# Fast weighted summation of erfc functions

VIKAS CHANDRAKANT RAYKAR♠, RAMANI DURAISWAMI♠,
and BALAJI KRISHNAPURAM♣

♠ Perceptual Interfaces and Reality Laboratory
Department of Computer Science and Institute for Advanced Computer Studies
University of Maryland, CollegePark, MD, USA
{vikas,ramani}@umiacs.umd.edu

♣ Computer Aided Diagnosis and Therapy Group
Siemens Medical Solutions, Malvern, PA, USA
balaji.krishnapuram@siemens.com

---

Direct computation of the weighted sum of $N$ complementary error functions at $M$ points scales as $\mathcal{O}(MN)$. We present a $\mathcal{O}(M+N)$ $\epsilon$-exact approximation algorithm to compute the same.

Categories and Subject Descriptors: Technical report [**CS-TR-4848/UMIACS-TR-2007-03**]: February 8, 2007

---

## 1. INTRODUCTION

The complementary error function is defined as follows (see Figure 1) [Abramowitz and Stegun 1972]

$$\operatorname{erfc}(z) = \frac{2}{\sqrt{\pi}} \int_z^\infty e^{-t^2} dt. \tag{1}$$

Consider the following weighted summation of $N$ erfc functions each centered at $\{x_i\}_{i=1}^N$.

$$E(y) = \sum_{i=1}^{N} q_i \operatorname{erfc}(y - x_i). \tag{2}$$

The scalars $q_i$ will be referred to as the *weights*. Direct computation of (2) at $M$ points $\{y_j\}_{j=1}^M$ is $\mathcal{O}(MN)$. In this report we will derive an $\epsilon$-exact approximation algorithm to compute the same in $\mathcal{O}(M+N)$ time.

For any given $\epsilon > 0$, $\widehat{E}$ is an $\epsilon - exact$ approximation to $E$ if the maximum absolute error relative to the total weight $Q_{abs} = \sum_{i=1}^N |q_i|$ is upper bounded by $\epsilon$, *i.e.*,

$$\max_{y_j} \left[ \frac{|\hat{E}(y_j) - E(y_j)|}{Q_{abs}} \right] \leq \epsilon. \tag{3}$$

The constant in $\mathcal{O}(M+N)$, depends on the desired *accuracy* $\epsilon$, which however can be *arbitrary*. In fact for machine precision accuracy there is no difference between the direct and the fast methods. The algorithm is inspired by the *fast multipole*

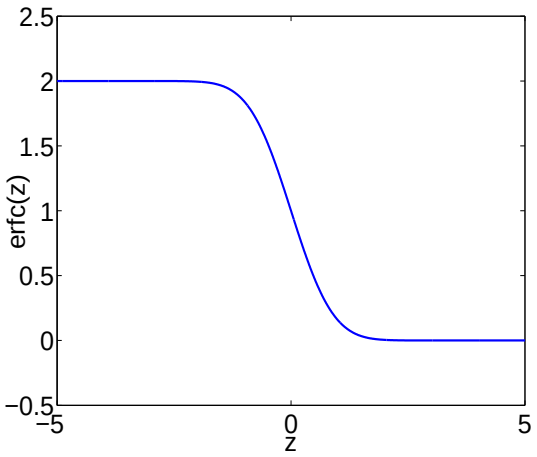



Fig. 1. The erfc function.

*methods* proposed in computational physics [Greengard 1994]. The fast algorithm is based on using a infinite series expansion for the erfc function and retaining only the first few terms contributing to the desired accuracy $\epsilon$.

## 2. SERIES EXPANSION

Several series exist for the erfc function (See for e.g. Chapter 7 in [Abramowitz and Stegun 1972]). Some are applicable only to a restricted interval, while other need a large number of terms to converge. We use the following series derived by Beauliu [Beauliu 1989; Tellambura and Annamalai 2000].

$$\operatorname{erfc}(x) = 1 - \frac{4}{\pi} \sum_{\substack{n=1 \\ n \text{ odd}}}^{2p-1} \frac{e^{-n^2h^2}}{n} \sin(2nhx) + \operatorname{error}(x), \tag{4}$$

