta_{qr})$. Within quadrature is an *interpolation* problem, namely that of approximation function values lying between the quadrature points. Taking the simplest form of *Newton-Cotes* formula, the two-point form, gives the familiar *trapezoidal rule*, as shown in Figure 2. Here the function values lying between the endpoints, the quadrature points in this case, are approximated by a linear function of the abscissa (polynomial in general). Recalling Taylor's theorem,

$$\begin{aligned} f(x) &= \sum_{p=0}^{\infty} \frac{f^{(p)}(a)}{p!}(x-a)^p \\ &= f(a) + f'(a)(x-a) + \ldots \end{aligned} \tag{9}$$

we see that this particular choice of interpolation corresponds to the Newton-Stirling formula (using central differences) or *Gregory-Newton* formula (using forward differences, shown), finite analogs of Taylor's theorem:

$$f(x) = f(x_i) + \frac{1}{2}\frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i) + \ldots \tag{10}$$

This is called the *finite-difference* approximation and is used in many forms throughout applied mathematics. The basic idea of approximating continuous functions with a finite set of pieces lies at the heart of all methods for numerical integration and the important special case of solving differential equations.

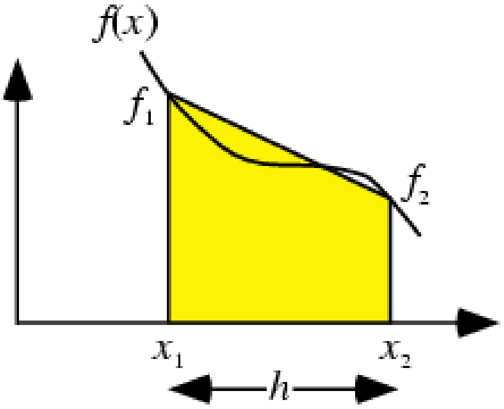

Figure 2: Trapezoidal rule for numerical integration.

Adaptive quadrature rule. Let $\delta_{QR}^{\min}$ and $\delta_{QR}^{\max}$ be our bounds on the distance between respective points in Q and R, yielding lower and upper bounds on R's mass contribution to Q of $K_h(\delta_{QR}^{\max})$ and $K_h(\delta_{QR}^{\min})$, respectively. An obvious special case of the Gregory-Newton formula is

$$\begin{aligned} K_h(\delta_{QR}) &= K_h(\delta_{QR}^{\min}) + \frac{1}{2}(\delta_{QR} - \delta_{QR}^{\min})\frac{\{K_h(\delta_{QR}^{\max}) - K_h(\delta_{QR}^{\min})\}}{\delta_{QR}^{\max} - \delta_{QR}^{\min}} \\ &\quad + O((\delta_{QR} - \delta_{QR}^{\min})^2) \end{aligned} \tag{11}$$

Since we are using $\delta_{QR}^{\min}$ and $\delta_{QR}^{\max}$ as our two quadrature points, and the placement of node boundaries is determined by the distribution of the data, we are in the realm of *adaptive quadrature* rather than the familiar fixed-width quadrature.

Intuitively, the closer $K_h(\delta_{QR}^{\max})$ and $K_h(\delta_{QR}^{\min})$ are to each other, the better we can approximate R's contribution by $N_R\overline{K_h}$ where $\overline{K_h} = \frac{1}{2}\{K_h(\delta_{QR}^{\max}) + K_h(\delta_{QR}^{\min})\}$.

Recall that our ultimate aim in this is not really to estimate the integral's value via some mean and standard deviation, say. Instead, we are interested in hard error guarantees in the current context, and thus we need *bounds* on the integral (actually, sum). In principle we could integrate the interpolation polynomial over the interval, but we don't know the placement of the evaluation points. However with the assumption $K_h()$ is *monotonic*, we can obtain simple bounds. The error of this linear approximation with respect to any point $\underline{x}_q \in Q$ is

$$\begin{aligned} e_{QR} &= \sum_r \left|K_h(\|\underline{x}_q - \underline{x}_r\|) - \overline{K_h}\right| \leq \frac{N_R}{\delta_{QR}^{\max} - \delta_{QR}^{\min}} \int_{\delta_{QR}^{\max}}^{\delta_{QR}^{\min}} \left|K_h(\delta) - \overline{K_h}\right| d\delta \\ &= \frac{N_R}{2}\left\{K_h(\delta_{QR}^{\min}) - K_h(\delta_{QR}^{\max})\right\}. \end{aligned} \tag{12}$$

