On the use of kernel functions in Minimal Learning Machines

Daniel Jean R. Vasconcelos, Ananda L. Freire and Amauri H. Souza Junior

Federal Institute of Ceará, Postgraduate Program in Computer Science
Fortaleza, Ceará, Brazil
daniel.jean@ppgcc.ifce.edu.br, anandalf@gmail.com,
amauriholanda@ifce.edu.br

Abstract. The Minimal Learning Machine is a recently proposed supervised method in which learning consists of fitting a multiresponse linear regression model between distances computed from the input and output spaces. This paper approaches the use of Minimal Learning Machines in combination with positive definite kernel functions. Particularly, we investigate if computing distances in feature spaces rather than in the original data space is beneficial for the MLM in the context of regression and classification tasks. This can be accomplished since the kernel trick allows us to calculate inner products (and consequently Euclidean distances) in feature spaces. We compare the standard MLM to its kernel variants on real-world problems.

1 Introduction

Over the last decade, kernel methods have become part of the state-of-the-art in machine learning. Basically, kernel methods are used to map data into high-dimensional feature spaces in which a learning algorithm is used to find linear patterns [1]. Usually, the feature space is implicit and the learning machine is formulated only based on inner products between data items in such space, a fact known as the kernel trick. Although the choice of the kernel depends on the specific data type and the application domain, some of the most commonly used kernels are the linear kernel, the polynomial kernel and the Gaussian kernel.

The Minimal Learning Machine (MLM, [2]) is a supervised learning algorithm based on distances. Recently, the MLM has gained attention due to its simple and easy implementation, additionally requiring the adjustment of only a single hyperparameter (the number of reference points). Learning in MLM consists in building a linear map between input and output distance matrices. In the test phase, the learned map is used to provide an estimate for the distances from the reference points outputs to the unknown target output value. Such estimates are then used to provide the actual output prediction, formulated as an optimization problem also known as multilateration.

This paper investigates the use of kernel functions in Minimal Learning Machines. More specifically, we are interested in analyzing the behavior of Minimal

Learning Machines when distances are computed in features spaces, defined implicitly by the kernel functions. This can be accomplished because the Minimal Learning machine is formulated using only pairwise distances so that they can be easily computed in feature spaces via the kernel trick.

Our empirical assessment is carried out through a comprehensive evaluation on 22 datasets, including both classification and regression tasks. Based on the experiments, we verify that the kernel functions have only a marginal impact on the performance of the regular MLM algorithm.

The remainder of the paper is organized as follows. Section 2 presents the formulation of the Minimal Learning Machine. Section 3 describes the use of MLMs in feature spaces, also providing an interpretation of the standard MLM as a kernel method. In Section 4, an experimental evaluation is conducted to show the performance of the MLM and its kernel-based variants. Conclusions are given in Section 5.

2 Minimal Learning Machine

The Minimal Learning Machine (MLM, [2, 3]) is a supervised method whose training step consists of fitting a multiresponse linear regression model between distances computed from the input and output spaces. Output prediction for new incoming inputs is achieved by estimating distances in the output spaces using the underlying linear model, followed by a search/optimization procedure in the space of possible outputs.

Let us define the learning problem as the problem of approximating a smooth continuous *target function* $f : \mathcal{X} \to \mathcal{Y}$ from (possibly) corrupted data $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i = f(\mathbf{x}_i) + \boldsymbol{\epsilon}_i)\}_{i=1}^{n}$, where $\mathbf{x}_i \in \mathcal{X}$ and $\mathbf{y}_i \in \mathcal{Y}$, and $\boldsymbol{\epsilon}_i$ corresponds to an error term to account for situations where the outputs are not perfectly observed. We call $\mathcal{X}$ and $\mathcal{Y}$ the input and output spaces, respectively.