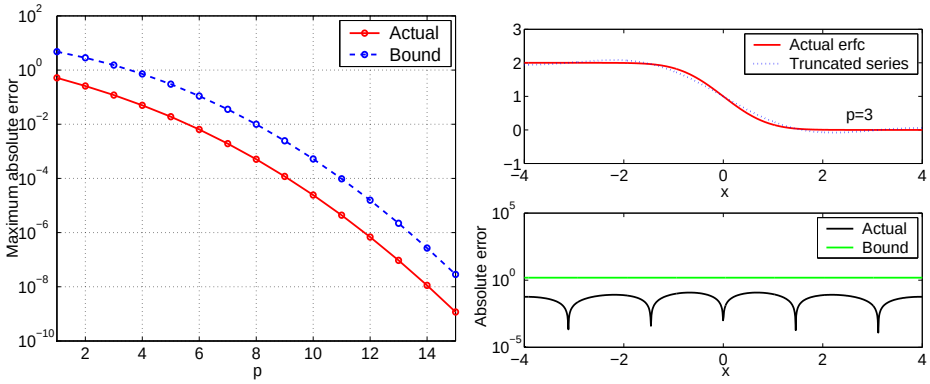
where

$$|\operatorname{error}(x)| < \left| \frac{4}{\pi} \sum_{\substack{n=2p+1 \\ n \text{ odd}}}^{\infty} \frac{e^{-n^2h^2}}{n} \sin(2nhx) \right| + \operatorname{erfc}\left(\frac{\pi}{2h} - |x|\right). \tag{5}$$

Here, $p$ is the *truncation number* and $h$ is a real number related to the sampling interval. These kind of series are of interest in the field digital communications wherein the noise is modeled as a Gaussian random variable. The series is derived by applying a Chernoff bound approach to an approximate Fourier series expansion of a periodic square waveform [Beauliu 1989].

This series converges rapidly especially as $x \to 0$. Figure 2(a) shows the maximum absolute error between the actual value of erfc [1] and the truncated series

[1]There is no closed form expression to compute erfc directly. The implementation in MATLAB uses a rational Chebyshev approximation and the accuracy is not adequate enough. In order to compare the error between the actual and the series approximation we use the Maple implementation(`feval(maple('erfc'),x))`) which provides very high precision using symbolic integration.




(a) (b)

Fig. 2. (a) The maximum absolute error between the actual value of erfc and the truncated series representation (Eq. 4) as a function of the truncation number $p$ for any $x \in [-4, 4]$. The error bound (Eq. 6) is also shown. (b) A sample plot of the actual erfc function and the $p = 3$ truncated series representation. The error as a function of $x$ is also shown in the lower panel.

representation as a function of $p$. For example for any $x \in [-4, 4]$ with $p = 12$ the error is less than $10^{-6}$. We have to choose $p$ and $h$ such that the error has to be less than $\epsilon$. We further bound the first term in (5) as follows (See Appendix 1 for a derivation).

$$|\text{error}(x)| < \frac{2}{\sqrt{\pi}h}\text{erfc}\left((2p+1)h\right) + \text{erfc}\left(\frac{\pi}{2h} - |x|\right). \tag{6}$$

For a fixed $p$ and $h$ as $|x|$ increases the error increases. Therefore as $|x|$ increases, $h$ should decrease and consequently the series converges slower leading to a large truncation number $p$.

## 3. FAST SUMMATION ALGORITHM

We will now derive a fast algorithm to compute $E(y)$ based on the series (4).

$$\begin{aligned} E(y) &= \sum_{i=1}^{N} q_i \text{erfc}(y - x_i) \\ &= \sum_{i=1}^{N} q_i \left[ 1 - \frac{4}{\pi} \sum_{\substack{n=1 \\ n \text{ odd}}}^{2p-1} \frac{e^{-n^2h^2}}{n} \sin\left\{2nh(y - x_i)\right\} + \text{error} \right]. \end{aligned} \tag{7}$$

Ignoring the error term the sum $E(y)$ can be approximated as

$$\widehat{E}(y) = Q - \frac{4}{\pi} \sum_{i=1}^{N} q_i \sum_{\substack{n=1 \\ n \text{ odd}}}^{2p-1} \frac{e^{-n^2h^2}}{n} \sin\left\{2nh(y - x_i)\right\}, \tag{8}$$

where $Q = \sum_{i=1}^{N} q_i$. The terms $y$ and $x_i$ appear together in the argument of the sin function. We separate them using the trignometric identity $\sin(\alpha - \beta) = \sin(\alpha)\cos(\beta) - \cos(\alpha)\sin(\beta)$.