To ensure that every $\phi(\underline{x}_q)$'s error e_q meets the user-specified ϵ tolerance, or $\frac{e_q}{\phi(\underline{x}_q)} \leq \epsilon$, we can enforce that $\frac{e_{QR}}{\phi(\underline{x}_q)} \leq \frac{N_R}{N}\epsilon$ by using the running lower bound $\phi_Q^{\min}$ for $\phi(\underline{x}_q)$, yielding

$$K_h(\delta_{QR}^{\min}) - K_h(\delta_{QR}^{\max}) \leq \frac{2\epsilon}{N}\phi_Q^{\min} \tag{13}$$

as a *local* pruning criterion which ensures the *global* error tolerance ϵ. It can be seen that exclusion and inclusion are also special cases of this generalized rule.

Arbitrary weights. For simplicity the algorithm is shown here assuming that all weights are 1. The generalization to arbitrary *positive* weights is fairly natural though minor insights are required. The generalization to comletely arbitrary weights is not completely straightforward and more complicated to explain. However the structure and behavior of the resulting algorithm is similar to that shown.

```
GP(Q,R,h)
  dl = N_R K_h(δ_QR^max), du = N_R K_h(δ_QR^min) − N_R.
  if K_h(δ_QR^min) − K_h(δ_QR^max) ≤ (2ε/N) φ_Q^min,
    foreach x_q ∈ Q, φ_q^min += dl, φ_q^max += du.
    return.
  else,
    if leaf(Q) and leaf(R), GPBase(Q,R,h), return.
    GP(Q.left,closer-of(Q.left,{R.left,R.right},h)).
    GP(Q.left,farther-of(Q.left,{R.left,R.right},h)).
    GP(Q.right,closer-of(Q.right,{R.left,R.right},h)).
    GP(Q.right,farther-of(Q.right,{R.left,R.right},h)).

GPBase(Q,R,h)
  foreach x_q ∈ Q,
    foreach x_r ∈ R,
      c = K_h(||x_q − x_r||), φ_q^min += c, φ_q^max += c.
    φ_q^max -= N_R.
  φ_Q^min = min_{q∈Q} φ_q^min, φ_Q^max = max_{q∈Q} φ_q^max − N_R.
```

Figure 3: Dual-tree algorithm, basic form. In the pseudocode a += b means a = a + b. A leaf's left or right child is defined to be itself. In the actual code repeated recursion cases are prevented.

N	Tree Build (sec)	Skilling with Direct Multiplies (sec)	Skilling with N-body Multiplies (sec)	Speedup Factor
2K	0.3	6	0.8	7.5
5K	0.7	89	1.6	56
10K	1.4	n/a	3.3	n/a
100K	15	n/a	34	n/a
1M	90	n/a	351	n/a

Figure 4: Computational comparison.

4 Experimental Example

We illustrate the performance of the approach using subsamples of increasing size from a large astronomical dataset from the Sloane Digital Sky Survey [16] consisting of two 'color' attributes of sky objects obtained from different bandwidth filters. We will predict the target variable *r-i* based on *g-r*.

Prediction is performed for N query points using N training points. The parameter σ was chosen using a very large test set, minimizing L_1 regression error. Note the tree need only be built once for the lifetime of each dataset, so its construction cost is amortized over all operations that will ever be done with that dataset.

The experiments were performed on a 1999-era Linux 500-MHz Pentium desktop workstation with 768Mb of RAM. The Skilling method with direct multiplies was not able to perform GP learning for 10,000 data and beyond as the explicit kernel matrix calculation required more RAM than available. The greater memory efficiency of the N-body method allowed prediction of 1 million test points with 1 million training points well within RAM limits (in fact datasets of several million are possible even on this machine).

