Computational tractability of machine learning algorithms for tall fat data

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

Huge data sets containing millions of training examples (*tall data*) with a large number of attributes (*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. In my thesis I propose to explore the scalability of machine learning algorithms both with respect to the number of training examples and the number of attributes per training example and propose linear $\mathcal{O}(N)$ time ϵ-exact approximation algorithms. The area lies at the interface of computational statistics, machine learning, approximation theory, and computational geometry.

[**Dissertation proposal**]: May 4, 2005

1. THE EVILS OF LEARNING WITH MASSIVE DATA SETS

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, experimental data from particle physics, medical databases, financial records, weather reports, audio and video data. We would like to build systems which can automatically extract useful information from the raw data. *Learning* is a principled method for distilling *predictive* and therefore scientific theories from the data [Poggio and Smale 2003].

In a *supervised learning* scenario we are given a collection of say N training examples along with the desired output based on which we learn a concept/model to draw inferences *for examples not in the training set.* A few canonical supervised learning tasks include *regression*, *classification*, and *density estimation.*

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.* The essence of the training examples have been captured in the model parameters, using which we can draw further inferences. 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. This is sometimes referred to as *'letting the data speak for themselves'* [Wand and Jones 1995]. All the available *data has to be retained* while making the inference. It should be noted that nonparametric does not mean the lack of parameters, but rather that the underlying function/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.

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 $\mathcal{O}(N^2)$ or $\mathcal{O}(N^3)$. This has seriously restricted the use of massive data sets. Current implementations can handle only a few thousands of training examples. During the last decade, data set size has outgrown processor speed. This thesis investigates computationally efficient ways to exploit such large data sets using nonparametric machine learning algorithms. We use ideas from scientific computing and computational geometry literature to design novel algorithms that scale as $\mathcal{O}(N)$.

1.1 Tall fat data

Consider a data set $\mathcal{D}$ of N training examples where each training instance has d attributes (for example $\mathcal{D} = \{x_i \in \mathbb{R}^d, y_i\}_{i=1}^N$). One of the other concerns of this thesis is that our algorithms should also scale well with d. We refer to $\mathcal{D}$ as *tall data* if $N >> d$ and *fat data* if $d >> N$. While most of the available data sets are tall, it is not uncommon to have *tall fat data.* A typical example for fat data would be gene expression data.

2. WEIGHTED SUPERPOSITION OF KERNELS

In most kernel based machine learning algorithms [Shawe-Taylor and Cristianini 2004], Gaussian processes [Rasmussen and Williams 2006], and non-parametric statistics [Izenman 1991] the 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.

The computational complexity to evaluate Equation 1 at a given test point is $\mathcal{O}(N)$.

For kernel machines (e.g. regularized least squares [Poggio and Smale 2003], support vector machines [Cristianini and Shawe-Taylor 2000], kernel regression [Wand and Jones 1995]) f is the regression/classification function. This is a consequence of the well known classical representer theorem [Wabha 1990] which states that the solutions of certain risk minimization problems involving an empirical risk term and a quadratic regularizer can be written as expansions in terms of the kernels centered on the training examples. In case of Gaussian process regression [Williams and Rasmussen 1996] f is the mean prediction. For non-parametric density estimation it is the kernel density estimate [Wand and Jones 1995].

Training these models scales as $\mathcal{O}(N^3)$ since most involve solving the linear system of equation

$$(\mathbf{K} + \lambda \mathbf{I})\xi = \mathbf{y}. \tag{2}$$

$\mathbf{K}$ 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 [Smola et al. 1996], spectral clustering [Chung 1997], 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.

3. ϵ-EXACT APPROXIMATION

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

$$f(y_j) = \sum_{i=1}^{N} q_i k(y_j, x_i) \;\; j = 1, \ldots, M, \tag{3}$$

leading to the quadratic complexity of $\mathcal{O}(MN)$. We will develop fast ϵ-exact algorithms that compute the sum 3 approximately in linear $\mathcal{O}(M+N)$ time. The algorithm is ϵ-exact in the sense made precise below.