$$\begin{aligned}\sin\{2nh(y-x_i)\} &= \sin\{2nh(y-x_*) - 2nh(x_i-x_*)\} \\ &= \sin\{2nh(y-x_*)\}\cos\{2nh(x_i-x_*)\} \\ &\quad - \cos\{2nh(y-x_*)\}\sin\{2nh(x_i-x_*)\}. \end{aligned} \tag{9}$$

Note that we have shifted all the points by $x_*$. Substituting the separated representation (9) in Eq. 8 we have

$$\begin{aligned}\widehat{E}(y) &= Q - \frac{4}{\pi}\sum_{i=1}^{N} q_i \sum_{\substack{n=1 \\ n \text{ odd}}}^{2p-1} \frac{e^{-n^2h^2}}{n} \sin\{2nh(y-x_*)\}\cos\{2nh(x_i-x_*)\} \\ &\quad + \frac{4}{\pi}\sum_{i=1}^{N} q_i \sum_{\substack{n=1 \\ n \text{ odd}}}^{2p-1} \frac{e^{-n^2h^2}}{n} \cos\{2nh(y-x_*)\}\sin\{2nh(x_i-x_*)\}. \end{aligned} \tag{10}$$

Exchanging the order of summation and regrouping the terms we have the following expression.

$$\widehat{E}(y) = Q - \frac{4}{\pi}\sum_{\substack{n=1 \\ n \text{ odd}}}^{2p-1} \left[A_n \sin\{2nh(y-x_*)\} - B_n \cos\{2nh(y-x_*)\}\right]. \tag{11}$$

where

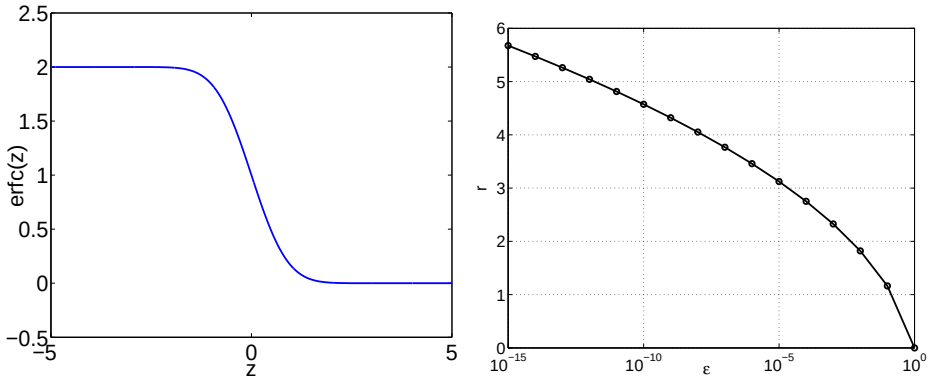
$$A_n = \frac{e^{-n^2h^2}}{n}\sum_{i=1}^{N} q_i \cos\{2nh(x_i-x_*)\} \text{ and } B_n = \frac{e^{-n^2h^2}}{n}\sum_{i=1}^{N} q_i \sin\{2nh(x_i-x_*)\} \tag{12}$$

## 4. RUNTIME AND STORAGE ANALYSIS

Note that the coefficients $\{A_n, B_n\}_{n=1(n \text{ odd})}^{2p-1}$ do not depend on $y$. Hence each of $A_n$ and $B_n$ can be evaluated separately is $\mathcal{O}(N)$ time. Since there are $p$ such coefficients the total complexity to compute $A$ and $B$ is $\mathcal{O}(2pN)$. The term $Q = \sum_{i=1}^{N} q_i$ can also be precomputed in $\mathcal{O}(N)$ time. Once $A$, $B$, and $Q$ have been precomputed, evaluation of $\widehat{E}(y)$ requires $\mathcal{O}(2p)$ operations. Evaluating at $M$ points is $\mathcal{O}(2pM)$. Hence the computational complexity has reduced from the quadratic $\mathcal{O}(NM)$ to the linear $\mathcal{O}((2p+1)N + 2pM)$. We need space to store the points and the coefficients $A$ and $B$. Hence the storage complexity is $\mathcal{O}(N + M + 2p)$.

