Similarity Domains Machine for Scale-invariant and Sparse Shape Modeling

Sedat Ozer

Abstract—We present an approach to extend the functionality and the use of kernel machines in image processing applications: we introduce a novel way to design spatial kernel machines with spatial properties and demonstrate how those newly introduced spatial properties enhance the possibilities of the use of kernel machines in image processing applications as a proof of concept. In this paper, we demonstrate four particular extensions: (1) how to model shapes efficiently with spatially computed kernel parameters in a geometrically scalable way; (2) how to visualize the kernel parameters precisely and intuitively on binary 2D shapes; (3) how to construct a one-class classifier from the binary classifier in a straightforward manner without re-training and (4) how to use the computed kernel parameters for filtering. The existing literature on kernel machines mostly focuses on estimating the optimal kernel parameters via *additional* cost function(s). In this work, instead of employing an additional cost function to estimate the kernel-related parameters, we investigate on an analytical solution to predict the actual kernel parameters locally and show how to build a spatial kernel machine with our analytical approach. Classical kernel machines do not perform well on precise shape modeling with low number of support vectors as demonstrated in this paper. However, we demonstrate and visualize that our analytical approach provides a natural means to relate the kernel parameters to the 2D shapes for sparse shape modeling (where the shape boundary represents the decision boundary). For that, we incorporate the selected kernel function's geometric properties as an additional constraint into the classifier's optimization problem by defining an easy-to-explain and an intuitive concept: similarity domains. In our experiments, we study and demonstrate how the resulting new kernel machine enhances the capabilities of the classical kernel machines with applications on shape modeling, (geometrically) scaling the non-linear decision boundary at various scales and precise visualization of the kernel parameters in 2D images.

Index Terms—Scalable kernel machine, local scale estimation, visualization of kernel parameters, explainable kernel machine, sparse spatial kernel machine.

1 Introduction

Kernel machines have long been used in image processing applications and they are provided with well-established theories and implementations. A particular existing challenge, however, is explaining the computed kernel parameters to non-expert users in an intuitive way. There is not much work available relating the kernel parameters to more common and understandable concepts for non-expert users. Furthermore, while kernel machines have been applied to many applications in the image-processing literature, the most predominant use of them has been the prediction of the image labels from sets of image datasets with many instances. Recent examples can be found in [3], [4], [9], [15], [21], [25], [26], [35], [38]. However, there is not much work done to learn shapes with kernel machines from a given *single* image data. In this work, our goal is to tackle with the above-mentioned problems and extend the possible use cases of kernel machines beyond their common use in image classification and categorization applications as in the above-given examples, while providing an easy to explain algorithm for non-experts. With those goals in mind, we propose a novel approach to relate the kernel parameters to the local properties of the data in a more intuitive way as opposed to the existing solutions focusing on computing the kernel parameters with additional cost functions. In our approach, we relate the kernel parameters to the local data properties analytically and demonstrate how our resulting new kernel machine enhances the capability of the classical kernel machines in a more intuitive and easy to explain way. Our experimental results show that our presented kernel machine yields significantly less number of support vectors when compared to the classical support vector machine (SVM) algorithm in almost all our experiments and yields similar or better accuracy than that of SVM. However, while we include accuracy comparisons to other classifiers on several benchmarks, our main motivation in this paper is beyond comparing only the classification accuracies: extending the functionality of kernel machines with new applications and demonstrating them. Consider Fig. 1. It is not trivial to use the learned parameters of any kernel machine implementation to obtain the image shown in Fig. 1e from the image shown in Fig. 1a via linear or straightforward operations. However, our proposed kernel machine can learn the foreground class (the white area in the image) in Fig. 1a by computing its own geometric kernel parameters and their numbers automatically. Since the kernel parameters have geometric meaning, they can now be used to extract each region of interest (objects) in the image individually. Furthermore, they also provide a mean to shift and (geometrically) scale those objects individually.

Recent literature reports that employing multiple kernels can yield higher performance when compared to using a single kernel with predetermined kernel parameters in kernel machines. Among those, while the earlier works studied this problem by assigning different kernel parameters to each training sample (each support vector) and by directly estimating the kernel parameters [8], [10], recent papers focused on this problem under the multiple kernel

• *This is author's version. Please download the final version from http://ieeexplore.ieee.org. DOI 10.1109/TIP.2018.2868380*
E-mail: see http://www.sedatozer.com

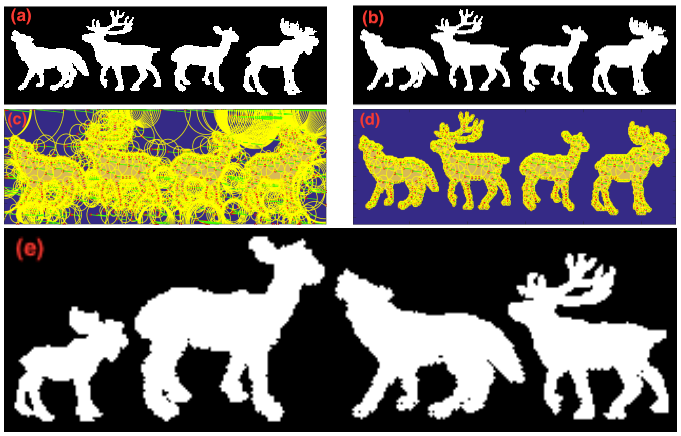


Fig. 1: Visualization of the kernel parameters of similarity domains machine (SDM) on the original binary image and use of them. **(a)** Original image: 280x85 pixels. **(b)** Trained SDM result with total of 1 pixel-error at T=0.1. **(c)** Visualization of all the computed kernel parameters. **(d)** Visualization of only the foreground kernel parameters. **(e)** Altered image with scaled and shifted objects. A region growing algorithm is applied on the foreground kernel parameters to extract each object. Then each object is individually scaled and shifted based on their foreground parameters. Finally, one-class approximation of SDM is applied on each of those scaled and shifted shapes to obtain the final altered image.

learning (MKL) framework [1], [7], [19]. MKL model, assumes that there is an available set of basis kernel functions provided by the user and that the optimal kernel is a (linear) combination of such basis kernel functions. The MKL framework includes both the estimation of the (linear) model parameters of MKL and the estimation of the parameters of the base learning algorithm such as support vector machine (SVM) [13], [14], [17], [34], [36]. Both direct-estimation and MKL-based techniques typically include at least one *additional* optimization step for the estimation of the optimal kernel parameters. In MKL-based techniques, the assumption is that all the basis kernel functions and their total number are already known or given by the user. The value of their final kernel estimate and the optimal kernel weights change based on the included (or user-given) set of basis kernel functions and their kernel parameters. In all such approaches, the answer for the question *"can we unify these individual optimization steps?"* remains a challenge.

In this paper, we present a geometry-based solution to the above-mentioned problems: we unify both the estimation of the kernel parameters and the hyper-parameters of the classifier in a single optimization problem by relating the kernel parameters to local data properties that are intuitive to explain for even non-experts. Furthermore, once the parameters and support vectors are learned by our binary classifier, we show that now it becomes possible to define a one-class classifier from the computed parameters of our binary classifier by using only *a subset of the computed support vectors* due to the geometric properties of our approach. As opposed to forming a dedicated cost function for the kernel parameters or for the kernel weights (as in the MKL model), we force the base algorithm to use the local relations between the data samples as a constraint (in this paper, we used Gaussian kernel and its properties to define such local relations between the data points).

We introduce the concept of similarity domains and use a kernel function to define similarity domains geometrically in the feature space. The concept of the similarity domains allows us to define the above-mentioned unified optimization problem with a single cost function in which both the kernel parameters and the hyper-parameters of a kernel machine are estimated automatically. This is also the first work that defines and computes the kernel parameters geometrically in kernel machines and relates them to the shapes intuitively. Due to the geometric nature of our algorithm, we demonstrate that the decision boundary becomes scale-invariant, i.e., the decision boundary (which can represent a shape) can be up- or down-scaled at a given scale while preserving the ratio between the SVs and the computed kernel parameters. This property is related to the domain adaptation problems and solves a sub-class of domain adaptation problems in kernel machines where the new domain is a scaled version of the original one. We demonstrate this property on scalable shapes in this paper. Our presented algorithm utilizes both the global and local properties of the training data. While the global properties are summarized by the decision function of SVM, the local properties are summarized by the kernel parameters. We call our approach *similarity domains machine* (SDM). Our presented approach in this paper will benefit many other learning algorithms and their applications. Our experimental results show that our algorithm's performance can reach to that of the SVM algorithm and that our algorithm yields lower number of support vectors almost in every experiment.

