A DC Algorithm Scheme For Generalized Eigenvalue Classifiers

1st Xing Gao
School of Mathematics and Statistics
Nanjing University of Science and Technology
Nanjing, China
gaoxing@njust.edu.cn

2nd Tao Yan *
School of Mathematics and Statistics
Nanjing University of Science and Technology
Nanjing, China
tyan@njust.edu.cn

3rd Kai Wang
School of Mathematics and Statistics
Nanjing University of Science and Technology
Nanjing, China
wangkaihawk@njust.edu.cn

***Abstract*—Generalized eigenvalue classifiers handle the classification problem very well. However, the major disadvantages of generalized eigenvalue based method lie in that they may be time-consuming and fail to get a stable or optimal solution due to the matrix singularity. This paper proposes a new method using a DC Algorithm scheme to solve a fractional optimization models which ensure the optimality of the solution. Numerical experiments show that the proposed method can get better results for public data sets.**

***Index Terms*—fractional optimization, DC Algorithm, generalized eigenvalue, classification**

I. Introduction

Support Vector Machines (SVM) [1] is widely used in classification, but it cannot effectively handle the XOR problem and need to solve a complex quadratic programming problem (QPP). Proximal Support Vector Machines via Generalized Eigenvalues (GEPSVM) [2] is an effective classification method overcoming the above two problems. Then two non-parallel proximal planes are obtained by solving two generalized eigenvalue problems(GEPs).

Inspired by GEPSVM, many generalized eigenvalue classifiers have been further studied extensively. Yang et al. proposed generalized eigenvalue extreme learning machine(GEPELM)[3] and laplacian generalized eigenvalues extreme learning machine(LapGELM)[4]. Sun et al.[5] first proposed Multiview learning with generalized eigenvalue proximal support vector machines(MvGSVM). Liang et al.[6] proposed manifold regularized GEPSVM(MRGEPSVM). Shao et al. proposed capped-norm proximal support vector machine(CPSVM)[7]. All of these classifiers obtained two hyperplanes by solving two GEPs.

When it comes to solving GEP, the related classifiers may fail to get a stable and optimal solution due to the matrix singularity occurring. At the same time, solving GEP may be very time-consuming when facing high-dimensional or huge data.Therefore, it is very important and meaningful to design a fast and robust algorithm for those classifiers without solving GEP.

The paper is organized as follows. In Section II the structure of identical fractional optimization model is introduced in detail. In Section III the algorithm is detailed.Numerical experiments are shown in Section IV. Finally, in Section V conclusions and future work are reported.

* Corresponding author.

II. The structure of fractional optimization model

Generalized eigenvalue classifiers have the similar structure as fractional optimization model. In this paper, we only take the GEPSVM model [2] as an example in detail and gives new algorithm without solving GEP.

Consider the binary classification problem of the training set $\boldsymbol{X} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$, where $\boldsymbol{x}_i \in \mathbb{R}^n$ and $y_i \in \{-1, 1\}$.In addition, let the matrices $\boldsymbol{A} \in \mathbb{R}^{m_1 \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{m_2 \times n}$ $(m_1 + m_2 = m)$ represent data points belonging to class 1 (positive class) and class 2 (negative class), respectively. The goal of the original GEPSVM with linear kernel is to look for two non-parallel hyperplanes in the $\mathbb{R}^n$:

$$\boldsymbol{\omega_1^T x} + \gamma_1 = 0 \text{ and } \boldsymbol{\omega_2^T x} + \gamma_2 = 0. \tag{1}$$

