ARL-TR-9604 ● Oct 2022

Subspace Learning Machine (SLM): A New Approach to Classification and Regression

by Vinod K Mishra and C-C Jay Kuo

ARL-TR-9604 ● Oct 2022

Subspace Learning Machine (SLM): A New Approach to Classification and Regression

Vinod K Mishra
DEVCOM Army Research Laboratory

C-C Jay Kuo
University of Southern California

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* October 2022	**2. REPORT TYPE** Technical Report	**3. DATES COVERED (From - To)** 1 January–30 June 2022
4. TITLE AND SUBTITLE Subspace Learning Machine (SLM): A New Approach to Classification and Regression		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Vinod K Mishra and C-C Jay Kuo		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) DEVCOM Army Research Laboratory ATTN: FCDD-RLC-NC Aberdeen Proving Ground, MD 21005		**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TR-9604
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCID ID: Vinod K Mishra, 0000-0001-9432-9082

14. ABSTRACT
Classification and regression are some of the most important tasks handled by supervised machine learning. Many approaches like feedforward multilayer perceptron, decision tree, support vector machines, and extreme learning machine methods have been proposed in the past for these tasks. Recently, a new approach called subspace learning machine/regressor (SLM/SLR) has been applied to data with low to moderate dimensions and it has shown substantial advantages over other similar methods. This technical report describes SLM/SLR and traces the reasons behind its superior performance.

15. SUBJECT TERMS
Machine Learning, Classification, Subspace Learning, Network and Computational Sciences

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Vinod K Mishra
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	UU	35	**19b. TELEPHONE NUMBER (Include area code)** (410) 278-0114

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

1. Introduction

Feature-based classification and regression tasks have been handled by deep learning (DL)-based models like FF-MLP,[1] DT,[2] SVM,[3] and ELM[4] for a long time. Such models have been found to be especially effective, but they also suffer from the lack of interpretability, high model complexity, and high computational cost. They all approach the common task of feature space partitioning in different ways.

We propose a new classification-oriented machine learning model named subspace learning machine (SLM). It finds a balance between simplicity and effectiveness by partitioning an input feature space into multiple discriminant subspaces in a hierarchical manner. SLM does not change the feature space at all. The probabilistic projection in SLM is simply used for feature space partitioning without generating new features. Each tree node splitting corresponds to a hyperplane partitioning through weights and bias learning. As a result, both half subspaces can be preserved. SLM learns partitioning parameters in a feedforward and probabilistic approach, which is efficient and transparent.

We explain the differences between SLM and other popular classification approaches in the following sections.

1.1 Feedforward Multilayer Perceptron (FF-MLP)

FF-MLP uses linear discriminant analysis (LDA) and the Gaussian mixture model (GMM) to capture feature distributions of multiple classes. It adopts neuron pairs with weights of opposite signed vectors to represent partitioning hyperplanes. It was introduced in 1958[3] and has been broadly applied to many classification and regression tasks. [5–7] Its universal approximation capability is studied in Hornik (1989), Stinchombe (1989), Glover (1986), and Leshno et al. (1993).[8–11]

There are two approaches to the design of a practical MLP solution:

- The parameters are finetuned at each layer through back propagation.[12] Architecture of MLP includes tabu search (a metaheuristic algorithm for solving combinatorial optimization problems)[13] and simulated annealing.[14] In the convolutional neural networks (CNNs)[15–18] variant of MLP, convolutional layers share neuron weights and biases across different spatial locations. The fully connected layers remain the same as in traditional MLPs. It also serves as the building blocks in transformer models.[19,20]
- MLP layers are constructed layer by layer.[21–24] In one optimization method,[25–27] the parameters of the newly added hidden layer are added

without back propagation.[28–30] In a method using CNN, the convolution operation changes the input feature space to an output feature space serving as the input to the next layer. Later the nonlinear activation in a neuron partitions the output feature space. Only one-half subspace is selected to resolve the sign confusion problem caused by the cascade of convolution operations.[31,4]

1.2 Decision Tree (DT)

A DT partitions one space into two subspaces by recursively selecting the most discriminant feature one at a time. Selecting a partition in DTs is easier, as it is conducted on a single feature, but its discriminant power is weak, so it is a weak classifier. For high tree depths, multiple DTs can be used to avoid overfitting the training data. In such situations, each of them individually is a weak classifier but their ensemble yields a strong one; for example, the random forest (RF) classifier.[32]

Classification and Regression Tree (CART)[2] (and similarly ID3[33]) are classical DT algorithms. They are weak classifiers but can achieve higher performance by using multiple DTs with bootstrap aggregation[32] and other boosting methods. [34] They may still fail due to poor split of training and test data and overfitting of the training data. As compared to them, one SLM tree (i.e., SLM Baseline) can exploit discriminant features obtained by probabilistic projections and achieve multiple splits at one node. SLM generally yields wider and shallower trees.

1.3 Random Forest (RF)

An RF consists of multiple decisions trees, and its predictive performance[35] depends on 1) the strength of individual trees, and 2) on a measure of their dependence that should be lower. RF training takes only a fraction of training samples and their features in building a tree. Thus, it trades the strength of each DT for the general ensemble performance for achieving a higher diversity. Several designs have been proposed for achieving uncorrelated individual trees, as follows:

- Bagging[36] builds each tree through random selection with replacement in the training set.
- Random split selection[37] selects a split at a node among the best splits at random.
- A random subset of features is selected[38] to grow each tree.

RF uses bagging and feature randomness to create uncorrelated trees in a forest, and their combined prediction is more accurate than that of an individual tree. In contrast, the SLM forest building process uses all training samples and the whole feature space. It utilizes feature randomness to achieve the diversity of each SLM tree (described in Section 3.1). The effective diversity and strength of an individual SLM tree remains unaffected in building an SLM forest. So, the SLM forest achieves better predictive performance and faster convergence in terms of the tree number.

1.4 Support Vector Machine (SVM)

SVM algorithm tries to find a hyperplane in an N-dimensional space (N = number of features). The optimum hyperplane has the maximum margin or the distance from the data points of all classes. Support vectors are the data points closer to the hyperplane and influence the position and orientation of the hyperplane. They are used to maximize the margin.

1.5 Extreme Learning Machine (ELM)

It projects a high-dimensional space to a 1D space randomly to find the optimal split point in the associated 1D space. Theory of random projection learning models and their properties (e.g., interpolation and universal approximation) has been investigated.[39–42] ELM is not efficient in practice for high feature dimensions as many trials are needed for finding good projections. It adopts random weights for training feed forward neural networks.[43] MLP can be built with ELM by adding new layers with randomly generated weights. However, it is inefficient in practice due to the requirements of long training time and large model size for a large search space.

