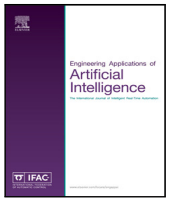
Engineering Applications of Artificial Intelligence 106 (2021) 104493

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Kernel optimization using conformal maps for the minimal complexity machine

Skyler Badge [a], Sumit Soman [a,1], Suresh Chandra [b], Jayadeva [a,*]

[a] Department of Electrical Engineering, Indian Institute of Technology Delhi, India
[b] Department of Mathematics, Indian Institute of Technology Delhi, India

ARTICLE INFO

Keywords:
Minimal complexity machine
Kernel optimization
Conformal kernel
Generalized eigenvalue problem
Support vector machines

ABSTRACT

The Minimal Complexity Machine (MCM) is a kernel-based learning model that can learn very sparse models that yield comparable or better performance than Support Vector Machines (SVMs). However, kernel optimization for the MCM has not yet been explored. It has been shown in prior work that a data dependent kernel helps improve generalization. We show results on data dependent optimized kernels for the MCM and a large-scale MCM variant. Results on benchmark datasets demonstrate both model sparsity and improved generalization.

1. Introduction

Deep learning architectures are methods of choice, particularly for image or high dimensional data. However, deep architectures consume large memory and computational resources. On the other hand, kernel methods are often more useful for Internet of Things (IoT) devices, smartphone, and embedded intelligence applications, where computation, memory, and power resources are constrained. Sparse models that generalize well are suitable for such applications, as sparsity results in lower computational cost and inference time while obtaining predictions, which are required for low-resource platforms.

Neural network training not only minimizes the empirical error, but also optimizes the feature map. This can potentially yield superior generalization. However, kernel based methods invariably employ a fixed kernel function. Thus, the performance (or generalization) of a kernel-based method such as the Support Vector Machine (SVM), is dependent on the kernel function chosen.

In this paper, we focus on kernel optimization for the recently proposed Minimal Complexity Machine (MCM) (Jayadeva, 2015). The MCM tries to minimize a bound on the Vapnik–Chervonenkis ((VC) dimension, yielding models that are usually 3x to 10x smaller than SVMs, and sometimes over a 100x smaller. We employ the data-dependent kernel optimization technique of Shah et al. (2009), which is based on conformal maps, to learn optimal kernels for the MCM. We show this leads to improved generalization and sparsity over baseline models. Further, a scalable MCM variant is also evaluated, to show scaleability to large datasets.

2. Related work

A recently developed class of kernel methods of interest is the Minimal Complexity Machine (MCM) (Jayadeva, 2015), that learns models by minimizing a combination of empirical error and an upper bound on the VC dimension (Vapnik et al., 1994). MCMs provide generalization comparable to SVMs while learning very sparse models. MCM variants have been developed for regression (Jayadeva et al., 2016), fuzzy classification (Jayadeva et al., 2015a), and least-squares methods (Jayadeva et al., 2018). Quadratic (Jayadeva et al., 2020) and large-scale (Sharma et al., 2017) versions, as well as hybrid variants (Jayadeva et al., 2015b; Pant et al., 2020), have been shown to yield sparse models exhibiting improved generalization. This paper presents the use of kernel optimization with the MCM.

Kernel methods typically employ a fixed kernel function (Smola et al., 1998). In this context, kernel methods can benefit from data dependent kernels, such as through kernel-target alignment (Cristianini et al., 2002) and semi-definite programming for learning the kernel (Lanckriet et al., 2004), among others. This is possibly the first work on data dependent kernel optimization for MCMs. The motivation is to further improve generalization while learning sparse models. Authors in Xiong et al. (2005) have presented kernel optimization approaches for SVMs. Our approach is essentially that of Shah et al. (2009), which solves a generalized eigenvalue problem that minimizes a type of Fischer discriminant. It follows the work of Amari and Wu (1999) and Xiong et al. (2005). Results on selected benchmark datasets show that using optimized kernels with the MCM yield superior results and sparse models.

* Corresponding author.
E-mail address: jayadeva@ee.iitd.ac.in (Jayadeva).
[1] Sumit Soman was involved in this work as a Ph.D. student at IIT Delhi.

https://doi.org/10.1016/j.engappai.2021.104493
Received 25 November 2020; Received in revised form 26 August 2021; Accepted 20 September 2021
Available online 21 October 2021
0952-1976/

A challenge faced when implementing kernel optimization methods is that they are not scalable to large datasets. The kernel matrix size is of the order of the number of samples in the dataset, and optimization usually involves large matrix operations that involve a significant computational overhead. A scalable variant of the MCM, based on the stochastic subgradient approach was proposed in Sharma et al. (2017). The SGD-MCM with kernel optimization is used, and it illustrates the accuracy improvements on large datasets as well.

The rest of the paper is organized as follows. Section 3 introduces the SVM and MCM formulations, as well as relevant kernel optimization approaches. In Section 4 the conformal kernel based optimization (Shah et al., 2009), which is used for the MCM, is presented. This is followed by application to the SGD-MCM, which is presented in Section 5. Results on various benchmark datasets are presented in Section 6. Section 7 contains concluding remarks.

