SANDIA REPORT
SAND2022-11061

Printed August 2022

Sandia National Laboratories

Support Vector Machines for Estimating Decision Boundaries with Numerical Simulations

Wilkins Aquino, Andrew Kurzawski, Cameron A. McCormick, Clay Sanders, Chandler Smith, Benjamin Treweek, Timothy Walsh

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185
Livermore, California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from

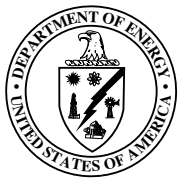
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods



ABSTRACT

Many engineering design problems can be formulated as decisions between two possible options. This is the case, for example, when a quantity of interest must be maintained below or above some threshold. The threshold thereby determines which input parameters lead to which option, and creates a boundary between the two options known as the decision boundary. This report details a machine learning approach for estimating decision boundaries, based on support vector machines (SVMs), that is amenable to large scale computational simulations. Because it is computationally expensive to evaluate each training sample, the approach iteratively estimates the decision boundary in a manner that requires relatively few training samples to glean useful estimates. The approach is then demonstrated on three example problems from structural mechanics and heat transport.

ACKNOWLEDGMENTS

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Many engineering design problems involve limiting a given quantity of interest to below or above some threshold. This quantity may be deterministic, such as the maximum stress within a structure; or it may be probabilistic, such as the expected likelihood that a component will fail after some amount of use. In the former example, the threshold might define the allowable operating conditions for the structure, thereby demarcating the conditions as 'safe' or 'unsafe'. In the latter example, it might define the lifespan of the component, thereby demarcating the component's status as 'no maintenance necessary' or 'maintenance required'. In both of these cases, the decision can be cast as a binary classification problem where the user must decide between two options. The threshold value in question forms a boundary between the classifications and is therefore called the decision boundary.

Given a set of input parameters (e.g., external loads, power spectral densities) it can be classified through the use of a decision function, which assigns the parameter set to one of the two classes. For example, this decision function might take in a set of external loads and classify them as 'safe' or 'unsafe'. In modern usage, the decision function may be based on a complex physical model which may be computationally expensive to evaluate. This will be the case for high-fidelity CFD or FEM simulations that may take hours, days, or even weeks to evaluate. Another use case is when the decision function utilizes empirical processes, such as physical experiments, which take a similarly long amount of time to carry out. Furthermore, there may be many parameters that affect the quantity of interest, leading to a decision boundary with high dimensionality. In practice, it is therefore necessary to limit the number of times the decision function is evaluated, which in turn will limit the number of samples from which the decision boundary can be estimated.

This report details the formulation of a machine learning algorithm for decision boundary estimation, based on the work of Cholette et al. [1] and Basudhar and Missoum [2]. The algorithm is designed to be efficient in the number of training samples, through the use of support vector machines (SVMs). First introduced by Vladimir Vapnik and his colleagues at AT&T Bell Labs [3], the SVM method is a machine learning technique for classification problems. SVMs have been demonstrated on classification problems including structural health monitoring [4], medical diagnosis [5], and others. SVMs have been shown to be among the most efficient algorithms for classification problems [6, 7, 8, 9].

The report is structured as follows. In Section 2, we formally define the classification problem. We also define support vectors and how they are used in solving the problem, before finally detailing our decision boundary estimation algorithm. In Section 3, we apply the algoirthm to three example problems. The first example is that of a bar under uniaxial load; the second example is a cone structure subjected to a spatially-varying pressure load; and the third example

is related to heat transport. Finally, in Section 4 we summarize the contributions of this report and suggest extensions to the algorithm.

2. FORMULATION

2.1. PROBLEM DESCRIPTION

Our main goal is to devise an efficient and general strategy to assess whether a given system design is acceptable under various operational conditions. To this end, we will postulate the latter acceptance/rejection problem abstractly as a classification task.

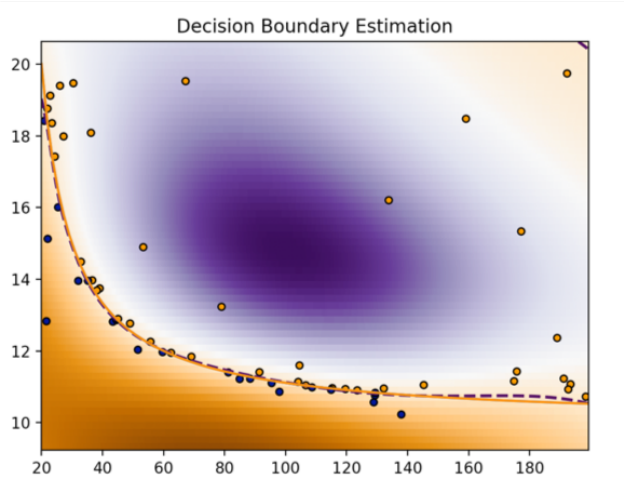


Figure 2.1-1. Decision Boundary Separating $\mathscr{A}^+$ (Blue) and $\mathscr{A}^-$ (Orange)

Consider a function $g : X \to \mathbb{R}$, where X is the parameter space. For instance, $g(x) = h(x) - a_o, x \in X$, where $x = \{V, \gamma\}$. Here, h can be a function that maps x to the maximum acceleration at a point of interest in a coupled fluid-structure interaction simulation, V and γ can be representative parameters in the fluid, and a_o an acceleration threshold. Now, consider the sets $\mathscr{A}^+ := \{x \in X : g(x) > 0\}$ and $\mathscr{A}^- := \{x \in X : g(x) < 0\}$ with $X = \mathscr{A}^+ \cup \mathscr{A}^-$, $\mathscr{A}^+ \cap \mathscr{A}^- = \emptyset$ as shown in Figure 2.1-1.