DEFINITION 3.1. *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.,*

$$\max_{y_j} \left[\frac{|\hat{f}(y_j) - f(y_j)|}{Q} \right] \leq \epsilon. \tag{4}$$

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.

3.1 Matrix vector multiplication

The sum in equation 3 can be thought of as a *matrix-vector multiplication*

$$f = \mathbf{K}q, \tag{5}$$

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.

DEFINITION 3.2. *A dense matrix of order $M \times N$ is called a* structured matrix *if its entries depend only on $\mathcal{O}(M+N)$ parameters.*

The reason we will be able to get $\mathcal{O}(M+N)$ 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 $\{y_j\}_{i=1}^{M}$. If the entries of the of the matrix $\mathbf{K}$ are completely random than we cannot do any better than $\mathcal{O}(MN)$ [1].

[1]This statement should be considered in a deterministic sense. It may be possible to derive ϵ-exact approximation algorithms in a probabilistic sense

4. GAUSSIAN KERNEL

The most commonly used kernel function is the *Gaussian kernel*

$$k(x, y) = e^{-\|x-y\|^2/h^2}, \tag{6}$$

where h is called the *bandwidth* of the kernel. The bandwidth h controls the degree of smoothing, of noise tolerance, and of generalization. The Gaussian kernel is a *local kernel* in the sense that $\lim_{\|x-y\|\to\infty} k(x, y) = 0$.

The sum of multivariate Gaussian kernels is known as the *discrete Gauss transform* in the scientific computing literature. More formally, for each *target point* $\{y_j \in \mathbb{R}^d\}_{j=1,\dots,M}$ the *discrete Gauss transform* is defined as,

$$G(y_j) = \sum_{i=1}^{N} q_i e^{-\|y_j-x_i\|^2/h^2}, \tag{7}$$

where $\{q_i \in \mathbb{R}\}_{i=1,\dots,N}$ are the *source weights*, $\{x_i \in \mathbb{R}^d\}_{i=1,\dots,N}$ are the *source points*, i.e., the center of the Gaussians, and $h \in \mathbb{R}^+$ is the *source scale* or *bandwidth*. In other words $G(y_j)$ is the total contribution at y_j of N Gaussians centered at x_i each with bandwidth h. Each Gaussian is weighted by the term q_i.

The computational complexity to evaluate the discrete Gauss transform (Equation 7) at M target points is $\mathcal{O}(MN)$. This makes the computation for large scale problems prohibitively expensive. In many machine learning tasks data-sets containing more than 10^4 points are already common and larger problems are of interest.

The *Fast Gauss Transform* (FGT) is an $\epsilon - exact$ approximation algorithm that reduces the computational complexity to $\mathcal{O}(M + N)$, at the expense of reduced precision, which however can be arbitrary. The constant depends on the desired precision, dimensionality of the problem, and the bandwidth. Given any $\epsilon > 0$, it computes an approximation $\hat{G}(y_j)$ to $G(y_j)$ such that the maximum absolute error relative to the total weight $Q = \sum_{i=1}^{N} |q_i|$ is upper bounded by ϵ, i.e.,

$$\max_{y_j} \left[\frac{|\hat{G}(y_j) - G(y_j)|}{Q} \right] \le \epsilon. \tag{8}$$

5. PROBLEMS SUCCESSFULLY ADDRESSED

As of now the following milestones have been achieved. Brief summaries of the problems are given below. The detailed reports are attached along with this proposal.

5.1 Fast computation of sums of Gaussians

[Raykar et al. 2005][Raykar and Duraiswami 2005a]

For the FGT the constant factor in $\mathcal{O}(M+N)$ grows exponentially with increasing dimensionality d, which makes the algorithm impractical for dimensions greater than three. The FGT was first proposed by [Greengard and Strain 1991] and applied successfully to a few lower dimensional applications in mathematics and physics. However the algorithm has not been widely used much in statistics, pattern recognition, and machine learning applications where higher dimensions occur commonly. An important reason for the lack of use of the algorithm in these areas is that the performance of the proposed FGT degrades exponentially with increasing dimensionality, which makes it impractical for the statistics, machine learning, and pattern recognition applications.

