Dual-Tree Fast Gauss Transforms

Dongryeol Lee (dongryel@cc.gatech.edu), Georgia Institute of Technology
Alexander Gray (agray@cc.gatech.edu), Georgia Institute of Technology
Andrew Moore (awm@cs.cmu.edu), Carnegie Mellon University

Problem

Kernel summations using the Gaussian kernel $K(||x_q - x_r||) = e^{\frac{-||x_q - x_r||^2}{2h^2}}$ with bandwidth h are common in many machine learning methods and physics problems. Given a set of reference points X_R and a set of query points X_Q in a metric space, the goal is to compute $\forall x_q \in X_Q$ the estimate $\tilde{G}(x_q)$ of $G(x_q) = \sum_{x_r \in X_R} K(||x_q - x_r||)$ as fast as possible while ensuring the user-specified error tolerance ϵ: $\forall x_q \in X_Q, \frac{|\tilde{G}(x_q) - G(x_q)|}{G(x_q)} \leq \epsilon$

Previous Approaches and Our Contribution

Discrete/geometric aspect The dual-tree framework [5], a recursive approach utilizing adaptive data structures (e.g. kd-tree), generalizes all of the well-known algorithms [1, 2, 4, 7] with automatic error control using finite-difference approximation.

Continous/approximation aspect The original Fast Gauss Transform [8] used multipole expansion on a flat grid, with associated error bounds (that were incorrect and later corrected by [3]). An approach based on a flat-clustering scheme [11] also utilized multipole-like approximation for Gaussian summations but also had incorrect error bounds. Both of these approaches, however, did not use a hierarchical structure as done in [7] for the Coulombic kernel, and require non-automatic parameter tweaking.

We propose a new method for Gaussian summation by combining the best tools from discrete algorithms (dual-tree methodology) and continuous approximation theory (multipole expansion). Performing multipole expansions for the Gaussian kernel within a hierarchical context is non-trivial and requires derivation of two new translational operators with two associated error bounds.

References

[1] A. W. Appel. An Efficient Program for Many-Body Simulations. *SIAM Journal on Scientific and Statistical Computing*, 6(1):85–103, 1985.

[2] J. Barnes and P. Hut. A Hierarchical $O(NlogN)$ Force-Calculation Algorithm. *Nature*, 324, 1986.

[3] B. Baxter and G. Roussos. A New Error Estimate for the Fast Gauss Transform. *SIAM Journal on Scientific Computing*, 24(1):257–259, 2002.

[4] P. Callahan and S. Kosaraju. A Decomposition of Multidimensional Point Sets with Applications to K-Nearest-Neighbors and N-Body Potential Fields. *Journal of the ACM*, 62(1):67–90, January 1995.

[5] A. Gray and A. W. Moore, N-Body Problems in Statistical Learning, In T. K. Lenn, T. G. Diettrich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13 (December 2000)*, MIT Press, 2001.

[6] A. Gray and A. W. Moore, Rapid Evaluation of Multiple Density Models, Artificial Intelligence and Statistics 2003, 2003

[7] L. Greengard and V. Rokhlin. A Fast Algorithm for Particle Simulations. *Journal of Computational Physics*, 73, 1987.

[8] L. Greengard and J. Strain, The Fast Gauss Transform, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 79–94.

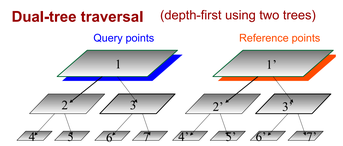
[9] D. Lee and A. Gray, Hierarchical Fast Gauss Transforms: Better Error Control and Higher Dimensions, Submitted to SIAM Data Mining, under review.

[10] B. W. Silverman, Density Estimation for Statistics and Data Analysis, Chapman and Hall, 1986.

[11] C. Yang, R. Duraiswami, N. A. Gumerov, and L. Davis, Improved Fast Gauss Transform and Efficient Kernel Density Estimation, *International Conference on Computer Vision*, 2003.

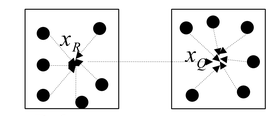
Three Translation Operators

The first translation operator transfers the contribution of a reference node R into the Taylor series centered about x_Q in a query node Q. The next two operators are newly derived. $H2H$ operator allows efficient precomputation of the Hermite moments in the reference tree in a bottom-up fashion from its children, whereas $L2L$ operator combines the approximations at different scales through one breadth-first traversal.