SLM Baseline does take the efficiency into account by building a general DT through probabilistic projections, which reduces the search space by leveraging most discriminant features with several hyper-parameters. We use the term "probabilistic projection" rather than "random projection" to emphasize their difference.

1.6 Gradient Boosting Decision Tree (GBDT)

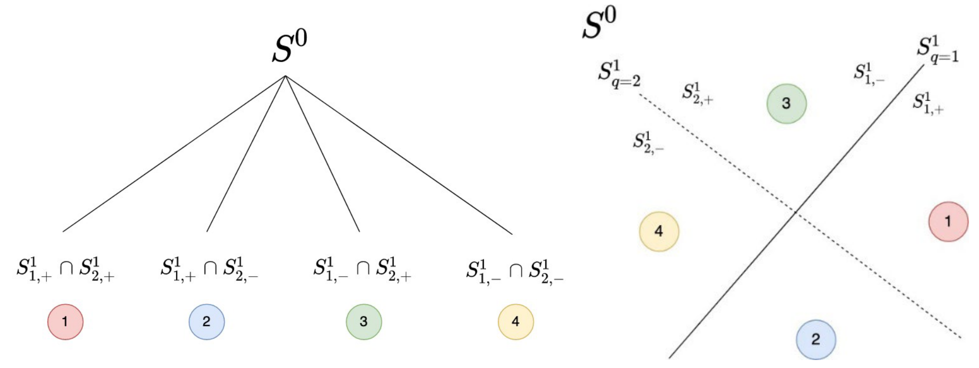
Gradient boosting is another ensemble method of weak learners. It builds a sequence of weak prediction models. Each new model attempts to compensate the prediction residual left in previous models. The gradient boosting decision tree (GBDT) method includes 1) the standard gradient boosting,[44] and 2) XGBoost.[45,46] It expands a general loss function in a Taylor series and defines a gain to perform more effective node splitting than standard DTs.

SLM Boost mimics the boosting process of XGBoost but replaces DTs with SLM trees. As compared with standard GBDT methods, SLM Boost achieves faster convergence and better performance because of stronger performance of an SLM tree.

2. Classification Using Subspace Learning Machine (SLM)

The SLM method is motivated by two basic ideas:

- Random projection is used to find more discriminant subspaces.
- In DT, one parent node is split into two child nodes. SLM allows an n-ary split instead. One example is shown in Fig. 1a and b, where space S^0 is split into four disjoint subspaces. Generally, the n-ary split gives wider and shallower decision trees so that overfitting can be avoided more easily.



S^0
$S^1_{1,+} \cap S^1_{2,+}$
$S^1_{1,+} \cap S^1_{2,-}$
$S^1_{1,-} \cap S^1_{2,+}$
$S^1_{1,-} \cap S^1_{2,-}$
1
2
3
4
S^0
$S^1_{q=2}$
$S^1_{2,+}$
$S^1_{2,-}$
$S^1_{1,-}$
$S^1_{q=1}$
$S^1_{1,+}$

Fig. 1 (a) An illustration of SLM, where space S^0 is partitioned into four subspaces with two splits, and (b) the corresponding SLM tree with a root node and four child nodes

We are given an input data set X.

- Denote a data sample as $\boldsymbol{x_l}$ which is a vector in a feature space (L = number of sample-vectors, D = number of features in each sample or the dimension of the feature space)

$$\boldsymbol{x_l} = \left(x_{l,1}, x_{l,2}, \dots, x_{l,d}, \dots, x_{l,D}\right)^T \in R^D, l = 1,2,\dots,L$$

An example for a point in ring and circle data set (D = 2) is $\boldsymbol{x_l} = \binom{x_{l,1}}{x_{l,2}}$.

- Each data vector $\boldsymbol{x_l}$ has the class label vector y_l, which has K classes. ($\boldsymbol{y} \sim K \times L$). Each y_l vector has K-dimensional label space of classes

$$y_l = \left(y_{l,1}, y_{l,2}, \dots, y_{l,k}, \dots, y_{l,K}\right)^T \in R^K, l = 1,2,\dots,L; k = 1,2,\dots,K$$

Here $y_{l,k}$= 1, and $y_{l,k'}$= 0 for $k \neq k'$.

- The combined sample and outcome are given by $D_l = (\boldsymbol{x_l}, y_l)$ having D+1 elements.

In the partitioning process, the root node is the whole sample space, and an intermediate or leaf node corresponds to a partitioned subspace. The goal of SLM is to hierarchically partition the feature space RD into multiple subspaces so that finally the samples at leaf nodes are as pure as possible. At the end of the process most samples at a node should be in the same class. Then all samples in the leaf node can be assigned to the majority class. The data classification by SLM has the following steps.

Step 1: $X \rightarrow S^0$: The feature subspace S^0 with dimension D_0 is determined and assigned to the root node of a DT.

For X with low dimensions $S^0 \approx$ X, and for X with high dimensions, less discriminant features from X are removed such that $D_0 < D$. For that, denote the *d*-th feature set of xl by $\boldsymbol{F_d} = \{x_{l,d} | 1 \leq l \leq L\}$.

Express the projection vectors (PV) in terms of basis vectors.

Express the PV as

$$\boldsymbol{a} = a_1\boldsymbol{e_1} + \cdots + a_d\boldsymbol{e_d} + \cdots + a_D\boldsymbol{e_D}, \left||\boldsymbol{a}|\right| = 1$$

Here $\boldsymbol{e_d}$ is the basis vector in which *d*-th value is 1 and the rest are 0.

Rewrite PV in terms of reordered basis vectors.