## 5. DIRECT INCLUSION AND EXCLUSION OF FARAWAY POINTS

Note that $z = (y - x_i) \in [-\infty, \infty]$. The truncation number $p$ required to approximate $\text{erfc}(z)$ can be quite large for large $|z|$. Luckily $\text{erfc}(z) \to 2$ as $z \to -\infty$ and $\text{erfc}(z) \to 0$ as $z \to \infty$ very quickly [See Figure 3(a)]. Since we are interested in the




(a) (b)

Fig. 3. (a) The erfc function (b) The value of $r$ for which $\text{erfc}(z) < \epsilon,\ \forall z > r$

result only to a certain precision $\epsilon$ we can use the following approximation.

$$\text{erfc}(z) \approx \begin{cases} 2 & \text{if } z < -r \\ p\text{-truncated series} & \text{if } -r \leq z \leq r \\ 0 & \text{if } z > r \end{cases} \tag{13}$$

The bound $r$ and the truncation number $p$ have to be chosen such that for any $z$ the error is always less than $\epsilon$. From Figure 3(b) we can see that for error of the order $10^{-15}$ we need to use the series expansion for $-6 \leq z \leq 6$.

However we cannot check the value of $(y - x_i)$ for all pairs of $x_i$ and $y$. This would lead us back to the quadratic complexity. To avoid this, we subdivide the points into clusters.

## 6. SPACE SUB-DIVISION

We uniformly sub-divide the space into $K$ intervals of length $2r_x$. The $N$ source points are assigned into $K$ clusters, $S_k$ for $k = 1, \ldots, K$ with $c_k$ being the center of each cluster. The aggregated coefficients are now computed for each cluster and the total contribution from all the influential clusters is summed up. For each cluster if $|y - c_k| \leq r_y$ then we will incorporate the series coefficients. If $(y - c_k) < -r_y$ then we will include a contribution of $2Q_k$. If $(y - c_k) > r_y$ then we will ignore that cluster. Hence

$$\begin{aligned} \widehat{E}(y) \;=\; & \sum_{|y-c_k| \leq r_y} \left( Q_k - \frac{4}{\pi} \sum_{\substack{n=1 \\ n \text{ odd}}}^{2p-1} \left[ A_n^k \sin\{2nh(y-c_k)\} - B_n^k \cos\{2nh(y-c_k)\} \right] \right) \\ & + \sum_{(y-c_k) < -r_y} 2Q_k. \end{aligned} \tag{14}$$

where

$$
\begin{aligned}
A_n^k &= \frac{e^{-n^2h^2}}{n} \sum_{i=1}^{N} q_i \cos\{2nh(x_i - c_k)\}, \\
B_n^k &= \frac{e^{-n^2h^2}}{n} \sum_{i=1}^{N} q_i \sin\{2nh(x_i - c_k)\}, \text{ and} \\
Q_k &= \sum_{\forall x_i \in S_k} q_i.
\end{aligned} \tag{15}
$$

The computational complexity to compute $A$,$B$, and $Q$ is still $\mathcal{O}((2p+1)N)$ since each $x_i$ belongs to only one cluster. Let $l$ be the number of influential clusters, *i.e.*, the clusters for which $|y - c_k| \leq r_y$. Evaluating $\widehat{E}(y)$ at $M$ points due to these $l$ clusters is $\mathcal{O}(2plM)$. Let $m$ be the number of clusters for which $(y - c_k) < -r_y$. Evaluating $\widehat{E}(y)$ at $M$ points due to these $m$ clusters is $\mathcal{O}(mM)$. Hence the total computational complexity is $\mathcal{O}((2p+1)N + (2pl+m)M)$. The storage complexity is $\mathcal{O}(N + M + (2p+1)K)$.

## 7. CHOOSING THE PARAMETERS

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

—$r_x$ (the interval length),
—$r$ (the cut off radius ),
—$p$ (the truncation number), and
—$h$ such that

for *any* target point $y$

$$
\left| \frac{\hat{E}(y) - E(y)}{Q_{abs}} \right| \leq \epsilon, \tag{16}
$$

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

Let us define $\Delta_i$ to be the point wise error in $\widehat{E}(y)$ contributed by the $i^{th}$ source $x_i$. We now require that

$$
|\hat{E}(y) - E(y)| = \left| \sum_{i=1}^{N} \Delta_i \right| \leq \sum_{i=1}^{N} |\Delta_i| \leq \sum_{i=1}^{N} |q_i| \epsilon. \tag{17}
$$

One way to achieve this is to let $|\Delta_i| \leq |q_i|\epsilon \ \forall i = 1, \ldots, N$.