Definition 1 (Minimal Learning Machine, [2]) *The Minimal Learning Machine denotes a class of functions $\mathcal{H}_{MLM}$ parametrized by a matrix $\mathbf{B} \in \mathbb{R}^{m \times m}$, and m pairs of input-output points $\{(\mathbf{r}_i, \mathbf{t}_i)\}_{i=1}^{m} \subseteq \mathcal{D}$, called reference points, such that a member of $\mathcal{H}_{MLM}$ is a function $h : \mathcal{X} \to \mathcal{Y}$ given by*

$$h(\mathbf{x}) = \underset{\mathbf{y}}{argmin} \sum_{j=1}^{m} \left[\|\mathbf{y} - \mathbf{t}_j\|^2 - \left(\sum_{i=1}^{m} B_{i,j} \|\mathbf{x} - \mathbf{r}_i\| \right)^2 \right]^2, \tag{1}$$

where $\|\mathbf{y} - \mathbf{t}_j\|$ represents the Euclidean distance between $\mathbf{y}$ and the j-th output reference point $\mathbf{t}_j$; similarly, $\|\mathbf{x} - \mathbf{r}_i\|$ denotes the Euclidean distance between $\mathbf{x}$ and the i-th input reference point $\mathbf{r}_i$.

Basically, the MLM assumes that distances computed in the output space can be approximated by a linear combination of distances taken from fixed located points (the reference points) in the input space. The Eq. (1) can be used to illustrate the main idea behind the MLM. The term $\sum_{i=1}^{m} B_{i,j} \|\mathbf{x} - \mathbf{r}_i\|$ consists

of a linear combination of distances taken from reference points $\mathbf{r}_i$, and represents an estimate to the actual distance $\|\mathbf{y}-\mathbf{t}_j\|$. The MLM then searches for the best location of such a point $\mathbf{y}$ considering the distances estimates for all the output reference points.

Before solving the optimization problem described in Eq. (1), we need to estimate the MLM parameters from training data, also known as the learning/training phase. The learning algorithm of the Minimal Learning Machine requires the *i*) selection of the set reference points $\{(\mathbf{r}_i, \mathbf{t}_i)\}$; and *ii*) determination of the parameters $\mathbf{B}$. With regard to the selection of reference points, in the original proposal, the MLM selects the reference points randomly from the available data points for learning, an approach also adopted in this work. The idea of matching distances in the input and output spaces drives the learning process of the parameters $\hat{\mathbf{B}}$.

In order to simplify notation, let us define the matrix $\mathbf{D} \in \mathbb{R}^{n \times m}$ as the pairwise distance matrix between the input samples $\{\mathbf{x}_i\}_{i=1}^{n}$ and the input reference points $\{\mathbf{r}_i\}_{i=1}^{m}$, i.e., $D_{ij} = \|\mathbf{x}_i - \mathbf{r}_j\|$. Similarly, $\boldsymbol{\Delta} \in \mathbb{R}^{n \times m}$ denotes the pairwise distance matrix in output space, that is $\Delta_{ij} = \|\mathbf{y}_i - \mathbf{t}_j\|$.

Once the reference points have been selected, the matrix $\mathbf{B}$ is estimated through the minimization of the following objective

$$J(\mathbf{B}) = \mathrm{Tr}\left[(\boldsymbol{\Delta} - \mathbf{D}\mathbf{B})^T(\boldsymbol{\Delta} - \mathbf{D}\mathbf{B})\right], \tag{2}$$

in which the optimal solution is achieved at $\hat{\mathbf{B}} = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T\boldsymbol{\Delta}$. This corresponds to the least-squares solution of a multiresponse linear regression problem between the distance matrices in the input and output space.

The test step in MLMs consists of predicting the outputs for new input data, and it refers to solving the minimization problem embedded in Eq. (1), that can also be interpreted as a multilateration problem [4]. In this regard, one may use any gradient-based optimization method, such as the Levenberg-Marquardt algorithm. Alternatively, [5] reports an efficient method to solve the multilateration problem for one-dimensional outputs.