If we generalize linear classifiers to nonlinear classifiers, the kernel based nonlinear surfaces is as following:

$$\boldsymbol{K}\left(\boldsymbol{x}^T, \boldsymbol{C}^T\right)\boldsymbol{u}_1 + \gamma_1 = 0, \boldsymbol{K}\left(\boldsymbol{x}^T, \boldsymbol{C}^T\right)\boldsymbol{u}_2 + \gamma_2 = 0, \tag{2}$$

where

$$\boldsymbol{C} := \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{B} \end{bmatrix} \tag{3}$$

and $\boldsymbol{K}$ is any kernel. The kernel commonly used in nonlinear classification is the Gaussian kernel(i.e.$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\mu \left\|\mathrm{x_i} - \mathrm{x_j}\right\|)$). Each of these hyperplanes is closest to its corresponding category of data points and as far away from the other as possible. To obtain the first surface of (1), the following optimization problem can be performed:

$$\min_{\boldsymbol{z}} r(\boldsymbol{z}) := \frac{\boldsymbol{z}^T\boldsymbol{G}\boldsymbol{z}}{\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z}}. \tag{4}$$

and we can get the first surface of (2) by solving:

$$\min_{\boldsymbol{z}} s(\boldsymbol{z}) := \frac{\boldsymbol{z}^T\boldsymbol{L}\boldsymbol{z}}{\boldsymbol{z}^T\boldsymbol{M}\boldsymbol{z}}, \tag{5}$$

where $\boldsymbol{z} = \begin{pmatrix} \boldsymbol{\omega} \\ \gamma \end{pmatrix}$ and $\boldsymbol{G}, \boldsymbol{H} \in \mathbb{R}^{(n+1)\times(n+1)}$; $\boldsymbol{L}, \boldsymbol{M} \in \mathbb{R}^{(m+1)\times(m+1)}$ are defined as:

$$\begin{aligned}
\boldsymbol{G} &= \left[\begin{array}{cc} \boldsymbol{A} & \boldsymbol{e} \end{array}\right]^T \left[\begin{array}{cc} \boldsymbol{A} & \boldsymbol{e} \end{array}\right] + \delta \boldsymbol{I} \\
\boldsymbol{H} &= \left[\begin{array}{cc} \boldsymbol{B} & \boldsymbol{e} \end{array}\right]^T \left[\begin{array}{cc} \boldsymbol{B} & \boldsymbol{e} \end{array}\right] \\
\boldsymbol{L} &= \left[\begin{array}{cc} \boldsymbol{K}\left(\boldsymbol{A}, \boldsymbol{C}^{\boldsymbol{T}}\right) & \boldsymbol{e} \end{array}\right]^T \left[\begin{array}{cc} \boldsymbol{K}\left(\boldsymbol{A}, \boldsymbol{C}^{\boldsymbol{T}}\right) & \boldsymbol{e} \end{array}\right] + \delta \boldsymbol{I} \\
\boldsymbol{M} &= \left[\begin{array}{cc} \boldsymbol{K}\left(\boldsymbol{B}, \boldsymbol{C}^{\boldsymbol{T}}\right) & \boldsymbol{e} \end{array}\right]^T \left[\begin{array}{cc} \boldsymbol{K}\left(\boldsymbol{B}, \boldsymbol{C}^{\boldsymbol{T}}\right) & \boldsymbol{e} \end{array}\right]
\end{aligned} \tag{6}$$

where $\delta > 0$, and $\boldsymbol{I}$ is a identity matrix of appropriate dimension and $\boldsymbol{e}$ is a column vector of appropriate dimension where each element is 1.

By the same way, generating the second surfaces of (1) and (2) leads to the following two minimization problems,respectively.

$$\min_{\boldsymbol{z}} \tilde{r}(\boldsymbol{z}) := \frac{\boldsymbol{z}^T \tilde{\boldsymbol{G}} \boldsymbol{z}}{\boldsymbol{z}^T \tilde{\boldsymbol{H}} \boldsymbol{z}} \tag{7}$$

$$\min_{\boldsymbol{z}} \tilde{s}(\boldsymbol{z}) := \frac{\boldsymbol{z}^T \tilde{\boldsymbol{L}} \boldsymbol{z}}{\boldsymbol{z}^T \tilde{\boldsymbol{M}} \boldsymbol{z}} \tag{8}$$

where $\tilde{\boldsymbol{G}}, \tilde{\boldsymbol{H}} \in \mathbb{R}^{(n+1)\times(n+1)}$ and $\tilde{\boldsymbol{L}}, \tilde{\boldsymbol{M}} \in \mathbb{R}^{(m+1)\times(m+1)}$ are defined as:

