Scalable machine learning for massive datasets: Fast summation algorithms

VIKAS CHANDRAKANT RAYKAR, University of Maryland, CollegePark
vikas@umiacs.umd.edu

Huge data sets containing millions of training examples with a large number of attributes (*tall fat data*) are relatively easy to gather. However one of the bottlenecks for successful inference of useful information from the data is the computational complexity of machine learning algorithms. Most state-of-the-art nonparametric machine learning algorithms have a computational complexity of either $\mathcal{O}(N^2)$ or $\mathcal{O}(N^3)$, where N is the number of training examples. This has seriously restricted the use of massive data sets. The bottleneck computational primitive at the heart of various algorithms is the multiplication of a structured matrix with a vector, which we refer to as *matrix-vector product* (MVP) primitive. The goal of my thesis is to speedup up these MVP primitives by *fast approximate algorithms* that scale as $\mathcal{O}(N)$ and also provide *high accuracy guarantees*. I use ideas from computational physics, scientific computing, and computational geometry to design these algorithms. Currently the proposed algorithms have been applied in kernel density estimation, optimal bandwidth estimation, projection pursuit, Gaussian process regression, implicit surface fitting, and ranking.

[**PhD research summary**]: January 19, 2007

1. COMPUTATIONAL CURSE OF NON-PARAMETRIC METHODS

During the past few decades is has become relatively easy to gather huge amount of data, apprehensively called-*massive data sets*. A few such examples include genome sequencing, astronomical databases, internet databases, medical databases, financial records, weather reports, audio and video data. *Learning* is a principled method for distilling *predictive* and therefore scientific theories from the data.

The *parametric approach* to learning assumes a functional form for the model to be learnt, and then estimates the unknown parameters. Once the model has been trained *the training examples can be discarded*. However, unless the form of the function is known a priori, assuming a certain form very often leads to erroneous inference. The *nonparametric methods* do not make any assumptions on the form of the underlying function. A price to be paid is that all the available *data has to be retained* while making the inference. Nonparametric does not mean the lack of parameters, but rather that the underlying model of a learning problem cannot be indexed with a finite number of parameters. The number of parameters usually grows with the number of training data. These are also known as *memory based methods*–the model is the entire training set.

One of the major bottlenecks for successful inference using nonparametric methods is their computational complexity. Most of the current stat-of-the-art nonparametric machine learning algorithms have the computational complexity of either $\mathcal{O}(N^2)$ (for prediction) or $\mathcal{O}(N^3)$ (for training). This has seriously restricted the use of massive data sets. For example, a simple kernel density estimation with 1 million points would take around 2 days.

2. BRINGING COMPUTATIONAL TRACTABILITY TO MASSIVE DATASETS

I see broadly three different approaches to cope with this quadratic scaling.

(1) ***Subset of data*** These methods are based on using a small representative subset of the training examples. Different schemes specify different strategies to effectively choose the subset. These methods can be considered to provide *exact inference in an approximate model*.

(2) ***Online learning*** This strategy uses sequential update methods which can find good solutions in single passes through the data. This cuts down the need for running very large scale batch optimizers.

(3) ***Fast matrix-vector product primitives*** At the heart of various algorithms is the multiplication of a structured matrix with a vector, which we refer to as *matrix-vector product* (MVP) primitive. This MVP is the bottleneck contributing to the $\mathcal{O}(N^2)$ quadratic complexity. In my thesis I use ideas and techniques from computational physics (fast multipole methods), scientific computing (Krylov subspace methods), and computational geometry (*kd*-trees,clustering) to speed up *approximate* calculation of these primitives to $\mathcal{O}(N)$ and also provide *high accuracy guarantees*. In analogy these methods provide *approximate inference in an exact model*.

3. WEIGHTED SUPERPOSITION OF KERNELS

In most kernel based machine learning algorithms, Gaussian processes, and non-parametric statistics. a key computationally intensive task is to compute a linear combination of local kernel functions centered on the training data, *i.e.*,