For all $x_i$ such that $|y - x_i| \leq r$ we have

$$
|\Delta_i| < \underbrace{|q_i| \frac{2}{\sqrt{\pi}h} \operatorname{erfc}\left((2p+1)h\right)}_{T_e} + \underbrace{|q_i| \operatorname{erfc}\left(\frac{\pi}{2h} - r\right)}_{S_e}. \tag{18}
$$

We have to choose the parameters such that $|\Delta_i| < |q_i|\epsilon$. We will let $S_e < |q_i|\epsilon/2$. This implies that

$$
\frac{\pi}{2h} - r > \operatorname{erfc}^{-1}(\epsilon/2). \tag{19}
$$

Hence we have to choose

$$h < \frac{\pi}{2\left(r + \text{erfc}^{-1}\left(\epsilon/2\right)\right)}. \tag{20}$$

We will choose

$$h = \frac{\pi}{3\left(r + \text{erfc}^{-1}\left(\epsilon/2\right)\right)}. \tag{21}$$

We will choose $p$ such that $T_e < |q_i|\epsilon/2$. This implies that

$$2p + 1 > \frac{1}{h}\text{erfc}^{-1}\left(\frac{\sqrt{\pi}h\epsilon}{4}\right). \tag{22}$$

We choose

$$p = \left\lceil \frac{1}{2h}\text{erfc}^{-1}\left(\frac{\sqrt{\pi}h\epsilon}{4}\right)\right\rceil. \tag{23}$$

Note that as $r$ increases $h$ decreases and consequently $p$ increases. If $x \in (r, \infty]$ we approximate $\text{erfc}(x)$ by 0 and if $x \in [-\infty, -r)$ then approximate $\text{erfc}(x)$ by 2. If we choose

$$r > \text{erfc}^{-1}(\epsilon), \tag{24}$$

then the approximation will result in a error $< \epsilon$. In practice we choose

$$r = \text{erfc}^{-1}(\epsilon) + 2r_x, \tag{25}$$

where $r_x$ is the cluster radius. For a target point $y$ the number of influential clusters

$$(2l + 1) = \left\lceil \frac{2r}{2r_x}\right\rceil. \tag{26}$$

Let us choose $r_x = 0.1\text{erfc}^{-1}(\epsilon)$. This implies $2l + 1 = 12$. So we have to consider $n = 6$ clusters on either side of the target point. Summarizing the parameters are given by

(1) $r_x = 0.1\text{erfc}^{-1}(\epsilon)$.
(2) $r = \text{erfc}^{-1}(\epsilon) + 2r_x$.
(3) $h = \pi/3\left(r + \text{erfc}^{-1}\left(\epsilon/2\right)\right)$.
(4) $p = \left\lceil \frac{1}{2h}\text{erfc}^{-1}\left(\frac{\sqrt{\pi}h\epsilon}{4}\right)\right\rceil$.
(5) $(2l + 1) = \lceil r/r_x \rceil$.

## 8. NUMERICAL EXPERIMENTS

In this section we present some numerical studies of the speedup and error as a function of the number of data points and the desired error $\epsilon$. The algorithm was programmed in C++ with MATLAB bindings and was run on a 1.6 GHz Pentium M processor with 512MB of RAM.

Figure 4 shows the running time and the maximum absolute error relative to $Q_{abs}$ for both the direct and the fast methods as a function of $N = M$. The points were normally distributed with zero mean and unit variance. The weights $q_i$ were set to 1. We see that the running time of the fast method grows linearly, while that of the

