Indian Institute of Technology Kanpur
Department of Mathematics and Statistics

MTH599A: Course Project on

"Binary Classification Using Some Notions of Kernelized Data Depths"

Under the supervision of

Dr. Subhajit Dutta

Submitted by

Kaushik Thakkar (221329)

Acknowledgement

I would like to thank all the people who have helped and supported me during this project. First and foremost, I would like to thank my project supervisor Dr. Subhajit Dutta for his constant support and guidance at every step of my project. I would also like to thank my classmates, who have many times patiently heard and suggested me regarding various aspects in this project.

Abstract

In this project, we investigate the application of kernelized versions of data depth functions for binary classification. We show how these kernelized version of depth functions are able to capture multi-modality in a data cloud, a characteristic that the traditional depth functions such as half-space depth and spatial depth are not able to capture.

We introduce two novel depth measures derived from kernelized version of half-space depth and spatial depth, viz. sphere depth and kernelized spatial depth. Further, algorithms for the computation of these depth measures have been given followed by an assessment of the computational time needed for these kernelized measures relative to their conventional counterparts, with increase in the sample size. These depth measures serve as features for classification, capturing the intrinsic structure of the data in a non-parametric manner. We compare the efficacy of these depth-based features in binary classification tasks using various machine learning algorithms viz. support vector machines, K-nearest neighbors, and generalized additive models with efficacy of these classifiers on usual data.

To conduct our study, we simulate from four location and scale models, from both the multivariate normal and t distributions. We build the classifiers on depth values (w.r.t. each class) based on the training data and evaluate their performance on the test data.

Contents

1 Introduction

Data depth[6] is a non-parametric approach that measures the relative position of a point with respect to the given data cloud (or, a probability distribution). For a distribution $F \in \mathbb{R}^d$, a corresponding depth function is any function $D(\mathbf{x}; F)$ which provides a F-based center-outward ordering of points $\mathbf{x} \in \mathbb{R}^d$.

There are various notions of data depth proposed in literature (viz. half-space depth, spatial depth, Mahalanobis depth etc.) that can be employed in procedures of multivariate data analysis such as classification, outlier detection, etc.

We denote by $\mathcal{F}$ the class of distributions on the Borel set of $\mathbb{R}^d$ and by $F_{\mathbf{X}}$ (or, simply F) the distribution of a given random vector $\mathbf{X}$.

Let $D(.;.) : \mathbb{R}^d \times \mathcal{F} \to [0, 1]$ be the bounded, non-negative depth function. The depth function should ideally satisfy the following properties:

- **Affine invariance:** The depth of a point $\mathbf{x} \in \mathbb{R}^d$ doesn't depend on the scales of the underlying measurements, i.e., $D(A\mathbf{x} + \mathbf{b}; F_{A\mathbf{X}+\mathbf{b}}) = D(\mathbf{x}; F_{\mathbf{X}})$ holds for any random vector $\mathbf{X} \in \mathbb{R}^d$, any $d \times d$ non-singular matrix A and any $\mathbf{b} \in \mathbb{R}^d$.
- **Maximality at centre:** The depth function should attain maximum value at the center, i.e., $D(\boldsymbol{\theta}; F) = \sup_{\mathbf{x} \in \mathbb{R}^d} D(\mathbf{x}; F)$ holds for any F having center $\boldsymbol{\theta}$.
- **Monotonicity relative to deepest point:** For a symmetric distribution along any fixed ray through the center, the depth at $\mathbf{x}$ should decrease monotonically. i.e., for any F having deepest point $\boldsymbol{\theta}$, $D(\mathbf{x}; F) \leq D(\alpha \cdot \mathbf{x} + (1 - \alpha) \cdot \boldsymbol{\theta})$ holds for $\alpha \in [0, 1]$.
- **Vanishing at infinity:** The depth of a point $\mathbf{x}$ should approach zero as $\|\mathbf{x}\|_2$ approaches infinity. i.e. $D(\mathbf{x}; F) \to 0$ as $\|\mathbf{x}\|_2 \to \infty$, for each $F \in \mathcal{F}$.

Moreover, the ultimate depth function should be: robust, adapted for non-convex support and scalable.

A sample version of $D(\mathbf{x}; F)$, denoted by $D_n(\mathbf{x}; F_n)$, may be defined by replacing F by a suitable empirical measure F_n. The depth values, computed from sample depth functions, represent the centrality or outlyingness of a data point w.r.t. the dataset. By employing these depth values as features, classification algorithms can distinguish between different classes based on the intrinsic structure of the data. In this report, we will investigate this approach.

2 Half-space depth (HD) and spatial depth (SPD)

2.1 HD

Half-space depth[6] at $\mathbf{x} \in \mathbb{R}^d$ with respect to the distribution F is defined to be

$$HD(\mathbf{x}; F) = \inf_{H}\{P(H) : \text{H is a closed half-space in } \mathbb{R}^d \text{ and } \mathbf{x} \in H\} \tag{1}$$

Alternatively (1) can also be written as:

$$HD(\mathbf{x}; F) = \inf_{\mathbf{u}\in\mathbb{S}(0,1)} \mathbb{P}\left(\langle \mathbf{u}, \mathbf{X}\rangle \geq \langle \mathbf{u}, \mathbf{x}\rangle\right) \text{ where } \mathbf{X} \sim F. \tag{2}$$

Here, $\mathbb{S}(\mathbf{0}, 1)$ is the sphere of radius 1 centered at 0, i.e. $\|\mathbf{u}\|_2 = 1$.

The sample version of $HD(\mathbf{x}; F)$ is $HD(\mathbf{x}; F_n)$. Here F_n denotes the empirical distribution function of the sample $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$ and is given by:

$$HD(\mathbf{x}; F_n) = \inf_{\mathbf{u}\in\mathbb{S}(0,1)} \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\left(\langle \mathbf{u}, \mathbf{x_i}\rangle \geq \langle \mathbf{u}, \mathbf{x}\rangle\right) \text{ where } \mathbb{I}(.) : \text{ indicator function.} \tag{3}$$

$HD(\mathbf{x}; F)$ and $HD(\mathbf{x}; F_n) \in [0, \frac{1}{2}]$. HD has properties of invariance, robustness and non-parametricity but it suffers from two major limitations, which are:

1. assumes that the underlying distribution has convex support, and
2. requires solving a non-convex and non-smooth optimization problem for which derivative-free algorithms are the only available option.

HD provides a simple measure of centrality. However it is not suited for distributions with non-convex supports, since it is based on Euclidean scalar products i.e. linear projections, so it cannot capture centrality for multi-modal distributions. This is evident from Figure 1.