$$f(x) = \sum_{i=1}^{N} q_i k(x, x_i), \tag{1}$$

where $\{x_i \in \mathbb{R}^d, i = 1, \ldots, N\}$ are the N training data points, $\{q_i \in \mathbb{R}, i = 1, \ldots, N\}$ are the weights, $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is the local kernel function, and $x \in \mathbb{R}^d$ is the test point at which $f(.)$ is to be computed [1]. The computational complexity to evaluate (1) at a given test point is $\mathcal{O}(N)$. Training these models scales as $\mathcal{O}(N^3)$ since most involve solving the linear system of equation $(\mathbf{K} + \lambda\mathbf{I})\xi = \mathbf{y}$, where $\mathbf{K}$ is the $N \times N$ Gram matrix where $[\mathbf{K}]_{ij} = k(x_i, x_j)$. Also many kernel methods in unsupervised learning like kernel principal component analysis and Laplacian eigenmaps involve computing the eigen values of the Gram matrix. Solutions to such problems can be obtained using iterative methods, where the dominant computation is evaluation of $f(x)$. Recently, such nonparametric problems have been collectively referred to as *N-body problems in learning* [Gray and Moore 2001], in analogy with the coulombic, magnetostatic, and gravitational N-body potential problems occurring in computational physics [Greengard 1994]. These problems require the calculation of all pairwise interactions in a large ensemble of particles.

[1] For kernel machines (e.g. regularized least squares, support vector machines, kernel regression) f is the regression/classification function. In case of Gaussian process regression f is the mean prediction. For non-parametric density estimation it is the kernel density estimate.

4. FAST APPROXIMATE MATRIX-VECTOR PRODUCT

In general we need to evaluate (1) at M points $\{y_j \in \mathbb{R}^d, j = 1, \ldots, M\}$, *i.e.*,

$$f(y_j) = \sum_{i=1}^{N} q_i k(y_j, x_i) \;\; j = 1, \ldots, M, \tag{2}$$

leading to the quadratic $\mathcal{O}(MN)$ cost. The sum can be thought of as a *matrix-vector product* $f = \mathbf{K}q$, where $\mathbf{K}$ is a $M \times N$ matrix the entries of which are of the form $[\mathbf{K}]_{ij} = k(y_j, x_i)$ and $q = [q_1, \ldots, q_N]^T$ is a $N \times 1$ column vector. We develop fast ϵ-exact algorithms that compute the sum 2 approximately in linear $\mathcal{O}(M + N)$ time. The algorithm is ϵ-exact, *i.e.*, for any given $\epsilon > 0$, $\widehat{f}$ is an $\epsilon - exact$ approximation to f if the maximum absolute error relative to the total weight $Q = \sum_{i=1}^{N} |q_i|$ is upper bounded by ϵ, *i.e.*, $\max_{y_j} \left[|\hat{f}(y_j) - f(y_j)|/Q \right] \leq \epsilon$. The constant in $\mathcal{O}(M+N)$, depends on the desired *accuracy* ϵ, which however can be *arbitrary*. In fact for machine precision accuracy there is no difference between the direct and the fast methods.

The fast algorithm is based on series expansion of the kernel and retaining only the first few terms contributing to the desired accuracy. Philosophically, the reason we will be able to achieve $\mathcal{O}(M + N)$ algorithms to compute the matrix-vector multiplication is that the matrix $\mathbf{K}$ is a structured matrix, since all the entries of the matrix are determined by the set of $M + N$ points $\{x_i\}_{i=1}^{N}$ and $\{y_j\}_{i=1}^{M}$. The algorithms are in the spirit of *fast multipole methods* used in computational physics. The fast multipole method (FMM) has been called one of the ten most significant algorithms [Dongarra and Sullivan 2000] in scientific computation discovered in the 20th century. Originally this method was developed for the fast summation of the potential fields generated by a large number of sources (charges), such as those arising in gravitational or electrostatic potential problems [Greengard and Rokhlin 1987]. Since then FMM has also found application in many other problems and can be viewed as a fast matrix-vector product algorithm for particular structured matrices.

5. CURRENT THESIS CONTRIBUTIONS

The thesis consists of two core contributions–(1) design of fast summation algorithms and (2) applying these fast primitives to certain large scale machine learning problems. Table I summarizes the contributions of this thesis. The source code for all the fast summation algorithms are released under the Lesser GPL. Below we present a brief summary of the key contributions.

