Schedae Informaticae Vol. 23 (2014): 57–67
doi: 10.4467/20838476SI.14.005.3022

Fast Optimization of Multithreshold Entropy Linear Classifier

Rafał Józefowicz[1] and Wojciech Marian Czarnecki[2]
[1]Google Inc.
e-mail: *rafjoz@gmail.com*
[2]Faculty of Mathematics and Computer Science
Jagiellonian University
ul. Łojasiewicza 6, 30-389 Kraków
e-mail: *wojciech.czarnecki@uj.edu.pl*

Abstract. Multithreshold Entropy Linear Classifier (MELC) is a density based model which searches for a linear projection maximizing the Cauchy-Schwarz Divergence of dataset kernel density estimation. Despite its good empirical results, one of its drawbacks is the optimization speed. In this paper we analyze how one can speed it up through solving an approximate problem. We analyze two methods, both similar to the approximate solutions of the Kernel Density Estimation querying and provide adaptive schemes for selecting a crucial parameters based on user-specified acceptable error. Furthermore we show how one can exploit well known conjugate gradients and L-BFGS optimizers despite the fact that the original optimization problem should be solved on the sphere. All above methods and modifications are tested on 10 real life datasets from UCI repository to confirm their practical usability.

Keywords: multithreshold classifier, entropy, approximation, optimization.

1. Introduction

Many methods of speeding up the kernel density estimator's (KDE) querying process has been proposed in the literature [6, 12, 14]. As optimization problem introduced in Multithreshold Entropy Linear Classifier [5] is closely related to the equations of KDE it appears natural that similar techniques can be used to simplify its computations with a bounded error. Importance of such reductions comes from the high (quadratic) complexity of the evaluation of functions required during training of this model which makes it hard to use for any dataset with more than a thousand points. In this paper we investigate two such approaches, first – sorting and discarding, which ignores computations of similarities between points that are too far away to have big impact on the function's value, second – binning, which smooths the function construction in order to heavily reduce amount of unique points. Both these methods are introduced in an adaptive manner so the optimization process have fixed error bound despite many different linear projections being analyzed during the training phase. We also show a very simple method which enables to use a wide range of optimization algorithms even though proposed model requires optimization with a specific constraints (sphere bounded).

2. Multithreshold Entropy Linear Classifier

Multithreshold Entropy Linear Classifier (MELC [5]) has been recently proposed as an information theoretic approach for building model from the multithreshold linear family [1]. It's core idea is to find a linear operator v (with unit norm) such that kernel density estimations of projected classes' training samples maximize the Cauchy-Schwarz Divergence ($\mathrm{D_{CS}}$ [9]). Let us recall the equation of $\mathrm{D_{CS}}$ in order to find the core computational bottleneck which appears in MELC optimization

$$\mathrm{D_{CS}}(f_-, f_+) = 2\mathrm{H}_2^\times(f_-, f_+) - \mathrm{H}_2(f_-) - \mathrm{H}_2(f_+),$$

for $f_\pm = [\![v^T X_\pm]\!]$ being a kernel density estimator of $v^T X_\pm$ with Silverman's rule [11], thus from the definition of Renyi's quadratic entropy, Renyi's quadratic cross entropy and the fact that $\mathrm{ip}^\times(f, g) = \int fg$ we have

$$\mathrm{D_{CS}}(f_-, f_+) = -2\log \mathrm{ip}^\times(f_-, f_+) + \log \mathrm{ip}^\times(f_-, f_-) + \log \mathrm{ip}^\times(f_+, f_+).$$

As whole $\mathrm{D_{CS}}$ function is composed of $\mathrm{ip}^\times$ evaluations, in the rest of our paper we focus purely on the $\mathrm{ip}^\times$, which we expand using Gaussian kernel density estimation [5] and denote $\mathrm{ip}^\times(v) = \mathrm{ip}^\times([\![v^T X_-]\!], [\![v^T X_+]\!])$.

$$\mathrm{ip}^\times(v) = \frac{1}{\sqrt{2\pi V(v)|X||Y|}} \sum_{x,y} \exp\left(-\frac{\langle v, x-y\rangle^2}{2V(v)}\right),$$

where $V(v)$ is a sum of each classes estimated variances using Silverman's rule [11].