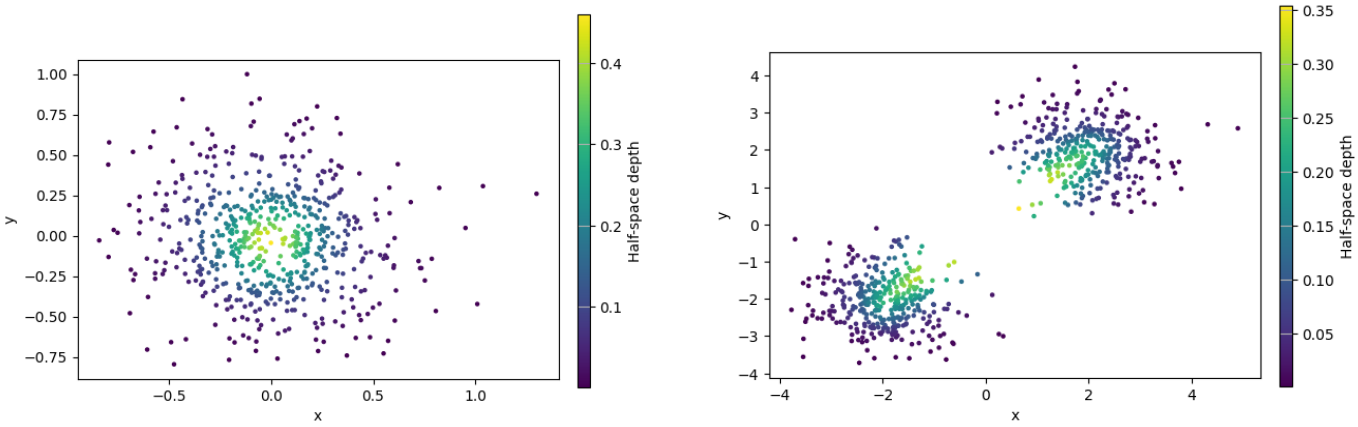
2.2 SPD

Spatial depth[2] at $\mathbf{x} \in \mathbb{R}^d$ with respect to the distribution F is defined to be:

$$SPD(\mathbf{x}; F) = 1 - \left\| \int S(\mathbf{y} - \mathbf{x}) dF(\mathbf{y}) \right\|_2, \tag{4}$$

where

$$S(\mathbf{x}) = \begin{cases} \frac{\mathbf{x}}{\|\mathbf{x}\|_2} & \text{if } \mathbf{x} \neq \mathbf{0} \\ \mathbf{0} & \text{otherwise .} \end{cases}$$



(a) HD captures the data cloud structure with convex support

(b) HD is unable to capture the data cloud structure with non-convex support

Figure 1: HD for data cloud with convex and non-convex support

For an unknown cdf F, the spatial depth is unknown and can be approximated by the sample spatial depth given by:

$$SPD(\mathbf{x}; F_n) = 1 - \frac{1}{|\mathscr{X} \cup \mathbf{x}| - 1} \left\| \sum_{\mathbf{y} \in \mathscr{X}} S(\mathbf{y} - \mathbf{x}) \right\|_2 , \tag{5}$$

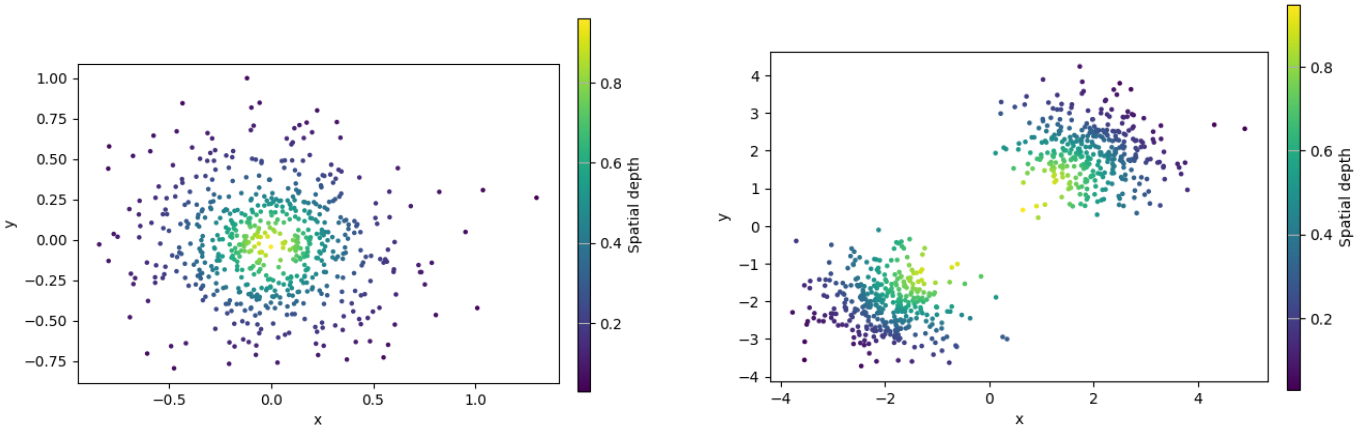
where $\mathscr{X} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$ is the sample and $|\mathscr{X} \cup \mathbf{x}|$ denotes cardinality of the union $\mathscr{X} \cup \mathbf{x}$.

After some simple algebra, we get

$$\left\| \sum_{\mathbf{y} \in \mathscr{X}} S(\mathbf{y} - \mathbf{x}) \right\|_2 = \sum_{\mathbf{y}, \mathbf{z} \in \mathscr{X}} \frac{\mathbf{x}^T\mathbf{x} + \mathbf{y}^T\mathbf{z} - \mathbf{x}^T\mathbf{y} - \mathbf{x}^T\mathbf{z}}{\sqrt{\mathbf{x}^T\mathbf{x} + \mathbf{y}^T\mathbf{y} - 2\mathbf{x}^T\mathbf{y}}\sqrt{\mathbf{x}^T\mathbf{x} + \mathbf{z}^T\mathbf{z} - 2\mathbf{x}^T\mathbf{z}}} . \tag{6}$$

We see that in the sample version of SPD the data appears only as inner products.

Both $SPD(\mathbf{x}; F)$ and $SPD(\mathbf{x}; F_n)$ has range between $[0,\ 1]$. For data generated from distribution with convex support the depth values captures the shape of the data cloud (i.e. depth contour matches with that of the data cloud). But this is not true in case of distribution with non-convex supports. This is evident from Figure 2.