$$\begin{aligned}
\tilde{\boldsymbol{G}} &= \left[\begin{array}{cc} \boldsymbol{B} & \boldsymbol{e} \end{array}\right]^T \left[\begin{array}{cc} \boldsymbol{B} & \boldsymbol{e} \end{array}\right] + \delta \boldsymbol{I} \\
\tilde{\boldsymbol{H}} &= \left[\begin{array}{cc} \boldsymbol{A} & \boldsymbol{e} \end{array}\right]^T \left[\begin{array}{cc} \boldsymbol{A} & \boldsymbol{e} \end{array}\right] \\
\tilde{\boldsymbol{L}} &= \left[\begin{array}{cc} \boldsymbol{K}\left(\boldsymbol{B}, \boldsymbol{C}^{\boldsymbol{T}}\right) & \boldsymbol{e} \end{array}\right]^T \left[\begin{array}{cc} \boldsymbol{K}\left(\boldsymbol{B}, \boldsymbol{C}^{\boldsymbol{T}}\right) & \boldsymbol{e} \end{array}\right] + \delta \boldsymbol{I} \\
\tilde{\boldsymbol{M}} &= \left[\begin{array}{cc} \boldsymbol{K}\left(\boldsymbol{A}, \boldsymbol{C}^{\boldsymbol{T}}\right) & \boldsymbol{e} \end{array}\right]^T \left[\begin{array}{cc} \boldsymbol{K}\left(\boldsymbol{A}, \boldsymbol{C}^{\boldsymbol{T}}\right) & \boldsymbol{e} \end{array}\right]
\end{aligned} \tag{9}$$

Generalized eigenvalue classifiers are only different in the construction of matrices $\boldsymbol{G}(\tilde{\boldsymbol{G}}), \boldsymbol{H}(\tilde{\boldsymbol{H}}), \boldsymbol{L}(\tilde{\boldsymbol{L}}), \boldsymbol{M}(\tilde{\boldsymbol{M}})$.Since the above four optimization models have the same fractional optimization structure, the methods for solving them are similar. Hence, we will only take (4) for discussion.

When $\boldsymbol{H}$ is positive definite, namely, the columns of the matrix $\left[\begin{array}{cc} \boldsymbol{A} & \boldsymbol{e} \end{array}\right]$ are linearly independent, the optimization objective (4) is a generalized Rayleigh quotient. Using the well-known properties of the generalized Rayleigh quotient, the solution of (4) comes directly from the solution of the following GEP: $\boldsymbol{G}\boldsymbol{z} = \lambda \boldsymbol{H}\boldsymbol{z}$.

However, in real-world problems, especially image classification, matrix $\boldsymbol{H}$ may be semi-positive, resulting in unstable solutions. So in next section,we will introduce a new method to solve (4).

III. PROPOSED METHOD

With regard to the fractional optimization model (4) , we rewrite it as:

$$\min_x f(x) = \frac{g(x)}{h(x)}, \tag{10}$$

where $g(x) = x^T G x, h(x) = x^T H x (x = z)$.

Many iterative fractional optimization algorithms are designed on the basis of parametric methods, which regard the following optimization problems:

$$\min_x \{F(x) = g(x) - c^* h(x)\} \tag{11}$$

as equivalent to the problem(10). Where c^* is the optimum value of the objective function for the fractional optimisation problem(10), and c^* is generally unknown. The commonly used Dinkelbach[8] method is an iterative parametric fractional optimization algorithm, which performs iterative point updates as follows:

$$\begin{aligned}
x_{k+1} &= \arg\min_x \{g(x) - c_k h(x)\} \\
c_k &= \frac{g(x_k)}{h(x_k)} > 0
\end{aligned} \tag{12}$$

In this way, the original problem (10) can be solved through the iterative algorithm of (12).

Next we focus on how to solve the following unconstrained subproblem in (12):

$$\min_x \{g(x) - c_k h(x)\}. \tag{13}$$

Noting that G is a positive defined matrix and H is a semi-positive defined matrix, therefore both $g(x)$ and $h(x)$ are convex quadratic functions. Then the subproblem(13) is a DC (Difference of Convex functions) programming problem,which can be resolved using the DC Algorithm (DCA)[9] that recently was used in SVM [10–14] .

