The improved fast Gauss transform with applications to machine learning

Vikas C. Raykar and Ramani Duraiswami
University of Maryland, CollegePark
{vikas,ramani}@cs.umd.edu

NIPS 2005 workshop
on
Large scale kernel machines
Whistler, December 9, 2005

Introduction

- Huge data sets containing
  - millions of training examples (*tall data*)
  - with large number of attributes (*fat data*)

  are relatively easy to gather.

- Nonparametric methods in machine leaning scale as either $\mathcal{O}(N^3)$ or $\mathcal{O}(N^2)$.

Supervised Learning

The key computational task is to compute a linear combination of local kernel functions centered on the training data, i.e.,

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

- Kernel machines (e.g. RLS, SVM) f is the regression/classification function. [Representer theorem]

- Gaussian processes f is the mean prediction.

- Density estimation f is the kernel density estimate.

Prediction

The computation complexity to predict at M points given N training examples scales as $\mathcal{O}(MN)$.

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

Training

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

$$(\mathbf{K} + \sigma^2\mathbf{I})\xi = \mathbf{y}.$$

$\mathbf{K}$ is the $N \times N$ Gram matrix where $[\mathbf{K}]_{ij} = k(x_i, x_j)$.

- Direct inversion is $\mathcal{O}(N^3)$.
- Iterative methods like conjugate-gradient can bring it down to $\mathcal{O}(kN^2)$.
- The quadratic complexity is due to the matrix-vector product $\mathbf{K}q$ for some q.

Unsupervised Learning

Methods like kernel principal component analysis, spectral clustering, or Laplacian eigenmaps involve computing the eigen vectors of the Gram/Laplacian matrix.

- Direct is $\mathcal{O}(N^3)$.
- Iterative methods can bring it down to $\mathcal{O}(kN^2)$.
- The quadratic complexity is due to the matrix-vector product $\mathbf{K}q$ for some q.

Gaussian kernel

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

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

where h is called the *bandwidth* of the kernel.

Sum of multivariate Gaussian kernels is called the discrete Gauss transform – $\mathcal{O}(MN)$.

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

Improved fast Gauss transform

Speed up of these tasks using rigorous ϵ-exact approximation algorithms to achieve

- $\mathcal{O}(N)$ training time.
- $\mathcal{O}(1)$ prediction/classification time.

One such algorithm was presented for the **Gaussian kernel** in NIPS2005*- the improved fast Gauss transform (IFGT).

*C. Yang, R. Duraiswami, and L. Davis. *Efficient kernel machines using the improved fast Gauss transform. In Advances in Neural Information Processing Systems, pages 15611568, 2005.*

ϵ can be arbitrarily small (e.g., machine precision) and speedup is maintained

Effectively we have an FFT like algorithm that is "drop-in," with no accuracy penalty.

Enables large scale kernel machines

Brief idea of the IFGT

- **Step 1a** Determine parameters of algorithm based on specified error bound, kernel bandwidth, and data distribution

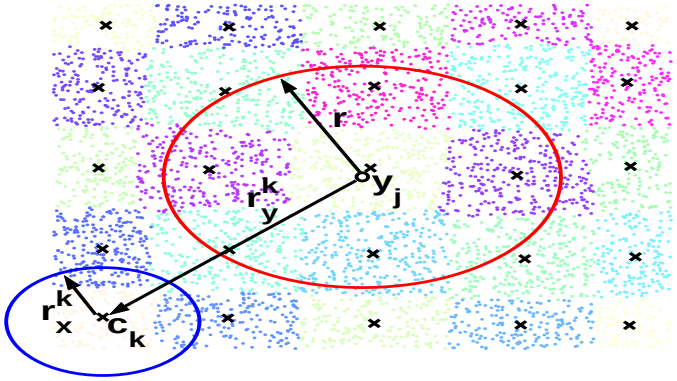
- **Step 1b** Subdivide the d-dimensional space using a k-center clustering based geometric data structure ($\mathcal{O}(N \log K)$).

- **Step 2** Build a p truncated representation of kernels inside each cluster using a set of decaying basis functions ($\mathcal{O}(Nd^p)$).

- **Step 3** Collect the influence of all the the data in a neighborhood using coefficients at cluster center and evaluate ($\mathcal{O}(Md^p)$).