(a) SPD captures the data cloud structure with convex support

(b) SPD is unable to capture the data cloud structure with non-convex support

Figure 2: SPD for Data with convex and non-convex support

3 Kernelized versions of HD and SPD

3.1 Kernel trick

As evident from Figure 1 and Figure 2, HD and SPD are not able to capture the structure of the data cloud when the underlying distribution has non-convex support. In such cases, the kernel trick comes to rescue.

Positive Definite kernels[5]:

Let $\mathbb{R}^d$ be the original feature space and let $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ be a positive definite kernel. Given any n points $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$ with $\mathbf{x_i}\ 's \in \mathbb{R}^d$, the $n \times n$ matrix with $(i,j)^{th}$ element as $\kappa(\mathbf{x_i}, \mathbf{x_j})$ is called the Gram matrix of κ.

Now, κ is a positive definite kernel if the gram matrix is positive semi definite $(p.s.d)$ $\forall\ n$ and $\forall\ \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$, i.e.,

$$\sum_{i,j=1}^{n} c_i c_j \kappa(\mathbf{x_i}, \mathbf{x_j}) \geq 0\ \forall\ c_i \in \mathbb{R} \text{ and } \mathbf{x_i} \in \mathbb{R}^d\ .$$

All positive definite kernels are also inner products on some appropriate space. This space is called the reproducing kernel hilbert space (RKHS) associated with the kernel κ, i.e., a positive definite kernel κ, implicitly defines an embedding map

$$\phi : \mathbf{x} \in \mathbb{R}^d \to \phi(\mathbf{x}) \in \mathbb{F}$$

via an inner product in the feature space $\mathbb{F}$, and thus we have

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle \ .$$

The kernel trick provides a computationally efficient way to recode the data. This is because instead of explicitly mapping the data points to the feature space and performing computations in that higher-dimensional space, the kernel trick allows us to compute the inner products directly in the original input space using the kernel function. This avoids the need to explicitly compute and store the high-dimensional feature vectors, which can be computationally expensive, especially when dealing with large datasets or complex feature spaces.

The basic idea of the kernelized versions of data depth functions is to evaluate the traditional depth values in a feature space induced by a positive definite kernel.

This approach is possible due to the fact that in traditional data depth functions, features appears as inner products (see equation(3) and equation(6)). Consequently, by substituting these features with ones from an induced space, calculated using kernels, it becomes possible to compute depth values within this transformed feature space. This is feasible because positive definite kernels implicitly define inner products in the higher-dimensional feature space, thereby enabling the application of traditional data depth concepts within this framework.

3.2 Sphere depth (SD)

Sphere depth[1] is the kernelized half-space depth (see equation(2)) and is given by:

$$SD^r(\mathbf{z}|\mathbf{F}) \stackrel{\text{def}}{=} \inf_{\mathbf{c} \in \mathbb{S}(\mathbf{z}, r)} \mathbb{P}(\kappa(\mathbf{c}, \mathbf{X}) \geq \kappa(\mathbf{c}, \mathbf{z})) \text{ where, } \mathbf{X} \sim F \tag{7}$$

where $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ is a positive definite kernel. The search space is constrained to a sphere of radius r (> 0) centered at z denoted by $\mathbb{S}(\mathbf{z}, r)$.

The kernelized depth of equation (7) has two main intuitive advantages:

1. data projections are non-linear and can be adapted for distributions with non-convex contours,
2. additional parameter r provides a flexible lever to control the depth's sensitivity depending on the data.

Proposition 1. If κ is the Gaussian kernel $\kappa(\mathbf{x},\mathbf{y}) \stackrel{def}{=} \exp(-\gamma\|\mathbf{x}-\mathbf{y}\|_2^2)$ with $\gamma > 0$, then $SD^r(\mathbf{z}|\mathbf{F})$ can be written as:

$$SD^r(\mathbf{z}|\mathbf{F}) = \inf_{\mathbf{c}\in\mathbb{S}(\mathbf{z},r)} \mathbb{P}_{\mathbf{X}}(\mathbb{B}(\mathbf{c},r)) = \inf_{\mathbf{c}\in\mathbb{S}(\mathbf{z},r)} \mathbb{E}\ [\ \mathbb{I}\{r^2 - \|\mathbf{X}-c\|_2^2 \geq 0\}] \tag{8}$$

Justification: For any $\mathbf{x},\ \mathbf{z} \in \mathbb{R}^d$ and $\mathbf{c} \in \mathbb{S}(\mathbf{z},r) : \kappa(\mathbf{c},\mathbf{x}) \geq \kappa(\mathbf{c},\mathbf{z}) \iff e^{-\gamma\|\mathbf{x}-\mathbf{c}\|_2^2} \geq e^{-\gamma r^2} \iff \|\mathbf{x}-\mathbf{c}\|_2 \leq r$. Therefore, $\mathbb{P}(\kappa(\mathbf{c},\mathbf{X}) \geq \kappa(\mathbf{c},\mathbf{z})) = \mathbb{P}_{\mathbf{X}}(\mathbb{B}(\mathbf{c},r))$.

The above equation is a non-smooth and non-convex optimization problem that can only be solved using gradient-free optimization algorithms. To tackle this limitation, the indicator function is smoothened using the sigmoid function ($sig_s : x \to \frac{1}{1+e^{-\frac{x}{s}}}$) and finally the sphere depth function looks like:

$$SD_s^r(\mathbf{z}|F) = \inf_{\mathbf{c}\in\mathbb{S}(\mathbf{z},r)} \mathbb{E}\ [sig_s(r^2 - \|\mathbf{X}-c\|_2^2)]\ . \tag{9}$$

This novel differentiable formulation allows for a fast manifold gradient descent, further it provides a better depth ranking since its depth values are smoothed and not constrained to multiples of $\frac{1}{n}$. Third, the behavior of the original sphere depth can still be recovered with small values of s since $\lim_{s\to 0} SD_s^r(\mathbf{z}|F) = SD^r(\mathbf{z}|F)$.