5.1 Fast computation of sums of Gaussians

The most commonly used kernel function is the *Gaussian kernel* $e^{-\|x-y\|^2/h^2}$, where h is called the *bandwidth* of the kernel. The computational cost of the direct evaluation of sums of multivariate Gaussian kernels scales as the product of the number of kernel functions and the evaluation points. The fast Gauss transform proposed by [Greengard and Strain 1991] is a ϵ-exact approximation algorithm that reduces the computational complexity of the evaluation of the sum of N Gaussians at M points in d dimensions from $\mathcal{O}(MN)$ to $\mathcal{O}(M + N)$. However, the constant factor

Kernel	Core MVP primitive	Applications
Gaussian	$G(y_j) = \sum_{i=1}^{N} q_i e^{-\|y_j - x_i\|^2/h^2}$	kernel density estimation Gaussian process regression implicit surface fitting
Hermite× Gaussian	$G(y_j) = \sum_{i=1}^{N} q_i H_r\left(\frac{y_j - x_i}{h_1}\right) e^{-(y_j - x_i)^2/h_2^2}$	optimal bandwidth estimation projection pursuit
error function	$G(y_j) = \sum_{i=1}^{N} q_i \operatorname{erfc}(y_j - x_i)$	ranking collaborative filtering

Table I. Summary of the thesis. The fast summation algorithms designed and tasks to which they were applied. Computation of each of these primitives at M points requires $\mathcal{O}(MN)$ time. The fast algorithms we design computes the same to a specified ϵ accuracy in $\mathcal{O}(M+N)$ time.

in $\mathcal{O}(M+N)$ grows exponentially with increasing dimensionality d, which makes the algorithm impractical for dimensions greater than three. We present a new algorithm where the constant factor is reduced to asymptotically polynomial order. The reduction is based on a new multivariate Taylor series expansion scheme combined with the efficient space subdivision using the k-center algorithm. Our experimental results indicate that the proposed algorithm gives good speedups in dimensions as high as tens for moderate bandwidths and as high as hundreds for large and small bandwidths. As an example we show how the proposed method can be used for very fast ϵ-exact multivariate kernel density estimation and fast Gaussian process regression. [Raykar et al. 2005; Raykar and Duraiswami 2007c; Raykar et al. 2007; Raykar and Duraiswami 2005a]

5.2 Fast optimal bandwidth estimation

Efficient use of kernel density estimation (KDE) requires the optimal selection of the smoothing parameter called the bandwidth h of the kernel. Small h leads to an estimator with small bias and large variance. Large h leads to a small variance at the expense of increase in bias. Most state-of-the-art automatic bandwidth selection procedures require estimation of quantities involving the density derivatives. The computational complexity of evaluating the density derivative at M evaluation points given N sample points from the density scales as $\mathcal{O}(MN)$. We propose a computationally efficient $\epsilon - exact$ approximation algorithm for the univariate Gaussian kernel based density derivative estimation that reduces the computational complexity from $\mathcal{O}(MN)$ to linear $\mathcal{O}(M+N)$. We apply the density derivative evaluation procedure to estimate the optimal bandwidth for kernel density estimation, a process that is often intractable for large data sets. We demonstrate the speedup achieved on the bandwidth selection using the solve-the-equation plug-in method. We also demonstrate that the proposed procedure can be extremely useful for speeding up exploratory projection pursuit techniques. [Raykar and Duraiswami 2005b; Raykar and Duraiswami 2006]

5.3 Large scale preference learning

We consider the problem of learning the ranking function that maximizes a generalization of the Wilcoxon-Mann-Whitney statistic on the training data. Relying on an ϵ-exact approximation for the error-function, we reduce the computational complexity of each iteration of a conjugate gradient algorithm for learning ranking functions from $\mathcal{O}(m^2)$, to $\mathcal{O}(m)$. Experiments on public benchmarks for ordinal regression and collaborative filtering indicate that the proposed algorithm is as accurate as the best available methods in terms of ranking accuracy, when the algorithms are trained on the same data. However, since it is several orders of magnitude faster than the current state-of-the-art approaches, it is able to leverage much larger training datasets containing tens to hundreds of thousands of samples (common in real-life applications such as book/movie recommender systems). [Raykar and Duraiswami 2007b; Raykar et al. 2007a; Raykar et al. 2007b]

5.4 Fast large scale Gaussian process regression

Gaussian processes allow the treatment of non-linear non-parametric regression problems in a Bayesian framework. However the computational cost of training such a model with N examples scales as $\mathcal{O}(N^3)$. Iterative methods for the solution of linear systems can bring this cost down to $\mathcal{O}(N^2)$, which is still prohibitive for large data sets. We consider the use of ϵ-exact matrix-vector product algorithms to reduce the computational complexity to $\mathcal{O}(N)$. Using the theory of inexact Krylov subspace methods we show how to choose ϵ to guarantee the convergence of the iterative methods. We demonstrate the speedup achieved on large data sets. For prediction of the mean the computational complexity is reduced from $\mathcal{O}(N)$ to $\mathcal{O}(1)$. Our experiments indicated that for low dimensional data ($d \leq 8$) the proposed method gives substantial speedups. [Raykar and Duraiswami 2007a]

6. FUTURE WORK

The following problems are among those that I wish to formulate well and solve in the course of this thesis, and in the future.