Choose an $\boldsymbol{e_d}$ and a corresponding partitioning point t_d. Calculate the loss function $L_d(t_d)$ (as given in Appendix B). Vary t_d, and find the optimum t_d^* out of them that minimizes $L_d(t_d)$.

$$t_d^* = argmin_{t_d} L_d(t_d)$$

Rank t_d^* according to their loss function values

$$L_1(t_1^*) \leq L_2(t_2^*) \ldots . \leq L_D(t_D^*)$$

and reorder basis vectors as ($\boldsymbol{e'_1}$, $\boldsymbol{e'_2}$,…, $\boldsymbol{e'_d}$,…, $\boldsymbol{e'_D}$) according to the loss function, which allows rewriting PV as

$$\boldsymbol{a} = a'_1\boldsymbol{e'_1} + \cdots + a'_d\boldsymbol{e'_d} + \cdots + a'_D\boldsymbol{e'_D}, \left||\boldsymbol{a}'|\right| = 1$$

Use hyperparameters to choose a'_d

The a'_d coefficients are chosen in a probabilistic manner using three hyperparameters (β, a, R).

i) β = Coefficient in the exponential distribution. It controls the probability of selecting a'_d using

$P_d = \beta_0 e^{-\beta d}$ (β_0 is the normalization). It is higher for smaller *d.*

ii) α = Coefficient in the exponential distribution. It controls the probability of selecting the dynamic range of a'_d using envelope parameter $A_d = \alpha_0 e^{-\alpha d}$ (α_0 is the normalization)., $\alpha > 0$. Only the integer values are used due to ease of computation.

$a'_d = 0, \pm 1, \pm 2, \ldots, \pm A_d$ (Integer part)

iii) R = Number of selected a'_d coefficients such that R < D (for large D, R << D). The rest of the $D - R$ coefficients are set to zero.

The data vector dimension is reduced, and it is $x_0 = \left(x_{0,1}, x_{0,2}, \ldots, x_{0,D_0}\right)$, $D_0 = D - R$.

Step 2: Generate *p* PVs using this process to get *p* 1D subspaces.

The three hyperparameters (β, a, R) define a collection of many PVs whose search space is bounded by

Lower bound = $\prod_{d=D+1-R}^{R}(2A_d + 1)$, and Upper bound = $\prod_{d=1}^{R}(2A_d + 1)$.

For each chosen PV in this set, the DFT or loss function is calculated to choose the best *p* PVs.

Step 3: $p \rightarrow q$ 1D subspaces

Select the best *q* subspaces from *p* candidate subspaces based on discriminability and correlation.

Step 4: Use *n*-ary splits to get 2*q* child nodes (example in Fig. 1).

3. Extension of Basic SLM to SLM Forest and SLM Boost

An ensemble model aims to obtain better performance than each constituent model alone, using methods like bootstrap aggregating ("bagging") and boosting—for example, RF and GBDT. The bagging applied to SLM gives SLM Forest and similarly boosting gives SLM Boost. The latter is inspired by XGBoost.[45,46]

3.1 SLM Forest

The RF contains a set of tree predictors, where each tree is built using the values of a random vector sampled independently. All trees in the forest[43] have the same distribution. According to the Strong Law of Large Numbers, the RF performance converges with increasing tree numbers. Combination of many weak decision trees leads to significant improvement over a single DT result.

Similarly, the SLM Forest uses the diversity of many single SLM trees through probabilistic selection to get better results than single tree. For partitioning at a node, D_0 dimensions are probabilistically selected from the D input feature dimensions by considering each feature's discriminant ability. A larger β denotes more discriminant features. All training samples and all feature dimensions are kept as the input at each node splitting to increase the strength of individual SLM trees. The decorrelating partitioning planes are used to get better performance and faster converge than RF.

3.2 SLM Boost

With standard DTs as weak learners, GBDT[44] and XGBoost[45,46] can deal with a large amount of data efficiently and achieve the state-of-the-art performance in many machine learning problems. They take the ensemble of standard DTs with boosting, that is, by defining an objective function and optimizing it with learning a sequence of DTs. By following the gradient boosting process, we propose SLM Boost to ensemble a sequence of SLM trees.

Let $f_t(x_l)$ denote the *t*-th SLM tree out of *T* trees in total. Then, the prediction of the ensemble is the sum of all trees, that is, each of the *L* samples is predicted as

$$\hat{y}_l = -\sum_{t=1}^{T} f_t(x_l), \text{l} = 1, 2, \ldots, \text{L}$$

Define the objective function for first *t*-trees

$$\Psi(\text{t}) = \sum_{l=1}^{L} \gamma(\hat{y}_l, \hat{y}_l^{(t)})$$

Here $\hat{y}_l^{(t)}$= prediction of sample *l* with all *t* trees, $\gamma(\hat{y}_l, \hat{y}_l^{(t)})$ = training loss for the model with

a sequence of t trees. Let initial model prediction be $\hat{y}_l^{(0)} = 0$, so one gets

$$\hat{y}_l^{(t)} = \hat{y}_l^{(t-1)} + f_t(x_l)$$

Then the objective function is

$$\Psi(\mathrm{t}) = \sum_{l=1}^{L} \gamma(\hat{y}_l, \hat{y}_l^{(t-1)} + f_t(x_l))$$

Following the XGBoost process, we use Taylor expansion up to the second order. With individual SLM trees stronger than individual DTs, SLM Boost achieves better performance and faster convergence than XGBoost.

4. Performance Evaluation of the SLM

4.1 Data Sets

To evaluate the performance of SLM, we conduct experiments on the nine data sets shown in Table 1 (also Appendix A for model parameters).

Table 1 Details of nine data sets used for classification

Data set	Properties	Description
Circle-and-Ring	Feature dimension (FD) = 2, synthetic	An inner circle as one class and an outer ring as the other class as shown in Fig. 4a,[47] 500 samples per class, 20% noisy samples in the decision boundary, samples randomly split into 60% training and 40% test sets
2-New-Moons	FD = 2, synthetic	Two interleaving new moons as shown in Fig. 4b.[47] Each new moon is a class, 500 samples per class, 30% noisy samples in the decision boundary, samples randomly split into 60% training and 40% test sets
4-New-Moons	FD = 2, synthetic	Four interleaving new moons as shown in Fig. 4c.[47] Each new moon is a class, 500 samples per class, 20% noisy samples in the decision boundary, samples randomly split into 60% training and 40% test sets
Iris	FD = 4, real-world	Iris plants data set[47,48] has 3 classes, 4D features, and 150 samples, samples randomly split into 60% training and 40% test sets
Wine	FD = 4, real-world	Wine recognition data set[47,49] has 3 classes, 13D features, and 178 samples, samples randomly split into 60% training and 40% test sets
B.C.W.	FD = 4, real-world	Breast cancer Wisconsin data set[47,49] has 2 classes, 30D features, and 569 samples, samples randomly split into 60% training and 40% test sets
Diabetes	FD = 4, real-world	The Pima Indians diabetes data set[50] is for diabetes prediction. It has 2 classes, 8D features, and 768 samples. By following,[4] we removed samples with the physically impossible zero value for glucose, diastolic blood pressure, triceps skin fold thickness, insulin, or BMI and used the remaining 392 samples for consistent experimental settings, samples randomly split into 60% training and 40% test sets