The sample version of $SD_s^r(\mathbf{x};F)$ is $SD_s^r(\mathbf{x};F_n)$. Here F_n denotes the empirical distribution function of the sample $\{\mathbf{x_1},\mathbf{x_2},\ldots,\mathbf{x_n}\}$ and is given by:

$$SD_s^r(\mathbf{z}|\mathbf{F_n}) = \inf_{\mathbf{c}\in\mathbb{S}(\mathbf{z},r)} \frac{1}{n}\sum_{i=1}^{n} [sig_s(r^2 - \|\mathbf{x_i}-c\|_2^2)]\ . \tag{10}$$

Note that $SD_s^r(\mathbf{z}|F)$ and $SD_s^r(\mathbf{z}|F_n) \in [0,1]$. In practice, for the Gaussian kernel (or, any radial basis kernel), computing $SD_0^r(\mathbf{z}|\mathbf{X_{1:n}})$ (here, $\mathbf{X_{1:n}} = \{\mathbf{x_1},\mathbf{x_2},\ldots,\mathbf{x_n}\}$, the sample) amounts to finding the ball of radius r with the least amount of data observations, centered at a distance r from $\mathbf{z}$. Moreover for any $\mathbf{z} \in \mathbb{R}^d$ and any $r > 0$, $SD_0^r(\mathbf{z}|\mathbf{X}) \leq HD(\mathbf{z}|\mathbf{X})$ and equality holds as $r \to \infty$.

3.3 Kernelized spatial depth (KSPD)

Kernelized spatial depth is the kernelized version of spatial depth (see equation(4)) and is defined as:

$$KSPD(\mathbf{x};F) = 1 - \left\| \int S(\phi(\mathbf{y}) - \phi(\mathbf{x}))dF(\phi(\mathbf{y})) \right\|_2\ , \tag{11}$$

where $\phi : \mathbf{x} \in \mathbb{R}^d \to \phi(\mathbf{x}) \in \mathbb{F}$ (here, $\mathbb{F}$ is the induced feature space) is the embedding map implicitly defined by a positive definite kernel $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$.

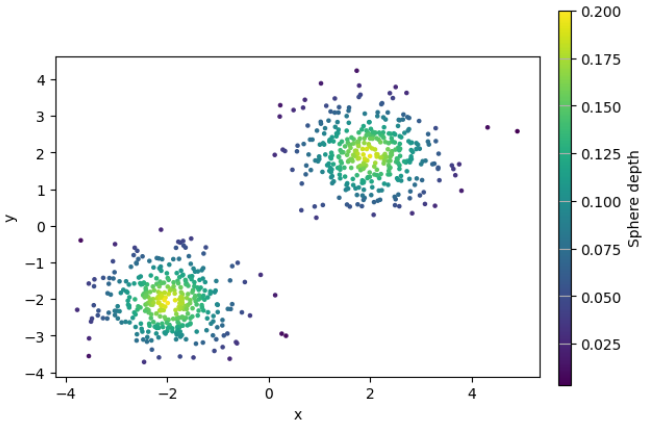
The sample version of $KSPD$ is obtained from the sample spatial depth (see equation(5) and equation(6)) after some simple algebra and replacing the inner products with the values of the kernel κ and is defined as:

$$KSPD_\kappa(\mathbf{x}; F_n) = 1 - \frac{1}{|\mathscr{X} \cup \mathbf{x}| - 1}\Big(\sum_{\mathbf{z},\mathbf{y} \in \mathscr{X}} \frac{\kappa(\mathbf{x},\mathbf{x}) + \kappa(\mathbf{y},\mathbf{z}) - \kappa(\mathbf{x},\mathbf{y}) - \kappa(\mathbf{y},\mathbf{z})}{\delta_\kappa(\mathbf{x},\mathbf{y}) \times \delta_\kappa(\mathbf{x},\mathbf{z})} \Big)^{\frac{1}{2}}, \tag{12}$$

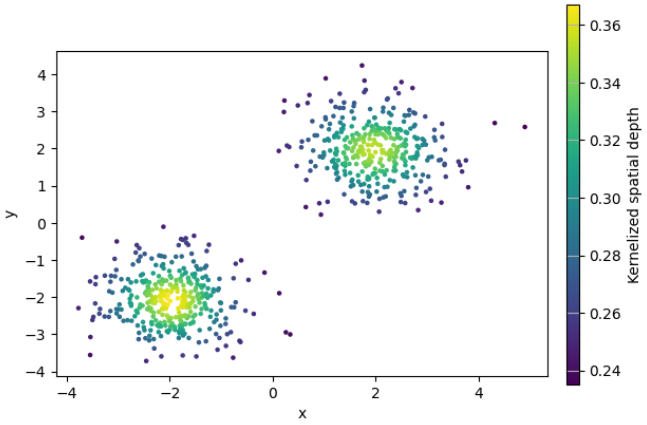
where, $\delta_\kappa(\mathbf{x},\mathbf{y}) = \sqrt{\kappa(\mathbf{x},\mathbf{x}) + \kappa(\mathbf{y},\mathbf{y}) - 2 \times \kappa(\mathbf{x},\mathbf{y})}$.

Here κ is a Gaussian kernel defined as: $\kappa(\mathbf{x},\mathbf{y}) = \exp(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2})$.

From figure 3 it is evident that SD and KSPD are able to capture the non-convex structure of the underlying probability distribution of the data cloud.



(a) SD captures the data cloud structure with non-convex support



(b) KSPD captures the data cloud structure with non-convex support

Figure 3: SD and KSPD captures the multimodality (non-convexity) of the data cloud

3.4 Algorithms to compute kernelized depths

3.4.1 Algorithm for SD

Given a sample $\mathbf{X_{1:n}} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$ the proposed depth of equation (10) is:

$$SD_s^r(\mathbf{z}|\mathbf{X_{1:n}}) = \inf_{\mathbf{c} \in \mathbb{S}(\mathbf{z},r)} \frac{1}{n} \sum_{i=1}^{n} \{sig_s(r^2 - \|\mathbf{x_i} - \mathbf{c}\|_2^2)\}$$

which upto a change of variable i.e. by setting $\mathbf{c} = \mathbf{z} + r \cdot \mathbf{u}$, where $\mathbf{u} \in \mathbb{S}(0,1)$, can be written as:

$$SD_s^r(\mathbf{z}|\mathbf{X_{1:n}}) = \inf_{\mathbf{u}\in\mathbb{S}(0,1)} \frac{1}{n} \sum_{i=1}^{n} \{sig_s(r^2 - \|\mathbf{x_i} - \mathbf{z} - r\mathbf{u}\|_2^2)\} \ .$$

This is solved through Reimannian gradient descent on the unit sphere (see Algorithm 1). For this the codes we available.

