Kernel Optimization using Conformal Maps for the Minimal Complexity Machine

Skyler Badge, Jayadeva * and Sumit Soman†
Department of Electrical Engineering, Indian Institute of Technology Delhi

***Abstract*—The Minimal Complexity Machine (MCM) has been shown to be kernel-based learning model that can learn sparse models while providing generalization comparable to the Support Vector Machine (SVM). One of the approaches for optimizing the SVM kernel includes selection of a set of empirical cores from the training data and constructing a conformal kernel map. This paper extends this approach to the MCM and shows that it can be used to learn robust kernel representations that generalize well on test data when compared to the conventional MCM without an optimized kernel.**

***Index Terms*—Minimal Complexity Machine, Kernel Optimization, Conformal Kernel, Generalized Eigenvalue Problem, Support Vector Machines**

I. Introduction

The success of Support Vector Machine (SVM) and learning models based on similar principles has largely been due to the use of kernel functions. These functions map the input samples to a high dimensional space that introduces separability, thereby enabling the function being optimized to find a hyperplane that generalizes well. The choice of the kernel function plays an important role, and several methods have been proposed in literature for kernel optimization. Indeed, they have also been applied to SVM based models and have shown improved performance.

One of the limitations arising out of using SVM is the fact that its model complexity in terms of the VapnikChervonenkis (VC) dimension can be unbounded [1]. To this end, the Minimal Complexity Machine (MCM)[2] and its variants [3], [4], [5], [6], [7], [8] have shown that it is possible to develop an alternate formulation that minimizes the VC dimension while also providing comparable generalization. This has resulted in the trained model being sparse, in terms of the number of support vectors found by the optimization routine, which has implicit benefits when such models have to be implemented on resource constrained platforms, where low memory footprints for representing the models are desirable. Our motivation stems from the fact that kernel optimization for the MCM has not been explored in available literature so far. Analogous to the SVM, we expect that choosing the optimal kernel would result in further improving the performance of the MCM-based models.

Our focus in this paper is on kernel optimization methods that use the training samples to optimize the kernel function learnt. Approaches in literature such as by Cristianini *et al.* [9] Lanckrict *et al.* [10] and Xiong *et al.* [11] have presented approaches in this direction. Motivated by Wu and Amari [12], Shah *et al.* [13] presented an approach for kernel optimization for the SVM where the set of empirical cores chosen from the training data resulted in construction of a kernel function whose solution was given by the Generalized Eigenvalue problem based on the proposed conformal transformation. We extend this idea to the MCM and show that it results in improved generalization on various benchmarking datasets.

* Corresponding Author, Web: http://web.iitd.ac.in/~jayadeva, Email: jayadeva@ee.iitd.ac.in
† Sumit Soman was involved in the initial phases of this work as a Ph.D. student.

II. Background

Given a set of M data samples in n dimensions $X \in \mathbb{R}^{M \times n}$ with labels $Y = y_i \in \{-1, +1\}, \forall i = 1, 2, ..., M$ called the training data, the binary classification problem involves prediction of the label $\hat{y}$ for a test sample $\hat{x} \in \mathbb{R}^{\aleph}$. One of the popular models used to solve this is the Support Vector Machine (SVM).

The SVM for binary classification solves the Quadratic Programming Problem (QPP) given by (1)-(3).

$$\min_{w,b,\xi} \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{M}\xi_i \tag{1}$$

subject to,

$$y_i(w^T x^i + b) + \xi_i \geq 1, \forall i = 1, 2, ..., M \tag{2}$$

$$\xi_i \geq 0, \forall i = 1, 2, ..., M. \tag{3}$$

Here, the model parameters are represented by the separating hyperplane $w^T x + b = 0$; ξ_i are the positively constrained slack variables that allow for *"softness"* of the constraints. Further, C is a hyperparameter that controls the trade-off between minimizing the L_2 norm of the weight vector and the slack variables. The class to which a test point $\hat{x} \in \mathbb{R}^n$ belongs is found by evaluating $sgn(w^T \hat{x} + b)$, where $sgn(\cdot)$ denotes the sign of the expression evaluated. The notion of the kernel function in case of the SVM is introduced by mapping the input samples x^i to a space spanned by the function $\phi(\cdot)$, such that the kernel function for every pair of input samples is denoted by $K(x^i, x^j) = \phi(x^i)^T \phi(x^j)$. The dual formulation of the SVM given by (4)-(6) allows the implicit use of such kernel functions.