In this paper, our focus is introducing new spatial capabilities to kernel machines rather than focusing on the accuracy improvements on multi-image categorization applications. Our main contributions include introduction of a novel technique to:

- Construct a novel spatial and scalable kernel machine with explainable kernel parameters;
- Model shapes in terms of the spatially defined kernel parameters;
- Reduce a binary classifier into a one-class classifier without re-training;
- Visualize the kernel parameters on shapes.

2 Related Work

Our algorithm is similar to various lines of works. While our algorithm is also similar to radial basis function (RBF) networks, it differs from RBF networks in many ways. A comparison between the radial basis function (RBF) networks and a large margin classifier (SVM) is already available in the literature in [30] and most of those differences are also valid for our work, since our algorithm is also derived from the large margin idea. In this paper, we will focus on comparisons to the kernel machines and provide a summary of the related work from the kernel machines perspective. We will first compare our algorithm to other kernel machines from the parameter computation perspective, then we will compare our algorithm to domain adaptation related works.

Kernel parameter computation: Recent advances in kernel-machines demonstrate that employing multiple kernels can yield higher performance than using a single (and in many practical applications hand picked) kernel parameter in kernel machines. The earlier related works in kernel parameter computation such as the ones reported by Chapelle et al. [8] and Bennett et al. [2] focused on involving an *additional* numerical optimization step for finding the optimal kernel parameters. Later on, related work mostly evolved around the MKL framework [19]. The main goal in MKL framework is estimating the optimal kernel function that

could give the highest accuracy as a linear combination of multiple kernel functions (i.e., in terms of t different kernel functions) as in Eq. ((3)) (see next section). In that model, each basis kernel $K_r(\mathbf{x_1}, \mathbf{x_2})$ defines a different kernel function with its own set of specific kernel parameters (where $r = 1, 2, 3, ..., t$). Each weight β_r reflects the wellness or the contribution of the r^{th} basis kernel function among t available kernel functions. Notice that if the included t number of functions in a MKL model cannot span the optimal solution space, then any optimization step included in the MKL model will yield a sub-optimal solution. Various MKL techniques have been used for visual object recognition, feature selection and data fusion applications. Examples can be found in [4], [13], [33].

Existing related similar techniques focus on using an additional optimization step for kernel parameters as used in the above-mentioned MKL techniques and in [8]. However, in our proposed technique, instead of utilizing an additional optimization step, we compute the kernel parameters analytically by utilizing the local and spatial properties of the data. Therefore, we use only one cost function in our algorithm. For example, the work in [8] proposes a two-step minimax approach in which they first maximize the margin of SVM to find the optimal α_i, and then minimize the model selection criterion T (see [8] for definitions of T) with respect to the (kernel) parameters by using the current estimates of the optimal α_i values found in the previous step. This two-step optimization procedure repeats itself recursively until it converges. MKL algorithms also use similar two-step optimization approaches. See some examples in [16]. However, as opposed to using additional optimization steps, we simply use the maximum margin (distance) idea locally to compute the kernel parameters analytically. By using that idea, we utilize the geometric properties of the kernel function *locally* within the SVM framework instead of constructing a cost function to minimize the *global* error (risk) for kernel parameter computation. We do not use the MKL model in our algorithm.

Domain Adaptation: Existing related models for domain adaptation problems include adapting the source (original) classifier $f^S(\mathbf{x})$ (the decision boundary) to the target classifier $f^T(\mathbf{x})$ in the target domain [11], [18], [20], [37]. Many of these works focus on including data from both the source and target domains to adopt a classifier to the target domain. While domain adaptation problems in computer vision varies, the focus of the existing literature on computer vision has been the accurate image categorization in different domains. One another important yet challenging domain adaptation problem is sparse shape modeling via learning and adopting that shape to different domains at different scales from a given single image. There is not much work done for the shape modeling problem under the domain adaptation related research. In this paper, we present a novel kernel machine specifically designed to deal with the sparse shape modeling and sparse shape scaling problems. Our kernel machine reconstructs the existing classifier, $\mathbf{f}(\mathbf{x})$, (i.e., the decision boundary) in new domains in a straightforward-manner (without re-training) as a scaled version of the original classifier at a given scale. This is illustrated in Fig. 2.

Another related subject is one-shot learning as in [12]. The main idea in one-shot learning is learning from one sample by modifying pre-trained models on unrelated classes. We demonstrate that it is possible to learn and model a binary shape from only one image with our algorithm without using additional pre-trained models. We also demonstrate how to use that learned model to scale the shape at different scales. Notice that while that concept is also similar to one-shot learning, we do not incorporate, require or use any additional model pre-trained (learned) on any other dataset. In this work, we assume that the data given to us is only a single image and we learn from that single image to learn the shape model.

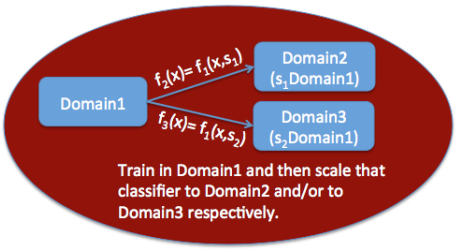


Fig. 2: This figure demonstrates a subset of the domain adaptation problems where the classifier is trained only on Domain1 and then needed to be applied in two new domains (Domain2 and Domain3) which are the scaled versions of Domain1 at different scales s_1 and s_2, respectively. The new decision function in each of those new domains can be obtained from the original decision function (trained on Domain1) as its scaled version at s_1 and s_2, respectively.

3 Preliminaries

In this section, we provide the necessary background upon which our proposed approach is built. Kernel machines estimate the class label y of the test vector $\mathbf{x}$ in terms of the set of training data $\mathbf{D}$=$\{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), ..., (\mathbf{x_n}, y_n)\}$, where $\mathbf{x_i}$ is the i^{th} training sample with the class label y_i and n the total number of training samples. The label y is estimated as $\overline{y}$ as below:

$$\overline{y} = sgn(f(\mathbf{x})) \text{ and } f(\mathbf{x}) = \sum_{i=1}^{k} \alpha_i y_i K(\mathbf{x}, \mathbf{x_i}) - b, \quad (1)$$

where the scalar α_i is a nonzero coefficient (Lagrange multiplier) for the support vector $\mathbf{x_i}$, $y_i \epsilon \{-1, +1\}$ the class label, k the total number of support vectors (SVs) and b the bias value for the hyperplane. $K(.)$ is a Mercer kernel function [32] and is defined as the inner product of two vectors in that higher dimensional space such that: $K(\mathbf{x}, \mathbf{x_i}) = < \phi(\mathbf{x}), \phi(\mathbf{x_i}) >$ where $\Phi(.)$ the mapping function from feature space into the higher dimensional space and where $< . >$ is the inner product. The training step of the classical SVM (the dual form) is formulated as follows:

$$\begin{aligned} &\max_{\alpha} Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x_i}, \mathbf{x_j}), \\ &\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{and } C \geq \alpha_i \geq 0 \end{aligned} \quad (2)$$

where C is a pre-specified upper bound (also known as the trade-off parameter) for all α_i. The training samples $\mathbf{x_i}$ with nonzero α_i are called support vectors. The kernel function $K(.)$ and its parameters are predetermined (user given) in the classical SVM [32]. MKL models the optimal kernel as a linear combination of several basis kernel functions:

$$K(\mathbf{x_1}, \mathbf{x_2}) = \sum_{r=1}^{t} \beta_r K_r(\mathbf{x_1}, \mathbf{x_2}), \quad (3)$$

where t is the total number of basis functions and β_r is the weight for the $K_r(.)$. When the MKL model given above is jointly used

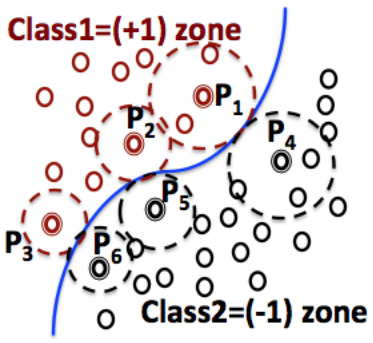


Fig. 3: An illustration of similarity domains. Red points are the samples from the (+1) class and the black points are from the (-1) class. The support vectors (SV) are shown with an additional circle around the point. The similarity domain of each SV is shown for the P_1, P_2, ..., P_6. The decision boundary (which may represent an edge in an image) is shown in blue. Notice that the similarity domain of a SV ends where it touches another SV from the other class.

with the SVM, the cost function $Q(\alpha)$ becomes $Q(\alpha, \beta)$ to be optimized with respect to both α and β:

$$\max_{\alpha,\beta} Q(\alpha,\beta) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \sum_{r=1}^{t}\beta_r K_r(\mathbf{x_i},\mathbf{x_j}),$$
$$\text{s.t. } \sum_{i=1}^{n}\alpha_i y_i = 0,\ \sum_{r=1}^{t}\beta_r = 1,\ C \geq \alpha_i \geq 0 \ and\ \beta_r \geq 0 \tag{4}$$

See [17] for an overview of MKL algorithms.