3. Background

Given a set of M training data samples in n dimensions $X \in \mathbb{R}^{M\times n}$ with labels $Y = y_i \in \{-1, +1\}, \forall i = 1, 2, \ldots, M$, a binary classification problem involves predicting the label $\hat{y}$ for a test sample $\hat{x} \in \mathbb{R}^n$. 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).

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

subject to,

$$y_i(w^T x_i + b) + q_i \geq 1, \forall i = 1, 2, \ldots, M \tag{2}$$

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

Here, the separating hyperplane is $w^T x + b = 0$; q_i is a non-negative slack variable that allows for *"softness"* of the constraint with respect to sample x_i. The hyperparameter C controls the trade-off between the L_2 norm of the weight vector and the extent of mis-classification. The class to which a test point $\hat{x} \in \mathbb{R}^n$ belongs is given by $sgn(w^T\hat{x} + b)$, where $sgn(\cdot)$ is the signum function.

The Minimal Complexity Machine (Jayadeva, 2015) is a novel kernel machine formulation, that attempts to learn a model with a small VC dimension. It does this by solving the following Linear Programming Problem (LPP).

$$\min_{h,w,b,q_i} h + C\sum_{i=1}^{M} q_i \tag{4}$$

subject to,

$$y_i(w^T x_i + b) + q_i \geq 1, \forall i = 1, 2, \ldots, M \tag{5}$$

$$h \geq y_i(w^T x_i + b) + q_i, \forall i = 1, 2, \ldots, M \tag{6}$$

$$q_i \geq 0,\ i = 1, 2, \ldots, M \tag{7}$$

The MCM objective function is a tradeoff between an upper bound on the VC dimension (h^2) and the misclassification error. Variants of the MCM, including formulations that are constructed through the Empirical Feature Space (EFS) route may be found in Jayadeva et al. (2018, 2016, 2015b), Sharma et al. (2017) and Jayadeva et al. (2020).

For the SVM, Shah et al. (2009) proposed a technique for computing an optimal data dependent kernel, that is based on Xiong et al. (2005); however, the approach of Shah et al. (2009) is provably optimal, avoids an iterative algorithm with gradient ascent, and does not need to tune any parameters. In this work, the kernel optimization approach of Shah et al. (2009) is used with the MCM, yielding sparse models with improved generalization.

4. MCM kernel optimization

The process for computing an optimal data dependent kernel for the MCM may be summarized as follows.

1. Transform the dataset so that its mean is zero and its variance is 1.
2. Run the following MCM formulation and obtain support vectors A from the training set whose $|\lambda| > \epsilon$, where ϵ is a small threshold.

$$\min_{\lambda,b,q} h + C\sum_{i=1}^{M} q_i \tag{8}$$

subject to,

$$y_i(\sum_{j=1}^{M} \lambda_j K(x_i, x_j) + b) + q_i \geq 1 \tag{9}$$

$$y_i(\sum_{j=1}^{M} \lambda_j K(x_i, x_j) + b) + q_i \leq h \tag{10}$$

$$q_i \geq 0 \tag{11}$$

Here, K is the kernel matrix. These support vectors form the set of empirical cores $A = \{a_i, i = 1, 2, \ldots, P\}$, where $P \leq M$ (number of training samples). MCM models use very few support vectors in comparison to SVMs. In the approach of Shah et al. (2009), a subset of support vectors (SVs) are used, since the number of SVs in the case of SVMs can be large. In the case of the MCM, the number is much smaller, allowing all to be used. The reduction is usually between 3x to 10x, but can be more than 100x in some cases. The region of maximum confusion lies in the neighborhood of support vectors. Kernel optimization distorts the feature map in these regions, so that the discrimination between samples of the two classes improves.
3. The training samples are arranged by writing the samples of the first class, followed by those of the second class. Compute K_0, W_0 and B_0 as follows:

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

$$= \left[\begin{array}{ccc|ccc} K(x_1^A, x_1^A) & \cdots & K(x_1^A, x_{M_1}^A) & K(x_1^A, x_1^B) & \cdots & K(x_1^A, x_{M_2}^B) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ K(x_{M_1}^A, x_1^A) & \cdots & K(x_{M_1}^A, x_{M_1}^A) & K(x_{M_1}^A, x_1^B) & \cdots & K(x_{M_1}^A, x_{M_2}^B) \\ \hline K(x_1^B, x_1^A) & \cdots & K(x_1^B, x_{M_1}^A) & K(x_1^B, x_1^B) & \cdots & K(x_1^B, x_{M_2}^B) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ K(x_{M_2}^B, x_1^A) & \cdots & K(x_{M_2}^B, x_{M_1}^A) & K(x_{M_2}^B, x_1^B) & \cdots & K(x_{M_2}^B, x_{M_2}^B) \end{array}\right] \tag{12}$$

$$B_0 = \begin{bmatrix} \frac{1}{M}K_{11} & 0 \\ 0 & \frac{1}{M}K_{22} \end{bmatrix} - \begin{bmatrix} \frac{1}{M}K_{11} & \frac{1}{M}K_{12} \\ \frac{1}{M}K_{21} & \frac{1}{M}K_{22} \end{bmatrix} \tag{13}$$

