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 15, 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.

Recently, such problems have been collectively referred to as

N-body problems in learning*

in analogy with the Coulombic N-body problems occurring in computational physics.

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

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.

Discrete Gauss transform

The sum of multivariate Gaussian kernels – $\mathcal{O}(MN)$.

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

$\{q_i \in \mathbf{R}\}_{i=1,\ldots,N}$ are the N *source weights*.

$\{x_i \in \mathbf{R}^d\}_{i=1,\ldots,N}$ are the N *source points*.

$\{y_j \in \mathbf{R}^d\}_{j=1,\ldots,M}$ are the M *target points*.

$h \in \mathbf{R}^+$ is the *source scale* or *bandwidth*.

ϵ-exact approximation

Given any $\epsilon > 0$

an approximation $\widehat{G}(y_j)$ 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.,

$$\max_{y_j} \left[\frac{|\widehat{G}(y_j) - G(y_j)|}{Q} \right] \leq \epsilon.$$

Outline

Fast Gauss transform

Improved fast Gauss transform

Error bounds and choosing parameters

Results

Extension to variable bandwidth.

Extension to derivative estimation.

Choosing ϵ for iterative methods

Gaussian process regression

Fast Gauss transform (FGT)

- $\epsilon - exact$ approximation algorithm – computational complexity is $\mathcal{O}(M + N)$.

- Proposed by Greengard and Strain * 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.

*Greengard, L. and Strain, J. 1991. The fast gauss transform. SIAM J. Sci. Stat. Comput. 12, 1,79-94.

FGT degrades for $d > 3$

1. The number of the terms in the Hermite expansion grows exponentially with dimensionality d.

2. The space subdivision scheme is a uniform box subdivision scheme which is tolerable in lower dimensions but is inefficient in higher dimensions.

3. The constant term due to the translation of the far-field Hermite series to the local Taylor series grows exponentially fast with dimension making it impractical for dimensions greater than three.

Improved fast Gauss Transform (IFGT)*

1. Different series expansion [Taylor's series]–reduces the number of the expansion terms to the polynomial order.

2. k-center algorithm is applied to subdivide the space which is more efficient.

3. No translation – Our expansion can act both as a far-field and local expansion.

*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.

IFGT–New improvements*

- A tighter point-wise error bound.
- Truncation number for each source is different.
- Automatic choice of the algorithm parameters.
- Careful comparison with the original FGT algorithm.

*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.

Separate out i and j

For any point $x_* \in \mathbf{R}^d$

$$
\begin{aligned}
G(y_j) &= \sum_{i=1}^{N} q_i e^{-\|y_j - x_i\|^2/h^2} \\
&= \sum_{i=1}^{N} q_i e^{-\|(y_j - x_*) - (x_i - x_*)\|^2/h^2}, \\
&= \sum_{i=1}^{N} q_i e^{-\|x_i - x_*\|^2/h^2} e^{-\|y_j - x_*\|^2/h^2} e^{2(y_j - x_*)\cdot(x_i - x_*)/h^2}.
\end{aligned}
$$

The crux of the algorithm is to separate this entanglement via the Taylor's series expansion of the exponentials.

Factorization via multivariate Taylor's series

$$e^{2(y_j - x_*)\cdot(x_i - x_*)/h^2} = \sum_{n=0}^{p_i - 1} \frac{2^n}{n!} \left[\left(\frac{y_j - x_*}{h} \right) \cdot \left(\frac{x_i - x_*}{h} \right) \right]^n + error_{p_i}.$$

The truncation number p_i for each source x_i is chosen based

on the prescribed error ϵ,

the bandwidth h,

and

the distance from the expansion center $\|x_i - x_*\|$.

Multi-index notation

A multi-index $\alpha = (\alpha_1, \alpha_2 \ldots, \alpha_d)$ is a d-tuple of nonnegative integers.

length $|\alpha| = \alpha_1 + \alpha_2 + \ldots + \alpha_d$.

factorial $\alpha! = \alpha_1!\alpha_2! \ldots \alpha_d!$.

d-variate monomial x^α is $x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_d^{\alpha_d}$.

Multi-index notation

$$e^{2(y_j-x_*)\cdot(x_i-x_*)/h^2} = \sum_{n=0}^{p_i-1} \frac{2^n}{n!}\left[\left(\frac{y_j-x_*}{h}\right)\cdot\left(\frac{x_i-x_*}{h}\right)\right]^n + error_{p_i}.$$

$$(x\cdot y)^n = \textstyle\sum_{|\alpha|=n} \frac{n!}{\alpha!} x^\alpha y^\alpha.$$

$$e^{2(y_j-x_*)\cdot(x_i-x_*)/h^2} = \sum_{|\alpha|\le p_i-1} \frac{2^\alpha}{\alpha!}\left(\frac{y_j-x_*}{h}\right)^\alpha\left(\frac{x_i-x_*}{h}\right)^\alpha + error_{p_i}.$$

Let us ignore the error and regroup

$$\begin{aligned}\widehat{G}(y_j) &= \sum_{i=1}^{N} q_i e^{-\|x_i - x_*\|^2/h^2} e^{-\|y_j - x_*\|^2/h^2} \left[\sum_{|\alpha| \le p_i - 1} \frac{2^\alpha}{\alpha!} \left(\frac{y_j - x_*}{h} \right)^\alpha \left(\frac{x_i - x_*}{h} \right)^\alpha \right] \\ &= \sum_{|\alpha| \le p_{max} - 1} C_\alpha e^{-\|y_j - x_*\|^2/h^2} \left(\frac{y_j - x_*}{h} \right)^\alpha .\end{aligned}$$

$$C_\alpha = \frac{2^\alpha}{\alpha!} \sum_{i=1}^{N} q_i e^{-\|x_i - x_*\|^2/h^2} \left(\frac{x_i - x_*}{h} \right)^\alpha 1_{|\alpha| \le p_i - 1}.$$

$$\hat{G}(y_j) \;=\; \sum_{|\alpha|\le p_{max}-1} C_\alpha e^{-\|y_j - x_*\|^2/h^2} \left(\frac{y_j - x_*}{h}\right)^\alpha .$$

$$C_\alpha = \frac{2^\alpha}{\alpha!} \sum_{i=1}^{N} q_i e^{-\|x_i - x_*\|^2/h^2} \left(\frac{x_i - x_*}{h}\right)^\alpha \mathbf{1}_{|\alpha|\le p_i - 1}.$$

The coefficients C_α can be evaluated separately is $\mathcal{O}(N)$.

Evaluation of $\hat{G}(y_j)$ at M points is $\mathcal{O}(M)$.

Hence the computational complexity has reduced from the quadratic $\mathcal{O}(NM)$ to the linear $\mathcal{O}(N+M)$.

Space subdivision

- Same x_* for all the points may require very high truncation numbers.

- We use an data adaptive space partitioning scheme like the farthest point clustering algorithm to divide the N sources into K clusters.