As mentioned earlier, instead of formalizing the kernel parameters within an additional optimization framework, we take an alternative approach and define them geometrically. While there are many kernel functions proposed and used in the literature (see [23], [24] for a set of sample kernel functions), in this work, we use Gaussian kernel function to define the local data properties spatially via similarity domains. The next section defines similarity domains concept to formalize the geometric properties used in our algorithm.

4 Similarity Domains

SVM is a learning algorithm that defines the decision boundary in terms of SVs; see Eq. (1). Similar to SVM, our algorithm: Similarity Domains Machine (SDM) also defines the decision boundary in terms of SVs. Furthermore, SDM defines a local similarity domain for each SV. Therefore, in this section, first we define the concept of the similarity domains for each support vector. We will then use that concept to design our novel classifier.

Similarity domains help us define a unified optimization problem in which the kernel parameters are computed automatically and geometrically. We define the similarity domain of $\mathbf{x_i} \epsilon \mathbf{R^d}$, as the ball in $\mathbf{R^d}$ where the center is the SV $\mathbf{x_i}$ and the ball radius is r_i. The r_i is defined as follows:

For any (+1) labelled support vector $\mathbf{x_i^+}$, where $\mathbf{x_i^+} \epsilon \mathbf{R^d}$ and superscript (+) represents the (+1) class:

$$r_i = min(\parallel \mathbf{x_i^+} - \mathbf{x_1^-} \parallel, ..., \parallel \mathbf{x_i^+} - \mathbf{x_k^-} \parallel)/2 \tag{5}$$

where superscript (-) means the (-1) class.

For any (-1) labelled support vector $\mathbf{x_i^-}$:

$$r_i = min(\parallel \mathbf{x_i^-} - \mathbf{x_1^+} \parallel, ..., \parallel \mathbf{x_i^-} - \mathbf{x_k^+} \parallel)/2. \tag{6}$$

In this work, we use Gaussian kernel function to represent similarities and similarity domains:

$$K_{\sigma i}(\mathbf{x}, \mathbf{x_i}) = \exp(- \parallel \mathbf{x} - \mathbf{x_i} \parallel^2 / \sigma_i^2) \tag{7}$$

where σ_i is the kernel parameter for SV $\mathbf{x_i}$. The similarity (kernel) function takes its maximum value where $\mathbf{x} = \mathbf{x_i}$. The relation between r_i and σ_i is as follows: $r_i^2 = a\sigma_i^2$ where a is a domain specific scalar constant. In our image experiments, the a value is found via a grid search and we noticed that setting $a = 2.85$ suffices for all the images used in our experiments.

Note that, in contrast to [5], [31], our similarity domain definition differs from the term "minimal enclosing sphere". In our approach, we define the term similarity domain as the dominant region of a SV in which the SV is the centroid and all the points within the domain are similar to the SV. The boundary of the similarity domain of a SV is defined based on its distance to the closest point from the other class. Thus any given vector within a similarity domain (a region) will be similar to the associated SV of that similarity domain.

To illustrate similarity domains, please refer to Fig. 3. In the figure, the dotted circles represent the similarity domains where red ones represents the similarity domains for the (+1) class, and the black ones represent the similarity domains from the (-1) class. The dominant area of a SV vanishes as it touches to the similarity domain of the closest SV from the other class. The kernel parameter σ_i is a function of the radius of the similarity domain of the SV $\mathbf{x_i}$. Notice that the point P_1 is the closest SV from the (+1) class to P_4 and thus they both have the same kernel parameter. Similarly, the SV pairs (P_3, P_6) and (P_2, P_5) share the same kernel parameters, respectively. This example primarily motivates our approach to assign different kernel parameters to different training samples.

What that illustration in Fig. 3 suggests is that, we can partition the classes on each side of the decision boundary in terms of similarity domains that are represented by kernel functions. We will use the similarity domain concept to define a kernel machine that computes its kernel parameters automatically and geometrically in the next section.

5 Similarity Domains Machine

The concept of similarity domains defines a distance-based geometric relation between the support vectors as mentioned in the earlier section. We will use such geometric relation as an additional constraint in our kernel machine and will represent similarity domains with the kernel function $K_{\sigma ij}(.)$ in our cost function. Below optimization problem defines our kernel machine: similarity domains machine with a kernel specific (geometric) constraint.

$$\max_{\alpha} Q(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K_{\sigma ij}(\mathbf{x_i},\mathbf{x_j}),$$
$$\text{subject to: } \sum_{i=1}^{n}\alpha_i y_i = 0, \quad C \geq \alpha_i \geq 0 \text{ for } i = 1, 2, ..., n, \tag{8}$$
$$\text{and } K_{\sigma ij}(\mathbf{x_i},\mathbf{x_j}) < T, \text{ if } y_i y_j = -1,\ \forall i, j$$

where T is a constant assuring that the similarity between the two closest samples from different classes remains smaller than the middle value of the kernel function. The kernel parameter σij is defined as $\sigma ij = min(\sigma_i, \sigma_j)$. For a given pair of closest vectors

$\mathbf{x_i}$ and $\mathbf{x_j}$ for which $y_i y_j = -1$, the kernel parameters are defined as follows:

$$\sigma_i^2 = \sigma_j^2 = \frac{- \parallel \mathbf{x_i} - \mathbf{x_j} \parallel^2}{\ln(K(\mathbf{x_i}, \mathbf{x_d}))} \tag{9}$$

where $\mathbf{x_d}$ is the most distant sample from $\mathbf{x_i}$. The decision function takes the following form:

$$f(\mathbf{x}) = \sum_{i=1}^{k} \alpha_i y_i \exp(-\frac{\parallel \mathbf{x} - \mathbf{x_i} \parallel^2}{\sigma_i^2}) \tag{10}$$

where k is the total number of support vectors.

The decision function $f(\mathbf{x})$ can be expanded as $f(\mathbf{x}) = \sum_{i=1}^{k_1} \alpha_i y_i K_i(\mathbf{x}, \mathbf{x_i}) + \sum_{j=1}^{k_2} \alpha_j y_j K_j(\mathbf{x}, \mathbf{x_j})$ where k_1 is the total number of SVs near vector $\mathbf{x}$ such that the Euclidian norm $\parallel \mathbf{x_i} - \mathbf{x} \parallel^2 - \sigma_i^2 >> 0$ and k_2 is the total number of SVs for which $\parallel \mathbf{x_j} - \mathbf{x} \parallel^2 - \sigma_j^2 \simeq 0$ such that $k_1 + k_2 = k$. As a result, local predictions can be made by the approximated decision function $f(\mathbf{x}) \simeq \sum_{i=1}^{k_1} \alpha_i y_i K_i(\mathbf{x}, \mathbf{x_i})$.

The parameter T defined in Eq. (8) has an upper and lower bound for SDM. The value of Gaussian kernel ranges between 0 and 1. Since T represents a low similarity value, it is bounded in the range: $0 < T < 0.5$ where 0.5 is the half of the maximum kernel value, (i.e., $max(K(.))/2 = 0.5$).