Table 1 Details of nine data sets used for classification (continued)

Ionosphere	FD = 4, real-world	Binary classification data set[51,49] is used to predict whether the radar return is good or bad. It has 2 classes, 34D features, and 351 instances. For consistency with,[4] we remove the feature dimension that has the zero variance from the data, Samples randomly split into 60% training and 40% test sets
Banknote	FD = 4, real-world	The banknote authentication data set[49] classifies whether a banknote is genuine or forged based on the features extracted from the wavelet transform of banknote images. It has 2 classes, 4D features, and 1372 samples, samples randomly split into 60% training and 40% test sets

We divide the 10 classifiers into two groups, 1) six basic methods: FF-MLP, BP-MLP, LSVM, SVM/RBF, DT, and SLM Baseline, and 2) four ensemble methods: RF, XGBoost, SLM Forest, and SLM Boost.

The best results for each group are shown in bold. SLM Baseline and SVM/RBF often outperform FF-MLP, BP-MLP, LSVM, and DT and give the best results.

Some observations follow.

- For the three synthetic 2D data sets (i.e., circle-and-ring, 2-new-moons, and 4-new-moons), the gain of SLM over MLP is relatively small due to noisy samples. The difference in sample distributions of training and test data plays a role in the performance upper bound. To demonstrate this point, we show sample distributions of their training and testing data in Fig. 2. For the data sets with high-dimensional input features, the SLM methods achieve better performance over the classical ones.
- The network architectures of FF-MLP and BP-MLP and their performance results are taken from Lin et al.[4] FF-MLP has a four-layer network architecture (one input layer, two hidden layers, and one output layer). The input neuron numbers equal feature dimension input and output one is the class number. The neuron numbers at each hidden layer are hyper-parameters determined adaptively by a data set. BP-MLP has the same architecture as FF-MLP against the same data set. Its model parameters are initialized by those of FF-MLP and trained for 50 epochs.
- For the two SVM models, we conduct grid search for hyper-parameter C in LSVM and hyper-parameters C and γ in SVM/RBF for each of the nine data sets to yield the optimal performance.
- For the DT model, the weighted entropy is used as the loss function in node splitting. We do not set the maximum depth limit of a tree, the minimum

sample number and the minimum loss decrease required as the stopping criteria. Instead, we allow the DT for each data set to split until the purest nodes are reached in the training. For the ensemble of DT models (i.e., RF and XGBoost), we conduct grid search for the optimal tree depth and the learning rate of XGBoost to ensure that they reach the optimal performance for each data set. The number of trees is set to 100 to ensure convergence.

- The hyper-parameters of SLM Baseline (i.e., with one SLM tree) include D_0, p, A_{int}, *alpha*, *beta*, and the minimum number of samples used in the stopping criterion. They are searched to achieve the performance as shown. The number of trees in SLM Forest is set to 20 due to the faster convergence of stronger individual SLM trees. The number of trees in SLM Boost is the same as that of XGBoost for fair comparison of learning curves.

Figure 2 shows visualization, and Table 2 shows the classification accuracy results of 10 classifiers.

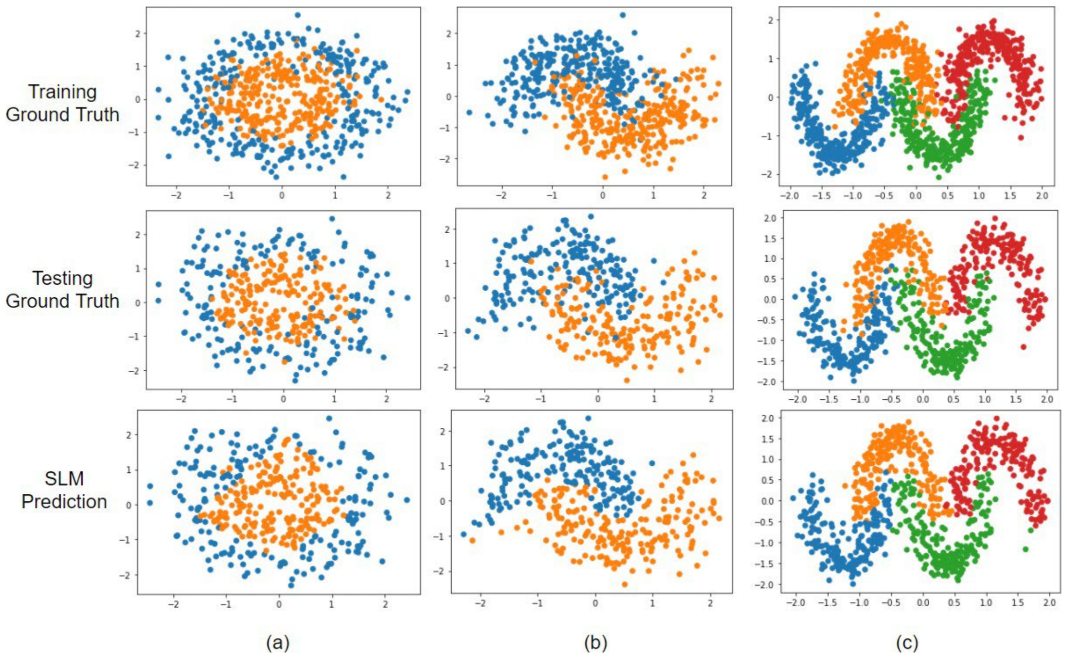


Fig. 2 Visualization of 2D feature data sets: (a) circle-and-ring, (b) 2-new-moon, and (c) 4-new-moon. One ground truth sample of the training data, the ground truth of the test data, and the SLM predicted results are shown in the first, second, and third rows, respectively.