For classification tasks, where outputs are represented using the 1-of-S encoding scheme[1], with S denoting the number of classes, [5] shows that, under the assumption that the classes are balanced, the minimization problem in Eq. (1) reduces to

$$h(\mathbf{x}) = \mathbf{t}_{m^*}, \tag{3}$$

where

$$m^* = \underset{j=1,\ldots,m}{\mathrm{argmin}} \left\{ \sum_{i=1}^{m} B_{i,j} \|\mathbf{x} - \mathbf{r}_i\| \right\}. \tag{4}$$

[1] A S-level qualitative variable is represented by a vector of S binary variables or bits, only one of which is *on* at a time. Thus, the j-th component of an output vector $\mathbf{y}$ is set to 1 if it belongs to class j and 0 otherwise.

As aforementioned, the term $\sum_{i=1}^{m} B_{i,j}\|\mathbf{x} - \mathbf{r}_i\|$ denotes an estimate for the distance between the j-th output reference point and the unknown target value $\mathbf{y}$ associated with $\mathbf{x}$. Thus, the output predictions for new incoming data can be carried out by simply selecting the output of the nearest reference point in the output space, estimated using the linear model $\hat{\mathbf{B}}$ [5]. The reader is referred to [2, 5, 3] for a comprehensive discussion about the MLM.

3 Minimal Learning Machines in Feature Spaces

We now turn our attention into the formulation of Minimal Learning Machines in feature spaces induced by kernel functions. For that, we need to define what we mean by kernel functions.

Definition 2 (Positive definite kernel) *A symmetric function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ which for all $m \in \mathbb{N}$, $c_i \in \mathbb{R}$, and $\mathbf{x}_i \in \mathcal{X}$, such that*

$$\sum_{i=1}^{m}\sum_{j=1}^{m} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0, \tag{5}$$

is a positive definite kernel.

The application of a positive definite kernel function on a sample from $\mathcal{X}$ induces a positive definite Gram matrix $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. Moreover, a well-known result on kernel methods [6, 7] is that we can construct kernel functions by defining arbitrary maps $\boldsymbol{\phi} : \mathcal{X} \rightarrow \mathcal{R}^{\mathcal{X}}$ and choosing

$$k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i)^T \boldsymbol{\phi}(\mathbf{x}_j) \tag{6}$$

for any $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$. In other words, kernels can be interpreted as inner products on feature spaces. This is also known as the kernel trick. Such a trick allows us to work in the feature space implicitly, i.e., a feature space in which the dot products can be evaluated directly using a nonlinear function in the input space.

A few examples of kernel functions are: Gaussian (Eq. 7), polynomial (Eq. 8) and quadratic, which is a special case of polynomial kernel, where its polynomial degree is set to $p = 2$. In this work, we assume a_1 and a_2 to be equal to 1.

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right). \tag{7}$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = (a_1 \mathbf{x}_i^T \mathbf{x}_j + a_2)^p. \tag{8}$$

The extension of the MLM to work in feature spaces is straightforward. This is because we can easily write distances as a function of inner products, i.e, for any arbitrary pair of point $\mathbf{x}_i$, $\mathbf{x}_j \in \mathcal{X}$ we have that

$$\|\mathbf{x}_i - \mathbf{x}_j\| = \sqrt{\mathbf{x}_i^T \mathbf{x}_i - 2\mathbf{x}_i^T \mathbf{x}_j + \mathbf{x}_j^T \mathbf{x}_j}, \tag{9}$$

and thus we can make use of the kernel trick to compute distances in feature spaces by

$$\|\boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{\phi}(\mathbf{x}_j)\| = \sqrt{\boldsymbol{\phi}(\mathbf{x}_i)^T\boldsymbol{\phi}(\mathbf{x}_i) - 2\boldsymbol{\phi}(\mathbf{x}_i)^T\boldsymbol{\phi}(\mathbf{x}_j) + \boldsymbol{\phi}(\mathbf{x}_j)^T\boldsymbol{\phi}(\mathbf{x}_j)}, \tag{10}$$

$$= \sqrt{k(\mathbf{x}_i, \mathbf{x}_i) - 2k(\mathbf{x}_i, \mathbf{x}_j) + k(\mathbf{x}_j, \mathbf{x}_j)}, \tag{11}$$

where $k(\cdot, \cdot)$ denotes a positive kernel.