Next, we will demonstrate how we utilize SDM to model shapes (edges) as decision function and how we can scale that decision function at different scales without the need for re-training the classifier. We will also show a method on how we can further reduce the computed parameters of SDM to a one-class classifier as a post-processing technique.

5.1 SDM for Sparse Shape Modeling as One-class Classifier

Sparse shape learning is a challenge for classical kernel machines as demonstrated in Fig. 4. Here, we demonstrate how we deal with that challenge by using the computed kernel parameters of SDM.

SDM can model shapes sparsely with its computed kernel parameters. For that, first the shape is learned as the decision boundary with SDM from the given binary image. The training of SDM can be done by labeling the shape (e.g., the white region in Fig. 4a) as foreground and by labeling everything else (e.g., the black region in Fig. 4a) as background while using each pixel coordinate as features. Once the image is learned, then, the computed kernel parameters of SDM are used to model the shape with our one-class classifier without performing any re-training. Next, we define our one class approximation of the decision boundary with SDM.

A One-class classifier can be obtained from the computed parameters of SDM without performing any re-training. Here, we show how using only the SVs and their associated kernel parameters from one class is enough to approximate $f(\mathbf{x})$ in SDM. The SVs of SDM can be grouped as $C_1 = \bigcup_{i=1, y_i \in +1}^{s_1} \mathbf{x_i}$ and $C_2 = \bigcup_{i=1, y_i \in -1}^{s_2} \mathbf{x_i}$, where $s_1 + s_2 = k$, s_1 is the total number of SVs from (+1) class and s_2 is the total number of the SVs from the (-1) class. Since kernel functions now represent local similarity domains geometrically, the original function $f(\mathbf{x})$ can be approximated by using only C_1 (or by using only C_2). Consequently, we define the one-class approximation by using only the SVs and their associated kernel parameters from the C_1 as follows:

$$\begin{aligned} &\overline{y} = +1, \; if \;\; \parallel \mathbf{x} - \mathbf{x_i} \parallel < r_i \; , \exists \mathbf{x_i} \in S_1 \\ &otherwise \; \overline{y} = -1, \end{aligned} \tag{11}$$

where r_i is the similarity domain radius for the i^{th} support vector $\mathbf{x_i}$ (where $\mathbf{x_i} \in C_1$ and $r_i^2 = a\sigma_i^2$), $\mathbf{x}$ is the testing sample and a is a domain specific constant. Since, a similarity domain of a support vector from (-) class can never overlap with a similarity domain from (+) class by definition (except one point where they touch each other), we can use the similarity domains from only C_1 to *approximate* the decision boundary (see Fig. 1 for an example).

Note that our one-class approximation is different than the one-class SVM in [29]. One-class SVM in [29] focuses on learning from the data of one class only via an optimization step. However, in our approach, we obtain an approximated one-class classifier for the same decision boundary that is learned by our binary classifier (SDM) by utilizing the already-computed kernel parameters. In our case, we do not need to perform (re)-training to obtain a one-class classifier. Furthermore, we demonstrate the relation between our binary classifier and our one-class classifier on binary shape images where the decision boundary represents the edge of the shape (for an example, please refer to Fig. 10 in Section VI).

5.2 Scale-invariant Kernel Parameters for Geometric Scaling

The recent advances in the domain transfer and domain adaptation problems do not address the geometric scalability of decision boundaries in kernel machines. In this section, we address the problem of scaling the shapes at different scales. By scale-invariant, we mean that the relative ratio of the kernel parameters of SDM does not change at different scales. Therefore, at any given scale factor s we can redefine the decision boundary in any new domain as follows:

$$f_{new}(\mathbf{x}) = f(\mathbf{x}, s) = \sum_{i=1}^{k} \alpha_i y_i \exp(-\frac{\parallel \mathbf{x} - s\mathbf{x_i} \parallel^2}{s^2 \sigma_i^2}) \tag{12}$$

where $f_{new}(\mathbf{x})$ is the desired decision function in the new domain. In Eq. (12), the input parameter s is given by the user.

The scaling of the decision boundary is explainable here, as we interpret the kernel parameters geometrically and compute them accordingly in the algorithm. That allows us to preserve the geometry of the shape by keeping the relative ratio between the kernel parameters at different scales as such ratio will be retained in the Euclidian space (see Section 6.1 for examples) as scale-invariant.

5.3 Implementation

The optimization process of the SDM returns k SVs and their associated scalar α_i and σ_i^2 values during the training. After the training, once all the σ_i^2 values are computed, we compute the corresponding r_i. One of the key properties of our proposed algorithm is that, it processes two training samples to compute the kernel parameters due to the constraint $K_{\sigma ij}(\mathbf{x_i}, \mathbf{x_j}) < T$, if $y_i y_j = -1$, $\forall i, j$. A similar approach of processing two samples at a time is also used in Platt's sequential minimal optimization (SMO) algorithm [27]. Therefore we use SMO as the base optimization technique for the optimization of Eq. (8) for SDM. Updating α_i and α_j are done similar to the technique given in [27]. SDM can

(a) Original image (b) SVM with σ^2=100, E= 309 (c) SVM with σ^2=20, E = 119 (d) SVM with σ^2=3.3, E = 51 (e) SVM with σ^2=1.4, E = 31 (f) SVM with σ^2=1.0, E = 3 (g) SDM with T=0.05, E = 0

Fig. 4: This figure shows what the classical SVM "learns" from the given training data containing 2D binary shape and how the SVM result changes with respect to the kernel parameter of the Gaussian kernel. **(a)** The original training image (83x64 = 5312 pixels) is shown. Each pixel coordinate (x_1,x_2) forms the training data and its color forms the label for that pixel. Once the SVM is trained, we used the same training data (pixel coordinates) to visualize what the SVM actually learned at different kernel parameters. The error (E) is computed as the total number of misclassified pixels. **(b)** Shows the SVM result with $\sigma^2 = 100$ with E=309. The total number of SV is **1036** (19.5% of all pixels). **(c)** Shows the SVM result with $\sigma^2 = 20$ with E=119. The total number of SV is **1084** (20.4% of all pixels). **(d)** Shows the SVM result with $\sigma^2 = 3.3$ with E=51. The total number of SV is **3261** (61.39% of all pixels). **(e)** Shows the SVM result with $\sigma^2 = 1.4$ with E=31. The total number of SV is **4778** (89.95% of all pixels). **(f)** Shows the SVM result with $\sigma^2 = 1.0$ with E=3. The total number of SV is **4974** (93.64% of all pixels); i.e., *SVM used almost all the training data as SVs*. **(g)** SDM result obtained with T=0.05 with E=0. The total number of SVs is **1082** (20.41% of all pixels).

be utilized in various applications and enhances kernel machines with its spatial properties. Further details and the analysis of the optimization procedure of SDM are out of the scope of this paper and will be reported in a future work.

5.4 Filtering via Kernel Parameters

Here, we demonstrate how geometrically computed kernel parameters can be utilized for a threshold-based filtering operation, instead of re-training a machine learning algorithm, to obtain different results at different parameters on a sample 2D image. The similarity domains vary in size and that is reflected in the computed kernel parameters as explained in the previous sections. We use such variance in kernel parameters to filter some SVs from the foreground. Once certain SVs are filtered, we can use the remaining foreground SVs and their associated kernel parameters to construct the filtered foreground (i.e., the new and altered decision boundary) via one-class approximation.

For example, consider the image shown in Fig. 4a. First, we train SDM using the full image data and obtain all the SVs and their corresponding kernel parameters at $T = 0.05$. Then, we select the foreground kernel parameters and their corresponding SVs. The foreground kernel parameters are quantized into n bins. See Table 1 for $n = 10$ case for Fig. 4a. In the table, the total counts show the total number of kernel parameters in each bin. We then simply threshold the kernel parameters to eliminate some of the foreground SVs. Applying one class approximation on those remaining foreground SVs yields different results at different thresholds. Fig. 5 demonstrates sample filtering results obtained at different thresholds.

TABLE 1: Bin centers for quantized kernel parameter values and total number of kernel parameters that fall in each bin for the image in Fig. 4a.