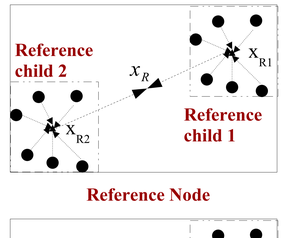


Order: (1, 1'), (2, 2'), (4, 4'), (4, 5'), (5, 4'), (5, 5'), (2, 3'), (4, 6'), (4, 7'), (5, 6'), (5, 7'), (3, 2'), (6, 4'), (6, 5'), (7, 4'), (7, 5'), (3, 3'), (6, 6'), (6, 7'), (7, 6'), (7, 7')

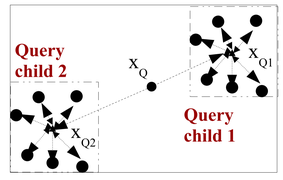
Lemma 1: *Hermite-to-local (H2L) translation operator* (Lemma 2.2 in [8]): Given a reference node R, a query node Q, and the Hermite expansion centered at a centroid x_R of R: $G(x_q) = \sum_{\alpha \geq 0} A_\alpha h_\alpha \left(\frac{x_q - x_R}{\sqrt{2h^2}}\right)$, where $A_\alpha = \frac{1}{\alpha!}\left(\frac{x_r - x_R}{\sqrt{2h^2}}\right)^\alpha$, the Taylor expansion of the Hermite expansion at the centroid x_Q of Q is given by: $G(x_q) = \sum_{\beta \geq 0} B_\beta \left(\frac{x_q - x_Q}{\sqrt{2h^2}}\right)^\beta$ where $B_\beta = \frac{(-1)^\beta}{\beta!} \sum_{\alpha \geq 0} A_\alpha h_{\alpha+\beta}\left(\frac{x_Q - x_R}{\sqrt{2h^2}}\right)$.



Lemma 2: *Hermite-to-Hermite (H2H) translation operator*: Given the Hermite expansion centered at a centroid $x_{R'}$ in a reference node R': $G(x_q) = \sum_{\alpha \geq 0} A'_\alpha h_\alpha \left(\frac{x_q - x_{R'}}{\sqrt{2h^2}}\right)$, this same Hermite expansion shifted to a new location x_R of the parent node R is given by: $G(x_q) = \sum_{\gamma \geq 0} A_\gamma h_\gamma \left(\frac{x_q - x_R}{\sqrt{2h^2}}\right)$, where $A_\gamma = \sum_{0 \leq \alpha \leq \gamma} \frac{1}{(\gamma-\alpha)!} A'_\alpha \left(\frac{x_{R'} - x_R}{\sqrt{2h^2}}\right)^{\gamma-\alpha}$.



Lemma 3: *Local-to-local (L2L) translation operator*: Given a Taylor expansion centered at a centroid $x_{Q'}$ of a query node Q': $G(x_q) = \sum_{\beta \geq 0} B_\beta \left(\frac{x_q - x_{Q'}}{\sqrt{2h^2}}\right)^\beta$, the Taylor expansion obtained by shifting this expansion to the new centroid x_Q of the child node Q is: $G(x_q) = \sum_{\alpha \geq 0} \left[\sum_{\beta \geq \alpha} \frac{\beta!}{\alpha!(\beta-\alpha)!} B_\beta \left(\frac{x_Q - x_{Q'}}{\sqrt{2h^2}}\right)^{\beta-\alpha}\right] \left(\frac{x_q - x_Q}{\sqrt{2h^2}}\right)^\alpha$.



Since we cannot store an infinite number of coefficients, we incur an error in approximation summarized below:

Lemma 4: *Error Bound for Truncating an Hermite Expansion* (as presented in [8, 3]) Given an Hermite expansion of a reference node R about its centroid x_R: $G(x_q) = \sum_{\alpha \geq 0} A_\alpha h_\alpha \left(\frac{x_q - x_R}{\sqrt{2h^2}}\right)$ where $A_\alpha = \sum_{r=1}^{N_R} \frac{1}{\alpha!}\left(\frac{x_r - x_R}{\sqrt{2h^2}}\right)^\alpha$, the maximum error due to truncating the series after the first p^D term for a fixed query point x_q is $|\epsilon_{DH}(p)| = \frac{N_R}{(1-r)^D} \sum_{k=0}^{D-1} \binom{D}{k} (1 - r^p)^k \left(\frac{r^p}{\sqrt{p!}}\right)^{D-k}$ for which $\forall x_r \in R$, $||x_r - x_R||_\infty \leq rh$ where $r < 1$.