$$\max_{\alpha} \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j y_i y_j \phi(x^i)^T \phi(x^j) \tag{4}$$

subject to,

$$\sum_{i=1}^{M} \alpha_i y_i = 0, \forall i = 1, 2, ..., M \tag{5}$$

$$0 \leq \alpha_i \leq C, \forall i = 1, 2, ..., M. \tag{6}$$

An alternative formulation to the SVM that results in sparse models with comparable generalization was the MCM, that requires solving the following Linear Programming Problem (LPP).

$$\min_{h,w,b,q^+,q^-} h + C \sum_{i=1}^{M} (q^+ + q^-) \tag{7}$$

subject to,

$$w^T x^i + b + q^+ \geq 1, \forall i = 1, 2, ..., M \tag{8}$$

$$h \geq w^T x^i + b + q^-, \forall i = 1, 2, ..., M \tag{9}$$

$$q^+, q^- \geq 0 \tag{10}$$

By introducing an upper bound on the distance that the separating hyperplane can have with the training samples, the MCM effectively leads to minimization of the VC dimension, which depends on the ratio of the radius-to-margin computed on the training samples. Details on minimization of the VC dimension by means of the MCM can be found in [2]. As the MCM solves a LPP, its dual is constructed through the Empirical Feature Space (EFS) route as discussed in [5].

For the SVM, Shah *et al.* [13] showed that their approach had additional advantages than [11], including avoidance of an iterative algorithm with an ascent procedure, while also eliminating the need to have tunable parameters in the kernel optimization strategy. Motivated by this, we apply the kernel optimization approach to the MCM and illustrate potential benefits of the proposed approach, including improved generalization.

III. Proposed Method

The approach presented in Shah *et al.* [13] involves selection of a subset of samples from the training data which are called the empirical cores. Using these, the kernel matrix computed on the traning samples can be split into sub-matrices which involve the "within-class" and "between-class" kernel scatter matrices. The optimal kernel can then be constructed as the Fisher Discriminant, which is subsequently solved using the Generalized Eigenvalue problem to obtain the desired solution.

The following is the proposed approach:

1) Given a binary classification dataset X and labels Y ($\in \mathbb{R}^{M \times n} \times \{\pm 1\}$), make the dataset zero mean and unit variance (normalize the training set and test set normalization should be done using training set mean and variance).
2) Run the following MCM formulation and obtain support vectors A from the training set whose $|\lambda| > \epsilon$, where ϵ is a small threshold.

$$\min_{\lambda,b,q} h + C \sum_{i} q_i \tag{11}$$

subject to,

$$y^i(\lambda^T K_0(x^i, x^j) + b) + q_i \geq 1 \tag{12}$$

$$y^i(\lambda^T K_0(x^i, x^j) + b) + q_i \leq h \tag{13}$$

$$q_i \geq 0 \tag{14}$$

These support vectors are the empirical cores $A = \{a^i, i = 1, 2, ..., P\}$, where $P \leq M$ (number of training samples).
3) Using the representation of K_0 as a combination of samples of two classes, compute W_0 and B_0 as follows

$$K_0 = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \tag{15}$$

$$B_0 = \begin{bmatrix} \frac{1}{M} K_{11} & 0 \\ 0 & \frac{1}{M} K_{22} \end{bmatrix} - \begin{bmatrix} \frac{1}{M} K_{11} & \frac{1}{M} K_{12} \\ \frac{1}{M} K_{21} & \frac{1}{M} K_{22} \end{bmatrix} \tag{16}$$

$$W_0 = \begin{bmatrix} k_{11} & 0 & \dots \\ 0 & k_{22} & \dots \\ \vdots & \vdots & \vdots \\ 0 & 0 & \dots k_{mm} \end{bmatrix} - \begin{bmatrix} \frac{1}{M} K_{11} & 0 \\ 0 & \frac{1}{M} K_{22} \end{bmatrix} \tag{17}$$