Table 2 Classification accuracy comparison of the 10 benchmarking methods on 9 data sets

Data sets→	circle-and-ring	2-new-moons	4-new-moons	Iris	Wine	B.C.W.	Pima	Ionosphere	Banknote
FF-MLP	87.25	91.25	95.38	98.33	94.44	94.30	73.89	89.36	98.18
BP-MLP	88.00	91.25	87.00	64.67	79.72	97.02	75.54	84.11	88.38
LSVM	48.50	85.25	85.00	96.67	**98.61**	96.49	76.43	86.52	99.09
SVM/RBF	**88.25**	89.75	88.38	**98.33**	**98.61**	**97.36**	75.15	**93.62**	**100.00**
DT	85.00	87.25	94.63	**98.33**	95.83	94.74	77.07	89.36	98.00
SLM Baseline	**88.25**	**91.50**	**95.63**	**98.33**	**98.61**	97.23	**77.71**	90.07	99.09
RF	87.00	90.50	**96.00**	**98.33**	**100.00**	95.61	**79.00**	94.33	98.91
XGBoost	87.50	91.25	**96.00**	**98.33**	**100.00**	98.25	75.80	91.49	99.82
SLM Forest	**88.25**	**91.50**	**96.00**	**98.33**	**100.00**	97.36	**79.00**	**95.71**	**100.00**
SLM Boost	**88.25**	**91.50**	**96.00**	**98.33**	**100.00**	**98.83**	77.71	94.33	**100.00**

4.2 Comparison of Model Sizes

The model size or the number of model parameters of FF/BP-MLP, LSVM, SVM/RBF, DT, and SLM Baseline are compared in Table 2.

- FF-MLP and BP-MLP share the same architecture, so their model sizes are the same. It is calculated by summing up the weight and bias numbers of all neurons.
- The model parameters of LSVM and SVM/RBF can be computed as

$$\text{SVM Parameter} = L + 1 + (D + 2)\, N_{SV} \tag{1}$$

Here L, D, and N_{SV} denote the number of training samples, the feature dimension, and the number of support vectors, respectively. The first term in Eq. 1 is the slack variable for each training sample. The second term denotes the bias. The last term comes from the fact that each support vector has D feature dimensions, one Lagrange dual coefficient, and one class label.

- The model sizes of DTs depend on the number of splits learned during the training process, and there are two parameters learned during each split for feature selection and split value, respectively; the sizes of DTs are calculated as two times the number of splits.
- The size of an SLM baseline model depends on the number of partitioning hyperplanes that are determined by the training stage. For given hyper-parameter $\boldsymbol{D}_0$, each partitioning hyperplane involves one weight matrix and a selected splitting threshold, with $\boldsymbol{q}_i$ decorrelated partitioning learned for partitioning each parent node. Then, the model size of the corresponding SLM can be calculated as

$$\text{SLM Parameter} = \sum_{i=1}^{M} q_i(D_0 + 1) \quad (2)$$

Here M = the number of partitioning hyperplanes. Table 3 compares the model sizes.

Table 3 ML model size comparison (four ML models against nine data sets, smallest model size in bold)

Data set	FF/BP-MLP	LSVM	SVM/RBF	DT	SLM Baseline	DT Depth	SLM Tree depth
Circle-and-Ring	125	2,965	1,425	350	**39**	14	4
2-new-moons	114	1,453	1,305	286	**42**	15	4
4-new-moons	702	2,853	2,869	298	**93**	11	5
Iris	47	235	343	34	**20**	6	3
Wine	147	453	963	**26**	99	4	2
B.C.W.	74	1,462	3,254	**54**	126	7	4
Pima	2,012	1,532	1,802	130	55	8	3
Ionosphere	278	1,017	2,207	**50**	78	10	2
Banknote	22	1,160	1,322	78	**40**	7	3

4.3 Comments on Performance

- For SVM, the training involves learning the dual coefficients and slack variables for each training sample and memorization of the support vectors. The model size increases linearly with the number of training samples and number of support vectors. For similar classification accuracy, the SVM models are much heavier with a large number of training samples.
- For MLPs, SLM models outperform the MLP models in all benchmarking data sets for high-dimension classification tasks.
- For the data sets with saturated performance such as Iris, Banknote, and Ionosphere, SLM achieves better or comparable performance with less than half of the parameters of MLP.
- For DTs, the SLM models give wider and shallower trees. The depth of SLM trees are overall smaller than the DT models, while the number of splitting can be comparable for the small data sets, like Wine. The SLM trees tend to make more splits to reach pure leaf nodes at shallower depth. While outperforming the DTs in all the data sets, the SLM model sizes are generally smaller than the DTs as they benefit from the subspace partitioning process.

4.4 Convergence Performance Comparison of DT Ensembles and SLM Ensembles

We compare the convergence performance of the ensemble and the boosting methods of DT and SLM for Wine, B.C.W., and Pima, three data sets in Fig. 3a–c.

- For RF and SLM Forest, which are ensembles of DT and SLM, respectively, we set their maximum tree depth and learning rate to the same. We show their accuracy curves as a function of the tree number in the left subfigure. We see that SLM Forest converges faster than RF.
- For XGBoost and SLM Boost, which are boosting methods of DT and SLM, respectively, we show the logloss value as a function of the tree number in the right subfigure. Again, we see that SLM Boost converges faster than XGBoost.

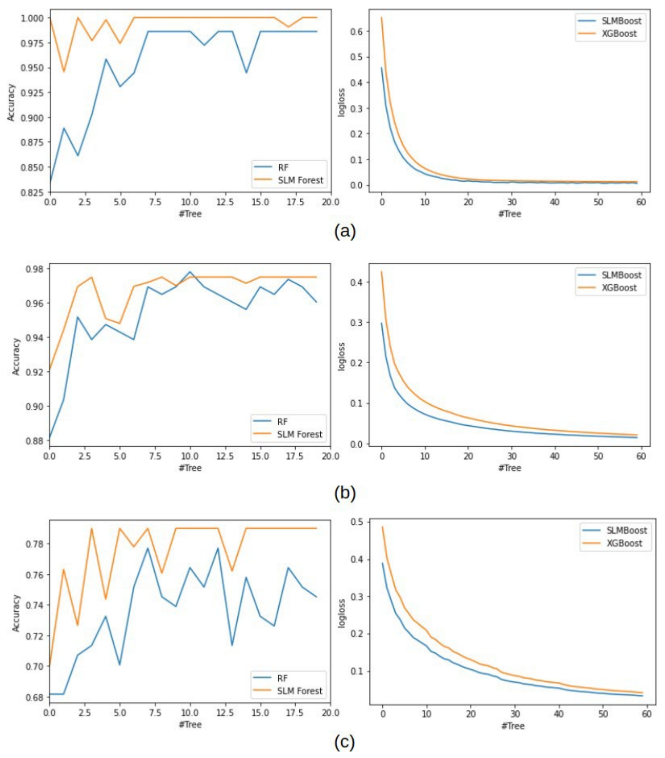


Fig. 3 Comparison of SLM and DT ensembles for three data sets: (a) Wine, (b) B.C.W., and (c) Pima. Each left subfigure compares the accuracy curves of SLM Forest and RF as a function of the tree number. Each right subfigure compares the logloss curves of SLM Boost and XGBoost as a function of the tree number.