Lemma 5: *Error Bound for Truncating a Local Expansion*: Given the following Taylor expansion about the centroid x_Q of a query node Q: $G(x_q) = \sum_{\beta \geq 0} B_\beta \left(\frac{x_q - x_Q}{\sqrt{2h^2}}\right)^\beta$ where $B_\beta = \frac{(-1)^\beta}{\beta!} \sum_{\alpha \geq 0} A_\alpha h_{\alpha+\beta}\left(\frac{x_Q - x_R}{\sqrt{2h^2}}\right)$ and A_α's are the Hermite coefficients of the expansion centered at the reference centroid x_R, truncating the series after p^D terms satisfies the error bound $|\epsilon_{DL}(p)| = \frac{N_R}{(1-r)^D} \sum_{k=0}^{D-1} \binom{D}{k} (1 - r^p)^k \left(\frac{r^p}{\sqrt{p!}}\right)^{D-k}$ where $\forall x_q \in Q$, $||x_q - x_Q||_\infty \leq rh$ for $r < 1$.

Lemma 6: *Error Bound for $H2L$ Operator*: A truncated Hermite expansion centered about the centroid x_R of a reference node R: $G(x_q) = \sum_{\alpha < p} A_\alpha h_\alpha \left(\frac{x_q - x_R}{\sqrt{2h^2}}\right)$ has the following Taylor expansion about the centroid x_Q of a query node Q: $G(x_q) = \sum_{\beta \geq 0} C_\beta \left(\frac{x_q - x_Q}{\sqrt{2h^2}}\right)^\beta$ where $C_\beta = \frac{(-1)^{|\beta|}}{\beta!} \sum_{\alpha < p} A_\alpha h_{\alpha+\beta}\left(\frac{x_Q - x_R}{\sqrt{2h^2}}\right)$. Truncating the series after p^D terms satisfies the error bound $|\epsilon_{H2L}(p)| = \frac{N_R}{(1-2r)^{2D}} \sum_{k=0}^{D-1} \binom{D}{k} ((1 - (2r)^p)^2)^k \left(\frac{((2r)^p)(2-(2r)^p)}{\sqrt{p!}}\right)^{D-k}$ for which $\forall x_q \in Q$, $||x_q - x_Q||_\infty \leq rh$, and $\forall x_r \in R$, $||x_r - x_R||_\infty \leq rh$ where $r < \frac{1}{2}$.

Algorithm Description

buildInternal(R)
n = empty node
$\{R_l, R_r\}$ = Partition R into two
$n.left$ = **buildReferenceTree**(R_l)
$n.right$ = **buildReferenceTree**(R_r)
$n.mcoeffs$ = **H2H** ($n.left.mcoeffs$, $n.left.x_R$, $n.x_R$)
$n.mcoeffs$ = $n.mcoeffs$ + **H2H** ($n.right.mcoeffs$, $n.right.x_R$, $n.x_R$)
return n

buildLeaf(R)
n = empty node
$n.mcoeffs$ = Compute the truncated Hermite series of order $PLIMIT$ using each $x_r \in R$, centered about x_R
return n

buildReferenceTree(R)
if $size(R)$ is below leaf threshold
return **buildLeaf**(R)
else
return **buildInternal**(R)

bestMethod($Q, R, maxerr$)
if $R.maxside \geq 2h$, $p_{DH} = \infty$
else
p_{DH} = the smallest $1 \leq p \leq PLIMIT$ such that $\epsilon_{DH}(p) \leq maxerr$, otherwise $p_{DH} = \infty$
if $Q.maxside \geq 2h$, $p_{DL} = \infty$
else
p_{DL} = the smallest $1 \leq p \leq PLIMIT$ such that $\epsilon_{DL}(p) \leq maxerr$, otherwise $p_{DL} = \infty$
if $max(Q.maxside, R.maxside) \geq h$, $p_{H2L} = \infty$
else
p_{H2L} = the smallest $1 \leq p \leq PLIMIT$ such that $\epsilon_{H2L}(p) \leq maxerr$, otherwise $p_{H2L} = \infty$
$c_{DH} = N_Q p_{DH}^D$, $c_{DL} = N_R p_{DL}^D$, $c_{H2L} = p_{H2L}^{2D}$, $c_{Direct} = DN_Q N_R$
if $c = c_{DH}$, return $\{DH, p_{DH}, c_{DH}(p_{DH})\}$
if $c = c_{DL}$, return $\{DL, p_{DL}, c_{DL}(p_{DL})\}$
if $c = c_{H2L}$, return $\{H2L, p_{H2L}, c_{H2L}(p_{H2L})\}$
return $\{DIRECT, \infty, \infty\}$