k-center problem

Given

a set of N points in d dimensions and

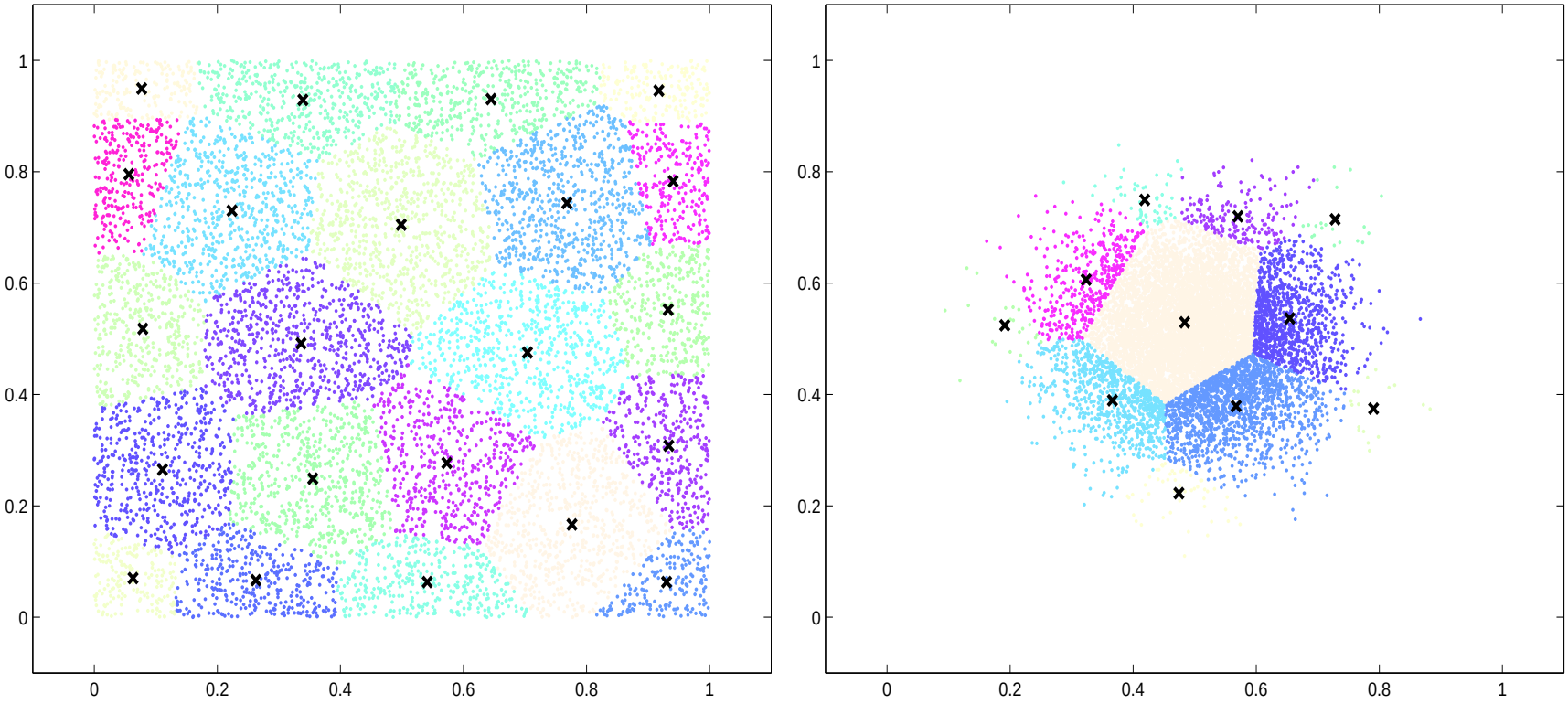
a predefined number of the clusters k,

find a partition of the points into clusters $S_1, \ldots, S_k$, and also the cluster centers $c_1, \ldots, c_k$,

so as to minimize the cost function – the maximum radius of clusters,

$$\max_i \max_{x \in S_i} \|x - c_i\|.$$

k-center clustering example



Rapid decay of the Gaussian

Consider only influential clusters.

$$\widehat{G}(y_j) = \sum_{\|y_j - c_k\| \leq r_y^k} \sum_{|\alpha| \leq p_{max}-1} C_\alpha^k e^{-\|y_j - c_k\|^2/h^2} \left(\frac{y_j - c_k}{h}\right)^\alpha,$$

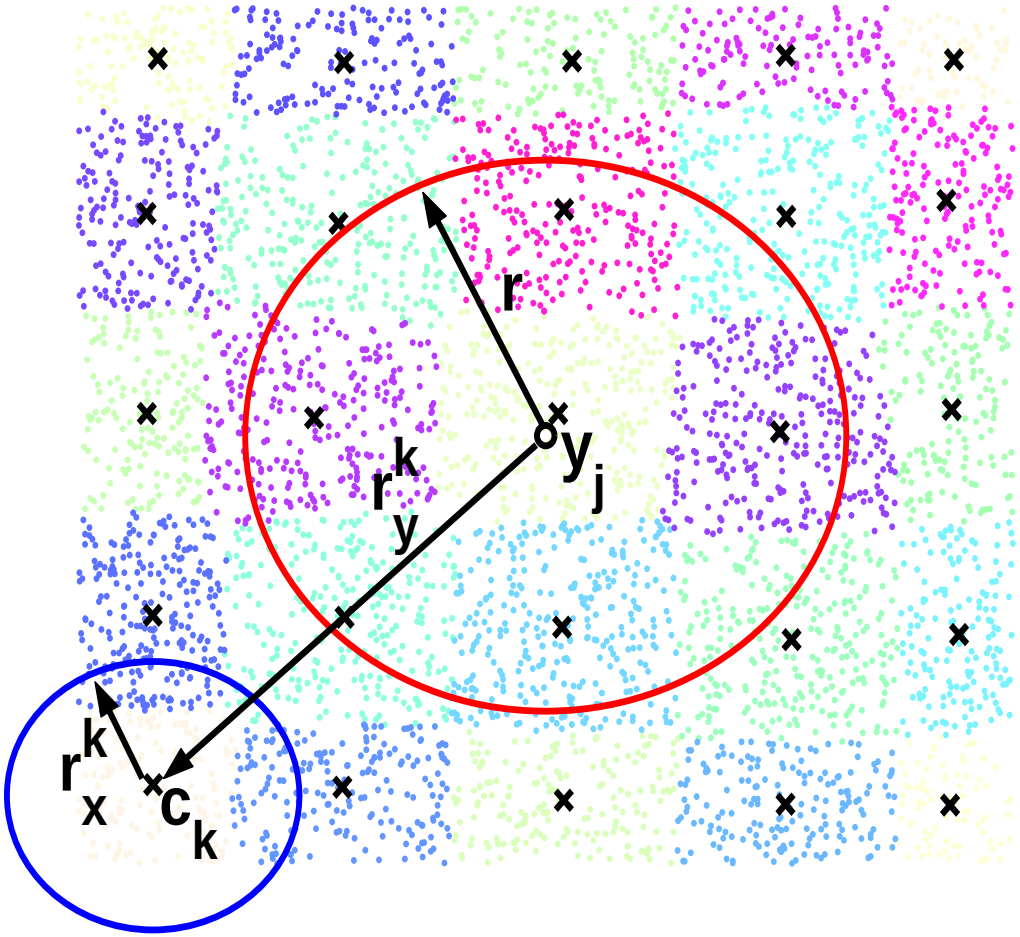
where,

$$C_\alpha^k = \frac{2^\alpha}{\alpha!} \sum_{x_i \in S_k} q_i e^{-\|x_i - c_k\|^2/h^2} \left(\frac{x_i - c_k}{h}\right)^\alpha 1_{|\alpha| \leq p_i - 1}.$$

IFGT

- **Step 0** Choose the parameters.
- **Step 1** Subdivide the source points into K clusters.
- **Step 2** Compute the cluster coefficients at the center of each cluster.
- **Step 3** For each target point sum the contribution from influential clusters.

IFGT Illustration



Computational complexity

$$\mathcal{O}\left(N \log K + N r_{(p_{max}-1)d} + M n r_{(p_{max}-1)d}\right).$$

$$r_{(p_{max}-1)d} = \binom{p_{max}+d-1}{d}$$

is

the total number of d-variate monomials of degree less than or equal to $p_{max} - 1$.

The d-variate monomials can be efficiently evaluated using the Horner's rule.

Storage complexity

$$\mathcal{O}(Kr_{(p_{max}-1)d} + N + M)$$

Choosing the parameters

Given any $\epsilon > 0$, we want to choose the following parameters

- K (the number of clusters),
- $\{p_i\}_{i=1}^N$ (the truncation number for each source point x_i),
- and the cut off radius $\{r_y^k\}_{k=1}^K$ for each cluster

  such that for **any** target point y_j we can guarantee that

$$\frac{|\widehat{G}(y_j) - G(y_j)|}{Q} \leq \epsilon,$$

where $Q = \sum_{i=1}^N |q_i|$.