Bin Center:	1.29	3.19	5.09	6.99	8.90	10.80	12.70	14.60	16.51	18.41
Total Counts:	303	16	12	2	3	1	0	2	2	1

6 Experiments

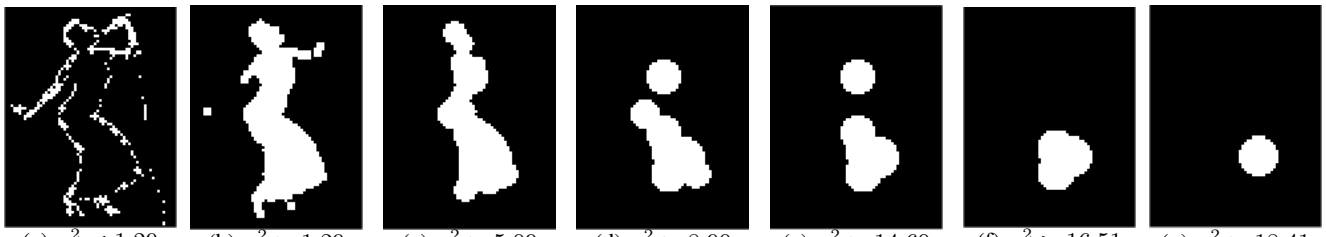
SDM is the first and only spatial kernel machine with automatically computed and visually explained kernel parameters. Therefore, in this section, our experiments focus on scalable shape modeling problems to demonstrate the new (spatial) capabilities of our kernel machine rather than focusing on the accuracy comparison between classifiers. However, in order to demonstrate that SDM is also a machine learning algorithm that can classify various datasets at a similar or better accuracy as SVM does, we included multiple UCI benchmark datasets [20] and compared SDM performance to that of SVM (we used libSVM as implementation from [6]) and SimpleMKL [28]. In those experiments, the accuracy is defined as percentage as follows: $100(1 - \sum_{i=1}^{l} sgn(\| \overline{y_i} - y_i \|))/l$ where l is the total number of test samples. In image experiments, we compute the error (E) as the total number of misclassified pixels for the given image. In all images, the radiuses are computed and visualized by using the following relation: $r_i^2 = 2.85\sigma_i^2$. All the shown images are resized to fit into the figures.

6.1 Shape Modeling and Scaling

Here, by sparse shape learning, we mean the sparse reconstruction of shapes. We assume the objects in all the images are already segmented and we use those segmentation results (binary masks) as inputs.

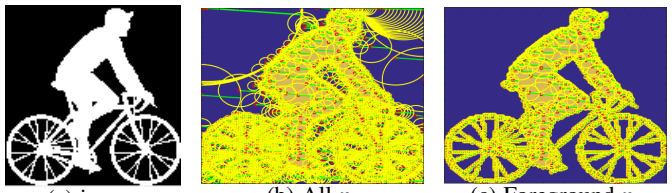
We first demonstrate the performances of both SVM and SDM on learning a shape as a decision boundary. SVM is not well suited for shape modeling in general. To demonstrate that, we picked the binary image shown in Fig. 4a. The training data consists of the coordinates (x_1, x_2) of all the pixels in the image. The pixel colors (being black or white) are used as labels (forming a total of labeled 5312 training samples). SVM is trained for σ^2= 100, 20, 3.3, 1.4 and 1.0 values, respectively. Then, for each of those σ^2 values, we predicted the label of each pixel in the image with the trained SVM (i.e., reconstructed the learned shape with the trained SVM). The figures 4b, 4c, 4d, 4e, 4f show the results for each of those σ^2 values respectively.

Fig. 4 demonstrates that it is hard for SVM to learn a shape as the learning accuracy (as well as the total number of SVs) depends on finding the appropriate kernel parameter. However, even if the correct kernel parameter is found for low error, as the figure suggest, SVM reduces the error value by converging the total number of SVs to the total number of training data. Fig. 4f yields 4974 SVs out of the total of 5312 training data. This



(a) $\sigma_i^2 < 1.29$ (b) $\sigma_i^2 > 1.29$ (c) $\sigma_i^2 > 5.09$ (d) $\sigma_i^2 > 8.90$ (e) $\sigma_i^2 > 14.60$ (f) $\sigma_i^2 > 16.51$ (g) $\sigma_i^2 > 18.41$

Fig. 5: This figure shows various one class approximations for the shape shown in Fig. 4a after filtering. The filtering is performed as thresholding where we kept and used only the set of kernel parameters (σ_i^2) that are either smaller or greater than the specified threshold.



(a) image (b) All r_i (c) Foreground r_i

Fig. 6: Visualization of the SDM kernel parameters at T= 0.1, E= 4. In the figure, blue color represents the background and the orange color represents the foreground. The red dots are the SVs and yellow circles around them show the boundaries of similarity domains as computed by SDM. The green lines are the radiuses (r_i) of similarity domains. Radiuses are obtained from the computed kernel parameters. **(a)** Original image: 111x114 pixels. **(b)** Visualization of all the radiuses from both background and foreground with total of 3050 SVs. **(c)** Visualization of only the foreground radiuses with total of 1438 foreground SVs, (only 11.36% of all pixels).

TABLE 2: Total number of SVs and the total pixel error of SDM at various T values are listed below for the images shown in Table 6.

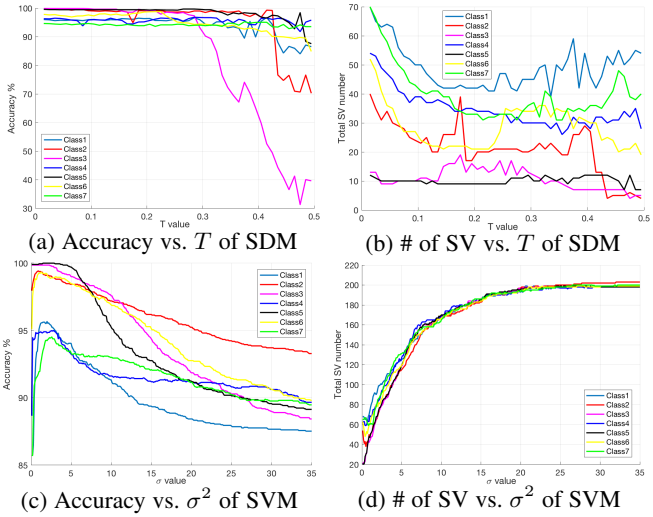
T value:	0.05	0.1	0.15	0.2	0.3	0.35	0.4
Image1:	4007/0	3563/1	3255/3	3111/7	3019/14	3014/42	2981/70
Image2:	2717/0	2375/0	2109/4	2108/1	1966/14	1939/37	1887/41
Image3:	1481/0	1311/0	1237/0	1175/1	1117/5	1188/15	1117/16
Image4:	1848/0	1513/0	1391/0	1387/1	1365/12	1279/19	1181/42
Image5:	1717/0	1425/0	1314/2	1307/3	1118/12	1146/21	1028/16
image6:	1189/0	1136/1	1031/5	873/7	944/6	820/11	775/9

suggests on how SVM can learn shapes (by considering almost all the training data as SVs). On the other hand, SDM was able to yield 0 pixel error with only 1082 SVs where T is set to 0.05. The number of total SVs (i.e., the total number of samples needed to construct the decision boundary) can be further reduced from that number by deriving a one-class SDM (see Section 6.3).

Fig. 6 visualizes the radiuses for similarity domains as computed by SDM for another image. Fig. 6c visualizes only the foreground similarity domains. For scaling the shapes, two examples are shown in Fig. 9, where first the original shape is represented as the decision function at a lower scale and then that decision function is scaled at three different scales by using Eq. (12).

TABLE 3: Test results on 7-class UCI image-segmentation data set. First two rows list the max accuracy (in %) and the total number of SVs for SDM and SVM, respectively. Next two rows list the AUC values for each classifier at their max accuracy parameters. The last row shows the significance test results: H=1 means that the accuracies are significantly different and H=0 means that the accuracies are not significantly different at 5% significance level.