5. Subspace Learning Regressor (SLR)

5.1 Method

A different loss function can be adopted in the subspace partitioning process for a different task. For example, to solve a regression problem, we can follow the same methodology as described in Section 2 but adopt the mean-squared-error (MSE) as the loss function. The resulting method is called subspace learning regression (SLR), and the corresponding regressor is the subspace learning regressor.

Mathematically, each training sample has a pair of input $\mathbf{x}$ and output y, where $\mathbf{x}$ is a D-dimensional feature vector and y is a scalar that denotes the regression target. Then, we build an SLR tree that partitions the D-dimensional feature space hierarchically into a set of leaf nodes. Each of them corresponds to a subspace. The mean of sample targets in a leaf node is set as the predicted regression value of these samples. The partition objective is to reduce the total MSE of sample targets as much as possible. In the partitioning process, the total MSE of all leaf nodes decreases gradually and saturates at a certain level.

The ensemble and boosting methods are applicable to SLR. The SLR Forest consists of multiple SLR trees through ensembles. Its final prediction is the mean of predictions from SLR trees in the forest. To derive SLR Boost, we apply the GBDT process and train a series of additive SLR trees to achieve gradient boosting, leading to further performance improvement. As compared with a decision tree, an SLR tree is wider, shallower, and more effective. As a result, SLR Forest and SLR Boost are more powerful than their counterparts as demonstrated in the next subsection.

5.2 Performance Evaluation

To evaluate the performance of SLR, we compare the root mean-squared-error (RMSE) performance of eight regressors on six data sets in Table 3. The five benchmarking regressors are linear SVR (LSVR), SVR with RBF kernel, DT, RF, and XGBoost. There are three variants of SLR: SLR Baseline (with one SLR tree), SLR Forest, and SLR Boost. The first three data sets are synthesized data sets as described by Quinlan.[52] We generate 1000 samples for each of them. The last three data sets are real world data sets. Samples in all six data sets are randomly split into 60% training samples and 40% test samples. Table 4 describes the regression data.

Table 4 Regression data sets

Data set	Description
Make Friedman 1	Its input vector, x, contains P (with $P > 5$) elements, which are independent and uniformly distributed on interval [0, 1]. Its output, y, is generated by the first five elements of the input. The remaining (P 5) elements are irrelevant features and can be treated as noise. We refer to Quinlan[52] for details. We choose $P = 10$ in the experiment.
Make Friedman 2-3	Their input vector, x, has four elements. They are independent and uniformly distributed on interval [0, 1]. Their associated output, y, can be calculated by all four input elements via mathematical formulas as described by Quinlan.[52]
Boston	It contains 506 samples, each of which has a 13D feature vector as the input. An element of the feature vector is a real positive number. Its output is a real number within interval [5, 50].
California Housing	It contains 20640 samples, each of which has an 8D feature vector. The regression target (or output) is a real number within interval [0.15, 5].
Diabetes	It contains 442 samples, each of which has a 10D feature vector. Its regression target is a real number within interval [25, 346].

The performance comparison is given in Table 5.

Table 5 Regression performance (eight regressors, six data sets)

Data sets	make friedman1	make friedman2	make friedman3	Boston	California housing	Diabetes
LSVR	2.49	138.43	0.22	4.90	0.76	53.78
SVR/RBF	**1.17**	**6.74**	**0.11**	**3.28**	**0.58**	**53.71**
DT	3.10	33.57	0.11	4.75	0.74	76.56
SLR Baseline	2.89	31.28	0.11	4.42	0.69	56.05
RF	2.01	22.32	0.08	3.24	0.52	54.34
XGBoost	1.17	32.34	0.07	2.67	0.48	53.99
SLR Forest	1.88	20.79	0.08	3.01	0.48	52.52
SLR Boost	**1.07**	**18.07**	**0.06**	**2.39**	**0.45**	**51.27**

Some observations follow.

- SLR Baseline outperforms DT in all data sets.
- Also, SLR Forest and SLR Boosting outperform RF and XGBoost, respectively.
- For make friedman1, make friedman3, California-housing, Boston, and diabetes data sets, SLR Boost achieves the best performance.
- For make friedman2, SVR/RBF achieves the best performance benefiting from the RBF on its specific data distribution. However, it is worthwhile to emphasize that, to achieve the optimal performance, SVR/RBF needs to overfit to the training data by fine-tuning the soft margin with a large regularization parameter (i.e., $C = 1000$). This leads to much higher

computational complexity. With stronger individual SLR trees and effective uncorrelated models, the ensemble of SLR can achieve better performance than DTs with efficiency.

6. Conclusion and Future Work

A novel ML model, the SLM, combines FF-MLP design and DT, and learns to discriminate subspace and make predictions. It utilizes the methods of hyperplane partitioning and random projection to achieve significantly better performance compared to other methods. It is lightweight, mathematically transparent, adaptive to high dimensional data, and achieves state-of-the-art benchmarking performance. Also, an SLM tree can serve as a weak classifier in general boosted and bootstrap aggregation methods as a more generalized model. In future, we will investigate SLM's extension to very high dimensional input data.

7. References

1. Cortes C, Vapnik V. Support-vector networks. Machine Learning. 1995;20(3):273–297.
2. Breiman L, Friedman J, Stone C, Olshen R. Classification and regression trees. Taylor & Francis; 1984.
3. Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review. 1958;65(6):386.
4. Lin R, Zhou Z, You S, Rao R, Kuo C-C J. From two-class linear discriminant analysis to interpretable multilayer perceptron design. arXiv preprint arXiv:2009.04442; 2020.
5. Devadoss AV, Ligori TAA. Forecasting of stock prices using multi layer perceptron. International Journal of Computing Algorithm. 2013;2(1):440–449.
6. Sivakumar K, Desai UB. Image restoration using a multilayer perceptron with a multilevel sigmoidal function. IEEE Transactions on Signal Processing. 1993;41(5)2018–2022.
7. Cybenko G. Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems. 1989;2(4):303–314.
8. Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. Neural Networks. 1989;2(5):359–366.
9. Stinchombe M. Universal approximation using feed-forward networks with nonsigmoid hidden layer activation functions. Proceedings of the IJCNN; 1989. p. 161–166.
10. Leshno M, Lin VY, Pinkus A, Schocken S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Networks. 1993;6(6):861–867.
11. Glover F. Future paths for integer programming and links to artificial intelligence. Computers & Operations Research. 1986;13(5):533–549.
12. LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD. Backpropagation applied to handwritten zip code recognition. Neural Computation. 1989;1(4):541–551.