Only 3 Skilling iterations were performed for both methods. At this number of iterations the test set error was the same for the Skilling method with direct matrix-vector multiplies and the proposed method with N-body matrix-vector multiplies, and furthermore was *not* significantly improved by increasing the number of iterations. In fact, in many cases the Skilling method (even with exact multiplies) degenerated numerically when many iterations were used. Fortunately, the early iterations are exactly those which can tolerate the small loss of accuracy introduced by our approximate method. We speculate that this behavior is fairly typical of the Skilling method in general, as it occurred in the several other datasets with which we experimented.

The computational behavior of the algorithm in terms of the data dimensionality, the size of the bandwidth σ, the form of the kernel function $K()$, and diversity of datasets is effectively explored in previous publications [8, 6] as the underlying algorithm is the same.

To summarize those studies, they demonstrate efficiency in dimensionalities up to 784, as ball-trees are sensitive to the instrinsic dimension of the data rather than the explicit dimension. The high performance of the method displays robustness across orders of magnitude in the range of bandwidth settings and various kernel functions such as the Gaussian, spherical, and Epanechnikov kernels. The primary limitations of this family of algorithms are their assumption that the datasets can fit comfortably in RAM, and the necessary property of low to moderate *intrinsic* dimension (arguably a property that typical real-world datasets tend to possess).

5 Conclusion

We have presented an approach for Gaussian process prediction which can scale to millions of points. We chose the GP example because the applicability of an N-body approach is not as obvious as it might be for other statistical models. We hope that our solution for GP's also serves to illustrate a more general kernel matrix-vector multiplication mechanism as well as motivate various special-case variants.

At least two other popular models deserve particular attention along these lines. The EM algorithm represents a dynamical form of N-body problem, much like the N-body problem of physics, where the moving class parameters pose the problem of efficient tree updates. A solution treating this problem is shown in [7], in the context of computational fluid dynamics. SVM prediction represents a special case of kernel matrix problem because the actual underlying problem (binary classification) does not strictly require the actual per-class summations, only a decision about which is larger. An approach taking advantage of this property has been submitted to this conference [11].

References

[1] B. Schlkopf and C. Burges and A. Smola, editor. *Advances in Kernel Methods - Support Vector Learning.* MIT Press, 1999.

[2] J. Barnes and P. Hut. A Hierarchical $O(NlogN)$ Force-Calculation Algorithm. *Nature*, 324, 1986.

[3] E. Chavez, G. Navarro, R. Baeza-Yates, and J. L. Marroquin. Proximity Searching in Metric Spaces. *ACM Computing Surveys*, 33:273–321, 2001.

[4] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, September 1977.

[5] M. Gibbs and D. J. C. MacKay. Efficient Implementation of Gaussian Processes. unpublished manuscript, 1997.

[6] A. G. Gray. *Bringing Tractability to Generalized N-Body Problems in Statisical and Scientific Computation.* PhD. Thesis, Carnegie Mellon University, Computer Science Department, 2003.

[7] A. G. Gray. Linear-time Smoothed Particle Hydrodynamics. to be submitted, 2003.

[8] A. G. Gray and A. W. Moore. Nonparametric Density Estimation: Toward Computational Tractability. In *SIAM International Conference on Data Mining 2003*, 2003.

[9] A. G. Gray and A. W. Moore. Rapid Evaluation of Multiple Density Models. In *Artificial Intelligence and Statistics 2003*, 2003.

[10] L. Greengard and V. Rokhlin. A Fast Algorithm for Particle Simulations. *Journal of Computational Physics*, 73, 1987.

[11] T. Liu, A. Moore, and A. G. Gray. New Algorithms for Efficient High Dimensional Non-parametric Classification. submitted to NIPS 2003, 2003.

[12] D. J. C. MacKay. Gaussian Processes: A Replacement for Supervised Neural Networks? NIPS 1997 tutorial lecture notes, 1997.

[13] G. Matheron. Principles of Geostatistics. *Economic Geology*, 58:1246–1266, 1963.

[14] S. M. Omohundro. Bumptrees for Efficient Function, Constraint, and Classification Learning. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*. Morgan Kaufmann, 1991.

[15] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Co., 1996.

[16] SDSS. *The Sloan Digital Sky Survey.* `www.sdss.org`.

[17] J. Skilling. Bayesian Numerical Analysis. In W. T. G. Jr. and P. Milonni, editor, *Physics and Probability.* Cambridge University Press, 1993.