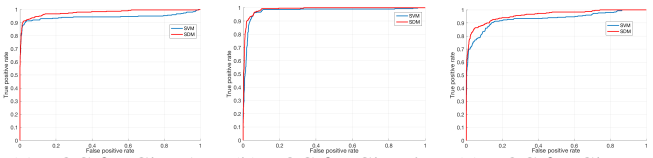
Classifier	Class1	Class2	Class3	Class4	Class5	Class6	Class7
SDM:	**96.71/42**	**99.67/35**	99.86/**13**	**96.81/30**	99.67/**11**	98.76/**21**	**95.24/34**
SVM:	95.67/96	99.43/50	99.86/20	95.00/95	**100.00**/60	**99.33**/69	94.52/93
SDM AUC:	**0.9778**	**0.9990**	**0.9995**	**0.9885**	0.9999	0.9985	**0.9600**
SVM AUC:	0.9431	0.9988	0.9965	0.9741	**1.0000**	**0.9995**	0.9305
Hypothesis (H):	1	0	0	1	1	1	0



(a) Accuracy vs. T of SDM (b) # of SV vs. T of SDM

(c) Accuracy vs. σ^2 of SVM (d) # of SV vs. σ^2 of SVM

Fig. 7: This figure shows how SDM and SVM accuracies as well as their total number of SVs change with respect to their individual kernel parameters (where the parameter for SDM is T and for the SVM it is the σ^2 of the Gaussian kernel). The dataset is the UCI image-segmentation dataset containing 18 dimensional data samples from 7 different image classes. Each color represents an individual class. Training is done as one vs. all for each class.



(a) ROC for Class1 (b) ROC for Class4 (c) ROC for Class7

Fig. 8: ROC plots for Class1, Class4 and Class7 for the 7-class UCI image-segmentation data set. See Table 3 for the AUC values.

6.2 Robustness: The T of SDM vs. σ^2 of SVM

Here we compare SDM to SVM in terms of accuracy & total number of SVs (see Fig. 7 and Table 3), area under the curve (AUC) values (see Table 3) and receiver operating characteristic (ROC) curves (see Fig. 8). For that, we use a benchmark dataset (image segmentation dataset) from the UCI repository [20]. The dataset has 7 different classes where each data has 18 features.

Fig. 7 demonstrates how the accuracy and the total number of SVs change with respect to T parameter of SDM and the σ^2 parameter of the Gaussian kernel of SVM. In the figure, the σ^2 value of SVM varies between 0.1 and 35 with increments of 0.1 and the T value of SDM varies between 0.015 and 0.5 with increments of 0.01. As the σ^2 value increases, the total number of SVs converges to the total number of training samples for SVM. However, increasing the T value does not change the total number

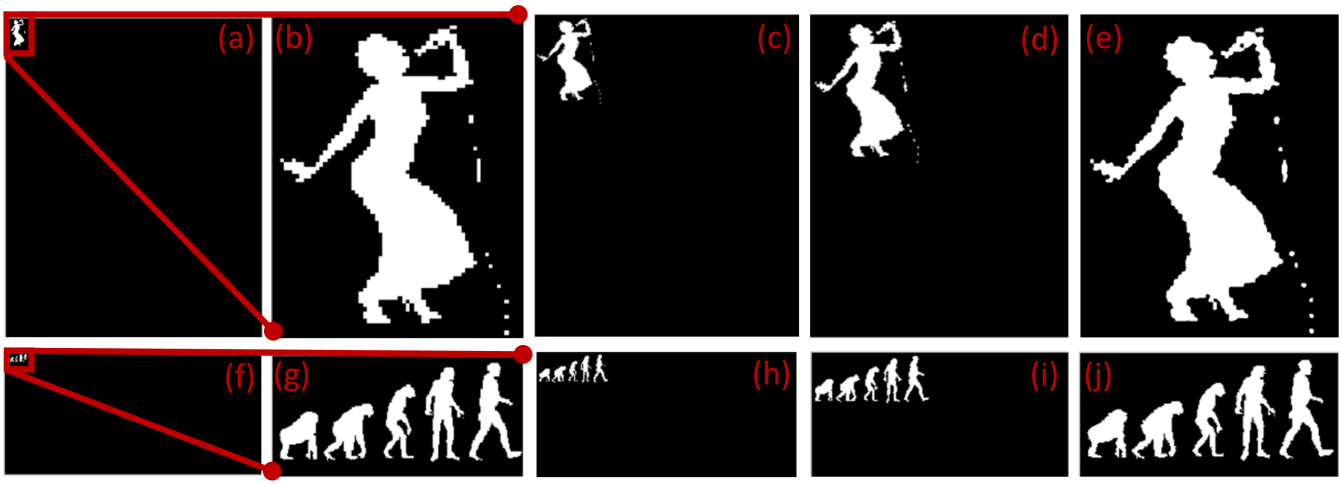


Fig. 9: Two binary shapes are represented by SDM as decision boundary and they are scaled at a higher resolution by using only the SVs at a given s value (instead of saving all the pixels). **(a)** Shows the 64x83 pixels original training data on 722x900 pixels black background. SDM learns the 64x83 pixels image with zero pixel error. **(b)** 64x83 pixels original training data (shown in red box in (a)) is visualized as a zoomed in image with standard interpolation (by using all of the 64x83 pixels). **(c)** the learned shape in 64x83 image is scaled at s=3 shown on 722x900 background by using only its SVs, **(d)** the learned decision boundary in 64x83 image is scaled at s=5 and shown on 722x900 background by using only its SVs, **(e)** the learned shape in 83x64 image is scaled at s=10.75 and shown on 722x900 background. **(f)** shows another 192x91 pixels image in red box (a new training data) on a 2100x1000 pixels black background. **(g)** shows the zoomed in version of the original image. **(h)** the scaled binary image is reconstructed at s=3 with SVs only, **(i)** the scaled binary image is reconstructed at s=5 with SVs only, **(j)** the scaled binary image is reconstructed at s=10.75 with SVs only, (all the images are resized to fit into the figure).

TABLE 4: Test results from various UCI datasets. (The best accuracy in % and the total number of SVs are shown).

Classifier	Ionosphere	SPECT	Wine1	Wine2	Wine3	Iris1	Iris2	Iris3	GermanCredit	BreastCancer
SDM:	**89.35% - 32**	89.84% - **59**	**91.30% - 9**	65.22% - **25**	76.09% - **24**	100% - **3**	**97.78**% -21	**98.89**% - **11**	**71.88% - 95**	97.26% - **14**
SVM:	75.60% - 40	89.84% - 73	82.61% - 40	65.94% - 40	**80.43**% - 40	100% - 9	96.67% - **20**	96.67% - 21	71.63% - 144	**97.60**% - 41
SimpleMKL:	72.16% - 55	89.84% - 80	90.58% - 19	**68.84**% - 39	69.57% - 39	100% - 24	96.67% - 44	93.33% - 43	69.23% - 200	95.71% - 72

TABLE 5: Additional test results from various UCI datasets. (The best accuracy in % and the total number of SVs are shown).

Classifier	Monk1	Monk2	Monk3	Liver	Tae1	Tae2	Tae3	Magic	Madelon	HillValley
SDM:	**89.81% - 60**	79.63% - **114**	94.44% -**57**	**70.67**% - **89**	**76.41**% - **35**	**74.53**% - **22**	75.47% - **21**	**74.52**% - **96**	**68.75% - 493**	**56.90**% - **177**
SVM:	87.93% - 124	**82.18**% - 169	**96.99**% - 68	**70.67**% - 110	75.47% - 44	71.70% - 38	**77.36**% - 35	69.42% - 172	64.40% - 600	51.97% - 199
SimpleMKL:	71.06% - 124	67.13% - 169	81.94% - 118	54.22% - 120	67.43% - 43	66.98% - 45	65.09% - 44	40.14% - 180	61.15% - 600	49.51% - 200

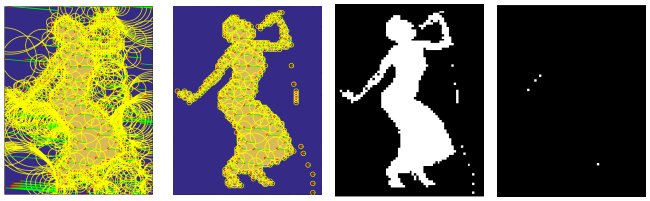
of SVs as drastically for SDM suggesting that the total number of SVs does not converge to the total number of training samples for none of the 7 classes in the data set. We also observe the same behavior in other experiments (where the total number of SVs did not converge to the total number of samples with respect to the change in T). Table 3 lists the maximum accuracy values and their associated total number of SVs for both SDM and SVM. For all the 7 classes, SDM yielded the minimum number of SVs when compared to SVM, while the accuracy values remained close to that of SVM for all 7 classes. Those results in Fig. 7 suggest that SDM is more robust with respect to its parameter T when the concern is the total number of SVs.