$$W_0 = \begin{bmatrix} k_{11} & 0 & \ldots \\ 0 & k_{22} & \ldots \\ \vdots & \vdots & \vdots \\ 0 & 0 & \ldots k_{MM} \end{bmatrix} - \begin{bmatrix} \frac{1}{M}K_{11} & 0 \\ 0 & \frac{1}{M}K_{22} \end{bmatrix} \tag{14}$$

where, B_0 is the between class kernel scatter matrix, W_0 is the within class kernel scatter matrix, and K_{11}, K_{12}, K_{21} and K_{22} are sub-matrices of K as shown in (12) and M_1 and M_2 are the number of samples in class A and B respectively.
4. Solve the generalized eigenvalue problem

$$B_0 z = \lambda W_0 z \tag{15}$$

Set α as the eigenvector corresponding to the largest eigenvalue.
5. Compute the optimal kernel matrix K with its elements given as

$$K_{ij} = K_{0ij}.r_i.r_j, \text{where} \tag{16}$$

$$R = \begin{bmatrix} 1 & k_1(x_1, a_1) & \dots & k_1(x_1, a_P) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & k_1(x_M, a_1) & \dots & k_1(x_M, a_P) \end{bmatrix}_{M\times(P+1)} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_P \end{bmatrix}_{(P+1)\times 1} \quad (17)$$

Here, r_i and r_j are the ith and jth elements of R respectively.

6. Train MCM using the optimal kernel K and obtain predictions on test samples.

4.1. Parameter tuning

For the MCM, after pre-processing the data, Algorithm 1 is used to tune the kernel parameters. γ is the RBF kernel width parameter. $CParams$ and $\gamma Params$ are the range of values over which C and γ are tuned, respectively.

Result: Tuned C and γ
Initialize $max\_accuracy = 0$;
for $C \in CParams$ **do**
  for $\gamma \in \gamma Params$ **do**
    Compute $[\lambda, b, h] = \text{MCM}(X, Y, C, \gamma)$;
    Compute $[predicted, accuracy] = \text{Predict}(X, Y, \gamma, \lambda, b)$;
    if $accuracy > max\_accuracy$ **then**
      $max\_accuracy = accuracy$;
      Save C and γ
    end
  end
end
return C and γ

Algorithm 1: Tuning Parameters for MCM using Grid Search

Post tuning, the following are executed:

1. $[\lambda, b, h] = MCM(X, Y, C, \gamma)$; (solves (8))
2. $[predicted, accuracy] = Predict(X, Y, \gamma, \lambda, b)$; $predicted$ is the list of predicted labels returned by the $Predict$ function.

$$predicted_i = \begin{cases} 1, & \frac{\sum_{j=1}^{M} \lambda_j K_0(x_i, x_j) + b}{\|\lambda\|} \geq 0 \\ 0, & otherwise \end{cases}, \forall i = 1, 2, \dots, M \quad (18)$$

Here, x_i is the ith test sample, while x_j denotes the jth training sample.

The vector λ is obtained by solving (8), using the tuned parameters, and is then used to generate predictions on test samples. In order to account for finite numerical precision, if $|\lambda_j| \leq \epsilon$, treat $|\lambda_j|$ as zero. The tolerance ϵ is typically chosen to be 10^{-6}.

4.2. Parameter tuning for conformal kernel

In this work, the tuning process is as shown in Algorithm 2. γ_c is the RBF kernel width parameter for the conformal kernel and $\gamma_c Params$ is the range of values over which it is tuned. The support vectors are determined, and constitute the empirical cores. The matrix K_0 as given by (12) is constructed. Using this, B_0 and W_0 are computed. On solving the generalized eigenvalue problem (15), the vector α is obtained, which is used to calculate the vector R (17). Finally the conformal kernel is calculated using (16). The optimized kernel is then used to learn a MCM model; this model is used on test samples.

The following are the function calls used in the pipeline:

1. $[\lambda, b, h] = MCM\_Conformal(X, Y, C, \gamma, R)$ (Here, R is used to calculate (16). Then (8) is solved.
2. $[predicted, accuracy] = Predict\_Conformal(X, Y, K_t, \lambda, b)$

Result: Tuned C, γ and γ_c and Accuracy
Initialize $max\_accuracy = 0$;
Initialize $max\_accuracy\_conformal = 0$;
for $C \in CParams$ **do**
  for $\gamma \in \gamma Params$ **do**
    Compute $[\lambda, b, h] = \text{MCM}(X, Y, C, \gamma)$;
    Compute $[predicted, accuracy] = \text{Predict}(X, Y, \gamma, \lambda, b)$;
    if $accuracy > max\_accuracy$ **then**
      $max\_accuracy = accuracy$;
      Save C and γ for MCM
    end
    for $\gamma_c \in \gamma_c Params$ **do**
      Compute $[\lambda, b, h] = \text{MCM\_Conformal}(X, Y, C, \gamma, R)$;
      Compute $[predicted\_conformal, accuracy\_conformal] =$
      $\text{Predict\_Conformal}(X, Y, \gamma, \gamma_c, C, \lambda, b)$;
      if $accuracy\_conformal > max\_accuracy\_conformal$ **then**
        $max\_accuracy\_conformal = accuracy\_conformal$;
        Save C, γ and γ_c for MCM Conformal
      end
    end
  end
end
return C,γ and γ_c