The code was publicly made available for non-commercial use.

IFGT Illustration



Sample results

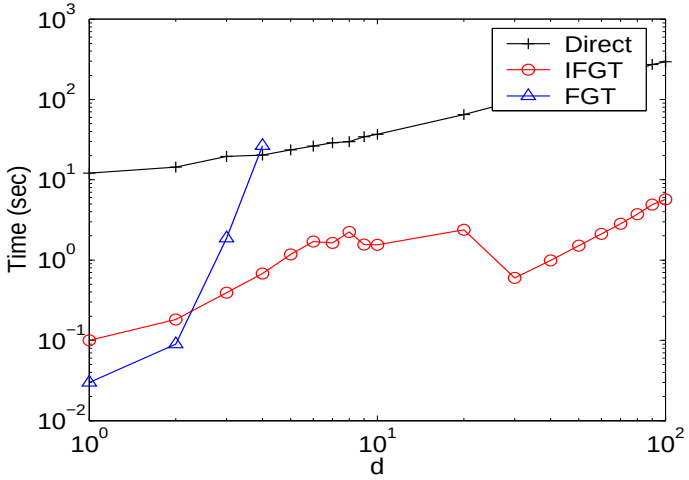
For example in three dimensions and 1 million training and test points [h=0.4]

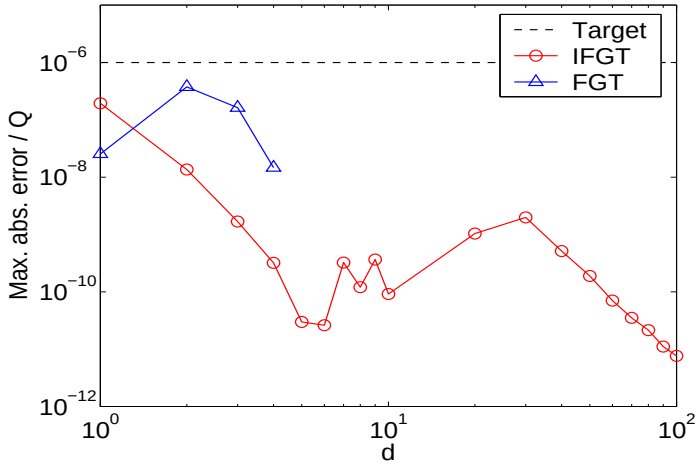
- IFGT – 6 minutes.

- Direct – 34 hours.

with an error of 10^{-8}.

IFGT can handle large dimensions [$h = \sqrt{d}$]





Segmentation using adaptive mean-shift

1.34 hours vs 2.1 minutes



Issues with IFGT presented in Yang et.al. 2005

- The error bounds were tight but very pessimistic.
- Parameter selection was not automatic.
- Users*† found the selection of parameters hard. Incorrect choice of algorithm parameters by these authors sometimes lead to poor reported performance of IFGT.
- Method for IFGT parameter selection presented in Lang et.al. is not optimal‡.

*Lee, D., Gray, A., & Moore, A., Dual-tree fast Gauss transforms, NIPS 2006

†de Freitas, N., Wang, Y., Mahdaviani, M., and Lang, D., Fast Kylov Methods for N-body learning, NIPS 2006.

‡Lang, D., Klaas, M., and Freitas, N. 2005. Empirical testing of fast kernel density estimation algorithms. Tech. Rep. UBC TR-2005-03.

Improvements to IFGT since NIPS 2005*

- A tighter point-wise error bound.
- Choice of the algorithm parameters completely automatic.
- Truncation number for each source is different.

*V. C. Raykar, C. Yang, R. Duraiswami, and N. Gumerov, Fast computation of sums of Gaussians in high dimensions. CS-TR-4767, Department of Computer Science, University of Maryland, CollegePark, 2005. https://drum.umd.edu/dspace/bitstream/1903/3020/1/CS-TR-4767.pdf