Algorithm 1 Riemannian gradient descent for computing SD

Input: $\mathbf{z}, \mathbf{X_{1:n}}, \text{tol}, \alpha$
Result: $l \stackrel{\text{def}}{=} SD_s^r(\mathbf{z}|\mathbf{X_{1:n}})$
Initialize $\mathbf{u} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x_i}$
$\mathbf{u} \leftarrow \frac{\mathbf{u}}{\|\mathbf{u}\|_2}$
$l \leftarrow L(u)$
for $i = 1$ to n_iter **do**
    $v \leftarrow -\nabla_u \mathbf{L}(u)$
    $v \leftarrow v - \langle v, u\rangle u$
    $v \leftarrow \frac{v}{\|v\|_2}$
    $u \leftarrow \cos(\alpha)u + \sin(\alpha)v$
    $l' \leftarrow \mathbf{L}(u)$
    dist $\leftarrow |l' - l|$
    if $l' > l$ **then**
        $\alpha \leftarrow \frac{\alpha}{2}$
    else if dist $<$ tol **then**
        break
    else
        $l \leftarrow l'$
    end if
end for
Return l

3.4.2 Algorithm for KSPD

The Algorithm 2 given below is used to compute the KSPD values (see equation (12)).

3.4.3 Computation time for various depth functions

Figure 4 shows the time taken by HD, SD, SPD and KSPD for various sample sizes. These computations are performed on data simulated from $2-$dimensional Gaussian distribution, $N_2([0,0]^T, I_2)$, I_2 is the 2×2 identity matrix.

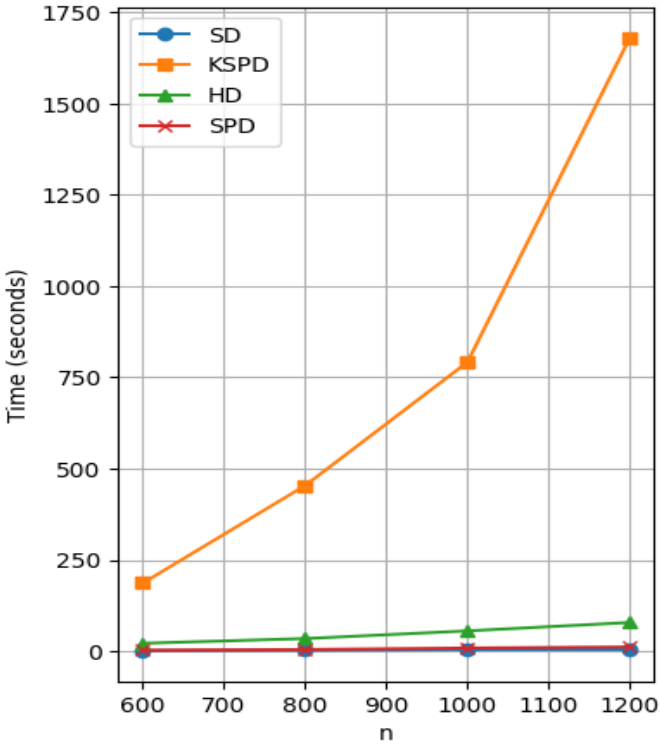


Figure 4: Computation time for HD, SD, SPD, KSPD for different sample sizes.

Algorithm 2 Computation of KSPD

for every pair of $\mathbf{x_i}$ and $\mathbf{x_j}$ in $\mathbf{X}_{1:n}$ **do**
    $K_{ij} = \kappa(\mathbf{x_i}, \mathbf{x_j})$
end for
given input $\mathbf{x}$
for every observation $\mathbf{x_i}$ in $\mathbf{X}_{1:n}$ **do**
    $\zeta_i = \kappa(\mathbf{x}, \mathbf{x_i})$
    $\delta_i = \sqrt{\kappa(\mathbf{x}, \mathbf{x}) + K_{ii} - 2\zeta_i}$
    if $\delta_i = 0$ **then**
        $z_i = 0$
    else
        $z_i = \frac{1}{\delta_i}$
    end if
end for
for every pair of $\mathbf{x_i}$ and $\mathbf{x_j}$ in $\mathbf{X}_{1:n}$ **do**
    $\tilde{K_{ij}} = \kappa(\mathbf{x}, \mathbf{x}) + K_{ij} - \zeta_i - \zeta_j$
end for
$D_\kappa(\mathbf{x}, \mathbf{X}_{1:n}) = 1 - \frac{1}{|\mathcal{X} \cup \{\mathbf{x}\}| - 1} \sqrt{\mathbf{z}^T \tilde{K} \mathbf{z}}$

4 Classification Methodology

The steps followed for building a binary classifier using SD and KSPD are as follows:

1. Simulated training dataset: $\{\mathbf{x}^i_{train}, y^i_{train}\}_{i=1}^{400}$, where $\mathbf{x}^i_{train} \in \mathbb{R}^d$ and $y^i_{train} \in \{0,1\}$, d is the dimension. Simulated test dataset: $\{\mathbf{x}^i_{test}, y^i_{test}\}_{i=1}^{400}$ where $\mathbf{x}^i_{test} \in \mathbb{R}^d$ and $y^i_{test} \in \{0,1\}$. The train and test set are simulated from location and scale models using the multivariate normal distribution and t distribution with 3 degrees of freedom. For $d = \{2, 5, 10\}$. The four models used for data generation are explained in Section 6.

2. The hyper parameters in SD (i.e., $r > 0$) and in KSPD (i.e. σ in the kernel) are tuned by held out cross-validation. From the training data, 30% of it is held out as the validation set. The temperature parameter of the sigmoid function in SD (i.e., s) is fixed to be 1.

3. Computed SD values of the training set with respect to the points in the first class (say, class 0) by setting $r = r_0$. Computed the SD of the training set with respect to the other class (say, class 1) by setting $r = r_1$. Similarly computed the kernelised spatial depth values of the training set with respect to class 0 and class 1 by setting $\sigma = s_0$ and $\sigma = s_1$, respectively. These depth values are used as extracted features from $\mathbb{R}^d$ to $[0,1]^2$ for training the classifiers.

4. Computed the SD values of points in the validation set with respect to training set points in class 0 and class 1 by setting $r = r_0$ and $r = r_1$, respectively. Similarly computed the KSPD values for the validation set with respect to class 0 and class 1 by setting $\sigma = s_0$ and $\sigma = s_1$, respectively.

5. Computed the misclassification rate for the predicted labels using depth values of the validation set as features for each of the r_0 and r_1 values (s_0 and s_1) and choose the r_0 and r_1 (s_0 and s_1) values that gives the minimum misclassification rate among all the r_0 and r_1 (s_0 and s_1) values. The competing r_0 and r_1 values are: $\{0.5, 1, 1.5, 2, 2^i; i = 2, \ldots, 6\}$. The competing s_0 and s_1 values are: $\{1, 4, 8, 16\}$.