Algorithm 2: Tuning Parameters for MCM Conformal Kernel using Grid Search

5. Scaling MCM kernel optimization for large datasets

For large datasets, solving the optimization problem given by (8) becomes difficult as matrix calculations require the entire data to be stored in memory. To tackle this, the approach of the stochastic sub-gradient descent MCM (Sharma et al., 2017) is followed to incorporate kernel optimization. In this version of the MCM, the data may be processed in batches, allowing large datasets to be processed.

Stochastic sub-gradient descent MCM is first run to tune the hyper-parameters C and γ by using a grid search. After choosing appropriate values of C and γ, we determine the corresponding support vectors. These support vectors are set as empirical cores. Some cores from the larger class are discarded so that both classes have equal number of cores. Steps 3 to 5 are now used to compute the optimal conformal kernel. This step also involves tuning of the hyper-parameter γ_c of the conformal kernel. Finally, stochastic sub-gradient descent is run again using the conformal kernel to learn a new model. This is summarized in Fig. 1.

6. Results

Results with data-dependent optimized kernels on various benchmark datasets are presented in the following sub-section. All simulations have been carried out using MATLAB R2018b running on a PC with an Intel Core i7-7560U CPU and 16 GB RAM, running a 64 bit Windows 10 Home OS. First, a comprehensive evaluation on various datasets in terms of classification accuracy is presented to illustrate the performance. Following that, a visual representation of the conformal kernels generated using our approach are compared with the non-optimized kernel.

6.1. Quantitative results on datasets

Kernel optimization results have been evaluated on a set of 27 datasets from the UCI machine learning repository (Asuncion and Newman, 2007). A list of these datasets along with the number of samples and features are shown in Table 1. The size of these datasets ranges from 100 to 1000 samples.

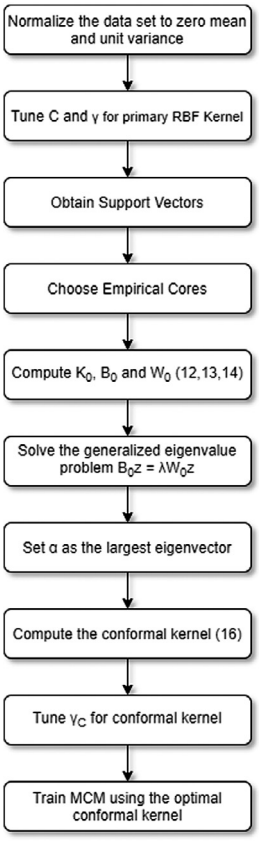


Fig. 1. Training MCM and large scale MCM with a conformal kernel.

Table 1
Dataset description.

Sr.No.	DataSet Name	# Samples	# Features
1	Ionosphere	351	54
2	Promotors	106	57
3	Mammographic Masses	961	5
4	Bands	539	39
5	SONAR	208	60
6	Wholesale Customers	440	7
7	Seed	140	7
8	Blogger	100	6
9	Indian Liver Patient Dataset	583	10
10	Blood Transfusion	748	4
11	Tic Tac Toe	958	9
12	Pima Indians	768	8
13	Heart Statlog	270	13
14	Habermann's Survival	306	3
15	Hepatitis	155	19
16	ECG	132	12
17	Congressional Voting Records	435	16
18	Horsecolic	300	27
19	Australian	690	14
20	Australian Credit Approval	690	15
21	German Credit	1000	20
22	Breast Cancer (W)	568	31
23	Planning Relax	182	12
24	Housevotes	436	16
25	Balance	576	4
26	Ecoli	220	6
27	Glass	146	10

Each dataset has been normalized to have zero mean and unit variance. Five fold cross-validation (Refaeilzadeh et al., 2009) results have been provided. Hyperparameters are tuned by using a grid search (Han et al., 2012) within each fold. The hyperparameter C is tuned in the range 2^{-19} to 2^{2} with a multiplicative step size of 2^{2}; γ is varied in the range $\{2^{-10}, 2^{-11}, 2^{-12}, 2^{-8}, 2^{-9}\}$, while the value of γ_c is varied based on a multiplicative factor on the value of $\gamma \times \{2, 2.2, 2.4, 2.6, 2.8, 3, 4, 10, 15, 20, 25, 30, 40, 50, 60, 70, 80, 100, 500, 800, 1000, 5000\}$. For large datasets, γ_c is additionally varied from 10^{-11} to 10^{-1} with a multiplicative step size of 10.

Table 2
Test accuracy on benchmark datasets.