13. Kirkpatrick S, Gelatt Jr CD, Vecchi MP. Optimization by simulated annealing. Science. 1983;220(4598):671–680.

14. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998;86(11):2278–2324.

15. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems. 2012;25.

16. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436–444.

17. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł. Polosukhin I. Attention is all you need. Advances in Neural Information Processing Systems. 2017;30.

18. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, et al. An image is worth 16x16 words: transformers for image recognition at scale. arXiv preprint arXiv:2010.11929; 2020.

19. Parekh R, Yang J, Honavar V. Constructive neural network learning algorithms for multi-category real-valued pattern classification. Dept Comput Sci, Iowa State University; 1997. Report No.: ISU-CS-TR97-06.

20. Me´zard M, Nadal J-P. Learning in feedforward layered networks: the tiling algorithm. Journal of Physics A: Mathematical and General. 1989;22(12):2191.

21. Frean M. The upstart algorithm: a method for constructing and training feedforward neural networks. Neural Computation. 1990;2(2):198–209.

22. Parekh R, Yang J, Honavar V. Constructive neural-network learning algorithms for pattern classification. IEEE Transactions on Neural Networks. 2000;11(2):436–451.

23. Kwok T-Y, Yeung D-Y. Objective functions for training new hidden units in constructive neural networks. IEEE Transactions on Neural Networks. 1997;8(5):1131–1148.

24. Gallant SI, et al. Perceptron-based learning algorithms. IEEE Transactions on Neural Networks. 1990;1(2):179–191.

25. Mascioli FF, Martinelli G. A constructive algorithm for binary neural networks: the oil-spot algorithm. IEEE Transactions on Neural Networks. 1995;6(3):794–797.

26. Parekh R, Yang J, Honavar V. Constructive neural-network learning algorithms for pattern classification. IEEE Transactions on Neural Networks. 2000;11(2):436–451.

27. Yang J, Parekh R, Honavar V. Distal: an inter-pattern distance-based constructive learning algorithm. Intelligent Data Analysis. 1999;3(1):55–73.

28. Marchand M. Learning by minimizing resources in neural networks. Complex Systems. 1989;3:229–241.

29. Kuo C-C J. Understanding convolutional neural networks with a mathematical model. Journal of Visual Communication and Image Representation. 2016;41:406–413.

30. Huang G-B, Zhu Q-Y, Siew C-K. Extreme learning machine: theory and applications. Neurocomputing. 2006;70(1-3):489–501.

31. Huang G-B, Chen L, and Siew CK. Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Networks. 2006;17(4):879–892.

32. Friedman JH. Greedy function approximation: a gradient boosting machine. Annals of statistics. 2001;1189–1232.

33. Chen T, Guestrin C. XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016. p. 785–794.

34. Breiman L. Bagging predictors. Machine Learning. 1996;24(2):123–140.

35. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: machine learning in Python. The Journal of Machine Learning Research. 2011;12:2825–2830.

36. Dietterich TG. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. Machine Learning. 2000;40(2):139–157.

37. Ho TK. The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1998;20(8):832–844.

38. Ahad A, Fayyaz A, Mehmood T. Speech recognition using multilayer perceptron. In: IEEE Students Conference, ISCON'02. Proceedings, vol. 1. IEEE; 2002, p. 103–109.

39. Huang G-B, Chen L. Convex incremental extreme learning machine. Neurocomputing. 2007;70(16-18):3056–3062.

40. Huang G-B, Chen L. Enhanced random search based incremental extreme learning machine. Neurocomputing. 2008;71(16-18):3460–3468.

41. Chen Y, Xu Z, Cai S, Lang Y, Kuo C-C J. A SAAK transform approach to efficient, scalable and robust handwritten digits recognition. In: 2018 Picture Coding Symposium (PCS). IEEE; 2018. p. 174–178.

42. Chen Y, Kuo C-C J. Pixelhop: A successive subspace learning (ssl) method for object recognition. Journal of Visual Communication and Image Representation. 2020;70:102749.

43. Huang G-B, Zhu Q-Y, Siew C-K. Extreme learning machine: theory and applications. Neurocomputing. 2006;70(1-3):489–501.

44. Yang Y, Wang W, Fu H, Kuo C-C J. On supervised feature selection from high dimensional feature spaces. arXiv preprint arXiv:2203.11924; 2022.

45. Chen T, Guestrin C. XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016. p. 785–794.

46. Amit Y, Geman D. Shape quantization and recognition with randomized trees. Neural computation. 1997;9(7):1545–1588.

47. Fisher RA. The use of multiple measurements in taxonomic problems. Annals of Eugenics. 1936;7(2):179–188.

48. Asuncion A, Newman D. UCI machine learning repository. Irvine University of California; 2007.

49. Smith JW, Everhart JE, Dickson W, Knowler WC, Johannes RS. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. Proceedings of the Annual Symposium on Computer Application in Medical Care. American Medical Informatics Association; 1988. p. 261.

50. Göpfert JP, Wersing H, Hammer B. Interpretable locally adaptive nearest neighbors. Neurocomputing. 2022;470:344–351.

51. Friedman JH. Multivariate adaptive regression splines. The Annals of Statistics. 1991;19(1):1–67.

52. Quinlan JR. Induction of decision trees. Machine Learning. 1986;1(1):81–106.

Appendix A. Model Specifications for Data Sets

FF-MLP = Feed-forward multilayer perceptron, LSVM = Lagrangian support vector machine

SVM/RBF = Support vector machine/restricted Boltzmann functions, DT = Decision tree