Fig. 8 shows the ROC curves for the classes 1, 4 and 7. For the other classes, ROC curves were almost identical, as both SDM and SVM yielded AUC value closer to 1.

In Table 3 (last row), we also show the McNemar test results for the significance. In the test, the null hypothesis is that the predicted labels of SDM and SVM have the equal accuracy for predicting the ground truth labels and the alternative hypothesis is that predicted labels have unequal accuracy. In the table, hypothesis (H) is 1 when to reject the null hypothesis at the 5% significance level and H is 0 when to not reject the null hypothesis at the 5% level.

6.3 Approximation and Image Altering with One-class SDM

Here we demonstrate how we can approximate the decision boundary with *one-class SDM approximation* and alter the order and the size of the objects individually in an image. The training image is shown in Fig. 1a. SDM is trained by using the 2D pixel coordinates and the result (reconstructed image with SDM) is shown in Fig. 1b. Fig. 1c visualizes all the similarity domains for all the SVs computed by SDM. The foreground SVs are selected and kept along with their associated kernel parameters to form one-class approximation. The selected SVs and their similarity domains are visualized in Fig. 1d. As there are 5 distinct objects in the figure, we further grouped the SVs and their associated kernel parameters for each of those five objects from the foreground by using a region growing algorithm. Then, we shifted and scaled each of those objects individually at different scales by using only their foreground SVs and their associated kernel parameters. The scaling is performed by using Eq. (12). The resulting altered binary image in Fig. 1e is obtained by using one-class SDM classification by using Eq. (11) on the combination of those scaled and shifted similarity domains. Fig. 10 demonstrates how much error is obtained by using only the foreground SVs (6.4% of all the pixels) with Eq. (11) for the image shown in Fig. 4a. To see further results on the ratio of the foreground SVs to the total pixel numbers, please refer to the D values in Table 6 for various images.



(a) All the r_i from SDM (b) Only foreground r_i (c) One-class approx. (d) Approx. error, E = 4

Fig. 10: Visualization of the SDM kernel parameters (at $T = 0.05$, E = 0) and its one-class approximation are shown for Fig. 4a. **(a)** visualizes all the similarity domains from both classes (total of 1082 SVs). **(b)** visualizes only the similarity domains of the foreground with total of 342 SVs. **(c)** shows the one-class approximation by using the Eq. (11). E = 4. **(d)** shows the approximation error in pixels by using only the foreground SVs (342 SVs = 6.4% of all the pixel data).

6.4 Benchmark Comparisons on Higher Dimensional Datasets

Here, we compare SDM's classification performance to both SVM and MKL on various UCI benchmark datasets. The results of these experiments are summarized in Tables 4 and 5. The best accuracies found by each algorithm (as percentage) along with the computed SV numbers are listed for each algorithm. Each class has been trained separately with one vs. all approach (for multi-class datasets) and the individual results for each class are listed in the tables. As demonstrated, SDM yields similar or better result when compared to SVM and MKL, while yielding lower number of SVs in almost all experiments.

7 Discussions and Conclusion

We introduced a novel scale-invariant kernel machine: Similarity Domains Machine (SDM) that computes its own scale-invariant kernel parameters automatically and geometrically. The existing work related to kernel parameters mostly focused on utilizing additional cost function(s) to estimate the kernel parameters. In this work, as an alternative to those techniques, we introduced a geometric approach to eliminate the additional cost function(s) for estimating the kernel parameters and to explain the parameters to non-experts in a more intuitive way. Our proposed approach yields to obtain a scale-invariant classifier with scale-invariant kernel parameters.

Our introduced similarity domains concept helps explaining the kernel parameters intuitively to non-experts. Due to its geometric properties, our novel classifier simplifies many problems that are *not trivial* to solve with the classical kernel machines in various image processing applications. In this paper, as a proof of concept, we demonstrated how to deal with four of those problems on 2D shapes: (1) a subset of the domain adaptation problems where the classifier is trained on only the source domain while that classifier needed to be scaled in new (target) domains; (2) forming a one-class classifier from the already trained binary classifier in a straightforward manner to approximate the decision boundary without re-training; (3) how to use the computed kernel parameters for sparse shape modeling; and (4) how to use kernel parameters to further alter (filter) the decision boundary.

We demonstrated that SDM is not only capable of modeling shapes efficiently, which can be considered as vectorizing the pixel-based binary shapes at low SV numbers (see the D values in Table 6), but also capable of scaling those shapes at different given resolutions with its sparse modeling. That property can be useful in many image processing applications (e.g., in computation of image pyramids).

Note that there are some visualization tools already available in the literature for machine learning such as t-SNE [22]. t-SNE essentially is used as a dimensionality reduction algorithm that reduces multidimensional data onto 2D or 3D space for visualization through an optimization process in the literature. However, in this paper, we use a classifier to utilize the classifier's kernel parameters on 2D datasets (images) without the need for an additional tool. By visualization, one goal in this paper is to demonstrate the accuracy of our proposed machine learning algorithm on computing the kernel parameters geometrically on 2D shapes.

SDM forms local clusters defined by similarity domains within each class. In this paper, while we limited our work with the use of similarity domains in 2D shape modeling applications, we believe that our presented approach will benefit many additional image processing applications including (but not limited to), graph construction from the labeled (image) dataset, outlier/noise detection and feature matching applications.

Our approach can also be extended to some other kernel functions that are a function of both distance and a scaling parameter β such that: $K(\mathbf{x}, \mathbf{x_i}) = f(\| \mathbf{x} - \mathbf{x_i} \|, \beta)$. A future work may involve studying the performance of other kernels within our SDM framework.

SDM is a spatial kernel machine and as a classifier, it can classify higher dimensional data yielding comparable results to that of SVM with less number of SVs (see tables 3, 4 and 5). In many cases, SDM yields even less then 50% of the total number of SVs found by SVM. That makes SDM a good alternative to SVM and to other kernel machines, where the concern is reducing the total number of SVs.

Acknowledgement

The author would like to thank Prof. Scott T. Acton from Univ. of Virginia for his valuable feedback. This work was done & completed before the author moved to MIT and was submitted at MIT.

References

[1] F. R. Bach, G. R. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, page 6. ACM, 2004.

[2] K. P. Bennett, M. Momma, and M. J. Embrechts. Mark: A boosting algorithm for heterogeneous kernel models. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 24–31. ACM, 2002.

[3] S. K. Biswas and P. Milanfar. Linear support tensor machine with lsk channels: Pedestrian detection in thermal infrared images. *IEEE Transactions on Image Processing*, 26(9):4229–4242, Sept 2017.

[4] S. S. Bucak, R. Jin, and A. K. Jain. Multiple kernel learning for visual object recognition: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(7):1354–1369, 2014.

[5] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[6] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[7] O. Chapelle, B. Schölkopf, A. Zien, et al. *Semi-supervised learning*, volume 2. MIT press Cambridge, 2006.

[8] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine learning*, 46(1-3):131–159, 2002.

[9] T. Chen and S. Lu. Subcategory-aware feature selection and svm optimization for automatic aerial image-based oil spill inspection. *IEEE Transactions on Geoscience and Remote Sensing*, 55(9):5264–5273, Sept 2017.

[10] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. *On kernel target alignment*, volume 2. NIPS, 2001.

[11] L. Duan, I. W. Tsang, D. Xu, and S. J. Maybank. Domain transfer svm for video concept detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1375–1381. IEEE, 2009.

[12] L. Fe-Fei et al. A bayesian approach to unsupervised one-shot learning of object categories. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1134–1141. IEEE, 2003.

[13] K. Gai, G. Chen, and C. Zhang. Learning kernels with radiuses of minimum enclosing balls. In *NIPS*, pages 649–657, 2010.