Sr.No.	DataSet Name	MCM	MCM Conformal	SVM
1	Ionosphere	88.87 ± 4.12	**93.72 ± 2.61**	89.74 ± 2.37
2	Promotors	71.44 ± 15.18	**82.15 ± 3.40**	82.06 ± 7.69
3	Mammographic Masses	75.33 ± 12.35	**83.77 ± 3.92**	82.42 ± 2.88
4	Bands	76.28 ± 7.03	**80.54 ± 2.99**	74.23 ± 5.25
5	SONAR	76.06 ± 8.13	**86.13 ± 7.12**	79.31 ± 6.46
6	Wholesale Customers	86.13 ± 2.12	**90.69 ± 1.19**	90.23 ± 2.05
7	Seed	95.71 ± 6.39	**99.29 ± 1.60**	97.14 ± 3.91
8	Blogger	84.82 ± 5.52	**84.82 ± 5.52**	71.04 ± 3.49
9	Indian Liver Patient Dataset	70.67 ± 1.48	**71.55 ± 0.00**	73.07 ± 1.37
10	Blood Transfusion	**78.60 ± 2.28**	75.80 ± 1.22	76.87 ± 1.11
11	Tic Tac Toe	**98.85 ± 1.13**	98.75 ± 0.28	65.34 ± 0.23
12	Pima Indians	75.78 ± 3.48	**76.43 ± 3.88**	77.86 ± 3.05
13	Heart Statlog	73.33 ± 6.75	**81.85 ± 3.80**	87.04 ± 1.31
14	Habermann's Survival	72.22 ± 2.85	**74.18 ± 0.82**	75.81 ± 2.26
15	Hepatitis	75.48 ± 10.04	**81.36 ± 2.96**	83.94 ± 5.79
16	ECG	83.39 ± 10.93	**87.15 ± 5.04**	87.95 ± 4.68
17	Congressional Voting Records	93.57 ± 2.99	**95.18 ± 0.93**	96.33 ± 0.94
18	Horsecolic	74.67 ± 8.77	**80.00 ± 4.25**	82.67 ± 3.84
19	Australian	83.76 ± 4.32	**86.37 ± 2.28**	87.68 ± 3.48
20	Australian Credit Approval	68.41 ± 2.21	**68.48 ± 1.54**	69.57 ± 0.00
21	German Credit	**74.10 ± 1.43**	71.60 ± 0.89	75.10 ± 2.72
22	Breast Cancer (W)	94.37 ± 1.72	**97.19 ± 0.96**	97.71 ± 1.33
23	Planning Relax	62.09 ± 8.53	**71.02 ± 2.92**	74.19 ± 3.58
24	Housevotes	94.04 ± 0.94	**95.87 ± 0.60**	96.79 ± 1.25
25	Balance	**100.00 ± 0.00**	99.83 ± 0.39	96.00 ± 1.01
26	Ecoli	96.36 ± 2.03	**98.18 ± 1.02**	98.64 ± 2.03
27	Glass	90.39 ± 6.66	**95.20 ± 3.94**	98.64 ± 1.86

Hyper-parameters need to be finely tuned; this seems to be a characteristic of sparse models. Table 2 reports test accuracies in terms of the mean and standard deviation over five folds. Baseline test accuracies using SVM with a RBF Kernel have been included. SVM results were generated by using LIBSVM (Chang and Lin, 2011). The results indicate that kernel optimization provides consistent and demonstrable improvements.

The number of support vectors in the case of the MCM and the SVM are reported. The values of λ have been obtained using Eq. (8). The number of support vectors is computed by counting the total number of samples for which $|\lambda| > \epsilon$, where ϵ is chosen as 10^{-6}. The support vectors are recorded along with the test accuracies, as they are generated using 5-fold cross-validation. The average value over the 5-folds, along with their respective standard deviations, are presented in Table 3. In almost all cases, it is observed that the MCM uses significantly fewer support vectors than the SVM.

Results on large datasets from the LIBSVM repository were also evaluated. Chosen datasets are listed in Table 4, indicating the number of samples and features. The datasets were normalized to lie between zero mean and unit variance. A train–test split was done randomly (random state = 42 in sklearn.model_selection.train_test_split()), with 20% of the data in the testing set. Hyperparameter tuning was performed using grid search. For the conventional large scale MCM, the hyperparameters tuned are C and γ (RBF kernel width parameter). In the case of the kernel optimized large scale MCM, the additional hyperparameter γ_c is tuned over the range 10^{-8} to 10^{2}. Furthermore, 20% of the training data was used for validation while tuning the hyperparameters. Results in Table 4 indicate that kernel optimization yields improved or comparable generalization in 12 of 14 datasets.

The Wilcoxon signed rank test (Rey and Neuhäuser, 2011) was used to determine if the improvements are statistically significant. The test yields a p-value of 3.6869×10^{-5} which is much less than 0.05, indicating that the improvements are statistically significant.

Table 3
Number of support vectors on benchmark datasets.