SLM = Subspace learning machine

Data sets	**FF-MLP**	**LSVM**	**SVM/RBF**	**DT**	**SLM**
Circle-and-Ring	Gaussian components = 4 for ring, 1 for circle, 8 and 9 neurons in the two hidden layers, total 125 parameters	Slack variables = 600, support vectors = 591, bias = 1, total parameters = 2965	Slack variables = 600, support vectors = 206, bias = 1, total parameters = 1425	Splits = 175 tree depth = 14, total parameters = 350	Input features = 2, tree depth = 4. node numbers at each level = 1, 4, 4, 10, and 8, partitions = 13, total parameters = 39
2-New-Moons	Gaussian components = 2 for each class, 8 neurons in each of two hidden layers, total parameters = 114	Slack variables = 600, support vectors = 213, bias = 1, total parameters = 1453	Slack variables = 600, support vectors = 176, bias = 1, total parameters = 1305	Splits = 143 tree depth = 15, total parameters = 286	Tree depth = 4. node numbers at each level = 1, 4, 8, 12, and 4, partitions = 14, total parameters = 42
4-New-Moons	Gaussian components = 3 for each class, 18 and 28 neurons in two hidden layers, total parameters = 702	Slack variables = 1200, support vectors = 413, bias =1, total parameters = 2853	Slack variables = 1200, support vectors = 417, bias =1, total parameters = 2869	Splits = 149 tree depth = 11, total parameters = 298	Tree depth = 5. node numbers at each level = 1, 4, 16, 22, 16, and 4, partitions = 31, total parameters = 93
Iris	Gaussian components = 2 for each class, 4 and 3 neurons in two hidden layers, total parameters = 47	Slack variables = 90, support vectors = 24, bias =1, total parameters = 235	Slack variables = 90, support vectors = 42, bias =1, total parameters = 343	Splits = 17 tree depth = 6, total parameters = 34	Input features = 4 tree depth = 3. Node numbers at each level = 1, 2, 2, and 4, partitions = 4, total parameters = 20

Wine	Gaussian components = 2 for each class, 6 neurons in each of two hidden layers, total parameters = 147	Slack variables = 107, support vectors = 23, bias =1, total parameters = 453	Slack variables = 107, support vectors = 57, bias =1, total parameters = 963	Splits = 13 tree depth = 4, total parameters = 26	Input features = 8 tree depth = 2. Node numbers at each level = 1, 8, and 256, partitions = 11, total parameters =99
B.C.W.	Gaussian components = 2 for each class, 2 neurons in each of two hidden layers, total parameters = 74	Slack variables = 341, support vectors = 35, bias =1, total parameters = 1462	Slack variables = 341, support vectors = 91, bias =1, total parameters = 3254	Splits = 27 tree depth = 7, total parameters = 54	Input features = 5 tree depth = 4. Node numbers at each level = 1, 8, 16, 8, and 32, partitions = 21, total parameters =126
Diabetes	Gaussian components = 2 for each class, 18 and 88 neurons in two hidden layers, total parameters = 2012	Slack variables = 461, support vectors = 107, bias = 1, total parameters = 1532	Slack variables = 461, support vectors = 134, bias = 1, total parameters = 1802	Splits = 65 tree depth = 8, total parameters = 130	Input features = 4 tree depth = 3. Node numbers at each level = 1, 2, 16, and 20, partitions = 11, total parameters = 55
Ionosphere	Gaussian components = 2 for each class, 6 and 8 neurons in two hidden layers, total parameters = 278	Slack variables = 211, support vectors = 23, bias =1, total parameters = 1017	Slack variables = 211, support vectors = 57, bias =1, total parameters = 2207	Splits = 25 tree depth = 10, total parameters = 50	Input features = 5 tree depth = 2. Node numbers at each level = 1, 4, and 20, partitions = 13, total parameters = 78
Banknote	Gaussian components = 2 for each class, 2 neurons at each of the two hidden layers, total parameters = 22	Slack variables = 823, support vectors = 56, bias =1, total parameters = 1160	Slack variables = 823, support vectors = 834, bias = 1, total parameters = 1322	Splits = 39 tree depth = 7, total parameters = 78	Input features = all tree depth = 3. Node numbers at each level = 1, 2, 8, and 18, partitions = 8, total parameters = 40

Appendix B. Loss Function

The process for calculating Discriminant Feature Test (DFT) or loss function is as follows.

(i) Denote d-th feature set of $\boldsymbol{x}_l$ by $\boldsymbol{F_d} = \{x_{l,d} | 1 \leq l \leq L\}$. Choose a partitioning point t_d corresponding to the feature x_d so that the data vectors are divided in two classes.

$D_d^{left} = \{(\boldsymbol{x}, y) | x_d \leq t_d\}$. Number of data vectors = N^{left}

$D_d^{right} = \{(\boldsymbol{x}, y) | x_d > t_d\}$. Number of data vectors = N^{right}

Next, let

$N_k^{left} \, or \, N_k^{right}$= #data vectors of class k in the left or right division

$N^{left} = \sum_{k=1}^{K} N_k^{left}$, $N^{right} = \sum_{k=1}^{K} N_k^{right}$, $N = N^{left} + N^{right}$

The class-specific probabilities are

$$p_k^{left} = \frac{N_k^{left}}{N^{left}}, p_k^{right} = \frac{N_k^{right}}{N^{right}}$$

The class-specific entropies are

$$H\left(D_d^{left}\right) = -\sum_{k=1}^{K} p_k^{left} \log\, p_k^{left} \,, H\left(D_d^{right}\right) = -\sum_{k=1}^{K} p_k^{right} \log\, p_k^{right}$$

Then the DFT function is

$$L_d(t_d) = \frac{N^{left}}{N} H\left(D_d^{left}\right) + \frac{N^{right}}{N} H(D_d^{right})$$

Thus, dimension D is reduced to D_0 or the dimension of the feature space satisfying the discriminant bound ($D_0 < D$). Out of many partitioning points t_d, the optimum t_d^* minimizes $L_d(t_d)$.

$$t_d^* = argmin_{t_d} L_d(t_d)$$

List of Symbols, Abbreviations, and Acronyms

CART	Classification and Regression Tree
CNN	convolutional neural network
DL	deep learning
DT	decision tree
ELM	extreme learning machine
FF-MLP	feedforward multilayer perceptron
GBDT	gradient boosting decision tree
GMM	Gaussian mixture model
LDA	linear discriminant analysis
ML	machine learning
PV	projection vectors
RF	random forest
SLM	subspace learning machine
SLM/SLR	subspace learning machine/regressor
SVM	support vector machines

1	DEFENSE TECHNICAL
(PDF)	INFORMATION CTR
	DTIC OCA
1	DEVCOM ARL
(PDF)	FCDD RLD DCI
	TECH LIB
1	DEVCOM ARL
(PDF)	FCDD RLC NC
	V MISHRA
1	UNIVERSITY OF SOUTHERN CALIFORNIA
(PDF)	C-C J KUO