[14] W. Gao and Y. Peng. Multiple kernel-learning based hyperspectral data classification. In *Advances in Intelligent Information Hiding and Multimedia Signal Processing: Proceeding of the Twelfth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Nov., 21-23, 2016, Kaohsiung, Taiwan, Volume 2*, pages 69–76. Springer, 2017.

[15] P. Ghamisi and B. Höfle. Lidar data classification using extinction profiles and a composite kernel support vector machine. *IEEE Geoscience and Remote Sensing Letters*, 14(5):659–663, 2017.

[16] M. Gönen and E. Alpaydin. Localized multiple kernel learning. In *Proceedings of the 25th international conference on Machine learning*, pages 352–359. ACM, 2008.

[17] M. Gönen and E. Alpaydın. Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268, 2011.

[18] B. Kulis, K. Saenko, and T. Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1785–1792. IEEE, 2011.

[19] G. R. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *The Journal of Machine Learning Research*, 5:27–72, 2004.

[20] M. Lichman. UCI machine learning repository, 2013.

[21] C. Liu, X. Wu, and Y. Jia. Transfer latent svm for joint recognition and localization of actions in videos. *IEEE Transactions on Cybernetics*, 46(11):2596–2608, Nov 2016.

[22] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[23] S. Ozer. On the classification performance of support vector machines using chebyshev kernel functions. *Master's Thesis, University of Massachusetts, Dartmouth*, 2007.

[24] S. Ozer, C. H. Chen, and H. A. Cirpan. A set of new chebyshev kernel functions for support vector machine pattern classification. *Pattern Recognition*, 44(7):1435–1447, 2011.

[25] S. Ozer, M. A. Haider, D. L. Langer, T. H. van der Kwast, A. J. Evans, M. N. Wernick, J. Trachtenberg, and I. S. Yetik. Prostate cancer localization with multispectral mri based on relevance vector machines. In *Biomedical Imaging: From Nano to Macro, 2009. ISBI'09. IEEE International Symposium on*, pages 73–76. IEEE, 2009.

[26] S. Ozer, D. L. Langer, X. Liu, M. A. Haider, T. H. van der Kwast, A. J. Evans, Y. Yang, M. N. Wernick, and I. S. Yetik. Supervised and unsupervised methods for prostate cancer segmentation with multispectral mri. *Medical physics*, 37(4):1873–1883, 2010.

[27] J. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods support vector learning*, 3, 1999.

[28] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 9(11), 2008.

[29] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[30] B. Scholkopf, K.-K. Sung, C. J. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE transactions on Signal Processing*, 45(11):2758–2765, 1997.

[31] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[32] V. Vapnik. The nature of statistical learning theory. *Data Mining and Knowledge Discovery*, pages 1–47, 1995.

[33] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 606–613. IEEE, 2009.

[34] J. Wang, H. Do, A. Woznica, and A. Kalousis. Metric learning with multiple kernels. In *NIPS*, pages 1170–1178, 2011.

[35] L. Wang, P. C. Pedersen, E. Agu, D. M. Strong, and B. Tulu. Area determination of diabetic foot ulcer images using a cascaded two-stage svm-based classification. *IEEE Transactions on Biomedical Engineering*, 64(9):2098–2109, Sept 2017.

[36] J. Yang, Y. Tian, L.-Y. Duan, T. Huang, and W. Gao. Group-sensitive multiple kernel learning for object recognition. *Image Processing, IEEE Transactions on*, 21(5):2838–2852, 2012.

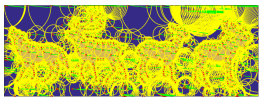
[37] J. Yang, R. Yan, and A. G. Hauptmann. Cross-domain video concept detection using adaptive svms. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 188–197. ACM, 2007.

[38] Y.-H. Yuan, Y. Li, J. Liu, C.-F. Li, X.-B. Shen, G. Zhang, and Q.-S. Sun. Learning multi-kernel multi-view canonical correlations for image recognition. *Computational Visual Media*, 2(2):153–162, 2016.

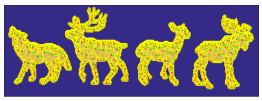
TABLE 6: Visualization of similarity domains on multiple shapes (images). Similarity domain boundaries are highlighted with yellow circles. The *first column* shows the original binary image, the *second column* shows what SDM learns at $T = 0.1$ where E is the total pixel error. The *third column* visualizes all the radiuses and SVs computed by SDM and the *fourth column* visualizes only the SVs and their radiuses for the foreground (+1) class. D (shown in percentage) is the ratio of the total number of foreground SVs to the total number of image pixels.

(a) image1: 280x85 pixels

(b) SDM result, T=0.1, E=1.



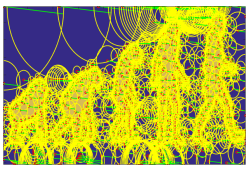
(c) All r_i, 3563 SVs.



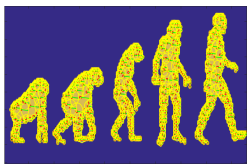
(d) 1459 SVs, D=6.13%

(e) image2: 192x91 pixels

(f) SDM result, T=0.1, E=0.



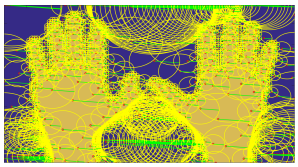
(g) All r_i, 2375 SVs.



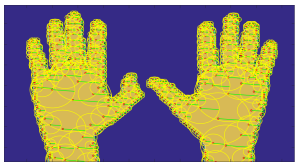
(h) 1025 SVs, D=5.87%

(i) image3: 115x77 pixels

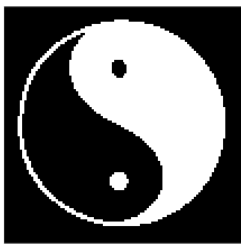
(j) SDM result, T=0.1, E=0.



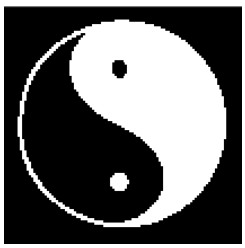
(k) All r_i, 1311 SVs.



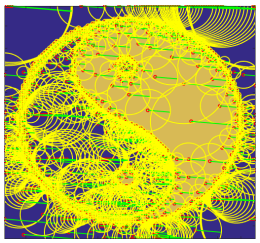
(l) 569 SVs, D=6.43%



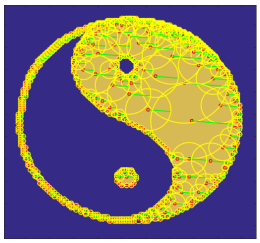
(m) image4: 106x106 pixels



(n) SDM result, T=0.1, E=0.



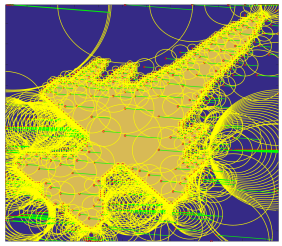
(o) All r_i, 1513 SVs.



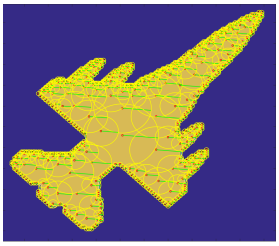
(p) 580 SVs, D=5.16%

(q) image5: 115x115 pixels

(r) SDM result, T=0.1, E=0.

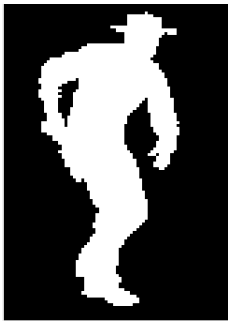


(s) All r_i, 1425 SVs.

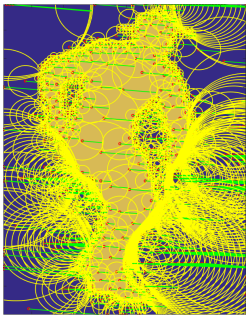


(t) 544 SVs, D=4.11%

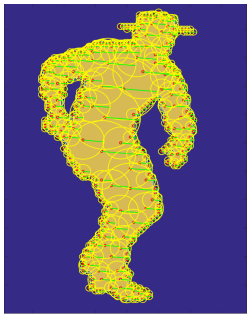
(u) image6: 78x110 pixels



(v) SDM result, T=0.1, E=1.



(w) All r_i, 1136 SVs.



(x) 381 SVs, D=4.4%