In an obvious way, naive computation of $\mathrm{ip}^\times$ is $\mathcal{O}(N^2)$, where $N = \max\{|X_-|, |X_+|\}$ due to the summation over all possible pairs (x, y). In the following sections we focus on methods which reduce this computational bottleneck while still preserving given approximation of $\mathrm{ip}^\times$ value.

3. Reduction of ip$^\times$ computational complexity

Sorting and discarding

Let us begin with the very simple conception of computing values of only those (x, y) pairs which are close enough to have an impact on the value of ip$^\times$. If we assume that points projections are sorted (which can be done in general in $\mathcal{O}(N \log N)$[1]) we can search the dataset in linear time and identify for each point x indices of first and last point which are at most at distance T from x. Following theorem shows what T to choose in order to obtain at most ϵ error.

Theorem 1 *Using adaptive sorting and discarding with distance threshold in each iteration of at least*

$$\sqrt{\max\left\{0, -V(v)\ln\left(2(\tfrac{\epsilon}{p})^2 \pi V(v)\right)\right\}},$$

where $V(v)$ is a sum of each classes estimated variances, leads to the computation of the ip$^\times$ function with at most ϵ error, assuming that at most fraction of p points is located closer than T.

Proof. We assume that $|\langle v, x-y\rangle| \geq T$ for N_T pairs of points which are being ignored during computation of ip$^\times$ so $-\langle v, x-y\rangle^2 \leq -T^2$, thus

$$\frac{1}{\sqrt{2\pi V(v)}|X||Y|}\sum_{x,y}\exp\left(-\frac{\langle v, x-y\rangle^2}{2V(v)}\right) \leq$$

$$\frac{1}{\sqrt{2\pi V(v)}|X||Y|}\sum_{x,y}\exp\left(-\frac{T^2}{2V(v)}\right) = \frac{N_T}{\sqrt{2\pi V(v)}|X||Y|}\exp\left(-\frac{T^2}{2V(v)}\right).$$

If we look for an ϵ approximation of non-regularized MELC objective we put $0 \leq p = N_T/(|X||Y|) \leq 1$ and consequently

$$p\frac{1}{\sqrt{2\pi V(v)}}\exp\left(-\frac{T^2}{2V(v)}\right) \leq \epsilon,$$

thus

$$\exp\left(-\frac{T^2}{2V(v)}\right) \leq \tfrac{\epsilon}{p}\sqrt{2\pi V(v)}$$

$$T^2 \geq -2V(v)\ln\left(\tfrac{\epsilon}{p}\sqrt{2\pi V(v)}\right),$$

obviously if $\ln\left(\tfrac{\epsilon}{p}\sqrt{2\pi V(v)}\right) > 0$ then any T satisfies this inequality (as it can only happen if we choose very big acceptable error ϵ), so for simplicity we add the

[1] In fact for iterative optimization techniques points ordering does not change much between subsequent calls so after initial sorting it can be done in linear time using insertion sort.

maximum of this value with 0.

$$T \geq \sqrt{\max\left\{0, -2V(v)\ln\left(\tfrac{\epsilon}{p}\sqrt{2\pi V(v)}\right)\right\}} = \sqrt{\max\left\{0, -V(v)\ln\left(2(\tfrac{\epsilon}{p})^2\pi V(v)\right)\right\}}.$$

□

Binning

While sorting and discarding technique is quite easy to implement and analyze its practical speedup might be limited for densely packed datasets. In such cases it might be more valuable to perform a binning of our projected points, so those located near each other are approximated by their empirical mean. Such an approach works well for densely packed datasets which makes it a complementary approach to the previous one.

Let us assume that we have some partitioning of the $\mathbb{R} = \bigcup_{i=1}^{k} a_i$ where each a_i is an interval. We define a binning operator as $b(x) = \text{mean}\{x \in v^T X \cap a_{i(x)}\}$, where $x \in a_{i(x)}$. We use following notation for simplicity $\langle v, x\rangle_b = b(\langle v, x\rangle)$. Similarly to the previous strategy, in order to preserve good approximation, bins width ($B = \max_i |a_i|$) needs to be adapted in each iteration and the exact equation is given in the following theorem.

Theorem 2 *Using adaptive binning technique with bin width in each iteration at most*

$$\sqrt{-2V(v)\ln\left(\max\left\{0, 1 - \epsilon\sqrt{2\pi V(v)}\right\}\right)},$$

where $V(v)$ is a sum of each classes estimated variances, leads to the computation of the ip$^\times$ function with at most ϵ error.