Choice of parameters is crucial

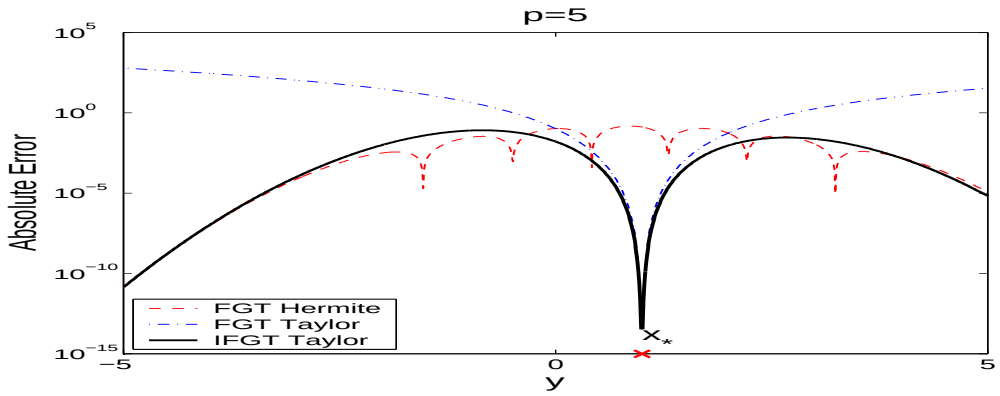
For 3 dimensions using $N = 50,000$ and $h = 0.76$

- Lee et.al.* report that IFGT takes greater than two times the direct time.

- With the parameters automatically chosen by the algorithm we take only 1.522 secs.

*Lee, D., Gray, A., & Moore, A.,Dual-tree fast Gauss transforms, NIPS 2006

IFGT expansion is both local as well as far-field!!



Hence we avoid the expensive translation operation that was required in the original FGT, and in some recently proposed algorithms.

Effect of bandwidth

- For small kernel bandwidth (h) where each training point only influences the immediate vicinity speedup is poor.

- Can use dual-tree methods*.

- In high dimensions the tails of the density contribute significantly to the total probability mass and is unlikely we have a dense sampling in tails.

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

Some recent extensions

IFGT with variable source scales

$$G(y_j) = \sum_{i=1}^{N} q_i e^{-\|y_j - x_i\|^2 / h_i^2}.$$

Approach: Build a composite factorization that builds a Taylor series for h_i as well.

Resulting IFGT runs at about the same speed.

For example for N = M = 1,024,000 while the direct evaluation takes around 2.6 days the fast evaluation requires only 4.65 minutes with an error of around 10^{-5}.

Derivative of kernel sums Many procedures (e.g., those involving optimal parameter estimation) involve taking the derivative of kernel sums.

- Automatic bandwidth selection for kernel density estimation.
- Selecting hyperparameters in Gaussian Process regression.

The derivatives of Gaussian sums involve sums of products of polynomials and Gaussians. IFGT algorithms have been developed for such kernels

$$G_r(y_j) = \sum_{i=1}^{N} q_i H_r\left(\frac{y_j - x_i}{h_1}\right) e^{-(y_j - x_i)^2/h_2^2}$$

Gaussian process regression

- Gaussian processes handle nonparametric regression in a Bayesian framework.

- The regression function is represented by an ensemble of functions, on which we place a Gaussian prior.

- This prior is updated in the light of the training data.

- As a result we obtain predictions together with valid estimates of uncertainty.

Speedup GP regression via IFGT

$\widetilde{\mathbf{K}} = \mathbf{K} + \sigma^2\mathbf{I}$	*Direct Inversion*		*Conjugate gradient*		*Conjugate gradient +IFGT*	
	Time	Space	Time	Space	Time	Space
Training phase $\xi = \widetilde{\mathbf{K}}^{-1}\mathbf{y}$	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$
Mean prediction $y = \mathbf{k}(x)^T\xi$	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$
Uncertainty $\mathbf{k}(x,x)$ $-\mathbf{k}(x)^T\widetilde{\mathbf{K}}^{-1}\mathbf{k}(x)$	$\mathcal{O}(N^3)$	$\mathcal{O}(N)$	$\mathcal{O}(N^3)$	$\mathcal{O}(N)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$

Conclusions

- The IFGT can reduce the computational complexity of numerous machine learning algorithms to linear time.

- Unlike methods which rely on choosing a subset of the dataset we use all the available points and still achieve $\mathcal{O}(N)$ complexity.

- The IFGT is now completely tweak free.

- Extensions to variable bandwidth, derivative estimation, and Gaussian processes.

- Good speedup for large bandwidths. For small bandwidths dual-tree methods can be used.