6. These tuned r_0 and r_1 (s_0 and s_1) values are used to calculate the depth values again for the training set and test set with respect to class 0 and class 1, respectively. The misclassification rate is noted down. These steps are done for all the 3 classifiers under study viz. support vector machines, generalized addi-

tive models and k-nearest neighbors. The whole procedure is repeated 10 times and the mean misclassification rate along with the standard error is reported in Section 7.

5 Classifiers

5.1 Support vector machines (SVM)

The SVM is a non-parametric supervised machine learning algorithm used for classification and regression tasks. SVM works by mapping the input data into a higher-dimensional feature space, where the classes can be separated by a hyperplane. The algorithm uses a kernel function such as the Radial Basis Function kernel (also known as SVM-RBF) to measure the similarity between pairs of data points in the feature space. It is defined as: $\kappa(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|_2^2)$.

In this there are 2 hyper parameters viz. C (the slack variable that allows for classification of data that are not linearly separable.) and γ. The candidate values for these hyper parameters are taken to be: $C = [0.1, 1, 10, 100, 1000]$ and $\gamma = [0.5, 1, 2, 3, 4]$. The values selected are the ones that gives minimum misclassification rate over the validation set. This task is performed by using **GridSearchCV** from scikit-learn library in Python.

5.2 K-nearest neighbors (KNN)

The KNN is a non-parametric supervised machine learning algorithm used for classification and regression tasks. The idea behind this algorithm is to identify the k number of nearest neighbors of a given query point, and assign a class label to that point by means of majority voting. Here, distance between two points is taken to be the standard Euclidean distance with $p = 2$.

As, k increases the model's bias typically increases while variance decreases. Increasing k tends to have a smoothing effect on the decision boundaries. This can be particularly useful for reducing the impact of noise and outliers in the data. However, excessively large values of k may lead to under fitting, where the model is too simple to capture the underlying patterns in the data. For this reason k is a hyper-parameter that needs to been tuned using held-out cross validation. The candidate values of k has been chosen from 1 to 10 in this study. The optimum value of k is selected to be the one that gives minimum miss classification on the validation set.

5.3 Generalized additive models (GAM) with logistic link

Generalized additive models (GAM)[5] are more automatic flexible statistical methods that may be used to identify and characterize nonlinear regression effects. In our setting, we are dealing with a binary classification problem. As, done in logistic regression setting we relate the mean of binary response $\mu(\mathbf{x}) = Pr(Y = 1|\mathbf{X} = \mathbf{x})$ to the predictors via a linear regression model using the *logit link* function as

$$\log\left[\frac{\mu(\mathbf{x})}{1-\mu(\mathbf{x})}\right] = \alpha + \beta_1 X_1 + \cdots + \beta_p X_p \ .$$

The additive logistic regression model replaces each linear term by a more general functional form as:

$$\log\left[\frac{\mu(\mathbf{x})}{1-\mu(\mathbf{x})}\right] = \alpha + f_1(X_1) + \cdots + f_p(X_p) \ ,$$

where each of the functions f_j are unspecified smooth functions. While the non parametric form for the functions f_j makes the model more flexible, the additivity is retained and hence the model is interpretable. The additive logistic regression model is an example of a generalized additive model. The model is fit using the backfitting algorithm for additive models. Subsequently, we classify a test point by the following decision rule:

$$\hat{Y} = \begin{cases} 1 & \text{if } (1 + e^{-(\hat{\alpha}+\hat{f}_1(X_1)+\cdots+\hat{f}_p(X_p))})^{-1} > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

In this study the feature is a $2-$dimensional vector ($p = 2$) of depth values with respect to each of the classes using different parameter values (viz. r and s in our case).

6 Simulation models

The training and test datasets are simulated from four different location and scale models as described below. Here, d denotes the dimension and $d = \{2, 5, 10\}$. $\mathbf{1}_d$ is a d-dimensional vector with all entries as 1. $\mathbf{I}$ is the $d \times d$ identity matrix.

- Model 1. F_{N0} (the distribution of class 0) is a d-dimensional multivariate Gaussian distribution, $\mathrm{N_d}(\mathbf{0}_d, \mathbf{I})$. F_{N1} (the distribution of class 1) is a d-dimensional multivariate Gaussian distribution, $\mathrm{N_d}(2 \cdot \mathbf{1}_d, \mathbf{I})$.

  F_{t0} (the distribution of class 0) is a d-dimensional multivariate t distribution with 3 degrees of freedom, $t_3(\mathbf{0}_d, \mathbf{I})$. F_{t1} (the distribution of class 1) is a d-dimensional multivariate t distribution, $t_3(2 \cdot \mathbf{1}_d, \mathbf{I})$.

- Model 2. F_{N0} is a d-dimensional multivariate Gaussian distribution, $\mathrm{N_d}(\mathbf{0}_d, \mathbf{I})$. F_{N1} is a mixture of two d-dimensional multivariate Gaussian distribution (with equal weights), $\mathrm{N_d}(2 \cdot \mathbf{1}_d, \mathbf{I})$ and $\mathrm{N_d}(-2 \cdot \mathbf{1}_d, \mathbf{I})$.

  F_{t0} is a d-dimensional multivariate t distribution with 3 degrees of freedom, $t_3(\mathbf{0}_d, \mathbf{I})$. F_{t1} is a mixture of two d-dimensional multivariate t distribution, $t_3(2 \cdot \mathbf{1}_d, \mathbf{I})$ and $t_3(-2 \cdot \mathbf{1}_d, \mathbf{I})$.

- Model 3. F_{N0} is mixture of three d-dimensional multivariate Gaussian distribution (with equal weights), $\mathrm{N_d}(\mathbf{0}_d, \mathbf{I})$, $\mathrm{N_d}(-6 \cdot \mathbf{1}_d, \mathbf{I})$ and $(6 \cdot \mathbf{1}_d, \mathbf{I})$. F_{N1} is a mixture of two d-dimensional multivariate Gaussian distribution (with equal weights), $\mathrm{N_d}(3 \cdot \mathbf{1}_d, \mathbf{I})$ and $\mathrm{N_d}(-3 \cdot \mathbf{1}_d, \mathbf{I})$.

  F_{t0} is a mixture of three d-dimensional multivariate t distribution with 3 degrees of freedom (with equal weights), $t_3(\mathbf{0}_d, \mathbf{I})$, $t_3(-6 \cdot \mathbf{1}_d, \mathbf{I})$ and $t_3(6 \cdot \mathbf{1}_d, \mathbf{I})$. F_{t1} is a mixture of two d-dimensional multivariate t distribution, $t_3(3 \cdot \mathbf{1}_d, \mathbf{I})$ and $t_3(-3 \cdot \mathbf{1}_d, \mathbf{I})$.