Sr.No.	DataSet Name	MCM conformal	SVM
1	Ionosphere	**74.00 ± 6.40**	89.62 ± 2.14
2	Promotors	**38.60 ± 5.98**	79.89 ± 8.70
3	Mammographic Masses	173.00 ± 333.29	**120.51 ± 93.35**
4	Bands	**102.60 ± 44.12**	112.86 ± 94.74
5	SONAR	**61.40 ± 18.96**	87.76 ± 21.48
6	Wholesale Customers	**38.20 ± 11.58**	85.36 ± 12.07
7	Seed	**17.40 ± 6.47**	82.95 ± 34.94
8	Blogger	**14.40 ± 4.39**	67.37 ± 9.52
9	Indian Liver Patient Dataset	**5.60 ± 12.52**	91.39 ± 44.89
10	Blood Transfusion	**11.80 ± 3.03**	112.06 ± 86.20
11	Tic Tac Toe	**71.60 ± 36.17**	144.29 ± 193.37
12	Pima Indians	**30.20 ± 7.50**	129.05 ± 125.42
13	Heart Statlog	**53.00 ± 29.30**	93.20 ± 15.14
14	Habermann's Survival	**10.00 ± 8.40**	82.01 ± 15.32
15	Hepatitis	**19.40 ± 15.45**	76.28 ± 19.46
16	ECG	**37.40 ± 10.26**	78.62 ± 23.22
17	Congressional Voting Records	**81.40 ± 25.86**	89.94 ± 15.67
18	Horsecolic	**67.00 ± 28.00**	86.72 ± 10.51
19	Australian	**98.60 ± 41.04**	134.07 ± 113.67
20	Australian Credit Approval	**16.00 ± 24.99**	125.64 ± 137.35
21	German Credit	**122.40 ± 46.44**	143.42 ± 167.36
22	Breast Cancer (W)	**72.20 ± 22.58**	98.93 ± 3.20
23	Planning Relax	**34.40 ± 29.04**	76.49 ± 6.48
24	Housevotes	**61.80 ± 6.57**	89.50 ± 17.91
25	Balance	**7.40 ± 0.89**	90.34 ± 13.91
26	Ecoli	**9.20 ± 6.83**	85.20 ± 32.97
27	Glass	**27.00 ± 9.57**	88.70 ± 24.41

Table 4
Data description and test accuracy of large scale MCM using conformal kernel.

Sr.No.	DataSet Name	# Samples	# Features	LS MCM	LS Conf MCM
1	w4a	49 749	300	97.39	**97.40**
2	diabetes	768	8	**76.62**	75.97
3	breast-cancer	683	10	96.35	**97.08**
4	australian	690	14	81.16	**83.33**
5	german-numer	1 000	24	71.50	**71.50**
6	fourclass	862	2	98.84	**98.84**
7	a3a	5 096	123	84.16	**84.18**
8	a4a	7 648	123	84.44	**84.46**
9	a8a	36 312	123	84.83	**84.86**
10	EEG Eye State	14 980	15	**78.77**	78.70
11	HTRU2	17 898	9	98.04	**98.04**
12	MAGIC Gamma Telescope	19 020	11	87.40	**87.51**
13	phishing	11 055	68	94.93	**94.93**
14	Default of Credit Card Clients	30 000	24	80.46	**80.52**

6.2. Visualization results

In order to help visualize how kernel optimization helps, we use the 'Glass' UCI dataset as a representative dataset. Figs. 2a and 2b show the projection of the training and test samples, respectively, along the first two principal components (Jolliffe and Cadima, 2016), without any kernel optimization. The data is then transformed using a RBF kernel with a tuned kernel parameter (γ). Projections on the two most significant kernel principal components are shown in Figs. 2c and 2d for train and test data respectively. In the kernel optimization pipeline, the kernel parameter γ_c for the conformal kernel is also tuned, and the conformal kernel matrix is computed by using (16). Figs. 2e and 2f represent the projections of the train and test data along the first two kernel principal components of the optimized kernel. The hyperparameters for which the graphs have been plotted are $C = 2$, $\gamma = 0.002$ and $\gamma_c = 0.0078$. Refer Appendix for the procedure used to generate the graphs.

Kernel optimization aims to enlarge the separation boundary by expanding the region around the support vectors obtained by an initial MCM run. This can be observed in Fig. 2e depicting the training data, where the margin between the two classes has increased as compared to Fig. 2c, with the increased separation clearly visible. A similar trend is observed for graphs with the testing data. The figures provide visual evidence that kernel optimization improves separability.

7. Conclusion & future work

This paper presented the use of kernel optimization for the Minimal Complexity Machine. The optimal kernel is found using the conformal kernel based approach of Shah et al. (2009). Results on benchmark datasets show significant improvement in test set accuracies. The advantage of using this with the MCM is that the approach retains model sparsity. Since the number of support vectors is much smaller in the case of the MCM, all support vectors can be used as empirical cores; in the case of Shah et al. (2009), a subset is used as the set of empirical cores. Future work involves extending this approach for other learning settings such as regression. Recent work on new variants of the MCM (Jayadeva et al., 2020) suggests additional directions in which optimized kernels may prove beneficial.