Point-wise error bounds

Define Δ_{ij} to be the error contributed by the i^{th} source x_i.

$$|\widehat{G}(y_j) - G(y_j)| = \left| \sum_{i=1}^{N} \Delta_{ij} \right| \leq \sum_{i=1}^{N} |\Delta_{ij}| \leq Q\epsilon = \sum_{i=1}^{N} |q_i|\epsilon.$$

One way to achieve this is to let

$$|\Delta_{ij}| \leq |q_i|\epsilon \ \forall i = 1, \ldots, N.$$

- Can get tighter bounds.
- Easier to choose a different truncation number for each source.

A source can err in two ways

1. Due to ignoring the cluster to which it belongs.

   $\Delta_{ij} = q_i e^{-\|y_j - x_i\|^2/h^2}$ if $\|y_j - c_k\| > r_y^k$.

2. Due to truncation of the Taylor's series.

   $\Delta_{ij} = q_i e^{-\|x_i - c_k\|^2/h^2} e^{-\|y_j - c_k\|^2/h^2} error_{p_i}$ if $\|y_j - c_k\| \leq r_y^k$.

Cutoff radius for each cluster

$$r_y^k = r_x^k + \min\left(R, h\sqrt{\ln(1/\epsilon)}\right).$$

- r_x^k is the radius of the cluster S_k.

- R is the maximum distance between any source and target point.

Truncation number for each source

$$error_{p_i} \leq \frac{2^{p_i}}{p_i!} \left(\frac{\|x_i - c_k\|}{h} \right)^{p_i} \left(\frac{\|y_j - c_k\|}{h} \right)^{p_i} e^{2\|x_i - c_k\|\|y_j - c_k\|/h^2}.$$

Hence

$$\Delta_{ij} \leq q_i \frac{2^{p_i}}{p_i!} \left(\frac{\|x_i - c_k\|}{h} \right)^{p_i} \left(\frac{\|y_j - c_k\|}{h} \right)^{p_i} e^{-(\|x_i - c_k\| - \|y_j - c_k\|)^2/h^2}.$$

We will have to bound this such that it is independent of $\|y_j - c_k\|$.

Truncation number for each source

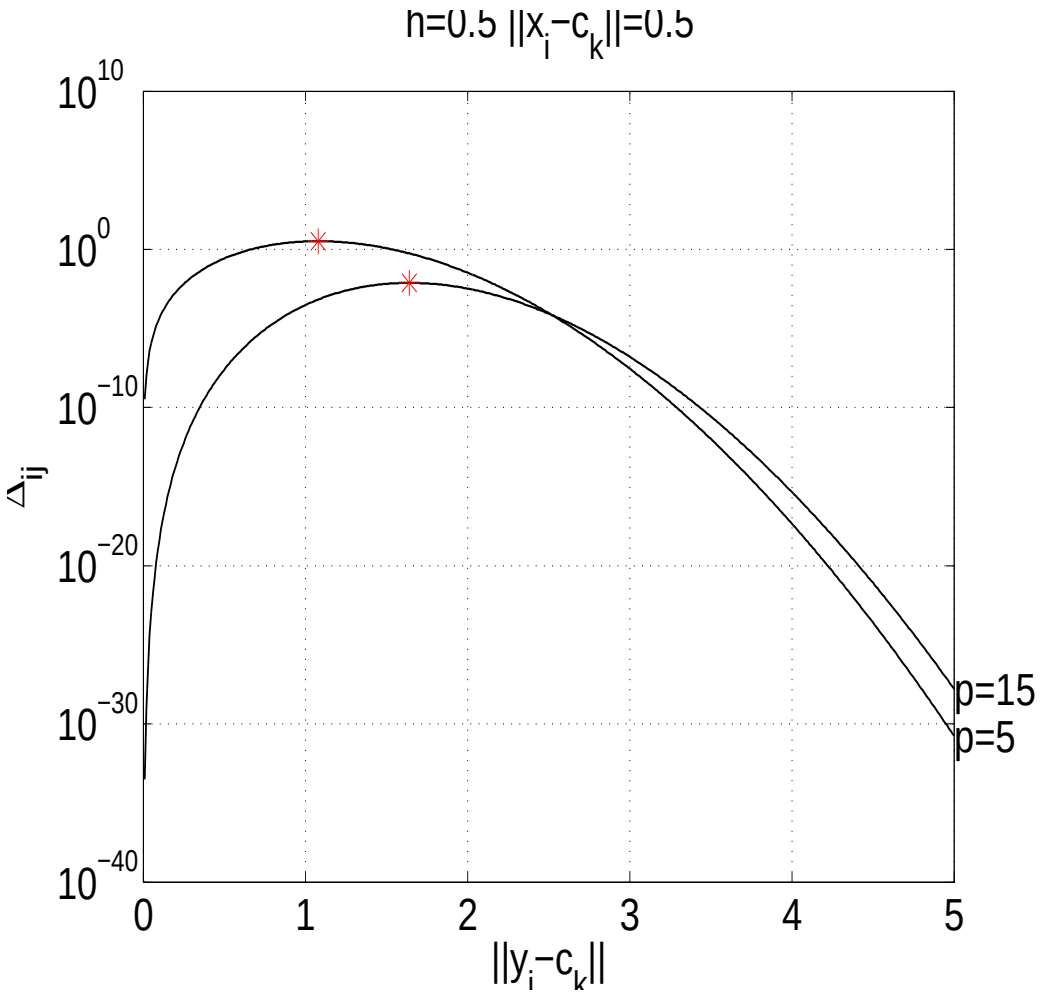
The error increases as a function of $\|y_j - c_k\|$,

reaches a maximum

and then starts decreasing.

The maximum is attained at

$$\|y_j - c_k\| = \|y_j - c_k\|_* = \frac{\|x_i - c_k\| + \sqrt{\|x_i - c_k\|^2 + 2p_i h^2}}{2}.$$

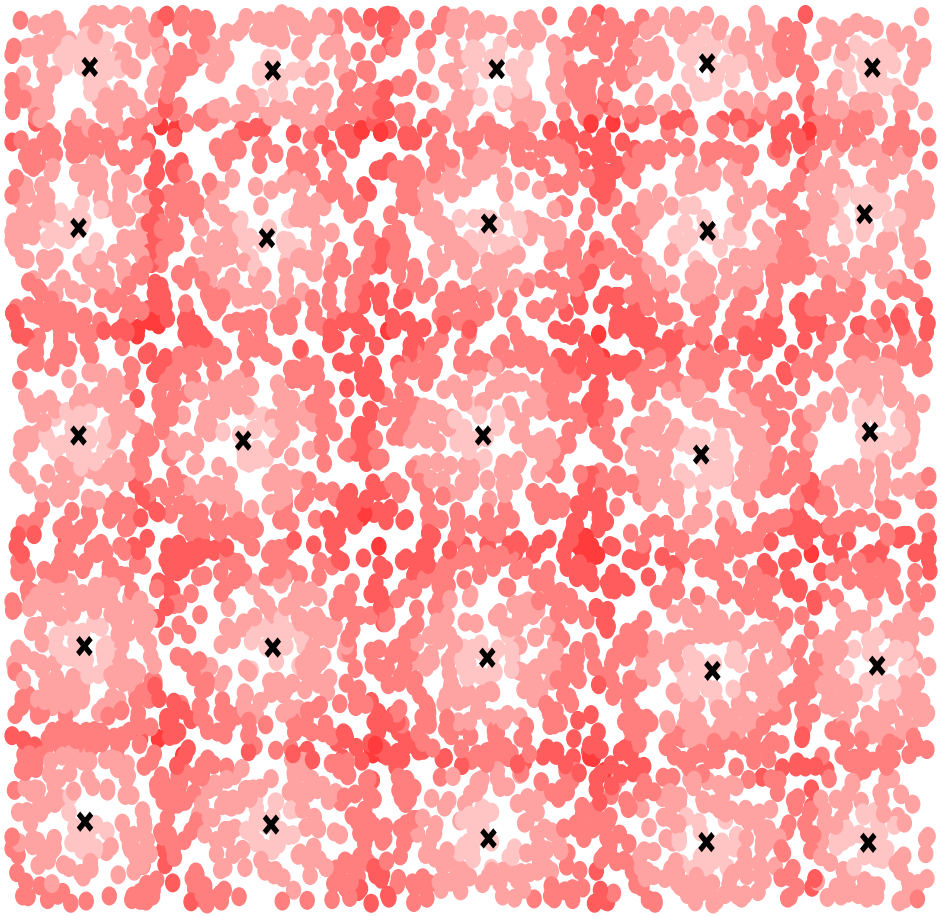

h=0.5 ||x_i−c_k||=0.5
10^10
10^0
10^−10
10^−20
10^−30
10^−40
Δ_ij
0
1
2
3
4
5
||y_i−c_k||
p=15
p=5