We propose a linear time $\epsilon - exact$ approximation algorithm for the Gaussian kernel which scales well with dimensionality. The constant factor is reduced to asymptotically polynomial order. The reduction is based on a new multivariate Taylor's series expansion (which can act both as a local as well as a far field expansion) scheme combined with the efficient space subdivision using the k-center algorithm. The proposed method differs from the original fast Gauss transform in terms of a different factorization, efficient space subdivision, and the use of point-wise error bounds. Algorithm details, error bounds, procedure to choose the parameters and numerical experiments are presented.

5.2 Multivariate kernel density estimation

[Raykar et al. 2005]
As an example we show how the IFGT can be used for very fast ϵ-exact multivariate kernel density estimation. We also show the effect of ϵ-exact approximation on the performance of the estimator.

5.3 Optimal bandwidth estimation for KDE

[Raykar and Duraiswami 2005b][Raykar and Duraiswami 2006]
Efficient use of KDE requires the optimal selection of the smoothing parameter called the bandwidth h of the kernel. For a practical implementation of KDE the choice of the bandwidth h is very important. 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. The bandwidth h has to be chosen optimally.

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.

6. WORK IN PROGRESS

6.1 Fast large scale Gaussian process regression

Gaussian processes allow 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 can bring this cost down to $\mathcal{O}(N^2)$, which is still prohibitive for large datasets. In this paper we use an ϵ-exact approximation technique, the improved fast Gauss transform to reduce the computational complexity to $\mathcal{O}(N)$, for the squared exponential covariance function. Using the theory of inexact Krylov subspace methods we show how to adaptively increase ϵ as the iteration progresses. For prediction at M points the computational complexity is reduced from $\mathcal{O}(NM)$ to $\mathcal{O}(M+N)$. We demonstrate the speedup achieved on large synthetic and real datasets. We also show how the hyperparameters can be chosen in $\mathcal{O}(N)$ time. Unlike methods which rely on selecting a subset of the data set we use all the available data and still get substantial speedups.

6.2 Implicit surface fitting via Gaussian processes

We use *Gaussian process regression*, an extremely powerful technique from machine learning, for surface reconstruction from point cloud data. The surface is defined implicity as the zero level set of a suitable function. This function is learnt from the given point cloud data. Rather than estimating the value of a function at a point, this method gives the entire predictive distribution in a Bayesian framework. As a result we obtain surface representations together with *valid estimates of uncertainty*. The computed uncertainty not only reflects the *noise* in the point cloud data but also the effect of *irregular sampling* of the surface during the scanning process. This fact is shown to be very useful for sparse representation of surface, mesh saliency representation, mesh simplification, and real-time re-scanning the surface based on the computed uncertainties. One of the major bottlenecks for most implicit surface methods is that their prohibitive computational complexity. This is no different for Gaussian process regression, the computational complexity of which scales as $O(N^3)$. In this paper we use two different ϵ-exact approximation techniques, the improved fast Gauss transform and the dual-tree methods both of which reduce the computational complexity to $O(N)$. Thus we are able to handle data-sets containing millions of points.

7. 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.

(1) ***Dual-tree methods*** Another class of methods recently proposed for the same problems are the dual-tree methods [Gray and Moore 2003]. These methods rely only on data structures like kd-trees and ball tress and not on series expansions. The dual-tree methods give good speedup for small bandwidths while the series based methods such as IFGT give good speedup for large bandwidths. I would like to explore whether these two different techniques can be combined together to give good speedups both for large and small bandwidths.

(2) ***Inexact eigenvalue methods*** Many kernel methods in unsupervised learning like kernel principal component analysis [Smola et al. 1996], spectral clustering [Chung 1997], and Laplacian eigenmaps involve computing the eigen values of the Gram matrix. Solutions to such problems can be obtained using iterative methods [Saad 1992]. The dominant computation is the matrix-vector multiplication for which we use the IFGT. We need to explore how the approximations affect the convergence of the iterative methods.