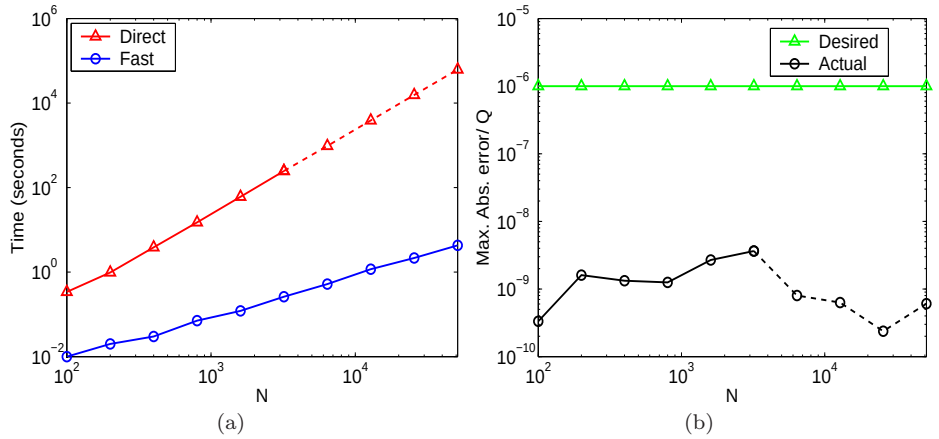



Fig. 4. (a) The running time in seconds and (b) maximum absolute error relative to $Q_{abs}$ for the direct and the fast methods as a function of $N = M$. For $N > 3,200$ the timing results for the direct evaluation were obtained by evaluating the sum at $M = 100$ points and then extrapolating (shown as dotted line).

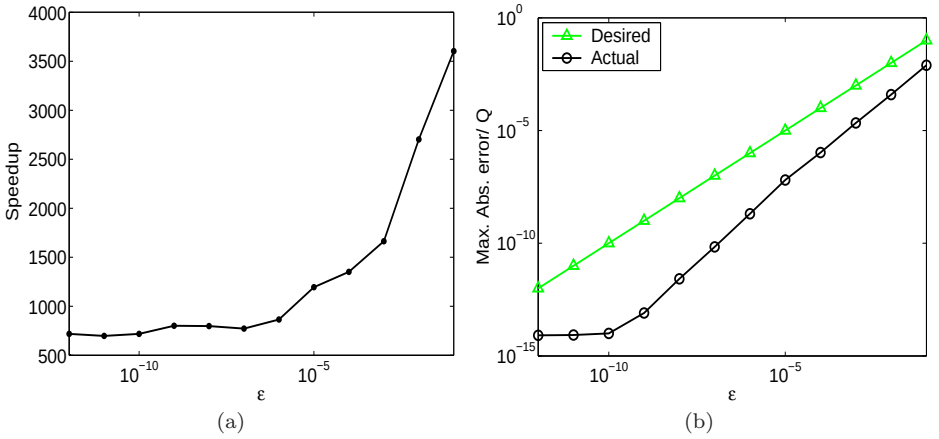



Fig. 5. (a) The speedup achieved and (b) maximum absolute error relative to $Q$ for the direct and the fast methods as a function of $\epsilon$ for $N = M = 3,000$.

direct evaluation grows quadratically. We also observe that the error is way below the desired error thus validating our bound. For example for $N = M = 51,200$ points while the direct evaluation takes around 17.26 hours the fast evaluation requires only 4.29 seconds with an error of around $10^{-10}$. Figure 5 shows the tradeoff between precision and speedup. An increase in speedup is obtained at the cost of reduced accuracy.

## Appendix 1 : Error bound

We will bound the first term in (5) as follows.

$$\begin{aligned}
\left|\frac{4}{\pi}\sum_{\substack{n=2p+1\\ n\text{ odd}}}^{\infty}\frac{e^{-n^2h^2}}{n}\sin(2nhx)\right| &\leq \frac{4}{\pi}\sum_{\substack{n=2p+1\\ n\text{ odd}}}^{\infty}\frac{e^{-n^2h^2}}{n}|\sin(2nhx)| \\
&\leq \frac{4}{\pi}\sum_{\substack{n=2p+1\\ n\text{ odd}}}^{\infty}\frac{e^{-n^2h^2}}{n} \quad [\text{ Since } |\sin(2nhx)|\leq 1] \\
&< \frac{4}{\pi}\sum_{\substack{n=2p+1\\ n\text{ odd}}}^{\infty}e^{-n^2h^2} \\
&< \frac{4}{\pi}\int_{2p+1}^{\infty}e^{-x^2h^2}dx < \frac{2}{\sqrt{\pi}h}\left[\frac{2}{\sqrt{\pi}}\int_{(2p+1)h}^{\infty}e^{-t^2}dt\right] \\
&= \frac{2}{\sqrt{\pi}h}\text{erfc}((2p+1)h)
\end{aligned}$$

## Appendix 2 : Gradient computation in ranking problem

This is an appendix to the paper [Raykar et al. 2007]. We show how the gradient computation in a ranking problem boils down to summation of erfc functions. Refer to the paper for more details regarding the ranking problem.

We will isolate the key computational primitive contributing to the quadratic complexity in the gradient computation. The following summarizes the different variables in analyzing the computational complexity of evaluating the gradient.