Truncation number for each source

Hence out strategy for choosing p_i is

$$|\Delta_{ij}| \Big|_{\|y_j - c_k\| = \|y_j - c_k\|_*} \leq |q_i|\epsilon.$$

Truncation numbers for different source points



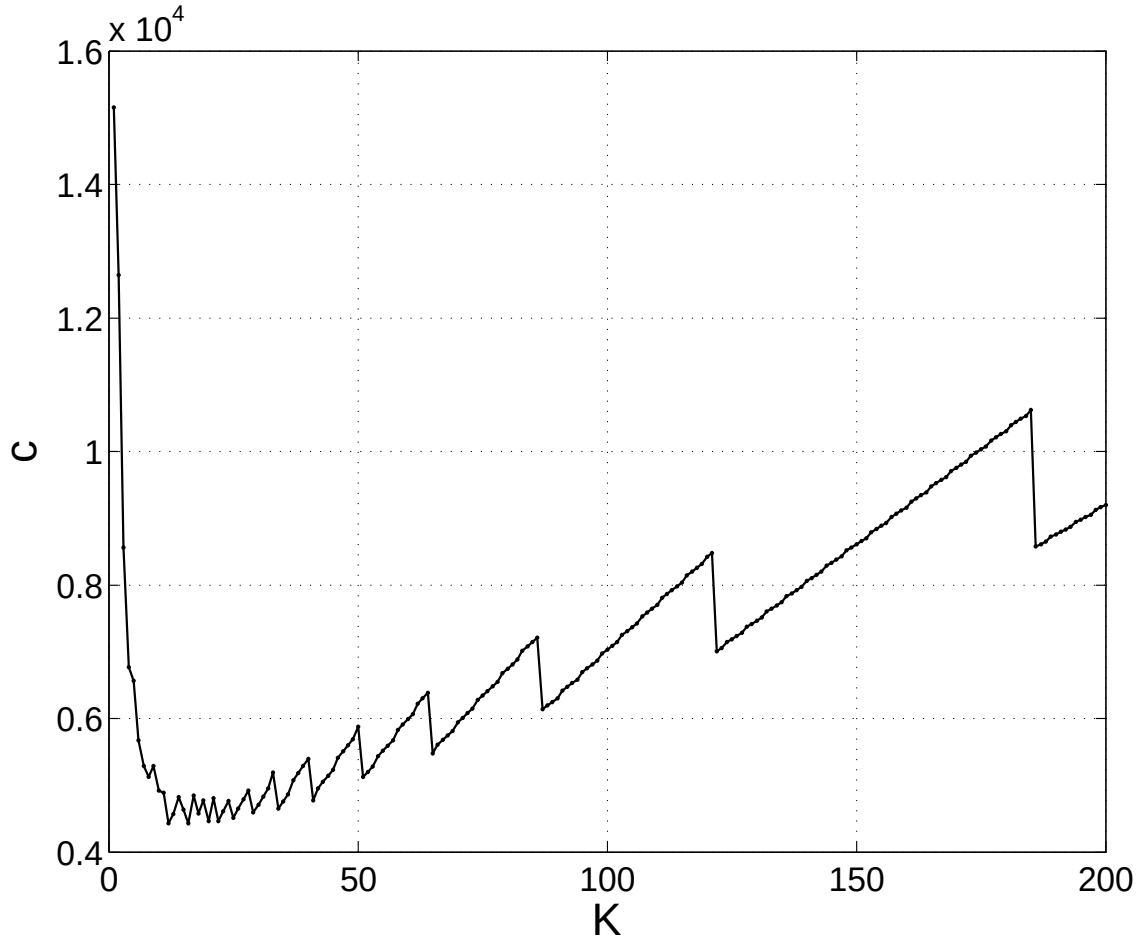
Choosing the number of clusters.

We optimize K assuming a uniform distribution of source points.

Choose K such that the constant term in the complexity is minimum

$$c = \log K + (1 + n) r_{(p_{max}-1)d}.$$

Constant term as a function of K



Fast multipole methods

- The FGT belongs to a more general class of methods called fast multipole methods *.

- The fast multipole method has been called one of the ten most significant algorithms in scientific computation discovered in the 20th century† and won its inventors, Vladimir Rokhlin and Leslie Greengard, the 2001 Steele prize, in addition to getting Greengard the ACM 1987 best dissertation award.

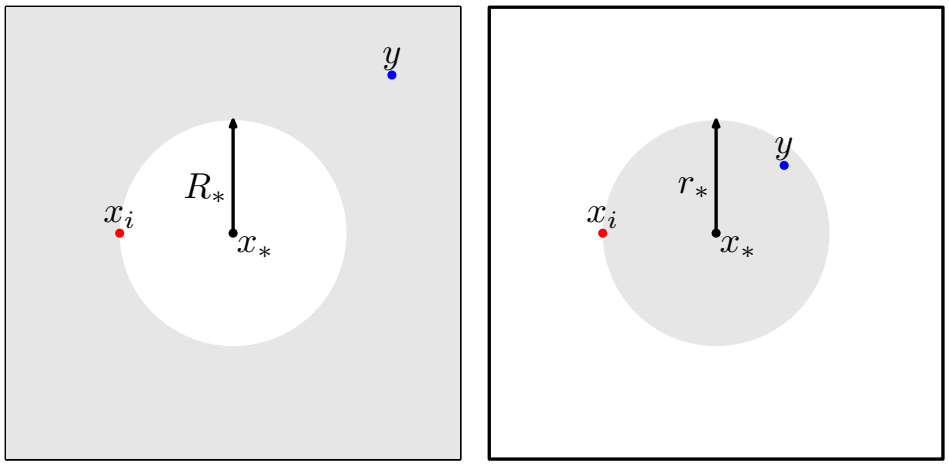
*L. F. Greengard and V. Rokhlin. A fast algorithm for particle simulation. Journal of Computational Physics, 73(2):325–348, 1987.

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

Expansions

The general fast multipole methods use two kinds of factorization

Far-field expansion and Local expansion.



Comparison with FGT expansions

$$e^{-\|y-x_i\|^2/h^2} = \sum_{\alpha \geq 0} \left[\frac{1}{\alpha!} \left(\frac{x_i - x_*}{h} \right)^\alpha \right] h_\alpha \left(\frac{y - x_*}{h} \right) \text{ [far-field Hermite expansion]}$$