- Model 4. F_{N0} is a mixture of two d-dimensional multivariate Gaussian distribution (with equal weights), $\mathrm{N_d}(\mathbf{0}_d, \mathbf{I})$ and $\mathrm{N_d}(\mathbf{0}_d, 10 \cdot \mathbf{I})$.$F_{N1}$ is a d-dimensional multivariate Gaussian distribution, $\mathrm{N_d}(\mathbf{0}_d, 5 \cdot \mathbf{I})$.

  F_{t0} is a mixture of two d-dimensional multivariate t distribution with 3 degrees of freedom (with equal weights), $\mathrm{t_d}(\mathbf{0}_d, \mathbf{I})$ and $\mathrm{t_d}(\mathbf{0}_d, 10 \cdot \mathbf{I})$.$F_{t1}$ is a d-dimensional multivariate t distribution, $\mathrm{t_d}(\mathbf{0}_d, 5 \cdot \mathbf{I})$.

7 Numerical Results

7.1 Bayes classifier

The Bayes classifier[3] classifies a test point based on the product of the likelihood and the prior probability. It is the classifier having the smallest probability of misclassification of all classifiers using the same set of features.

Let $\mathbf{x}$ be the test point, $C(\mathbf{x})$ be the Bayes classifier. Then

$$\begin{aligned} C(\mathbf{x}) &= \arg\max_{r\in\{0,1\}} \mathbb{P}(Y=r|\mathbf{X}=\mathbf{x}) \\ &= \arg\max_{r\in\{0,1\}} \underbrace{\mathbb{P}(\mathbf{X}=\mathbf{x}|Y=r)}_{\text{likelihood}} \cdot \underbrace{\mathbb{P}(Y=r)}_{\text{prior}} \end{aligned}$$

Assuming the likelihood function and the prior is known (which is true here since the datasets are simulated from density functions) the bayes classifier is used as benchmark.

7.2 Results

Dim	Bayes	On $\mathbb{R}^d$			SD			KSD		
		SVM	KNN	GAM	SVM	KNN	GAM	SVM	KNN	GAM
2	8.53 (1.01)	8.93 (1.23)	9.37 (1.41)	8.80 (1.18)	11.85 (1.20)	13.03 (2.84)	12.30 (2.15)	13.57 (3.87)	14.30 (2.22)	18.70 (6.16)
5	1.28 (0.59)	1.75 (0.77)	1.57 (0.64)	1.67 (0.75)	3.47 (0.85)	9.63 (14.30)	3.88 (2.38)	4.20 (2.70)	3.43 (1.92)	6.60 (5.80)
10	0.12 (0.13)	3.55 (2.68)	0.18 (0.22)	0.23 (0.17)	0.48 (0.58)	0.85 (1.75)	5.55 (15.90)	6.43 (0.12)	8.45 (9.02)	4.32 (3.60)

Table 1: Mean misclassification rates (standard deviations) on the test data generated from model 1 with multivariate normal distribution using various classifiers (SVM, KNN, GAM) on usual data, SD and KSPD. Bayes misclassification is computed on usual data for benchmarking.

1. From Table 1 and table 2 we see that when the data is 10−dimensional then SVM trained on SD performs better as compared to SVM trained on usual data, where as KNN and GAM performs better on usual data than using SD and KSPD as features.

2. From Table 3 we see SVM trained on KSPD as features performs better than SVN trained on SD and usual data for dimension 10. Similar results are obtained on t_3, a heavy tailed distribution as evident from Table 4.

Dim	**Bayes**	**On $\mathbb{R}^d$**			**SD**			**KSD**		
		SVM	KNN	GAM	SVM	KNN	GAM	SVM	KNN	GAM
2	13.68 (2.06)	14.23 (2.37)	14.75 (2.32)	14.00 (2.27)	18.60 (3.08)	18.58 (2.73)	17.68 (3.52)	18.25 (5.21)	18.00 (2.97)	20.63 (4.37)
5	6.48 (1.30)	10.20 (1.97)	7.30 (1.38)	7.17 (1.40)	10.35 (3.07)	14.10 (4.96)	9.50 (2.14)	10.10 (3.55)	11.30 (1.40)	13.03 (4.67)
10	2.38 (0.95)	19.08 (3.45)	2.65 (1.09)	3.44 (1.31)	5.00 (1.26)	8.58 (5.05)	11.18 (10.34)	12.33 (9.41)	11.65 (3.80)	11.53 (6.31)

Table 2: Mean misclassification rates (standard deviations) on the test data generated from model 1 with multivariate t distribution using various classifiers (SVM, KNN, GAM) on usual data, SD and KSPD. Bayes misclassification is computed on usual data for benchmarking.

Dim	**Bayes**	**On $\mathbb{R}^d$**			**SD**			**KSD**		
		SVM	KNN	GAM	SVM	KNN	GAM	SVM	KNN	GAM
2	10.90 (1.80)	11.50 (1.22)	12.20 (1.50)	13.90 (2.50)	16.15 (2.80)	18.20 (4.30)	16.40 (2.40)	19.00 (6.00)	23.90 (5.20)	25.10 (6.70)
5	2.00 (0.55)	3.05 (0.74)	2.45 (0.82)	5.80 (1.36)	10.80 (2.93)	18.75 (16.90)	11.67 (5.30)	12.47 (6.37)	16.75 (4.50)	12.25 (7.25)
10	0.10 (0.12)	7.80 (3.89)	0.25 (0.27)	1.50 (0.50)	7.97 (5.20)	17.85 (18.48)	8.33 (5.49)	6.00 (5.32)	18.12 (5.82)	9.05 (7.11)

Table 3: Mean misclassification rates (standard deviations) on the test data generated from model 2 with multivariate normal distribution using various classifiers (SVM, KNN, GAM) on usual data, SD and KSPD. Bayes misclassification is computed on usual data for benchmarking.

3. From Table 5 we see that all the classifiers trained on usual data perform better than classifier trained on depth features. However with data from t_3 SVM trained on KSPD and SD on $10-$dimensional data is better than SVM when trained on usual data (see Table 6).

4. From Table 7 we see that all the classifiers whether trained on usual data or SD or KSPD give competitive performance. Moreover, we see that for $10-$dimensional data SVM trained on SD and KSPD perform better than SVM trained on usual data for normal distribution, however this is not the case for data simulated from t_3 distribution (see , Table 8).

