Automatic online tuning for fast Gaussian summation

Vlad I. Morariu, Balaji V. Srinivasan, Vikas C. Raykar, Ramani Duraiswami, Larry S. Davis

UMIACS University of Maryland Institute for Advanced Computer Studies

Problem

$$G(\mathbf{y}_j) = \sum_{i=1}^{N} q_i \exp\left(-\frac{||\mathbf{x}_i - \mathbf{y}_j||^2}{h^2}\right), \quad j = 1, \ldots, M$$

- Direct computation is quadratic, O(MN)
- Slows down kernel machines
- Algorithms such as FGT [1] and IFGT [2] obtain speedup by approximating to specified error ε, but do not perform well across all bandwidths *h*, point distributions, and error ε and need tuning

Our solution

- Uses the IFGT and a tree data structure
- Four methods, each optimal in different situations
- **Black box approach:** Automatically predicts fastest method and tunes its parameters for the given dataset
- Compares favorably to Dual-Tree methods [3]

Improved Fast Gauss Transform

- Reduces cost of Gauss Transform to O(*N*+*M*) [2]
- Uses truncated Taylor expansion of exponential term

$$e^{-\|y_j - x_i\|^2/h^2} = e^{-\|x_i - x_*\|^2/h^2} e^{-\|y_j - x_*\|^2/h^2} \left[\sum_{|\alpha| \le p-1} \frac{2^\alpha}{\alpha!} \left(\frac{y_j - x_*}{h}\right)^\alpha \left(\frac{x_i - x_*}{h}\right)^\alpha\right] + \Delta_{ij}$$

- The error of truncating series starting at p^{th} term is bounded by

$$\Delta_{ij} \le \frac{2^p}{p!} \left(\frac{||x_i - x_*||}{h}\right)^p \left(\frac{||y_j - x_*||}{h}\right)^p e^{-(||x_i - x_*|| - ||y_j - x_*||)^2/h^2}$$

- Truncation number is chosen for each source point by assuming worst possible placement of a target point and choosing p s.t.

  $\Delta_{ij} \le \epsilon$ which guarantees that $|\hat{g}(y_j) - g(y_j)|/Q \le \epsilon$

  where Q is the sum of all q_i

Summary of approach:

1) Choose number of centers *K* and max truncation p_{max}
2) Cluster source points using *K*-center algorithm
3) Pre-compute contributions from sources

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

4) For each target y_j evaluate by collecting contributions from clusters within cut-off radius

$$\hat{g}(y_j) = \sum_{||y_i - c_k|| \le r_k^y} \sum_{|\alpha| \le p_{\max}-1} C_\alpha^k e^{-\frac{||y_j - c_k||^2}{h^2}} \left(\frac{y_j - c_k}{h}\right)^\alpha$$

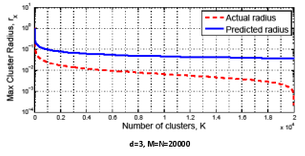
Problems:

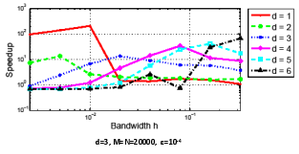
- Parameters are optimized for uniform distributions
- IFGT performs poorly for low bandwidths where kernels are local

Automatic IFGT parameter tuning

Choosing number of clusters without assuming a uniform distribution

- Instead of estimating cluster radius as $k^{-1/d}$, use incremental clustering to evaluate actual cluster radius, thus choosing optimal K based on *actual* distribution





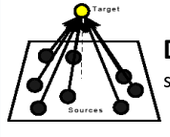
Choosing cluster-wise truncation numbers

- Allow some points to have higher error as long as other points in the same cluster have low enough error to compensate
- Ensure that for each cluster S_k

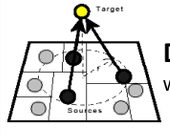
$$\sum_{x_i \in S_k} |q_i| \Delta_{ij} \le \sum_{x_i \in S_k} |q_i| \epsilon = Q_k \epsilon$$