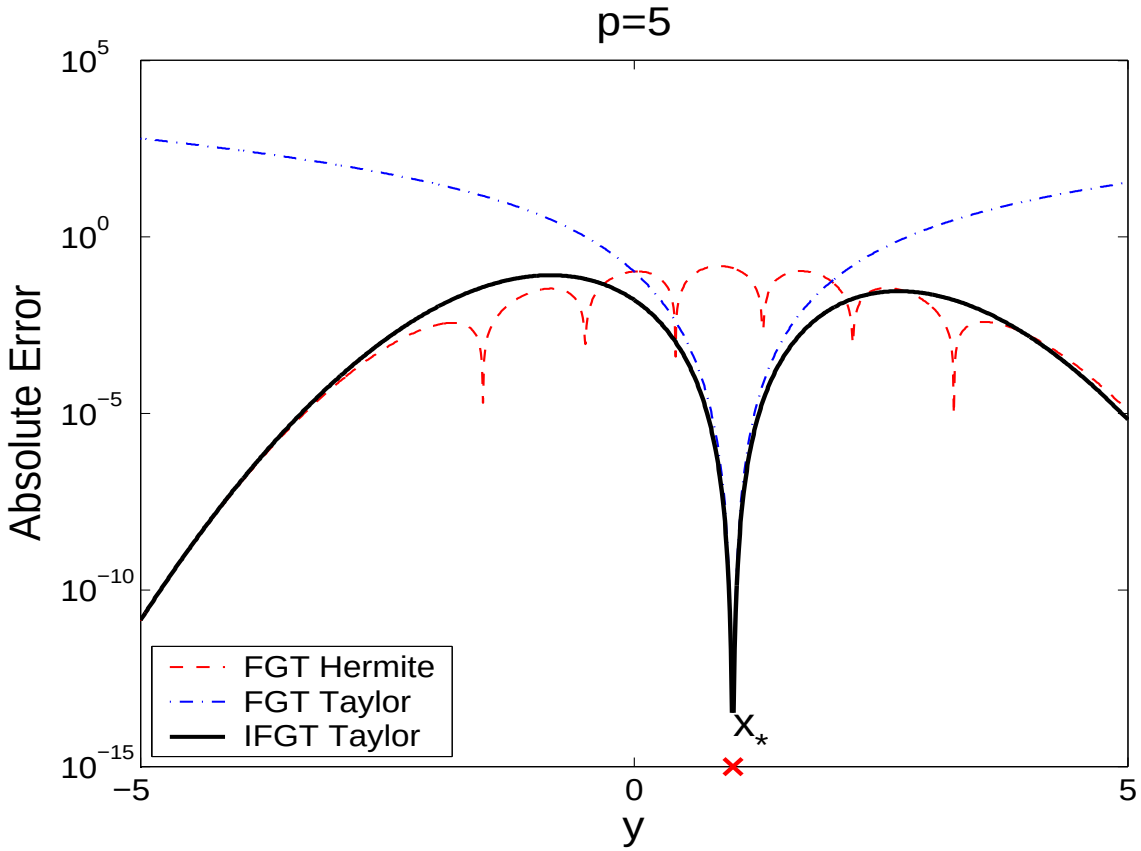
$$e^{-\|y-x_i\|^2/h^2} = \sum_{\beta \geq 0} \left[\frac{1}{\beta!} h_\beta \left(\frac{x_i - x_*}{h} \right) \right] \left(\frac{y - x_*}{h} \right)^\beta \text{ [local Taylor expansion]}$$

Compare this with the single IFGT expansion

$$e^{-\|y-x_i\|^2/h^2} = \sum_{|\alpha| \geq 0} \left[\frac{2^\alpha}{\alpha!} e^{-\|x_i - x_*\|^2/h^2} \left(\frac{x_i - x_*}{h} \right)^\alpha \right] e^{-\|y_j - x_*\|^2/h^2} \left(\frac{y_j - x_*}{h} \right)^\alpha .$$

IFGT expansion is both local as well as far-field

Hence we avoid the expensive translation operation.



FGT vs IFGT complexity

FGT

$$O(p^d N) + O(p^d M) + O(dp^{d+1}(2n+1)^d min((\sqrt{2}rh)^{-d/2}, M)).$$

IFGT

$$O(\log KN) + O(r_{(p-1)^d} N) + O(nr_{(p-1)^d} M).$$

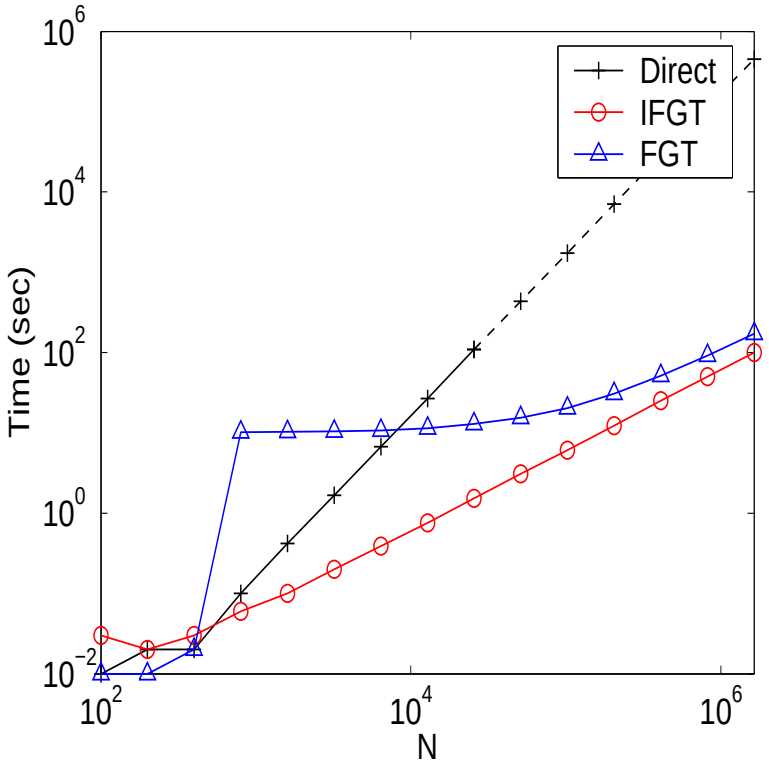
FGT-Explosive growth with d

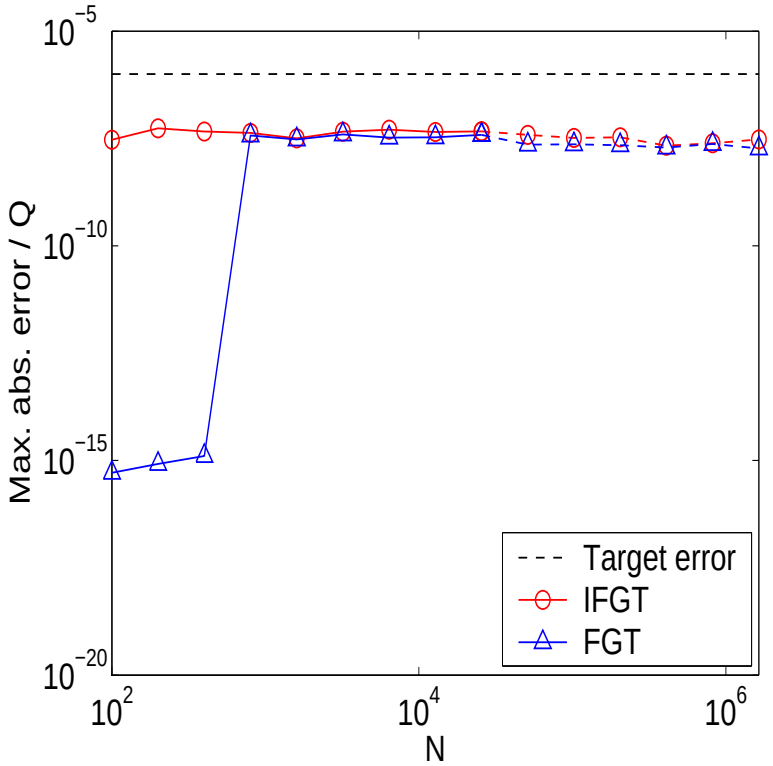
d	FGT					IFGT		
	# of boxes (N_{side}^d)	p	# of terms (p^d)	n	Constant term	K	p	# of terms ($r_{(p-1)d}$)
1	3	9	9	2	7.0+002	5	9	9
2	9	10	100	2	1.5e+005	7	15	120
3	27	10	1000	2	1.9e+007	15	16	816
4	81	11	14641	2	3.6e+009	29	17	4845
5	243	11	161051	2	4.3e+011	31	20	42504
6	729	12	2985984	2	9.0e+013	62	20	177100
7	2187	14	105413504	2	3.7e+016	67	22	1184040