—***Core algorithms*** Development of these kind of fast approximate algorithms for more kernels–e.g., the Epanechnikov kernel for kernel density estimation and the Matèrn class of kernels used in Gaussian process regression.

—***Convergence issues*** In many applications these fast MVP primitives are embedded in a optimization routine–e.g., in ranking problem we embedded it in a conjugate-gradient procedure. A theoretical issue which we have barely touched upon concerns the convergence of these optimization routines when using approximate MVP primitives.

—***Applications*** A few applications which I would like to further explore include–hyperparameter selection for Gaussian processes, implicit surface fitting via Gaussian processes, Nadarya-Watson kernel regression, and inexact eigenvalue methods for unsupervised learning.

7. OPEN PROBLEMS

Following are a few open problems which may require much time and thought.

—***The curse of dimensionality.*** For the Gaussian kernel our experimental results indicate that it easy to get good speedups at very large or very small bandwidths. For moderate bandwidths and moderate dimensions ($d \leq 10$) our proposed algorithm is capable of giving good speedups. However getting good speedups for *moderate bandwidths and large dimensions* remains an important open research problem.

—***The paradox of the curse of dimensionality.*** For most machine learning tasks even though the data is very high dimensional, the true intrinsic dimensionality is typically very small. I intend to explore if dimensionality reduction approaches like PCA and manifold learning methods can be directly incorporated into our fast algorithms.

—***Structure, Inference, and Computation*** A more ambitious task would be to explore if there are any deeper connections between structure in the data, computation, and inference.

REFERENCES

Dongarra, J. and Sullivan, F. 2000. The top ten algorithms of the century. *Computing in Science and Engineering 2,* 1, 22–23. 3

Gray, A. and Moore, A. 2001. N-body problems in statistical learning. In *Advances in Neural Information Processing Systems*. 521–527. 2

Greengard, L. 1994. Fast algorithms for classical physics. *Science 265,* 5174, 909–914. 2

Greengard, L. and Rokhlin, V. 1987. A fast algorithm for particle simulations. *J. of Comp. Physics 73,* 2, 325–348. 3

Greengard, L. and Strain, J. 1991. The fast Gauss transform. *SIAM Journal of Scientific and Statistical Computing 12,* 1, 79–94. 3

Raykar, V. C. and Duraiswami, R. 2005a. The improved fast Gauss transform with applications to machine learning. Presented at the NIPS 2005 workshop on Large scale kernel machines. 4

Raykar, V. C. and Duraiswami, R. 2005b. Very fast optimal bandwidth selection for univariate kernel density estimation. Tech. Rep. CS-TR-4774, Department of computer science, University of Maryland, Collegepark. 4

Raykar, V. C. and Duraiswami, R. 2006. Fast optimal bandwidth selection for kernel density estimation. In *Proceedings of the sixth SIAM International Conference on Data Mining*, J. Ghosh, D. Lambert, D. Skillicorn, and J. Srivastava, Eds. 524–528. 4

Raykar, V. C. and Duraiswami, R. 2007a. Fast large scale gaussian process regression using approximate matrix-vector products. *submitted*. 5

Raykar, V. C. and Duraiswami, R. 2007b. Fast weighted summation of erfc functions. Tech. Rep. CS-TR-4848, Department of computer science, University of Maryland, Collegepark. 5

Raykar, V. C. and Duraiswami, R. 2007c. *Large Scale Kernel Machines*. MIT Press, Chapter The Improved Fast Gauss Transform with applications to machine learning. 4

Raykar, V. C., Duraiswami, R., and Gumerov, N. 2007. Fast computation of sums of Gaussians. *Journal of Machine Learning Research (submitted)*. 4

Raykar, V. C., Duraiswami, R., and Krishnapuram, B. 2007a. A fast algorithm for learning large scale preference relations. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS 2007)*. 5

Raykar, V. C., Duraiswami, R., and Krishnapuram, B. 2007b. A fast algorithm for learning the large scale ranking function. *IEEE Transactions on Pattern Analysis and Machine Intelligence (submitted)*. 5

Raykar, V. C., Yang, C., Duraiswami, R., and Gumerov, N. 2005. Fast computation of sums of Gaussians in high dimensions. Tech. Rep. CS-TR-4767, Department of Computer Science, University of Maryland, CollegePark. 4