Thus, the functional form of MLMs in feature space induced by the kernel $k(\cdot, \cdot)$ is given by

$$h_k(\mathbf{x}) = \underset{\mathbf{y}}{\operatorname{argmin}} \sum_{j=1}^{m} \left[\|\mathbf{y} - \mathbf{t}_j\|^2 - \left(\sum_{i=1}^{m} B_{i,j} (k(\mathbf{x}, \mathbf{x}) - 2k(\mathbf{x}, \mathbf{r}_i) + k(\mathbf{r}_i, \mathbf{r}_i))^{\frac{1}{2}} \right)^2 \right]^2 . \tag{12}$$

The kernel functions also affect the MLM training phase in a very simple manner. To train MLMs, we replace the input distance matrix $\mathbf{D}$ by $\boldsymbol{\Phi} \in \mathbb{R}^{n \times m}$ such that its (i, j)-entry is

$$\Phi_{i,j} = \sqrt{k(\mathbf{x}_i, \mathbf{x}_i) - 2k(\mathbf{x}_i, \mathbf{r}_j) + k(\mathbf{r}_j, \mathbf{r}_j)}, \tag{13}$$

and, consequently, the optimal estimate of the parameters can be written as $\hat{\mathbf{B}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}\boldsymbol{\Delta}$. The Algorithm 1 summarizes the learning process of the MLM using kernels.

Algorithm 1 Kernel MLM - training procedure

Input: Training datasets $\mathcal{X}_n$ and $\mathcal{Y}_n$, and m.
Output: $\hat{\mathbf{B}}$, $\mathcal{R}$ and $\mathcal{T}$.

1. Randomly select m reference points, $\mathcal{R}$, from $\mathcal{X}_n$ and their corresponding outputs, $\mathcal{T}$, from $\mathcal{Y}_n$;
2. Compute $\boldsymbol{\Phi}$ according to Eq. (13): The distance matrix between the elements of $\mathcal{X}_n$ and $\mathcal{R}$;
3. Compute $\boldsymbol{\Delta}$: The distance matrix between $\mathcal{Y}_n$ and $\mathcal{T}$;
4. Calculate $\hat{\mathbf{B}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{\Delta}$.

In this work, we decided to restrict the use of kernel methods to the input space, thus the geometrical interpretation of the output estimation step is kept, and only the training phase significantly changes.

3.1 Minimal Learning Machines and conditionally positive kernels

There is a larger class of kernel functions, named conditionally positive definite (cpd) kernels, which can be defined similar to positive kernels, with the difference being that Eq. (5) only needs to be satisfied for values of c_i such that $\sum_i c_i = 0$.

The importance of conditionally positive kernels in this work comes from the fact that the function given by the negative of the Euclidean distance is a cpd kernel, i.e., kernels functions of the form $k(\mathbf{x}_i, \mathbf{x}_j) = -\|\mathbf{x}_i - \mathbf{x}_j\|$ are cpd.

In fact, the MLM can be interpreted as a cpd kernel method, even though the MLM uses proper distances rather than its negative values. We argue that the MLM algorithm is not affected if all computed distances are replaced by negative Euclidean distances. To show that, let us first consider the MLM function given in Eq. (1). As one can easily observe, setting the distances to their negative values does not change anything because all the terms are squared. Second, the cost function in Eq. (2) does not change as well, since $(-\boldsymbol{\Delta} + \mathbf{DB})^T(-\boldsymbol{\Delta} + \mathbf{DB}) = (\boldsymbol{\Delta} - \mathbf{DB})^T(\boldsymbol{\Delta} - \mathbf{DB})$. Therefore, the optimal solution for the parameters $\hat{\mathbf{B}}$ remains the same.

Differently from other kernel methods proposed in the literature where kernels are applied to input and output spaces [8, 9], in the MLM, model complexity is controlled by the number of reference points rather than a regularization term (ridge regression). In addition, the preimage problem in kernel methods reduces to the multilateration problem, but it has a geometric interpretation in the MLM.