CRediT authorship contribution statement

Skyler Badge: Methodology, Software, Writing – review & editing, Validation. **Sumit Soman:** Methodology, Investigation, Writing – original draft. **Suresh Chandra:** Conceptualization. **Jayadeva:** Conceptualization, Writing – review & editing, Resources, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix. Generating plots

A.1. Plots for input data

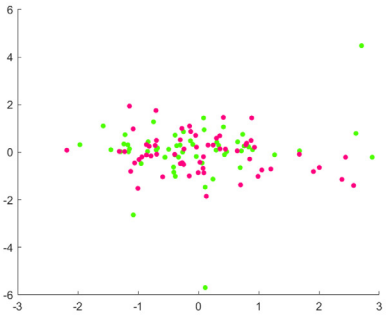
Given a set of centered input data $X \in \mathbb{R}^{M \times n}$, with M samples and n features, PCA (Schölkopf et al., 1998) is performed to obtain the two most significant dimensions of the data. The samples are projected along these dimensions and plotted. The steps are:

1. Compute covariance matrix $C = X^T X$.
2. Compute eigenvalues and eigenvectors of C:
   solve the eigenvalue problem $C\alpha = \lambda\alpha$
   Here, the columns of matrix α are eigenvectors and λ is a diagonal matrix of the corresponding eigenvalues.
3. Select 2 eigenvectors $\alpha^1, \alpha^2 \in \mathbb{R}^n$ corresponding to highest two eigenvalues.
4. Normalize α^i using $\alpha^i \leftarrow \frac{1}{\sqrt{\lambda_i}}\alpha^i$ for $i = 1, 2$.
   λ_i is the eigenvalue corresponding to eigenvector α^i.
5. Compute projection of each sample $x^i \in \mathbb{R}^n$ as $(x^i \cdot \alpha^1, x^i \cdot \alpha^2)$.
6. Plot the projections for M samples.

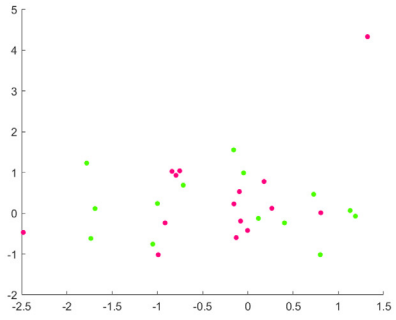
A.2. Plots for transformed data

Given a Kernel matrix K, Kernel PCA (Schölkopf et al., 1998) is performed to obtain the two most significant dimensions of the data. The samples are projected along these dimensions and plotted. The steps are:

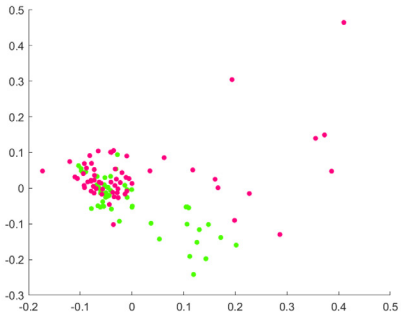
1. K is centered as given in Appendix B of the work by Schölkopf et al. (1998).
2. Compute eigenvalues and eigenvectors of K:
   solve the eigenvalue problem $K\alpha = M\lambda\alpha$
   Here, the columns of matrix α are eigenvectors and λ is a diagonal matrix of the corresponding eigenvalues.



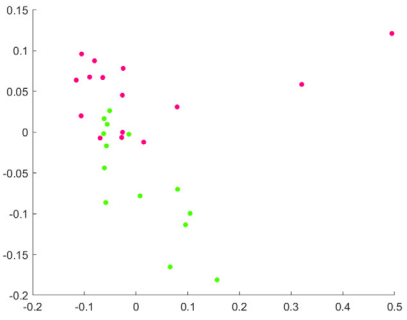
(a) Train data embedding without any kernel optimization



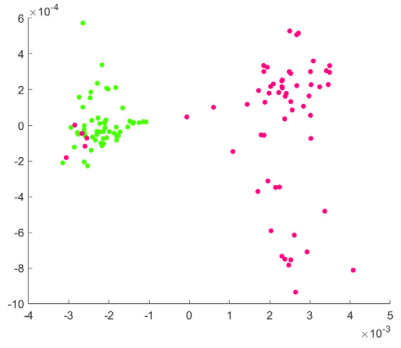
(b) Test data embedding without any kernel optimization



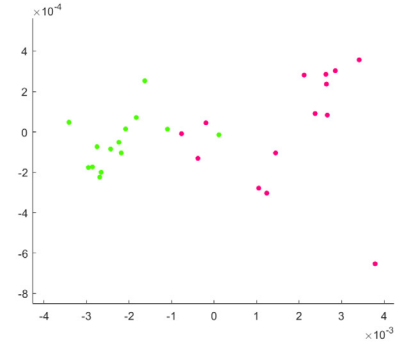
(c) Train data embedding with primary RBF kernel



(d) Test data embedding with primary RBF kernel



(e) Training data embedding using kernel optimization



(f) Test data embedding after kernel optimization

Fig. 2. Plots showing data embedding on train and test sets.