Dim	**Bayes**	**On $\mathbb{R}^d$**			**SD**			**KSD**		
		SVM	KNN	GAM	SVM	KNN	GAM	SVM	KNN	GAM
2	18.52 (1.40)	19.27 (1.80)	19.72 (2.47)	23.15 (1.45)	25.45 (2.50)	27.85 (4.70)	25.22 (4.31)	27.27 (5.91)	31.93 (4.60)	31.32 (4.20)
5	7.90 (1.43)	11.87 (1.88)	8.45 (1.26)	15.86 (1.07)	25.10 (5.45)	25.20 (4.35)	21.30 (3.66)	23.18 (8.55)	28.68 (8.46)	22.75 (8.61)
10	3.65 (0.88)	22.30 (2.41)	4.20 (0.82)	12.28 (1.54)	21.35 (4.38)	24.30 (10.12)	20.48 (5.27)	14.80 (4.74)	22.20 (4.52)	17.25 (5.61)

Table 4: Mean misclassification rates (standard deviations) on the test data generated from model 2 with multivariate t distribution using various classifiers (SVM, KNN, GAM) on usual data, SD and KSPD. Bayes misclassification is computed on usual data for benchmarking.

Dim	**Bayes**	**On $\mathbb{R}^d$**			**SD**			**KSD**		
		SVM	KNN	GAM	SVM	KNN	GAM	SVM	KNN	GAM
2	2.95 (0.45)	3.63 (1.03)	3.30 (0.75)	5.20 (0.94)	13.98 (8.30)	6.63 (2.00)	12.15 (3.02)	15.00 (7.25)	18.95 (7.53)	17.03 (9.03)
5	0.10 (0.12)	0.53 (0.40)	0.20 (0.21)	0.83 (0.54)	8.18 (13.12)	12.83 (14.48)	4.43 (4.14)	10.55 (8.60)	6.50 (4.06)	5.60 (8.46)
10	0.00 (0.00)	9.55 (8.21)	0.00 (0.00)	0.08 (0.11)	17.85 (17.85)	9.23 (7.76)	13.93 (6.67)	11.95 (11.30)	14.57 (9.80)	8.38 (9.00)

Table 5: Mean misclassification rates (standard deviations) on the test data generated from model 3 with multivariate normal distribution using various classifiers (SVM, KNN, GAM) on usual data, SD and KSPD. Bayes misclassification is computed on usual data for benchmarking.

Dim	**Bayes**	**On $\mathbb{R}^d$**			**SD**			**KSD**		
		SVM	KNN	GAM	SVM	KNN	GAM	SVM	KNN	GAM
2	10.22 (1.45)	11.85 (1.28)	11.42 (1.71)	15.12 (1.94)	21.30 (6.54)	18.55 (5.14)	21.80 (5.37)	19.48 (6.30)	27.07 (6.81)	24.37 (7.29)
5	3.20 (0.81)	7.27 (1.70)	3.55 (1.08)	10.15 (1.86)	17.82 (9.40)	20.28 (10.76)	14.52 (6.83)	19.98 (4.40)	18.85 (8.46)	18.37 (6.10)
10	1.27 (0.51)	21.4 (5.02)	1.35 (0.69)	8.07 (1.35)	17.23 (6.10)	22.13 (11.84)	19.07 (5.26)	15.03 (6.62)	16.60 (8.87)	14.37 (9.46)

Table 6: Mean misclassification rates (standard deviations) on the test data generated from model 3 with multivariate t distribution using various classifiers (SVM, KNN, GAM) on usual data, SD and KSPD. Bayes misclassification is computed on usual data for benchmarking.

Dim	**Bayes**	**On $\mathbb{R}^d$**			**SD**			**KSD**		
		SVM	KNN	GAM	SVM	KNN	GAM	SVM	KNN	GAM
2	36.37 (2.34)	42.55 (2.47)	43.78 (3.42)	40.13 (2.52)	44.00 (3.12)	44.00 (2.68)	42.38 (2.30)	44.82 (3.35)	45.45 (3.29)	45.70 (3.81)
5	26.15 (2.00)	36.00 (2.41)	39.03 (3.72)	34.50 (3.26)	38.67 (2.21)	39.20 (3.90)	36.35 (3.25)	34.15 (3.45)	40.40 (5.30)	39.75 (4.38)
10	17.80 (1.63)	38.50 (8.16)	40.40 (1.77)	29.75 (1.82)	35.90 (2.30)	39.15 (3.40)	36.44 (3.40)	32.90 (4.10)	38.90 (3.50)	36.44 (3.40)

Table 7: Mean misclassification rates (standard deviations) on the test data generated from model 4 with multivariate normal distribution using various classifiers (SVM, KNN, GAM) on usual data, SD and KSPD. Bayes misclassification is computed on usual data for benchmarking.

Dim	**Bayes**	**On $\mathbb{R}^d$**			**SD**			**KSD**		
		SVM	KNN	GAM	SVM	KNN	GAM	SVM	KNN	GAM
2	41.00 (1.74)	43.47 (3.02)	45.95 (2.11)	45.90 (3.78)	45.22 (2.87)	46.25 (2.09)	45.00 (1.71)	45.45 (3.21)	48.87 (1.80)	46.20 (2.62)
5	38.67 (2.43)	41.05 (2.27)	44.08 (1.77)	43.75 (3.68)	42.10 (1.84)	47.70 (3.55)	42.63 (2.25)	43.50 (2.28)	46.13 (1.76)	46.43 (2.04)
10	36.77 (4.14)	40.10 (4.12)	45.43 (1.54)	43.23 3.64)	41.28 (3.72)	45.00 (2.45)	42.65 (2.76)	42.33 (3.72)	45.80 (2.43)	43.43 (2.84)

Table 8: Mean misclassification rates (standard deviations) on the test data generated from model 4 with multivariate t distribution using various classifiers (SVM, KNN, GAM) on usual data, SD and KSPD. Bayes misclassification is computed on usual data for benchmarking.

References

[1] Arturo Castellanos et al. "Fast kernel half-space depth for data with non-convex supports". In: (2023). arXiv: 2312.14136 [stat.ML].

[2] Yixin Chen et al. "Outlier Detection with the Kernelized Spatial Depth Function". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31 (2009).

[3] Luc Devroye, László Györfi, and Gábor Lugosi. *A Probablistic Theory of Pattern Recognition*. Jan. 1996.

[4] Subhajit Dutta, Soham Sarkar, and Anil K. Ghosh. "Multi-scale Classification using Localized Spatial Depth". In: *Journal of Machine Learning Research* (2016).

[5] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[6] Robert Serfling and Yijun Zuo. "General notions of statistical depth function". In: *The Annals of Statistics* 28.2 (2000).