4 Experiments

The experiments were performed on classification and regression problems. We used 14 and 8 datasets, respectively, taken from UCI database [10] and StatLib [11]. In Tab. 1, the details of each dataset are described, such as problem category, the number of attributes, classes, and samples for training and test.

The code was implemented in Matlab R2014a and the tests were executed on a Intel Core i3-4130 processor CPU 3.4GHz with 8GB of RAM. The MLM and kernel implementations are available in goo.gl/wUvax3.

To assess the performance of MLM with different kernel functions, for both classification and regression, we executed 30 independent runs, where 80% of the data were used for training and the remaining 20% for testing. For each run, some parameters needed to be tuned, e.g., the percentage of reference points m, the spread of the Gaussian kernel (σ) and the order of the polynomial kernel (p). Those parameters went through a 10-fold cross-validation in a grid search to find the most suitable values.

For m, representing the percentage values, a grid from 0.1 to 1 (equivalent to 10% to 100%) with steps of 0.1 was designed. For each number, a linear kernel MLM was trained and validated by the cross-validation process. The term m is then chosen based on the best mean result (accuracy or RMSE). Once the percentage of reference points is defined, the points themselves are chosen randomly but kept the same to the other tested MLMs. The parameters $\sigma \in [5, 55]$, with steps of 5, and $p \in [3, 10]$, with steps of 1, are chosen after a similar process, but now using the Gaussian/Polynomial kernel, respectively, plus the reference points previously determined. With those parameters defined, we trained and tested MLMs with linear, Gaussian, quadratic and polynomial kernels.

Table 1: Description of classification (C) and regression (R) datasets.

Data	Problem	#Attributes	#Classes	#Train	#Test
Balance Scale (BS)	C	4	3	500	125
Breast Cancer Diagnostic (BCD)	C	30	2	455	114
Diabetes (DIA)	C	8	2	614	154
Ecoli (EC)	C	7	8	269	67
Hayes-Roth (HR)	C	3	3	128	32
Heart Disease (HD)	C	13	2	216	54
Iris (IRS)	C	4	3	120	30
Leaf Classification (LC)	C	14	30	272	68
Liver Disorders (LD)	C	6	2	276	69
MONK's Problem 1 (M1)	C	6	2	445	111
MONK's Problem 2 (M2)	C	6	2	481	120
MONK's Problem 3 (M3)	C	6	2	443	111
Connectionist Bench Sonar (SNR)	C	60	2	166	42
Wine Quality (WQ)	C	13	3	142	36
Auto MPG (MPG)	R	7	-	314	78
Body Fat (BF)	R	14	-	202	50
Boston Housing Corrected (BH)	R	18	-	405	101
Breast Cancer Prognostic (BCP)	R	33	-	155	39
Computer Hardware (CPU)	R	9	-	167	42
Forest Fires (FF)	R	4	-	414	103
Servo (SRV)	R	4	-	134	33
Red Wine Quality (RWQ)	R	11	-	1279	320

Tab. 2 and 3 provides the mean and standard deviation from the 30 runs for m, σ, p and accuracy (for classification) or RMSE (for regression). Moreover, we applied Friedman's nonparametric hypothesis test [12] with a significance level of $\alpha = 0.05$, followed by a Dunn-Sydák's post hoc test. We compared the performance of each nonlinear kernels to the linear one. Considering the null hypothesis to be that the different MLMs performances are all the same, if the hypothesis test result fails to reject it, we show a ✓, otherwise, we show an ✗.

For classification problems (Tab. 2) with nonlinear kernels, most of the results presented an "equivalent" performance to the linear one. As exceptions, the Gaussian kernel has four out of fourteen sets with nonequivalent results distributions, the quadratic kernel has two, and the polynomial kernel has seven of them. Nevertheless, within this exception pool, the results show that the linear kernel provides a slightly better accuracy overall.

For regression problems (Tab. 3), the nonlinear kernels, applied to datasets MPG and BH, provided nonequivalent results to the linear kernel. In those cases, also in four others, the linear kernel provided a slightly better average performance.