Numerical experiments

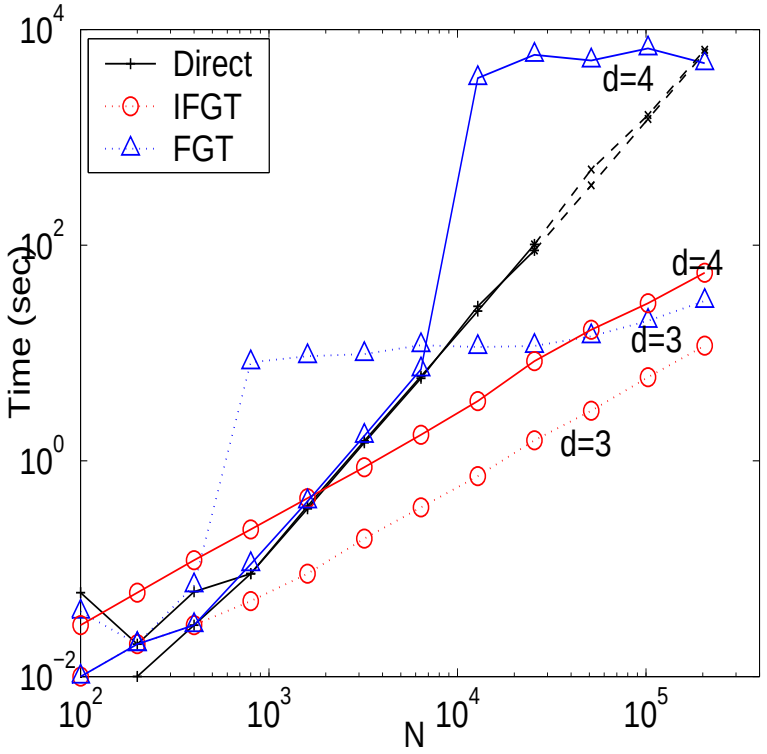
- Programmed in C++ with MATLAB bindings.
- Run on a 1.6 GHz Pentium M processor with 512Mb of RAM.
- Code available.

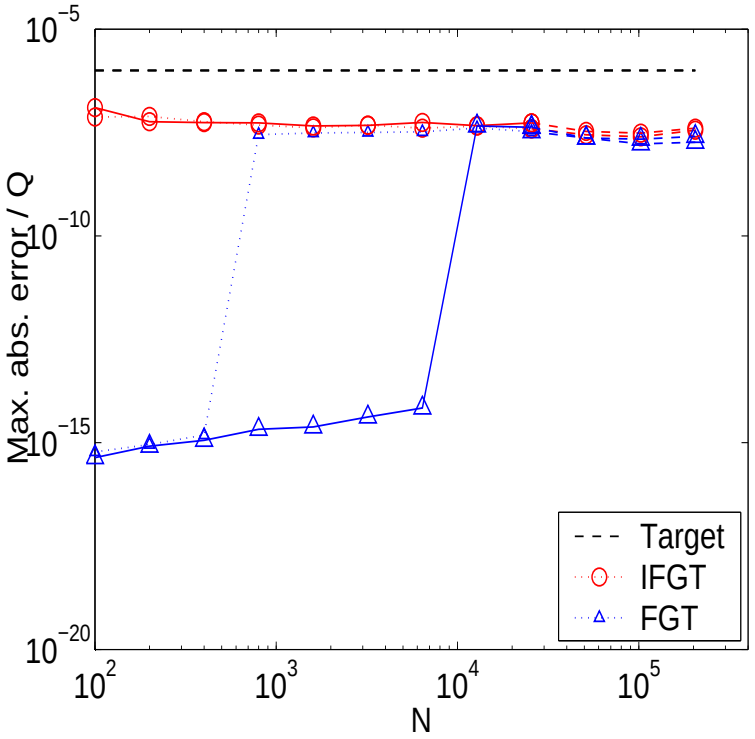
Speedup as a function of N [$d = 3$ and $h = 1.0$]