Tree data structure

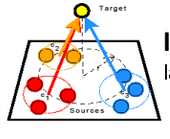
- Using a tree data structure, we can compute Gauss Transform efficiently across bandwidths
- Must choose one of four evaluation methods



Direct: evaluate equation directly; works well for small N and M, and does not have high overhead cost



Direct+Tree: build tree directly on sources; works well for small bandwidths



IFGT: works well for large N and M, and medium to large bandwidths

IFGT+Tree: build tree on cluster centers, works well for case where number of clusters K is large

Comparison of methods

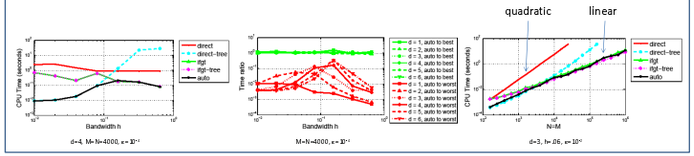
$Cost_{\text{direct}}(d, N, M)$	$O(dMN)$
$Cost_{\text{direct+tree}}(d, N, M, n_s)$	$O(d(N + Mn_s)\log N)$
$Cost_{\text{ifgt}}(d, N, M, K, n_c, p_{\max})$	$O(dN \log K + (N + Mn_c)r_{(p_{\max}-1)d} + dMK)$
$Cost_{\text{ifgt+tree}}(d, N, M, K, n_c, p_{\max})$	$O((N + Mn_c)(d \log K + r_{(p_{\max}-1)d}))$

Automatic method selection

- n_s and n_c are the average number of sources and cluster centers, respectively, that are within the cut-off radius of a query target
- n_s and n_c can be approximated from sub-sampled dataset
- Given *d, N, M, K,* n_c, and n_s, we can estimate cost of each method

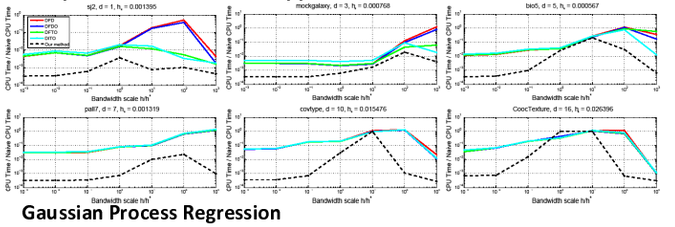
Summary of method selection approach:

1) Estimate n_s
2) Calculate $\text{Cost}_{\text{direct}}(d,N,M)$ and $\text{Cost}_{\text{direct+tree}}(d,N,M,n_s)$
3) Estimate highest K_{limit} for which *ifgt* and *ifgt+tree* could be faster
4) If $K_{limit} > 0$
   a) Compute IFGT parameters K and p_{max}
   b) Estimate n_c ; estimate $\text{Cost}_{\text{ifgt}}$ and $\text{Cost}_{\text{ifgt+tree}}$
5) Return arg $\min_i \text{Cost}_i$



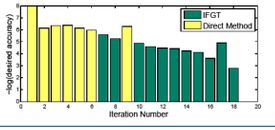
Results

Comparison with Dual-Tree approach



Gaussian Process Regression

	Robotarm	Abalone	Housing	Elevator
Dims	2	7	12	18
Size	1000	4177	506	8752
direct	0.578s	**16.1s**	**0.313s**	132s
ifgt	**0.0781s**	52.3s	1317s	133s
direct-tree	5.45s	328s	2.27s	**0.516s**
ifgt-tree	**0.0781s**	35.2s	549s	101s
auto	**0.0938s**	**14.5s**	**0.547s**	**0.797s**



References

[1] L. Greengard and J. Strain. The fast Gauss transform. SIAM J. Sci. Stat. Comput., 1991

[2] V. Raykar, C. Yang, R. Duraiswami, and N. Gumerov. Fast computation of sums of Gaussians in high dimensions. UMD-CS-TR-4767, 2005

[3] D. Lee and A. G. Gray. Faster Gaussian summation: Theory and experiment. In UAI, 2006

Open source C/C++ and MATLAB bindings

http://sourceforge.net/projects/figtree