It is important to highlight that the Gaussian kernel has proved itself to be very sensitive to the choice of its parameter σ. This feature imposes to cover a

Table 2: Results for classification problems using the same reference points.

Datasets	m		Linear	Gaussian	Quadratic	Polinomial
BS	0.45 ± 0.10	accuracy	0.903 ± 0.024	0.903 ± 0.023	0.897 ± 0.026	0.898 ± 0.025
		parameters	-	31.167 ± 13.817	-	3.067 ± 0.254
			-	✓	✓	✓
BCD	0.52 ± 0.24	accuracy	0.939 ± 0.015	0.932 ± 0.018	0.934 ± 0.019	0.933 ± 0.014
		parameters	-	49.833 ± 1.729	-	3.200 ± 0.610
			-	✗	✓	✗
DIA	0.11 ± 0.03	accuracy	0.750 ± 0.031	0.735 ± 0.036	0.753 ± 0.035	0.740 ± 0.036
		parameters	-	45.000 ± 5.776	-	3.800 ± 1.186
			-	✗	✓	✗
EC	0.33 ± 0.16	accuracy	0.841 ± 0.040	0.844 ± 0.035	0.836 ± 0.036	0.830 ± 0.039
		parameters	-	26.500 ± 16.049	-	4.767 ± 1.612
			-	✓	✓	✓
HR	0.87 ± 0.18	accuracy	0.855 ± 0.064	0.859 ± 0.051	0.858 ± 0.051	0.841 ± 0.069
		parameters	-	43.167 ± 9.072	-	4.200 ± 2.172
			-	✓	✓	✓
HD	0.93 ± 0.10	accuracy	0.696 ± 0.059	0.672 ± 0.046	0.690 ± 0.046	0.703 ± 0.040
		parameters	-	11.333 ± 10.178	-	3.700 ± 1.418
			-	✗	✓	✓
IRS	0.65 ± 0.20	accuracy	0.950 ± 0.040	0.951 ± 0.037	0.952 ± 0.037	0.951 ± 0.038
		parameters	-	45.833 ± 7.535	-	3.100 ± 0.403
			-	✓	✓	✓
LC	0.68 ± 0.21	accuracy	0.659 ± 0.069	0.660 ± 0.067	0.648 ± 0.063	0.643 ± 0.059
		parameters	-	30.167 ± 15.082	-	3.600 ± 1.037
			-	✓	✗	✗
LD	0.99 ± 0.03	accuracy	0.716 ± 0.046	0.706 ± 0.046	0.702 ± 0.046	0.700 ± 0.055
		parameters	-	6.667 ± 5.522	-	3.000 ± 0.000
			-	✓	✓	✓
M1	0.12 ± 0.04	accuracy	0.977 ± 0.027	0.977 ± 0.027	0.974 ± 0.025	0.965 ± 0.026
		parameters	-	37.000 ± 9.391	-	3.667 ± 1.124
			-	✓	✓	✗
M2	1.00 ± 0.00	accuracy	0.824 ± 0.033	0.819 ± 0.035	0.793 ± 0.030	0.774 ± 0.031
		parameters	-	5.500 ± 0.000	-	3.000 ± 0.000
			-	✓	✗	✗
M3	1.00 ± 0.00	accuracy	0.975 ± 0.012	0.978 ± 0.012	0.975 ± 0.012	0.975 ± 0.012
		parameters	-	32.167 ± 24.507	-	5.100 ± 3.263
			-	✓	✓	✓
SNR	0.78 ± 0.16	accuracy	0.868 ± 0.038	0.869 ± 0.040	0.864 ± 0.042	0.857 ± 0.044
		parameters	-	23.000 ± 15.852	-	5.367 ± 2.205
			-	✓	✓	✗
WQ	0.99 ± 0.04	accuracy	0.797 ± 0.072	0.777 ± 0.078	0.781 ± 0.082	0.778 ± 0.080
		parameters	-	7.333 ± 7.008	-	3.300 ± 0.877
			-	✗	✓	✗

large grid when looking for the best parameter value. In contrast, the polynomial kernel's order had an average of $p = 3$ or $p = 4$ in the majority of the cases.