—We have $S$ classes with $m_i$ training instances in the $i^{th}$ class.
—Hence we have a total of $m=\sum_{i=1}^{S}m_i$ training examples in $d$ dimensions.
—$|\mathcal{E}|$ is the number of edges in the preference graph, and
—$\mathcal{M}^2=\sum_{\mathcal{E}_{ij}}m_im_j$ is the total number of pairwise preference relations.

For any $x$ we will define $z=\sqrt{3}w^Tx/(\pi\sqrt{2})$. Note that $z$ is a scalar and for a given $w$ can be computed in $\mathcal{O}(dm)$ operations for the entire training set. We will now rewrite the gradient as

$$g(w)=-\lambda w-\Delta_1+\frac{1}{2}\Delta_2-\frac{1}{2}\Delta_3, \tag{27}$$

where the vectors $\Delta_1$, $\Delta_1$, and $\Delta_3$ are defined as follows–

$$\begin{aligned}
\Delta_1 &= \sum_{\mathcal{E}_{ij}}\sum_{k=1}^{m_i}\sum_{l=1}^{m_j}(x_k^i-x_l^j). \\
\Delta_2 &= \sum_{\mathcal{E}_{ij}}\sum_{k=1}^{m_i}\sum_{l=1}^{m_j}x_k^i\text{erfc}(z_k^i-z_l^j). \\
\Delta_3 &= \sum_{\mathcal{E}_{ij}}\sum_{k=1}^{m_i}\sum_{l=1}^{m_j}x_l^j\text{erfc}(z_k^i-z_l^j).
\end{aligned} \tag{28}$$

The vector $\Delta_1$ is independent of $w$ and can be written as follows–

$$\Delta_1 = \sum_{\mathcal{E}_{ij}} m_i m_j (x^i_{\text{mean}} - x^j_{\text{mean}}), \text{ where } x^i_{\text{mean}} = \frac{1}{m_i} \sum_{k=1}^{m_i} x^i_k$$

is the mean of all the training instances in the $i^{th}$ class. Hence $\Delta_1$ can be pre-computed in $\mathcal{O}(|\mathcal{E}|d + dm)$ operations.

The the other two terms $\Delta_2$ and $\Delta_3$ can be written as follows–

$$\Delta_2 = \sum_{\mathcal{E}_{ij}} \sum_{k=1}^{m_i} x^i_k E^j_-(z^i_k) \quad \Delta_3 = \sum_{\mathcal{E}_{ij}} \sum_{l=1}^{m_j} x^j_l E^i_+(-z^j_l) \tag{29}$$

where

$$\begin{aligned} E^j_-(y) &= \sum_{l=1}^{m_j} \text{erfc}(y - z^j_l). \\ E^i_+(y) &= \sum_{k=1}^{m_i} \text{erfc}(y + z^i_k). \end{aligned} \tag{30}$$

Note that $E^j_-(y)$ in the sum of $m_j$ erfc functions centered at $z^j_l$ and evaluated at $y$–which requires $\mathcal{O}(m_j)$ operations. In order to computed $\Delta_3$ we need to evaluate it at $m_i$ points, thus requiring $\mathcal{O}(m_i m_j)$ operations. Hence each of $\Delta_2$ and $\Delta_3$ can be computed in $\mathcal{O}(dSm + \mathcal{M}^2)$ operations.

Hence the core computational primitive contributing to the $\mathcal{O}(\mathcal{M}^2)$ cost is the summation of erfc functions. In the next section we will show how this sum can be computed in linear $\mathcal{O}(m_i + m_j)$ time, at the expense of reduced precision which however can be arbitrary. As a result of this $\Delta_2$ and $\Delta_3$ can be computed in linear $\mathcal{O}(dSm + (S-1)m)$ time. In terms of the optimization algorithm since the gradient is computed approximately the number of iterations required to converge will increase. However this is more than compensated by the cost per iteration which is drastically reduced.

## REFERENCES


Abramowitz, M. and Stegun, I. A. 1972. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* Dover.

Beauliu, N. C. 1989. A simple series for personal computer computation of the error function $Q(.)$. *IEEE Transactions on Communications 37,* 9, 989–991.

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

Raykar, V. C., Duraiswami, R., and Krishnapuram, B. 2007. A fast algorithm for learning large scale preference relations. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS 2007).*

Tellambura, C. and Annamalai, A. 2000. Efficient computation of erfc(x) for large arguments. *IEEE Transactions on Communications 48,* 4, 529–532.