Proof. We assume that $|\langle v, x-y\rangle - (\langle v, x\rangle_b - \langle v, y\rangle_b)| \leq B$, so

$$\left|\text{ip}^\times(v) - \frac{1}{\sqrt{2\pi V(v)}|X||Y|}\sum_{x,y}\exp\left(-\frac{(\langle v, x\rangle_b - \langle v, y\rangle_b)^2}{2V(v)}\right)\right| =$$

$$\left|\frac{1}{\sqrt{2\pi V(v)}|X||Y|}\sum_{x,y}\left[\exp\left(-\frac{\langle v, x-y\rangle^2}{2V(v)}\right) - \exp\left(-\frac{(\langle v, x\rangle_b - \langle v, y\rangle_b)^2}{2V(v)}\right)\right]\right| \leq$$

$$\left|\frac{1}{\sqrt{2\pi V(v)}|X||Y|}\sum_{x,y}\left[\exp(0) - \exp\left(-\frac{B^2}{2V(v)}\right)\right]\right| =$$

$$\left|\frac{1}{\sqrt{2\pi}V(v)}\left[1 - \exp\left(-\frac{B^2}{2V(v)}\right)\right]\right|.$$

Let us now assume that we are given some acceptable error $\epsilon \geq 0$. We will show how small bins have to be used based on our dataset and current projection.

$$\left|\frac{1}{\sqrt{2\pi V(v)}}\left[1-\exp\left(-\frac{B^2}{2V(v)}\right)\right]\right| \leq \epsilon,$$

but $\exp\left(-\frac{B^2}{2V(v)}\right) \leq 1$, so

$$\frac{1}{\sqrt{2\pi V(v)}}\left[1-\exp\left(-\frac{B^2}{2V(v)}\right)\right] \leq \epsilon,$$

thus

$$\exp\left(-\frac{B^2}{2V(v)}\right) \geq 1-\epsilon\sqrt{2\pi V(v)}.$$

Naturally if $1-\epsilon\sqrt{2\pi V(v)} < 0$ then any B satisfies this inequality (similarly to the sorting and discarding method, it may only happen if we choose very large acceptable error ϵ) so we introduce maximum function here

$$-\frac{B^2}{2V(v)} \geq \ln\left(\max\left\{0, 1-\epsilon\sqrt{2\pi V(v)}\right\}\right),$$

$$B \leq \sqrt{-2V(v)\ln\left(\max\left\{0, 1-\epsilon\sqrt{2\pi V(v)}\right\}\right)}.$$

□

Figure 1 shows how these two bounds behave with increasing size of the acceptable error. In particular one can see that both methods have very similar growth (up to the maximization/minimization symmetry) with changing ϵ. As a result, due to the fact that binning is much more aggressive technique we should expect that using these bounds as the actual bin width/discarding threshold will lead to much greater reduction of the computational complexity when using binning.

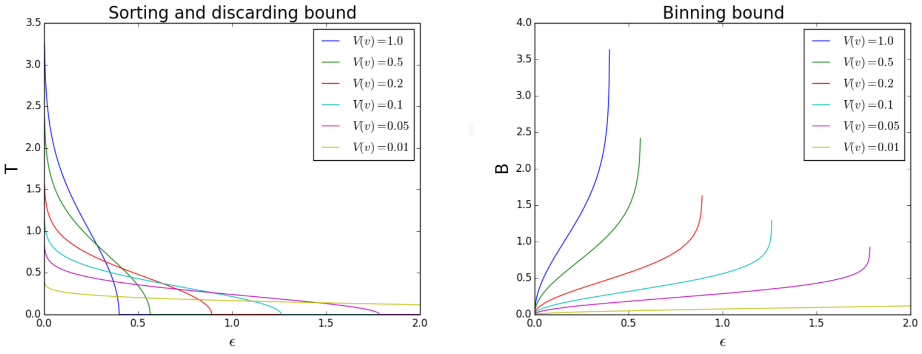

Fig. 1. Plots of the values of the discarding threshold (on the left) and bin width (on the right) as the function of the acceptable error ϵ.

4. Out of sphere optimization

Now we are going to show, that MELC objective function can be efficiently optimized in the whole R^d space by adding some custom regularization term. The importance of this result is the fact that it enables us to use vast amount of existing optimization techniques (such as Adaptive gradient descent, Conjugate Gradients, BFGS, L-BFGS etc.) without adapting them to the sphere constraints. The second important aspect is the fact that this modification does not involve adding any additional constants which have to be fitted. Following theorem describes modified objective function.