The whole DCA frame should update two columns of x^i and y^i. Since $g(x)$ and $h(x)$ are differentiable, $\partial h(x)$ is reduced to a singleton and one has $\partial h(x) = \{\nabla h(x)\}$,then we can get $y^i = \nabla h(x^i) = 2Hx^i$. Updating x^i only requires solving an unconstrained convex quadratic programming problem:

$$x^{i+1} \in \operatorname{argmin}\{g(x) - c_k x^T y^i : x \in \mathbb{R}^n\}. \tag{14}$$

Thus we can obtain the explicit iteration formula of x^i through first-order necessary conditions:

$$x^{i+1} = c_k (R^T)^{-1} R^{-1} H x^i, \tag{15}$$

where lower triangle matrix R is from the Cholesky decomposition of $G = RR^T$. Naturally, we obtain the DCA scheme for(13) in Algorithm1.

Algorithm 1 DCA scheme for(13)

Input: $x^0 \in \operatorname{dom} h$; c_k; G; H; I_{max}; ε_1;
Output: x^i;

```
i⇐ 0;
R = chol(G);
J = (R^T)^{-1} R^{-1} H;
while i ≤ I_max do
    x^{i+1} = c_k J x^i
    if ‖x^{i+1} − x^i‖ ≤ ε_1 then
        return
    end if
    i = i + 1
end while
```

Next, by applying Algorithm1, we propose the Algorithm2 to solve (10).

The main frame (12) has superlinear convergence [8], while the subproblem(13) has linear convergence [15]. So

Algorithm 2 Proposed Method for(10)

Input: $x_0 \in \operatorname{dom} h$; G; H; K_{max}; ε_2;
Output: x_k;

$k \Leftarrow 0$
while $k \leq K_{max}$ **do**
  $c_k = \frac{x_k^T G x_k}{x_k^T H x_k}$
  get x^i by Algorithm1
  $x_{k+1} = x^i$
  if $\|x_{k+1} - x_k\| \leq \varepsilon_2$ **then**
    return
  end if
  $k = k + 1$
end while

Algorithm2 can ensure the optimality of the solution and overcome the theoretical fallacy and imprecision caused by matrix singularity.

IV. Numerical Experiments

To show the performance of the proposed algorithm,we present the results of publicly accessible data sets from the UCI repositories [16] and MNIST Handwritten Digits [17].

The MNIST Handwritten Digits database(about 6K training examples of each class; 1K test examples) consists of 8-bit grayscale images of "0" through "9" as shown in Fig.1.

Fig. 1. An illustration of 10 subjects in the MNIST Handwritten Digits database.

UCI repositories is an open source suitable for pattern recognition and machine learning proposed by the University of California Irvine.

The SEMG Basic Hand Movements dataset from UCI repositories consists of the following six movements,which can be considered as daily hand grasps: a)Cylindrical(cyl)-for holding cylindrical tools; b)Tip(tip)-for holding small tools; c)Hook(hoo)-for supporting a heavy load; d)Palmar(pal)-for grasping with palm facing the object; e)Spherical(sph)-for holding spherical tools; f)Lateral(lat)-for holding thin, flat objects. An illustrative photo is shown in Fig.2.

For linear kernel,the parameter δ was selected from the values $\{10^i | i = -7, -6, \ldots, 7\}$.For nonlinear kernels, the Gaussian kernel has been selected (i.e.K($\boldsymbol{x}_i, \boldsymbol{x}_j$) = $\exp(-\mu \|\mathrm{x_i} - \mathrm{x_j}\|)$). The kernel parameter μ was chosen from the values$\{10^i | i = -4, -3, -2, -1\}$,while the parameter δ was selected from the values $\{10^i | i = -2, -1, \ldots, 4\}$.