In other experiments that we executed, we found that normalizing the data does not improve the accuracy or RMSE for nonlinear kernels, although it increases the number of equivalent results for the Gaussian kernel.

Table 3: Results for regression problems using the same reference points.

Datasets	m		Linear	Gaussian	Quadratic	Polinomial
MPG	0.89 ± 0.15	RMSE	3.960 ± 0.407	4.296 ± 0.516	4.182 ± 0.527	4.248 ± 0.554
		parameters	-	50.000 ± 1.526	-	3.133 ± 0.434
			-	✗	✗	✗
BF	0.51 ± 0.21	RMSE	5.165 ± 0.418	5.215 ± 0.474	5.272 ± 0.474	5.327 ± 0.441
		parameters	-	45.333 ± 6.363	-	3.067 ± 0.254
			-	✓	✓	✗
BH	1.00 ± 0.00	RMSE	0.016 ± 0.002	0.047 ± 0.004	0.018 ± 0.002	0.019 ± 0.002
		parameters	-	50.500 ± 0.000	-	3.000 ± 0.000
			-	✗	✗	✗
BCP	0.98 ± 0.05	RMSE	16.159 ± 2.488	25.304 ± 3.060	17.749 ± 3.113	19.974 ± 3.201
		parameters	-	50.500 ± 0.000	-	3.000 ± 0.000
			-	✗	✓	✗
CPU	0.70 ± 0.25	RMSE	57.885 ± 33.521	152.252 ± 70.371	51.857 ± 29.467	51.855 ± 27.611
		parameters	-	49.000 ± 3.511	-	3.933 ± 1.202
			-	✗	✓	✓
FF	0.32 ± 0.25	RMSE	57.097 ± 33.692	54.039 ± 34.500	57.897 ± 33.034	55.974 ± 33.922
		parameters	-	19.000 ± 21.015	-	5.067 ± 1.893
			-	✓	✓	✓
SRV	0.96 ± 0.08	RMSE	0.676 ± 0.268	0.677 ± 0.269	0.686 ± 0.276	0.680 ± 0.295
		parameters	-	41.333 ± 15.707	-	3.100 ± 0.305
			-	✓	✓	✓
RWQ	1.00 ± 0.02	RMSE	0.702 ± 0.024	0.704 ± 0.024	0.709 ± 0.025	0.714 ± 0.026
		parameters	-	49.500 ± 4.026	-	3.033 ± 0.183
			-	✓	✓	✗

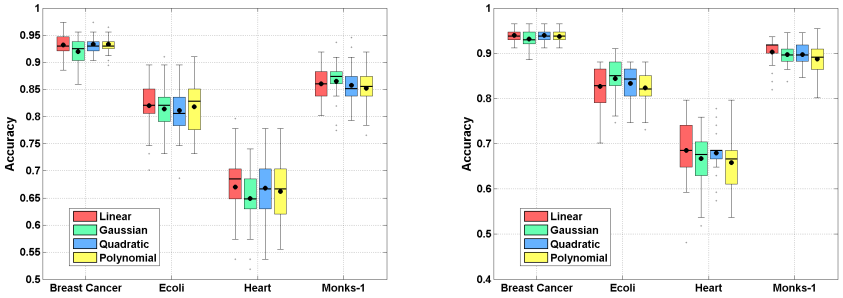
One may argue that the choice of m value in Tab. 2 and 3 benefits the linear kernel results, since it is used to define this value. So, to discard such possibility and also verify if nonlinear kernels thrive with smaller percentages, we performed another set of tests, where m is fixed in 10% or 20%. The reference points, in this case, are chosen randomly for each tested MLM.

The results are presented in Fig. 1 and 2. The datasets used here were solely chosen because of the similar range of their results, which improves their visualizations.

From Fig. 1a and 2a, we observe a decrease of performance with all kernels, due to insufficient references points. Besides that, in average, the results are still similar to the ones presented in Tab. 2 and 3 for both m values. Only for Ecoli dataset and $m = 20\%$, the Gaussian kernel provided a slight improvement.