DFGTHBase(Q, R)
for each $x_q \in Q$
for each $x_r \in R$
$c = K_h(||x_q - x_r||)$, $G_q^{min} += c$,
$G_q^{max} += c$, $G_q^{est} += c$
$G_q^{max} -= N_R$
$G_Q^{min} = \min_{q \in Q} G_q^{min}$, $G_Q^{max} = \max_{q \in Q} G_q^{max}$

DFGTH(Q, R)
$dl = N_R K_h(\delta_{QR}^{max})$, $du = N_R K_h(\delta_{QR}^{min}) - N_R$
$m = K_h(\delta_{QR}^{min}) - K_h(\delta_{QR}^{max})$
$N_T = N_R \left(\frac{N(K_h(\delta_{QR}^{min}) - K_h(\delta_{QR}^{max}))}{2m\epsilon_{DFGTH}^{min}} - 1\right)$
// Try finite-difference pruning.
if $m \leq \frac{\epsilon_T}{2}(G_Q^{min} + dl)$
$G_Q^{min} += dl$, $G_Q^{max} += du$,
$G_Q^{est} += 0.5(dl + du + N_R)$, return
// Otherwise, try multipole-type pruning.
else
$\{m, p, \epsilon_{FGT}\}$ = **bestMethod**($Q, R, \frac{\epsilon_T}{N} G_Q^{min}$)
if $m = DH$
for each $x_q \in Q$
$G_q^{est} +=$ **EVALM**($R.mcoeffs, p, x_R, x_q$)
else if $m = DL$
$Q.loceffs +=$ **DIRECTL**($\forall x_r \in R, p, x_Q$)
else if $m = H2L$
$Q.loceffs +=$ **H2L**($R.mcoeffs, p, x_R, x_Q$)
if $m \neq DIRECT$
$G_Q^{min} += dl, G_Q^{max} += du$, return
if leaf(Q) and leaf(R), **DFGTHBase**(Q, R)
else
DFGTH($Q.left, R.left$), **DFGTH**($Q.left, R.right$)
DFGTH($Q.right, R.left$), **DFGTH**($Q.right, R.right$)

Using both finite-difference and multipole prunings requires an extra field in each query node G_Q^{est} storing contributions from reference nodes obtained by finite-difference pruning and direct Hermite evaluations. The contributions from Taylor coefficients obtained via direct local accumulation and **H2L** translation opeator will be accounted for during the post-processing step.
The algorithm consists of three parts: constructing the query and the reference tree (preprocessing), invoking the recursive function **DFGTH**, then combining all contributions in a single breadth-first traversal sweep (post-processing).
In preprocessing, we construct trees for the query and the reference dataset. For the reference tree, the Hermite moments of order $PLIMIT$ is pre-computed. We used $PLIMIT = 8$ for $D = 2$, $PLIMIT = 6$ for $D = 3$, $PLIMIT = 4$ for $D = 5$.

Experimental Results and Conclusion

We studied the runtime performance of five algorithms on five real-world datasets (scaled to fit in $[0, 1]^D$ hypercube) for kernel density estimation at every query point with a range of bandwidths, from 3 orders of magnitude smaller than to three orders larger than optimal, according to the least-squares cross-validation score [10]. Our new method has the minimum time requirement for cross-validation over all bandwidth scales, but is suited only for low-dimensional problems (1 to 3). [a] [b]