Theorem 3 *Given arbitrary sets $X_-, X_+ \subset \mathbb{R}^d$ and corresponding $D_{CS}(v) = D_{CS}([\![v^T X_-]\!], [\![v^T X_+]\!])$ function we have:*

$$d := \max_{\|v\|=1} D_{CS}(v) = \max_{v} D_{CS}(v) - (\|v\|^2 - 1)^2$$

and

$$\{v : \|v\| = 1 \wedge D_{CS}(v) = d\} = \{v : D_{CS}(v) - (\|v\|^2 - 1)^2 = d\}.$$

Proof. According to [5], $\mathrm{D_{CS}}$ is scale invariant so for any $v \in \mathbb{R}^d, c \in \mathbb{R}_+$

$$\mathrm{D_{CS}}(v) = \mathrm{D_{CS}}(cv).$$

As a result also

$$\mathrm{D_{CS}}(v) - (\|v\|^2 - 1)^2 = \mathrm{D_{CS}}(cv) - (\|v\|^2 - 1)^2,$$

but as $-(\|v\|^2 - 1)^2 \leq 0$ and $-(\|v\|^2 - 1)^2 = 0 \iff \|v\| = 1$ we have that $\mathrm{D_{CS}}(v) - (\|v\|^2 - 1)^2$ is maximized for v with norm 1 and that it is equal to $\mathrm{D_{CS}}(v)$. As a result sets of solutions of both problems are identical.

□

Consequently we can apply any advanced optimization technique which is not designed to work on the sphere to optimize $\mathrm{D_{CS}}$ criterion. In particular we can use L-BFGS [3] instead of more complex and less popular RBFGS [10] and previously proposed [5] less efficient – gradient descent on sphere method. At the same time the norm of the candidate solution will stay close to 1 so we will not suffer from numerical problems [5].

It is worth noting that despite similarity to the L_2 regularization [13] of the additive loss function (or weight decay from neural networks) this additional terms serves no regularization purposes nor it affects the actual function value. It only guides the gradient based optimizers towards more informative regions of the state space.

From the practical point of view we also need a gradient of the new function but thanks to the additivity of derivative operator we get

$$\nabla \left[\mathrm{D_{CS}}(v) - (\|v\|^2 - 1)^2\right] = [\nabla \mathrm{D_{CS}}(v)] - 4v(\langle v, v\rangle - 1),$$

and we can use any optimization software able to maximize a function given $(f, \nabla f)$.

Tab. 1. Mean ratio of exp calls between approximated technique and original method during optimizations.

Method	CG		L-BFGS-B	
name	bin	dist	bin	dist
australian	0.11	0.44	0.11	0.45
breast-cancer	0.10	0.46	0.10	0.46
diabetes	0.21	0.56	0.22	0.54
fourclass	0.19	0.51	0.19	0.49
german.numer	0.15	0.47	0.19	0.46
heart	0.29	0.47	0.26	0.47
ionosphere	0.25	0.55	0.24	0.54
liver-disorders	0.29	0.65	0.31	0.67
sonar	0.32	0.53	0.29	0.50
splice	0.19	0.44	0.16	0.43

5. Evaluation

We evaluate proposed approximations on 10 datasets from UCI repository [2] and libSVM's repository [4, 7]. Both D_{CS} and approximations are coded in Python using numpy and scipy [8]. We use scipy's optimization module to perform training of all models using two optimization techniques – Conjugate Gradients (CG) and L-BFGS-B [3]. Each experiment is performed in cross validation manner with multiple starting points (randomly selected, but constant across methods to achieve comparable results) due to the convergence of MELC optimization to local optima. We analyze γ hyperparameter of D_{CS} in $[0.1, 0.5, 1.0, 1.5, 2.0]$ and acceptable error $\epsilon \in [0.01, 0.02, 0.03, 0.05, 0.1, 0.2, 0.5]$. Similarly to the original paper we use Balanced Accuracy (BAC[2]) as the measure of classification correctness due to MELC highly balanced formulation.

First, we investigate how big is mean reduction of computations using each of the approximating schemes. Table 1 reports mean ratio of exp function calls (which is equivalent to number of pairs analyzed in each $\mathrm{ip}^{\times}$ evaluation when optimizing whole D_{CS} function and its gradient) in given method to the original implementation.