Here are some noteworthy points:

- The best parameters, from which the algorithms got the best accuracy over the validation set(10 % of the training set at random), were selected.
- The final algorithms were trained by using the best parameters and all training points, and then tested on the testing data.
-The norm of the initial iteration point we selected for Algorithm1 and Algorithm2 should be as small as possible, in order to prevent excessive $g(x)$ and $h(x)$ in the iteration process, resulting in overflow of computer memory and unnecessary errors.
- We get R with the code $R = chol(G)$ and get J with the code $J = R\backslash$ (R$'\backslash$ H)in matlab.

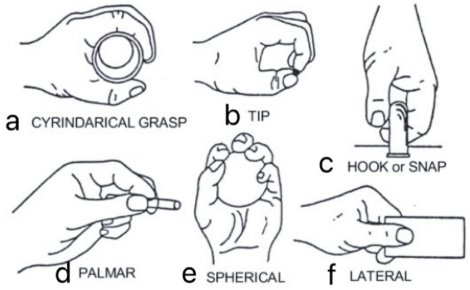

Fig. 2. An illustration of 6 movements in the SEMG Basic Hand Movements database.

First, we select eight pairwise digits from MINST of varing difficulty for odd vs. even digit classification.Selected classes are shown in Table I.

TABLE I
THE COMPARISON OF MNIST HANDWRITTEN DIGITS DATABASE

Selected class	DCA scheme		Generalized eigenvalue-based method	
	Accuracy(%)	Train time(s)	Accuracy(%)	Train time(s)
0 vs.1	**99.95**	**0.02**	99.87	2.07
1 vs.2	**98.66**	**0.02**	95.29	2.13
2 vs.3	**96.62**	**0.02**	95.98	2.15
2 vs.5	**97.04**	**0.02**	96.31	2.27
2 vs.7	**97.62**	**0.02**	96.84	2.36
3 vs.4	**99.15**	**0.02**	98.9	2.16
3 vs.8	**96.17**	**0.02**	95.21	2.45
5 vs.6	**96.32**	**0.02**	95.24	2.17

As shown in Table I, the DCA scheme outperforms Generalized eigenvalue-based method in terms of classification accuracy and train time.

Then, we select nine pairwise movements from SEMG of varing difficulty for classification.Selected classes are shown in Table II.

TABLE II
THE COMPARISON OF SEMG BASIC HAND MOVEMENTS DATABASE

Selected class	DCA scheme		Generalized eigenvalue-based method	
	Accuracy(%)	Train time(s)	Accuracy(%)	Train time(s)
cyl vs.tip	**71.67**	**0.48**	67.50	158.35
cyl vs.pal	**77.51**	**0.44**	76.67	158.63
cyl vs.lat	**76.67**	**0.44**	75.00	158.13
tip vs.hoo	**72.50**	**0.46**	69.17	159.96
tip vs.sph	**71.67**	**0.48**	70.00	158.83
hoo vs.pal	**75.00**	**0.46**	68.33	159.60
hoo vs.lat	**75.83**	**0.50**	69.17	163.82
pal vs.sph	**74.17**	**0.47**	72.50	163.77
sph vs.lat	**79.17**	**0.48**	75.00	163.97

As shown in Table II, the DCA scheme outperforms Generalized eigenvalue-based method in terms of classification accuracy and train time.Especially in terms of the time consumed, the former is far less than the latter.

Last, we prove the rationality of proposed algorithm by doing experimental results on the following data sets from UCI.

TABLE III
THE COMPARISON OF SOME UCI DATABASE

Data Set	DCA scheme		GE method	
m * n	Accuracy(%)	Train time(s)	Accuracy(%)	Train time(s)
Swarm Behavior 24016*2400	**97.75**	**0.37**	95.25	1.63
Mushroom 8124 * 22	**92.88**	0.08	86.46	**0.0004**
WDBC* 569 * 30	**78.09**	**0.013**	74.78	1.227
Malware* 4465 * 241	**81.03**	**1.54**	79.69	557.37

* on the upper right of the dataset represents using nonlinear kernel

Table III shows the linear and nonlinear kernel comparison of the DCA scheme versus generalized eigenvalue based method on publicly available datasets from the UCI database. From Table III, it is easily observed that some cases in the DCA scheme is faster than generalized eigenvalue-based method, especially in the Malware dataset. Then we can infer that the running time of generalized eigenvalue-based method is very sensitive to the dimension and structure of the matrix while our method is very robust. At the same time, we find the DCA scheme exhibits relatively superior classification performance to the other.