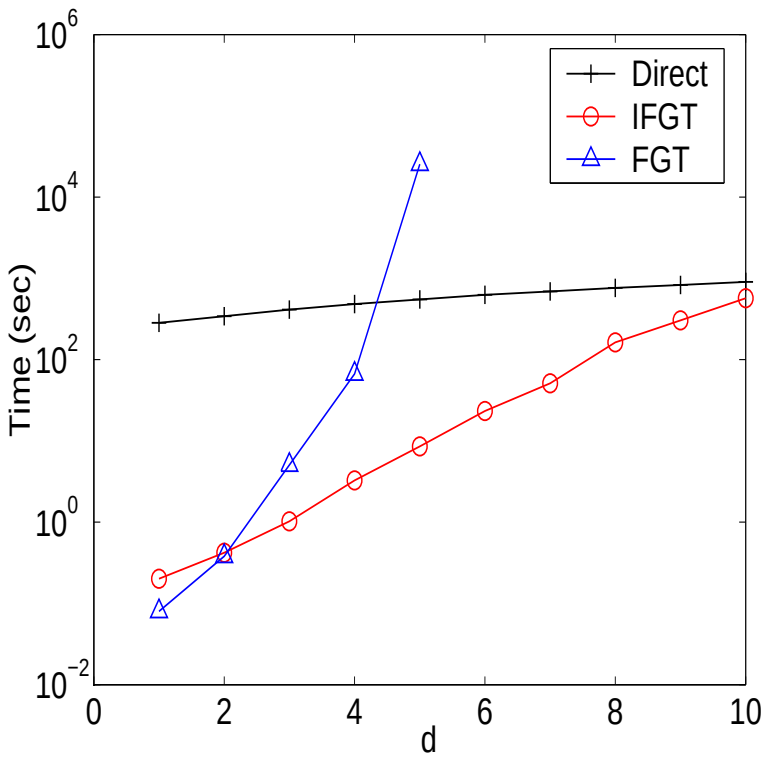


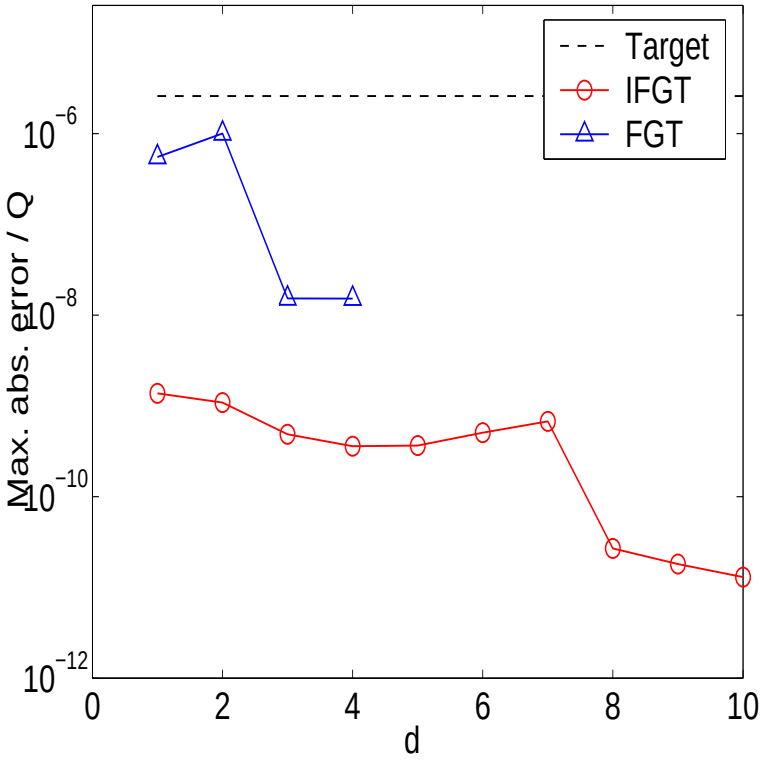
Speedup as a function of N and d



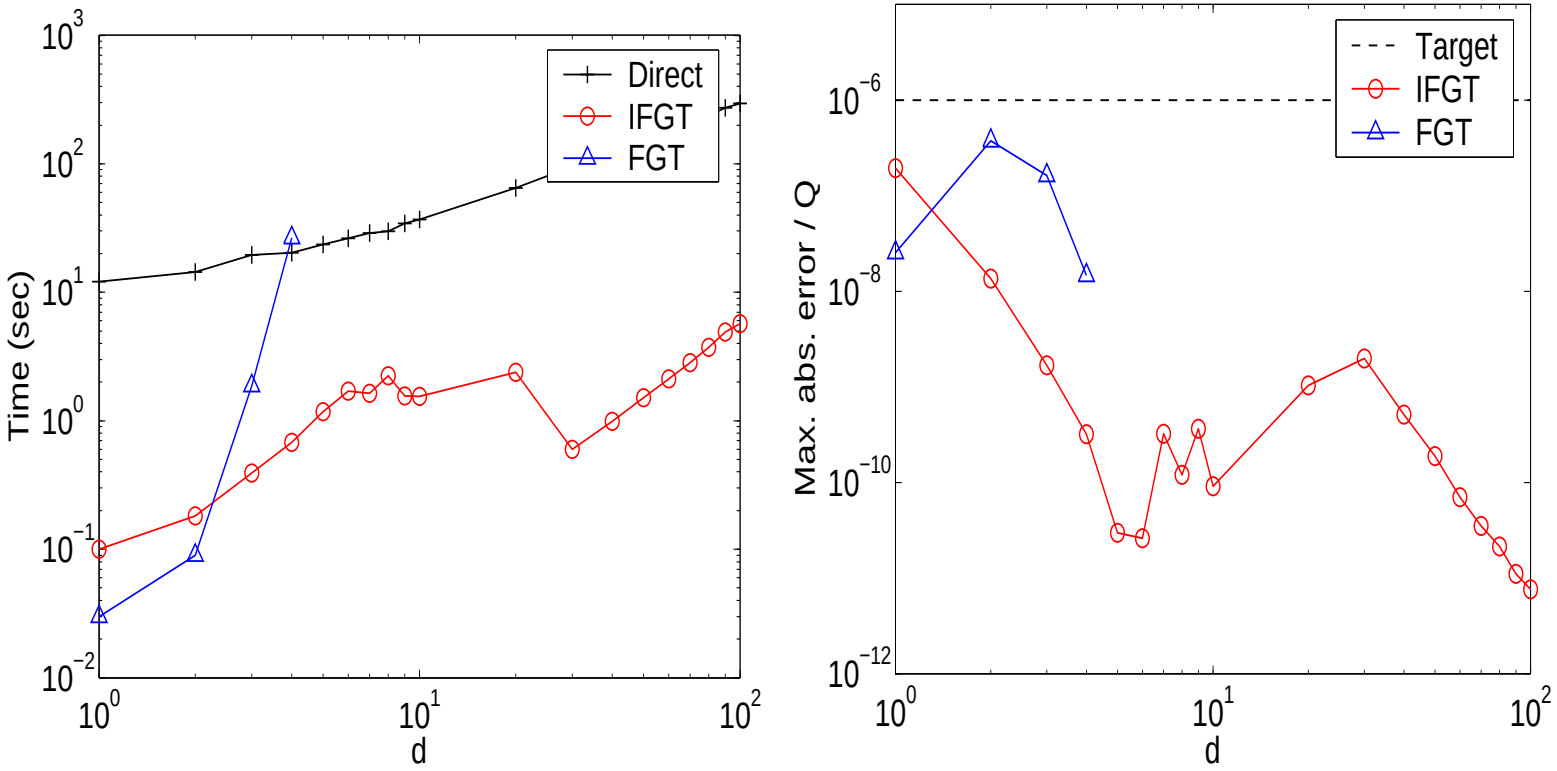


Speedup as a function of d [$h = 2.0$]

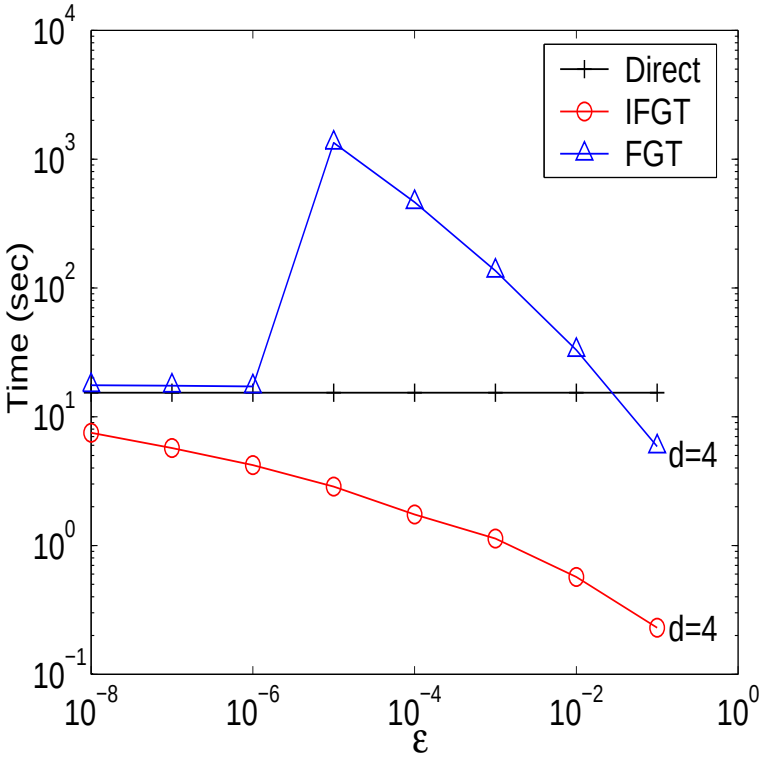


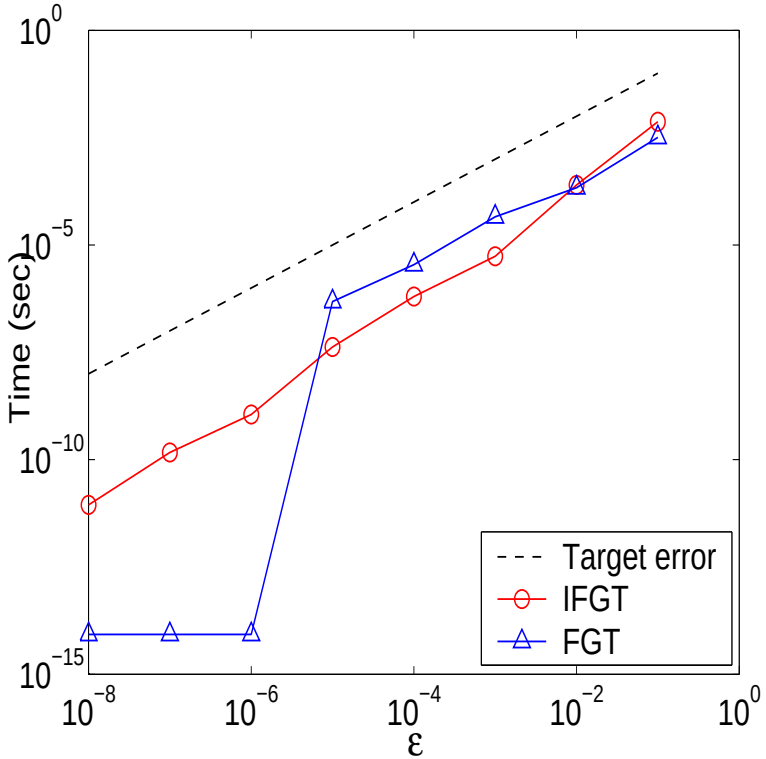


Speedup as a function of d [$h = \sqrt{d}$]