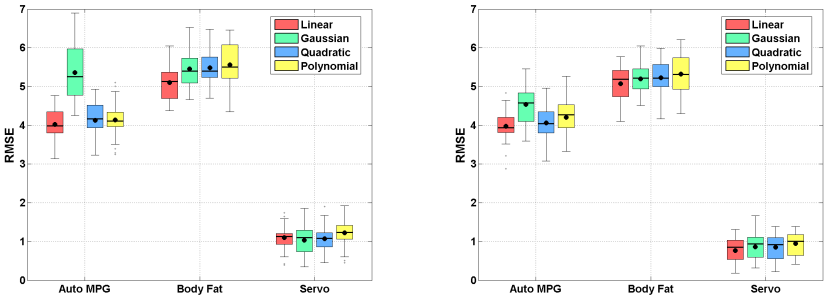
5 Conclusions

This paper presented an extensive comparison between linear, Gaussian, quadratic and polynomial kernels applied to the MLM with 22 datasets of classification and regression problems. From the first results in Tab. 2 and 3, one can notice that, in most of the cases, the average performance between them is similar which is also confirmed by the Friedman's statistical test. The next set of tests evaluated



(a) $m = 10\%$ (b) $m = 20\%$

Fig. 1: MLM with different kernels and fixed m value for classification problems.



(a) $m = 10\%$ (b) $m = 20\%$

Fig. 2: MLM with different kernels and fixed m value for regression problems.

their performance when the percentage of reference points is fixed and small. We then verified that a very small percentage of points diminishes the performance but maintain similar average results among the networks.

From those results, we can argue that the regular MLM (linear kernel) has the best trade-off features since it provides good results and does not have any other hyperparameter to be found. In contrast, the Gaussian kernel makes the MLM performance very sensitive to the choice of σ and did not provide better results than the other tested kernels. Thus, based on the results for the datasets considered here, computing distances in feature spaces does not improve over the regular MLM. The fact that the MLM is already a cpd kernel method may provide an explanation for that.

Acknowledgments

The authors acknowledge the support of CAPES.

References

1. J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.
2. A. H. Souza Junior, F. Corona, G. A. Barreto, Y. Miche, and A. Lendasse. Minimal learning machine: A novel supervised distance-based approach for regression and classification. *Neurocomputing*, 164:34–44, 2015.
3. A.H. Souza Junior, F. Corona, Y. Miché, A. Lendasse, G. Barreto, and O. Simula. Minimal learning machine: A new distance-based method for supervised learning. In *Proceedings of the 12th International Work Conference on Artificial Neural Networks (IWANN'2013)*, volume 7902, pages 408–416, 2013.
4. E. Niewiadomska-Szynkiewicz and M. Marks. Optimization schemes for wireless sensor network localization. *International Journal of Applied Mathematics and Computer Science*, 19:291–302, 2009.
5. Diego P. P. Mesquita, João P. P. Gomes, and Amauri H. Souza Junior. Ensemble of efficient minimal learning machines for classification and regression. *Neural Processing Letters*, 2017.
6. Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. Kernel methods in machine learning. *Annals of Statistics*, 36(3):1171–1220, 2008.
7. Bernhard Schölkopf. The kernel trick for distances. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 301–307. MIT Press, 2001.
8. Jason Weston, Olivier Chapelle, André Elisseeff, Bernhard Schölkopf, and Vladimir Vapnik. Kernel Dependency Estimation. In Suzanna Becker, Sebastian Thrun, Klaus Obermayer, Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *NIPS*, pages 873–880. MIT Press, 2002.
9. Corinna Cortes, Mehryar Mohri, and Jason Weston. A general regression technique for learning transductions. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 153–160, 2005.
10. M. Lichman. Uci machine learning repository. Irvine, CA: University of California, School of Information and Computer Science, 2013.
11. M Meyer and P. Vlachos. Statlib: Data, software and news from the statistics community, 1989.
12. J. Denšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.