One can easily notice that sorting and discarding method (denoted as "dist") roughly halves the number of analyzed pairs, while binning (denoted as "bin") reduces it 3–10 times. It is an obvious consequence of the fact that binning is much more aggressive method. It appears that strength of reduction depends only on the dataset, not on the optimization algorithm used which suggests, that projections for which particular level of possible reduction are uniformly distributed over the space of all projections. These effects are also heavily dependent[3] on the choice of γ and ϵ

[2] $\mathrm{BAC} = \frac{1}{2}\left(\frac{\mathrm{TP}}{\mathrm{TP+FN}} + \frac{\mathrm{TN}}{\mathrm{TN+FP}}\right)$

[3] We do not include the exact values in the Table for better readability.

which is the obvious consequence of Theorems 1 and 2 saying that with increasing variance (which is proportional to γ^2) the reduction strength decreases superlinearly.

The set of heat maps in Figure 2 shows differences between BAC obtained by the original D_{CS} and each approximation for a given dataset and γ, ϵ hyperparameters pair. In general, up to few isolated cases errors are on the level of $0.5\% - 3\%$. For small γ values errors introduced by the approximation are significantly higher and for sonar and splice datasets can grow to even 10%. Fortunately, these are very rare phenomena. Even more interesting is the fact that for many experiments we actually noticed increase in the BAC score (bluish elements). This might be the consequence of more rough evaluation of the function (and gradient) values leading to optimization less prone to falling into local maxima. Our hypothesis is that it acts like a regularization helping to train MELC model.

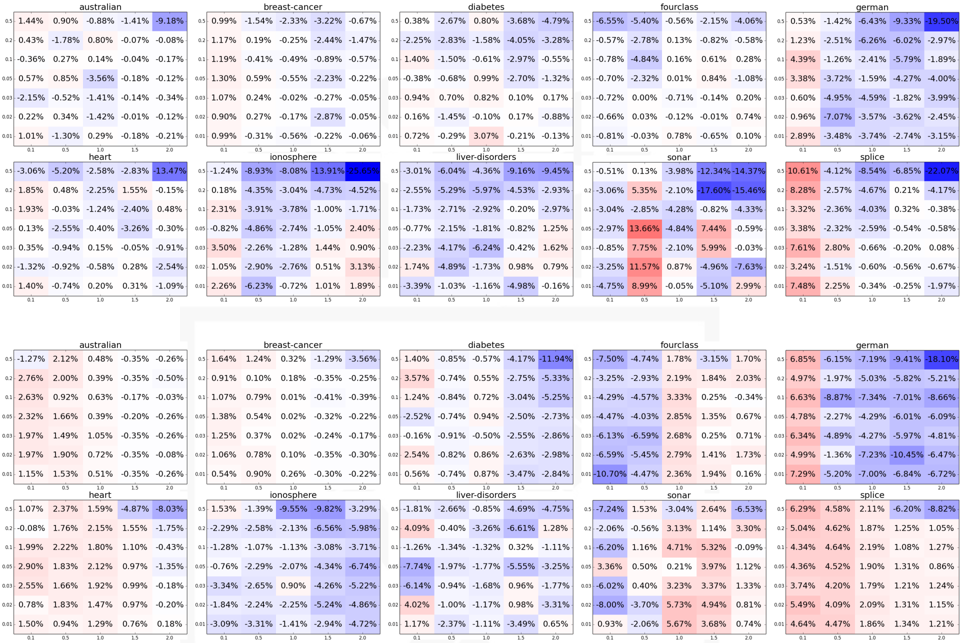

Fig. 2. Comparison of the cross validation BAC scores between given approximated strategy (two top rows sorting and discarding, two bottom ones binning), γ hyperparameter of D_{CS} (x-axis), accepted error ϵ (y-axis). Positive values (and corresponding red colors) represent decrease in BAC score while negative values and corresponding blue colors – increase after using approximated method.

Analysis of the number of iterations of each optimization method required to converge (see Table 2) shows that both approximations significantly simplify the problem. It is important to notice that the number of iterations is not the number of D_{CS} function evaluations (as both Conjugate Gradients and L-BFGS-B evaluate it multiple times in each iteration, especially during line searches). Consequently, number of iterations cannot be used as a measure of optimization speed but it says much about the complexity of the function being maximized. This seems to confirm

Tab. 2. Number of optimization methods' iterations.