3. Select 2 eigenvectors $\alpha^i, \alpha^2 \in \mathbb{R}^M$ corresponding to highest two eigenvalues.
4. Normalize α^i using $\alpha^i \leftarrow \frac{1}{\sqrt{(\lambda_i)}}\alpha^i$ for $i = 1, 2$.
5. The projection of a sample x, on the kth eigenvector, is given by $\sum_{j=1}^{M} \alpha_j^k K(x, x^j)$. Compute the projections for $k = 1, 2$.
6. Plot the projections for M samples.

References

Amari, S.-i., Wu, S., 1999. Improving support vector machine classifiers by modifying kernel functions. Neural Netw. 12 (6), 783–789. http://dx.doi.org/10.1016/S0893-6080(99)00032-5.

Asuncion, A., Newman, D., 2007. UCI machine learning repository.

Chang, C.-C., Lin, C.-J., 2011. LIBSVM: A library for support vector machines. ACM Trans. Intell. Syst. Technol. (TIST) 2, 27:1–27:27, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., Kandola, J.S., 2002. On kernel-target alignment. In: Advances in Neural Information Processing Systems. pp. 367–373. http://dx.doi.org/10.1007/3-540-33486-6_8.

Han, S., Qubo, C., Meng, H., 2012. Parameter selection in SVM with RBF kernel function. In: World Automation Congress 2012. pp. 1–4.

Jayadeva, 2015. Learning a hyperplane classifier by minimizing an exact bound on the VC dimension. Neurocomputing 149, 683–689. http://dx.doi.org/10.1016/j.neucom.2014.07.062.

Jayadeva, Batra, S.S., Sabharwal, S., 2015a. Learning a fuzzy hyperplane fat margin classifier with minimum VC dimension. arXiv preprint arXiv:1501.02432.

Jayadeva, Chandra, S., Batra, S.S., Sabharwal, S., 2016. Learning a hyperplane regressor through a tight bound on the VC dimension. Neurocomputing 171, 1610–1616. http://dx.doi.org/10.1016/j.neucom.2015.06.065.

Jayadeva, Sharma, M., Soman, S., Pant, H., 2018. Ultra-sparse classifiers through minimizing the VC dimension in the empirical feature space. Neural Process. Lett. 1–33. http://dx.doi.org/10.1007/s11063-018-9793-9.

Jayadeva, Soman, S., Bhaya, A., 2015b. The MC-ELM: Learning an ELM-like network with minimum VC dimension. In: Neural Networks (IJCNN), 2015 International Joint Conference on. IEEE, pp. 1–7. http://dx.doi.org/10.1109/IJCNN.2015.7280663.

Jayadeva, Soman, S., Pant, H., Sharma, M., 2020. QMCM: Minimizing Vapnik's bound on the [VC] dimension. Neurocomputing http://dx.doi.org/10.1007/s11063-018-9793-9.

Jolliffe, I.T., Cadima, J., 2016. Principal component analysis: a review and recent developments. Phil. Trans. R. Soc. A 374 (2065), 20150202. http://dx.doi.org/10.1098/rsta.2015.0202.

Lanckriet, G.R., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.I., 2004. Learning the kernel matrix with semidefinite programming. J. Mach. Learn. Res. 5 (Jan), 27–72. http://dx.doi.org/10.5555/1005332.1005334.

Pant, H., Soman, S., Bhaya, A., et al., 2020. Neurodynamical classifiers with low model complexity. Neural Netw. 132, 405–415. http://dx.doi.org/10.1016/j.neunet.2020.08.013.

Refaeilzadeh, P., Tang, L., Liu, H., 2009. Encyclopedia of Database Systems. Arizona State University, pp. 532–538.

Rey, D., Neuhäuser, M., 2011. Wilcoxon-signed-rank test.

Schölkopf, B., Smola, A., Müller, K.-R., 1998. Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput. 10 (5), 1299–1319. http://dx.doi.org/10.1162/089976698300017467.

Shah, S., Chandra, S., et al., 2009. Kernel optimization using a generalized eigenvalue approach. In: International Conference on Pattern Recognition and Machine Intelligence. Springer, pp. 32–37. http://dx.doi.org/10.1007/978-3-642-11164-8_6.

Sharma, M., Soman, S., Pant, H., et al., 2017. Large-scale minimal complexity machines using explicit feature maps. IEEE Trans. Syst. Man Cybern.: Syst. http://dx.doi.org/10.1109/TSMC.2017.2694321.

Smola, A.J., Schölkopf, B., Müller, K.-R., 1998. The connection between regularization operators and support vector kernels. Neural Netw. 11 (4), 637–649. http://dx.doi.org/10.1016/S0893-6080(98)00032-X.

Vapnik, V., Levin, E., Le Cun, Y., 1994. Measuring the VC-dimension of a learning machine. Neural Comput. 6 (5), 851–876. http://dx.doi.org/10.1162/neco.1994.6.5.851.

Xiong, H., Swamy, M., Ahmad, M.O., 2005. Optimizing the kernel in the empirical feature space. IEEE Trans. Neural Netw. 16 (2), 460–474. http://dx.doi.org/10.1109/TNN.2004.841784.