4) Solve the generalized eigenvalue problem $B_0 q = \lambda W_0 q$. Set α as the largest eigenvalue.
5) Compute the optimal kernel K as

$$K = QK_0Q, \text{where} \tag{18}$$

$$q = \begin{bmatrix} 1 & k_1(x^1, a^1) & k_1(x^1, a^2) & \dots \\ \vdots & \ddots & & \\ 1 & k_1(x^M, a^1) & & k_1(x^M, a^P) \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_P \end{bmatrix} \tag{19}$$

and $Q = \{q(x^1), q(x^2), ..., q(x^M)\}$
6) Train MCM using the optimal kernel K and obtain predictions on test samples.

IV. Results

The results of the proposed approach on various datasets can be seen in Table I.

TABLE I
ADD CAPTION

	MCM Baseline								**MCM Conformal Kernel**					
Dataset	Train Acc	Std(Train Acc)	Test Acc	Std(Test Acc)	C	Gamma0	Support Vectors	std(Support Vectors)	Train Acc	Std(Train Acc)	Test Acc	Std(Test Acc)	C	Gar
1	78.35264	2.255433718	75.78134284	3.476857598	0.00049	0.001953	30.2	7.496665926	78.60197227	0.636991651	**75.77455**	4.827348	0.000488	0.0
2	90.74074	9.316949906	73.33333333	6.753549085	1.9E-06	0.003906	53	29.30017065	90.05336701	4.884752688	**82.59259**	4.057204	0.000488	0.0
3	76.79491	2.634409224	72.22104707	2.846534464	0.125	0.000244	10	8.396427812	72.77967571	1.99810166	**74.59016**	0.946476	0.125	0.0
5	96.51017	1.732016057	88.87323944	4.117577316	0.00195	0.000244	74	6.403124237	97.08660257	2.89622196	**92.87324**	2.271745	7.63E-06	0.0
8	97.92093	2.250073896	83.39031339	10.9323408	0.00195	0.000244	37.4	10.26157883	86.11924937	2.237527066	**84.90028**	5.196014	0.125	0.0
10	98.85074	0.351106721	92.65121139	2.20780409	0.00195	0.001953	112.4	3.911521443	96.10373413	1.067149387	**93.32768**	1.297374	0.001953	0.0
13	90.54425	1.755932196	83.76141867	4.315510336	0.00049	0.003906	98.6	41.04022417	91.63094729	1.421931011	**85.21072**	2.936379	0.000488	0.0
14	70.72464	2.492506097	68.4057971	2.209847925	0.00012	0.000244	16	24.989998	70.30570354	0.289480235	**69.56522**	0	0.03125	0.0
18	95.7747	1.239639529	94.36578171	1.715970746	0.00195	0.000488	72.2	22.57653649	98.08382418	0.541764761	**97.88697**	0.484113	0.001953	0.0
19	84.6188	14.30089824	62.08708709	8.528872175	0.00781	0.000244	34.4	29.0396281	76.34459491	8.577832994	**68.95646**	5.310556	0.007813	0.0
20	95.54854	3.660331636	76.05623531	8.13219303	7.6E-06	0.001953	61.4	18.95521037	95.14101164	4.877032842	**80.8203**	7.813077	7.63E-06	0.0
21	96.21758	1.214901934	94.03835873	0.944479296	0.00781	0.000244	61.8	6.57267069	97.63666332	0.409316267	**95.41524**	2.430695	3.05E-05	0.0
24	96.02319	2.075923006	86.12532728	2.11922987	0.5	0.000244	38.2	11.58447237	91.53982445	1.224974158	**90.69423**	2.112525	0.125	0.0
25	98.52592	1.813019972	92.74700493	3.357363135	7.6E-06	0.001953	21	18.09696107	97.25908887	3.344197897	**97.7167**	1.607235	1.91E-06	0.0
26	99.48718	0.764467685	90.3908046	6.657544294	0.125	0.000488	27	9.565563235	98.42336935	2.987878931	**96.55172**	3.448276	0.125	0.0
27	100	0	95.71428571	6.38876565	0.125	0.001953	17.4	6.46529195	96.82775936	2.125111296	**97.85714**	3.194383	0.125	0.0
29	72.08315	1.603424841	70.66607722	1.481204704	0.03125	0.000244	5.6	12.52198067	70.1594533	0	**71.55172**	0	0.125	0.0
30	74.24367	7.979206197	61.52631579	6.473898156	2	0.000244	9.6	3.78153408	66.26888349	7.552300862	**73.78947**	3.816243	0.000122	0.0
4	88.55042	1.302218368	**83.8561828**	5.07645739	7.6E-06	0.001953	24.6	6.580273551	81.17015681	4.452228344	82.64785	5.544723	7.63E-06	0.0
6	78.87756	1.23566052	**78.60402685**	2.282420868	2	0.001953	11.8	3.033150178	74.19807753	3.203997438	74.73289	2.020515	0.007813	0.0
9	99.52941	1.052267284	**71.43722944**	15.18426886	2	0.003906	38.6	5.983310121	96.81871345	4.406113564	70.48918	11.41142	2	0.0
11	90.91667	6.423967708	**74.66666667**	8.771798245	3.1E-05	0.000977	67	28	84.92218543	4.619592052	73.66667	7.397447	3.05E-05	0.0
17	86.17786	5.441104494	**76.28397496**	7.03292692	0.00012	0.000488	102.6	44.11689019	85.63849537	5.849668521	75.52945	4.00255	0.000488	0.0
23	100	0	**100**	0	2	0.000244	7.4	0.894427191	95.67744036	1.109552176	95.65956	1.370361	2	0.0
28	92.24582	2.249678967	**84.82205514**	5.524414451	2	0.003906	14.4	4.393176527	85.0990235	7.656356705	83.82206	9.30661	2	0.0