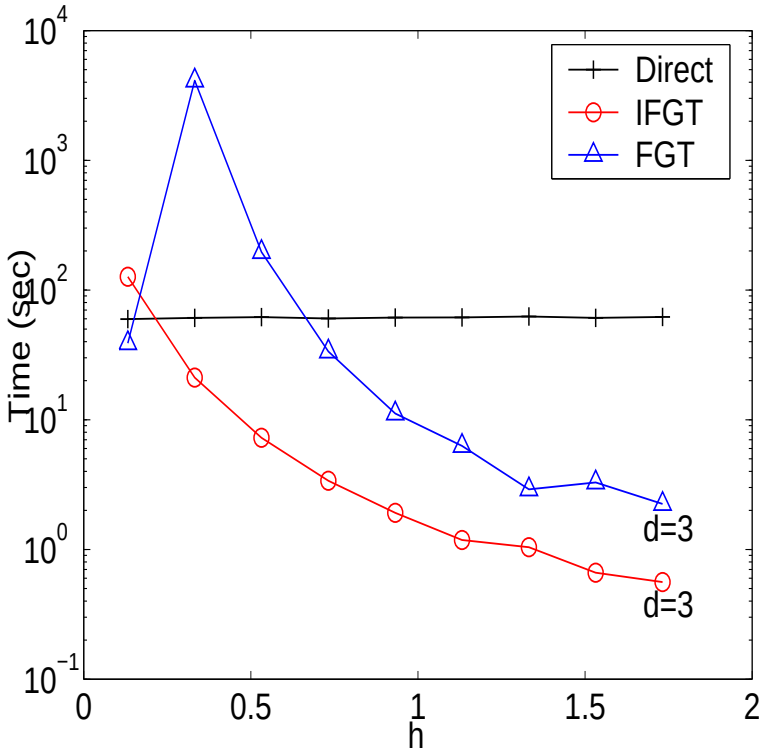


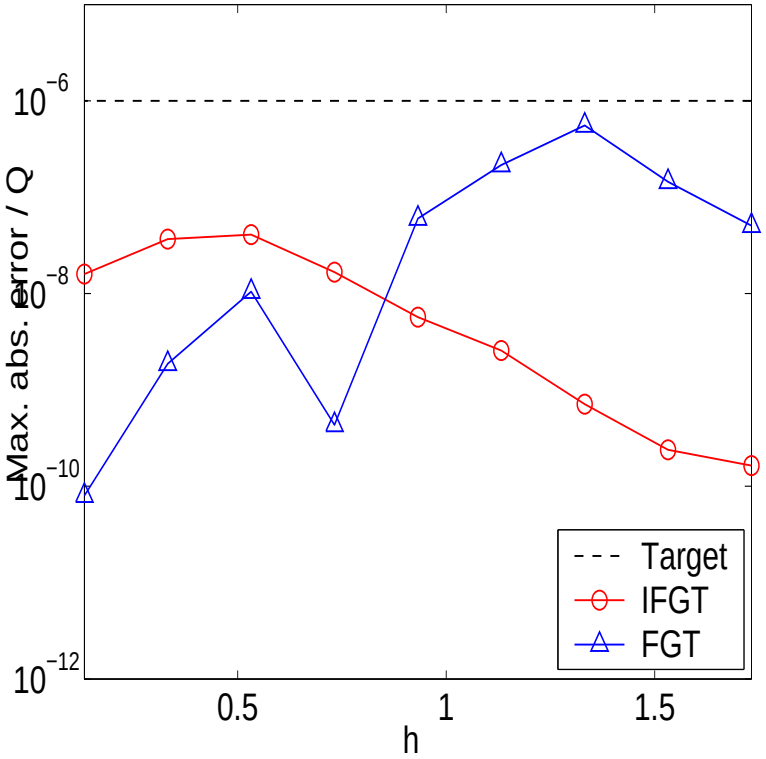
Speedup as a function of ϵ





Speedup as a function of h





Other related methods

- Methods based on sparse data-set representation.
- Binned Approximation based on FFT*.
- Dual-tree methods†.

*B. W. Silverman. Algorithm AS 176: Kernel density estimation using the fast Fourier transform. Journal of Royal Statistical society Series C: Applied statistics, 31(1):93–99, 1982.

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

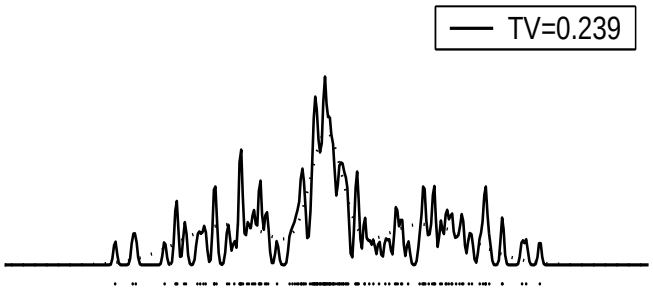
Some Extensions

IFGT with variable source scales

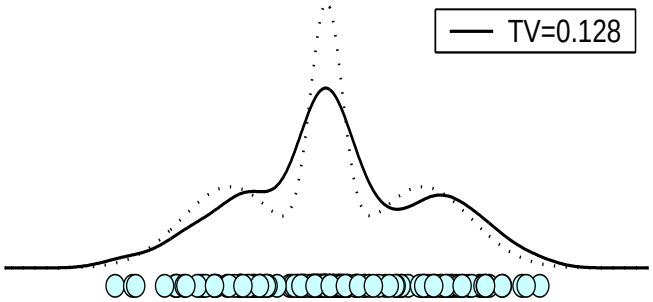
$$G(y_j) = \sum_{i=1}^{N} q_i e^{-\|y_j - x_i\|^2 / h_i^2}.$$

Variable bandwidth density estimation

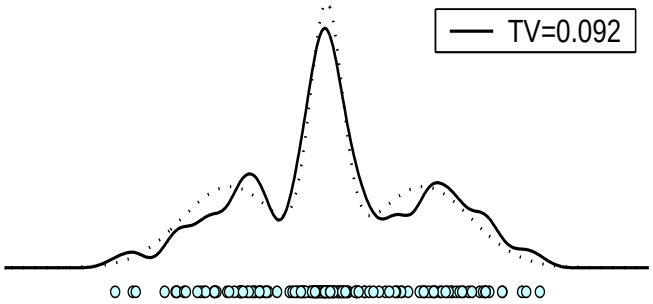
(a) h=0.05



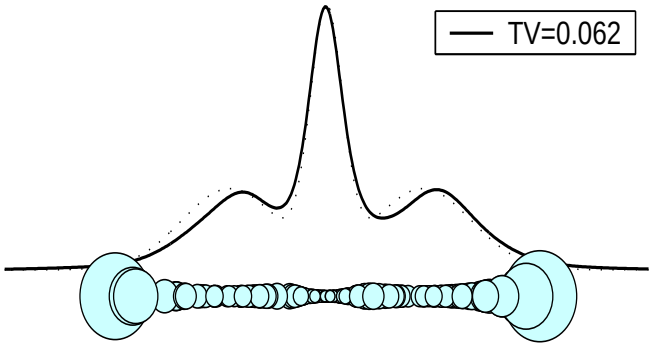
(b) h=0.70



(c) h=0.36



(d) Variable h



Segmentation using adaptive mean-shift

1.34 hours vs 2.1 minutes

Optimal bandwidth estimation for KDE

Most automatic bandwidth selection procedures for kernel density estimation require estimates of quantities involving the density derivatives.

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

Gaussian Process Regression

- Coupled with the Conjugate-gradient the IFGT reduces the computational cost of GP regression to $\mathcal{O}(N)$.

- For example for N=25,600 training takes around 3 secs. (compare to 10 hours[direct] or 17 minutes[CG]).

GP regression

$\widetilde{\mathbf{K}} = \mathbf{K} + \sigma^2\mathbf{I}$	*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 $\mathbf{k}(x,x)$ $-\mathbf{k}(x)^T\widetilde{\mathbf{K}}^{-1}\mathbf{k}(x)$	$\mathcal{O}(N^3)$	$\mathcal{O}(N)$	$\mathcal{O}(N^3)$	$\mathcal{O}(N)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$

How to choose ϵ

Use the theory of inexact Krylov subspace methods*

$$\epsilon_k \leq \frac{\delta}{N} \frac{\|\mathbf{y} - \widetilde{\mathbf{K}}\xi_0\|}{\|\tilde{r}_{k-1}\|}.$$

This guarantees that

$$\|\mathbf{y} - \widetilde{\mathbf{K}}\xi_k\|_2 \leq (\eta + \delta)\|\mathbf{y} - \widetilde{\mathbf{K}}\xi_0\|_2.$$

Matrix-vector product may be performed in an increasingly inexact manner as the iteration progresses and still allow convergence to the solution.

*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.