Algorithm \ scale	0.001	0.01	0.1	1	10	100	1000
sj2-50000-2 (astronomy: positions), $D = 2$, $N = 50000$, $h^* = 0.00139506$							
Naive	301.696	301.696	301.696	301.696	301.696	301.696	301.696
FGT	out of RAM	out of RAM	out of RAM	3.892312	2.01846	0.319538	0.183616
IFGT	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	7.576783
DFD	0.837724	1.087066	1.658592	6.018158	62.077669	151.590062	1.551019
DFGT	0.849935	1.11567	4.599235	72.435177	18.450387	2.777454	2.532401
DFGTH	**0.846294**	**1.10654**	**1.683913**	**6.265131**	**5.063365**	**1.036626**	**0.68471**
colors50k (astronomy: colors), $D = 2$, $N = 50000$, $h^* = 0.00016911$							
Naive	301.696	301.696	301.696	301.696	301.696	301.696	301.696
FGT	out of RAM	out of RAM	out of RAM	> 2×Naive	> 2×Naive	0.475281	0.114430
IFGT	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	7.55986
DFD	1.095838	1.469454	2.802112	30.294007	280.633106	81.373053	3.604753
DFGT	1.099828	1.983888	29.231309	285.719266	12.886239	5.336602	3.5638
DFGTH	**1.081216**	**1.47692**	**2.855083**	**24.598749**	**7.142465**	**1.79648**	**0.627554**
edsgc-radec-rnd (astronomy: angles), $D = 2$, $N = 50000$, $h^* = 0.00466204$							
Naive	301.696	301.696	301.696	301.696	301.696	301.696	301.696
FGT	out of RAM	out of RAM	out of RAM	2.859245	1.768738	0.210799	0.059664
IFGT	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	7.585585
DFD	0.812462	1.083528	1.682261	5.860172	63.849361	357.099354	0.743045
DFGT	0.84023	1.120015	4.346061	73.036687	21.652047	3.424304	1.977302
DFGTH	**0.821672**	**1.104545**	**1.737799**	**6.037217**	**5.7398**	**1.883216**	**0.436596**
mockgalaxy-D-1M-rnd (cosmology: positions), $D = 3$, $N = 50000$, $h^* = 0.000768201$							
Naive	354.868751	354.868751	354.868751	354.868751	354.868751	354.868751	354.868751
FGT	out of RAM	out of RAM	out of RAM	out of RAM	> 2×Naive	> 2×Naive	> 2×Naive
IFGT	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive
DFD	0.70054	0.701547	0.761524	0.843451	1.086608	42.022605	383.12048
DFGT	0.73007	0.733638	0.799711	0.999316	50.619588	125.059911	109.353701
DFGTH	**0.724004**	**0.719951**	**0.789002**	**0.877564**	**1.265064**	**22.6106**	**87.488392**
bio5-rnd (biology: drug activity), $D = 5$, $N = 50000$, $h^* = 0.000567161$							
Naive	364.439228	364.439228	364.439228	364.439228	364.439228	364.439228	364.439228
FGT	out of RAM	out of RAM	out of RAM	out of RAM	out of RAM	out of RAM	out of RAM
IFGT	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive
DFD	2.249868	2.4958865	4.70948	12.065697	94.345003	412.39142	107.675935
DFGT	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive
DFGTH	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive	> 2×Naive

We have already investigated a more efficient $O(D^p)$ expansion scheme (advocated in [11]) in [9]. Beyond statistical application, the dual-tree multipole methods appear to be the state-of-the-art for the corresponding kernel summation problem in computational physics.

[a] Times which include preprocessing costs are measured in CPU seconds on a dual-processor AMD Opteron 242 machine with 8 Gb of main memory and 1 Mb of CPU cache. All the codes that we have written and obtained are written in C and C++, and was compiled under `-O6 -funroll-loops` flags on Linux kernel 2.4.26. We have limited all datasets to 50K points, and set $\epsilon = 0.01$. When any method fails to achieve the error tolerance in less time than twice that of the naive method, we give up.

[b] FGT [8] and IFGT [11] codes were obtained from the authors' websites. For the FGT, the absolute deviation τ is used as an estimate for ϵ, and we keep halving τ until the relative error bound ϵ is met for all query points. For the IFGT, which has multiple parameters which must be tweaked simultaneously, an automatic scheme was created, based on the recommendations given in the paper and software documentation: For $D = 2$, use $p = 8$; for $D = 3$, use $p = 6$; set $\rho_x = 2.5$; start with $K = \sqrt{N}$ and double K until the error tolerance is met. When this failed to meet the tolerance, we resorted to additional trial and error by hand, again using knowledge which is unavailable in practice. The costs of parameter selection for these methods in both computer and human time is not included in the table. DFD refers to the depth-first dual-tree finite-difference method [6]. DFGT replaces finite-difference pruning in place of multipole pruning utilizing three translation operators, while DFGTH uses both finite-difference and multipole prunings.