(3) ***Preconditioners for the Gram matrix*** The larger the condition number (the more *ill-conditioned* the matrix) slower is the rate of convergence. An important way around this difficulty is to *pre-condition* the matrix, so that we work with a matrix with a small condition number. More specifically we find a positive-definite symmetric matrix $\mathbf{M}$ and solve the system $\mathbf{M}^{-1}\widetilde{\mathbf{K}}\xi = \mathbf{M}^{-1}\mathbf{y}$. If the condition number of $\mathbf{M}^{-1}\widetilde{\mathbf{K}}$ is less than that of $\widetilde{\mathbf{K}}$, then the convergence is faster. We will explore $\mathcal{O}(N)$ preconditioners for the Gram matrix.

(4) ***Fast large scale linear SVMs*** Linear SVMs are quite powerful for solving large-scale data-mining tasks such as those arising in the textual domain. There has been a lot of interest in directly training the SVM in the primal [Keerthi and DeCoste 2005]. I intend to combine a factorization technique with SVM primal training to obtain substantial speedups for linear SVMs. An advantage for the linear case would be that the computation would be exact and no approximations would be involved, unlike the Gaussian kernel.

(5) ***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. We will explore if we can integrate the dimensionality reduction approaches like PCA and manifold learning methods into our algorithm. We need to be a bit careful as to not go above our budget.

(6) ***Epanechnikov kernel*** For kernel density estimation the Epanechnikov kernel is the optimal kernel, in the sense that it minimizes the MISE [Wand and Jones 1995]. For the Epanechnikov kernel it is possible to devise a fast method with zero error. It is a little bit trickier than the IFGT because the kernel has a compact support. As a result the kernel is discontinuous at the boundary. We need to use multiresolution trees.

(7) ***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

Chapelle, O. 2005. Training a support vector machine in the primal.

Chung, F. 1997. *Spectral Graph Theory*. Amer. Math. Society Press. 3, 6

Cristianini, N. and Shawe-Taylor, J. 2000. *An Introduction to Support Vector Machines (and other kernel-based learning methods)*. Cambridge University Press. 2

Gray, A. and Moore, A. 2001. N-body problems in statistical learning. In *Advances in Neural Information Processing Systems*. 521–527. 3

Gray, A. G. and Moore, A. W. 2003. Nonparametric density estimation: Toward computational tractability. In *SIAM International conference on Data Mining*. 6

Greengard, L. 1994. Fast algorithms for classical physics. *Science 265,* 5174, 909–914. 3

Greengard, L. and Strain, J. 1991. The fast Gauss transform. *SIAM Journal of Scientific and Statistical Computing 12,* 1, 79–94. 4

Izenman, A. J. 1991. Recent developments in nonparametric density estimation. *Journal of American Staistical Association 86,* 413, 205–224. 2

Keerthi, S. S. and DeCoste, D. 2005. A modified finite Netwon method for fast solution of large scale linear SVMs. *Journal of Machine Leaning Research 6*, 341–361. 7

Poggio, T. and Smale, S. 2003. The mathematics of learning: Dealing with data. *Notices of the American Mathematical Society 50,* 5, 537–544. 1, 2

Rasmussen, C. E. and Williams, C. K. I. 2006. *Gaussian Processes for Machine Learning*. The MIT Press. 2

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. 5

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. 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, 5

Saad, Y. 1992. *Numerical Methods for Large Eigenvalue Problems*. Manchester University Press. 6

Shawe-Taylor, J. and Cristianini, N. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press. 2

Smola, A., Scholkopf, B., and Muller, K.-R. 1996. Nonlinear component analysis as a kernel eigenvalue problem. Tech. Rep. 44, Max-Planck-Institut fr biologische Kybernetik, Tubingen. 2, 6

Wabha, G. 1990. *Spline Models for Observational data*. SIAM. 2

Wand, M. P. and Jones, M. C. 1994. Multivariate plug-in bandwidth selection. *Computational Statistics 9*, 97–117.

Wand, M. P. and Jones, M. C. 1995. *Kernel Smoothing*. Chapman and Hall, London. 1, 2, 7

Williams, C. K. I. and Rasmussen, C. E. 1996. Gaussian processes for regression. In *Advances in Neural Information Processing Systems*. Vol. 8. 2