V. Conclusion

Generalized eigenvalue classifiers such as GEPSVM sometimes handle the classification problem very well. Their solutions can be obtained directly from solving GEP, but a major disadvantage of eigenvalue based method is that it may fails to get good generalization ability when the matrix singularity appears. In this paper, we propose a DCA scheme to avoid troubles in solving GEPs. Moreover, compared with classic GEPSVM using generalized eigenvalue based method, it not only has comparable or better classification performance on public datasets, but also computes faster. In future, the work includes research in more efficient DCA scheme and the extension of other generalized eigenvalue classifiers.

References

[1] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[2] Olvi L Mangasarian and Edward W Wild. Multisurface proximal support vector machine classification via generalized eigenvalues. *IEEE transactions on pattern analysis and machine intelligence*, 28(1):69–74, 2005.

[3] Ping Sun and Liming Yang. Generalized eigenvalue extreme learning machine for classification. *Applied Intelligence*, 1:30, 2022.

[4] Wang Xue and Liming Yang. Laplacian generalized eigenvalues extreme learning machine. *Neural Processing Letters*, 1:33, 2022.

[5] Shiliang Sun, Xijiong Xie, and Chao Dong. Multiview learning with generalized eigenvalue proximal support vector machines. *IEEE transactions on cybernetics*, 49(2):688–697, 2018.

[6] Jun Liang, Fei-yun Zhang, Xiao-xia Xiong, Xiao-bo Chen, Long Chen, and Guo-hui Lan. Manifold regularized proximal support vector machine via generalized eigenvalue. *International Journal of Computational Intelligence Systems*, 9(6):1041–1054, 2016.

[7] Pei-Wei Ren, Chun-Na Li, Yuan-Hai Shao, et al. Capped-norm proximal support vector machine. *Mathematical Problems in Engineering*, 2022, 2022.

[8] Werner Dinkelbach. On nonlinear fractional programming. *Management science*, 13(7):492–498, 1967.

[9] Hoai An Le Thi and Tao Pham Dinh. Open issues and recent advances in dc programming and dca. *Journal of Global Optimization*, pages 1–58, 2023.

[10] Bo Liu, Ruiguang Huang, Yanshan Xiao, Junrui Liu, Kai Wang, Liangjiao Li, and Qihang Chen. Adaptive robust adaboost-based twin support vector machine with universum data. *Information Sciences*, 609:1334–1352, 2022.

[11] Guoquan Li, Lin Yin, Linxi Yang, and Zhiyou Wu. Robust projection twin support vector machine via dc programming. *Optimization*, 71(4):1189–1211, 2022.

[12] Hossein Moosaei, Fatemeh Bazikar, Saeed Ketabchi, and Milan Hladík. Universum parametric-margin ν-support vector machine for classification using the difference of convex functions algorithm. *Applied Intelligence*, 52(3):2634–2654, 2022.

[13] Jun Sun and Wentao Qu. Dca for sparse quadratic kernel-free least squares semi-supervised support vector machine. *Mathematics*, 10(15):2714, 2022.

[14] Guoquan Li, Linxi Yang, Zhiyou Wu, and Changzhi Wu. Dc programming for sparse proximal support vector machines. *Information Sciences*, 547:187–201, 2021.

[15] Hadi Abbaszadehpeivasti, Etienne de Klerk, and Moslem Zamani. On the rate of convergence of the difference-of-convex algorithm (dca). *Journal of Optimization Theory and Applications*, pages 1–22, 2023.

[16] P.M. Murphy and D.W. Aha. Uci machine learning repository, 1992. www.ics.uci.edu/mlearn/MLRepository.html.

[17] S.T. Roweis. Mnist handwritten digits, 1998. https://cs.nyu.edu/~roweis/data.html.