V. Conclusion & Future Work

References

[1] Vladimir Vapnik, Esther Levin, and Yann Le Cun. Measuring the vc-dimension of a learning machine. *Neural computation*, 6(5):851–876, 1994.

[2] Jayadeva. Learning a hyperplane classifier by minimizing an exact bound on the VC dimension. *Neurocomputing*, 149:683–689, 2015.

[3] Jayadeva, Suresh Chandra, Sanjit S Batra, and Siddarth Sabharwal. Learning a hyperplane regressor through a tight bound on the VC dimension. *Neurocomputing*, 171:1610–1616, 2016.

[4] Jayadeva, Sanjit Singh Batra, and Siddarth Sabharwal. Learning a fuzzy hyperplane fat margin classifier with minimum vc dimension. *arXiv preprint arXiv:1501.02432*, 2015.

[5] Jayadeva, Mayank Sharma, Sumit Soman, and Himanshu Pant. Ultra-sparse classifiers through minimizing the VC dimension in the empirical feature space. *Neural Processing Letters*, pages 1–33, 2018.

[6] Jayadeva, Sumit Soman, and Amit Bhaya. The MC-ELM: Learning an ELM-like network with minimum VC dimension. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–7. IEEE, 2015.

[7] Mayank Sharma, Sumit Soman, Himanshu Pant, et al. Large-scale minimal complexity machines using explicit feature maps. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017.

[8] Jayadeva, Sumit Soman, Himanshu Pant, and Mayank Sharma. Qmcm: Minimizing vapniks bound on the vc dimension. *Neurocomputing*, 2020.

[9] Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz S Kandola. On kernel-target alignment. In *Advances in neural information processing systems*, pages 367–373, 2002.

[10] Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine learning research*, 5(Jan):27–72, 2004.

[11] Huilin Xiong, MNS Swamy, and M Omair Ahmad. Optimizing the kernel in the empirical feature space. *IEEE transactions on neural networks*, 16(2):460–474, 2005.

[12] Shun-ichi Amari and Si Wu. Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, 12(6):783–789, 1999.

[13] Sameena Shah, Suresh Chandra, et al. Kernel optimization using a generalized eigenvalue approach. In *International Conference on Pattern Recognition and Machine Intelligence*, pages 32–37. Springer, 2009.