Method	CG			L-BFGS-B		
name	bin	$\mathrm{D_{CS}}$	dist	bin	$\mathrm{D_{CS}}$	dist
australian	4	36	22	11	39	37
breast-cancer	4	35	8	6	39	14
diabetes	3	30	20	18	36	29
fourclass	4	12	10	6	15	14
german.numer	7	60	32	7	58	38
heart	3	40	19	12	34	20
ionosphere	5	600	216	18	384	152
liver-disorders	4	30	22	22	43	30
sonar	4	262	115	15	139	100
splice	4	92	26	14	65	41

our claim that approximation works similar to the regularization and thus it reduces small irregularities of the error surface due to the removal of small elements from the $\mathrm{ip}^{\times}$ internal summation.

Experiments also showed importance on the regularization technique added to perform out of sphere optimization. During maximization of $\mathrm{D_{CS}}$ in sonar and german datasets, norms of v rapidly grew to over 1000 if we turn off this modification and still use CG/L-BFGS-B. As a result the optimization problem became extremely hard and we needed tens of thousands $\mathrm{D_{CS}}$ evaluation in order to converge. Adding regularizing term reduced the norm to nearly 1 and number of required function calls by two orders of magnitude.

6. Conclusions

In this paper we proposed two simple approximation schemes for faster computation of MELC objective function and its gradient. We proved that in order to achieve constant error bound during optimization one needs a specific adaptive strategy for each of them and gave a simple, closed form equations for setting required parameters based on the user-specified acceptable level of error in the $\mathrm{ip}^{\times}$ function value. We also showed how one can easily change the objective function in order to use wide range of existing optimizers while at the same time still work near the unit sphere which, as described in the MELC theory [5], is important from the numerical point of view.

During extensive evaluation we confirmed that such approach is valid in terms of reducing the mean number of exp calls by even an order of magnitude while not sacrificing the resulting classifiers accuracy. In fact the experiments suggest that proposed method acts like some kind of regularization which might not only simplify the optimization problem but also slightly increase the obtained results.

7. References

[1] Anthony M., *Partitioning points by parallel planes,* Discrete mathematics 282 (1), 2004, pp. 17–21.

[2] Blake C., Merz Ch.J., {*UCI*} *repository of machine learning databases,* 1998.

[3] Byrd R.H., Lu P., Nocedal J., Zhu C., *A limited memory algorithm for bound constrained optimization,* SIAM Journal on Scientific Computing 16 (5), 1995, pp. 1190–1208.

[4] Chang C.C., Lin C.J., *Libsvm: A library for support vector machines,* ACM Transactions on Intelligent Systems and Technology (TIST) 2(3), 2011, pp. 27:1–27:27.

[5] Czarnecki W.M., Tabor J., *Multithreshold entropy linear classifier,* arXiv preprint arXiv:1408.1054, 2014.

[6] Elgammal A., Duraiswami R., Davis L.S., *Efficient kernel density estimation using the fast gauss transform with applications to color modeling and tracking,* IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (11), 2003, pp. 1499–1504.

[7] Ho T. K., Kleinberg E.M., *Building projectable classifiers of arbitrary complexity,* Pattern Recognition, 1996., IEEE Proceedings of the 13th International Conference on, 2, 1996, pp. 880–885.

[8] Jones E., Oliphant T., Peterson P., *Scipy: Open source scientific tools for python,* http://www. scipy. org/, 2001.

[9] Principe J.C., *Information theoretic learning: Rényi's entropy and kernel perspectives*, Springer Science & Business Media, New York, USA, 2010.

[10] Qi C., Gallivan K.A., Absil P.A. *Riemannian bfgs algorithm with applications,* Recent advances in optimization and its applications in engineering, Springer, 2010, pp. 183–192.

[11] Silverman B.W., *Density estimation for statistics and data analysis*, Monographs on Statistics and Applied Probability 26, CRC Press, 1986.

[12] Silverman B.W., *Algorithm as 176: Kernel density estimation using the fast fourier transform,* Applied Statistics, 1982, pp. 93–99.

[13] Vapnik V., *The nature of statistical learning theory,* Springer, New York, USA, 2000.

[14] Yang C., Duraiswami R., Gumerov N.A., Davis L., *Improved fast gauss transform and efficient kernel density estimation,* Proceedings of the Ninth IEEE International Conference on Computer Vision, 2003, pp. 664–671.