Now, let

$$y_T(x) = \text{sgn}(g(x)) \tag{2.1}$$

So, notice that $y_T(x) = -1, \forall x \in \mathscr{A}^-$ and $y(x) = 1, \forall x \in \mathscr{A}$. Assume we are given a data set $\{x_i, y_i\}, i = 1, .., N$. Our goal is to, using this data set, find a function f that approximates (2.1). That is, f classifies new observations x_n as being in $\mathscr{A}^+$ or $\mathscr{A}^-$.

One of the main challenges in many applications is that obtaining samples x_i, y_i is computationally expensive. Hence, our strategy should use data efficiently and effectively. As we will show later in the report, we adopt an adaptive approach for efficiently sampling the parameter space, hence, ameliorating the computational expense.

Support Vector Machines (SVMs) have been proposed as one of the most effective algorithms in solving acceptance boundary problems [10]. Therefore, we adopt this family of algorithms as our foundational component. We provide a brief overview of SVMs next and then introduce an adaptive strategy for sampling the parameter space X partly based on the approach presented in [11, 2].

2.2. SUPPORT VECTOR MACHINES (SVMS)

SVMs are among the most effective and efficient classification algorithms [6, 7, 8, 9]. The main idea behind SVMs is to find a hypersurface that creates maximum separation of classes in a data set.

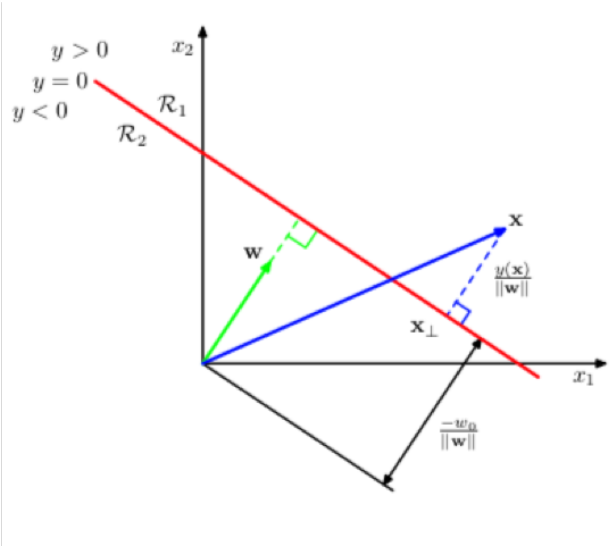


Figure 2.2-1. Separating plane.

We start by considering the simplest classification case: linearly separable data. That is, classes in the data set can be separated exactly by a hyperplane. First, consider the equation for a plane as shown in Figure 2.2-1 given as

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 \tag{2.2}$$

Notice that any point $\boldsymbol{x}_a$ on the plane satisfies $y(\boldsymbol{x}_a) = 0$. Then, it follows that $\boldsymbol{w}^T \boldsymbol{x}_a = -w_0$. The distance from any point $\boldsymbol{x}$ to the plane is given as

$$\gamma(\boldsymbol{x}) := \frac{\boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{x}_a)}{\|\boldsymbol{w}\|} = \frac{\boldsymbol{w}^T \boldsymbol{x} + w_0}{\|\boldsymbol{w}\|} = \frac{y(\boldsymbol{x})}{\|w\|} \tag{2.3}$$

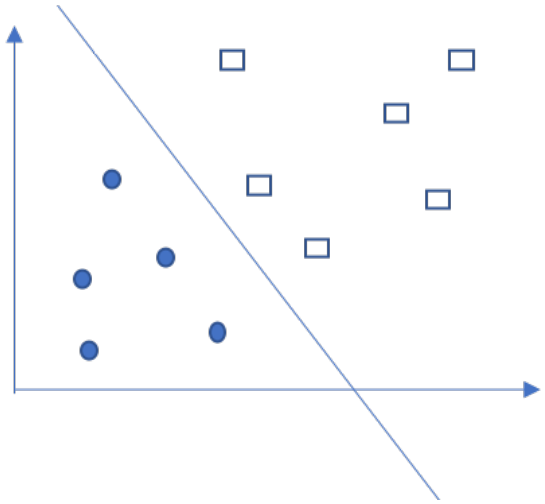


Figure 2.2-2. Linearly Separable Data.

Now consider a data set $\{\boldsymbol{x}_i, \hat{y}_i\}, i = 1...N$ where $\hat{y}_i \in \{-1, 1\}$ are class labels. Also, assume that all these points are perfectly separated by a plane represented by (2.2) as shown in Figure 2.2-2. The corresponding distance from any of these points, say $\boldsymbol{x}_i$, to the plane will be denoted as γ_i. Notice that this is a signed distance (e.g. $y(\boldsymbol{x}_i) > 0 \implies \gamma_i > 0$). We can make the distance unsigned by redefining γ_i as

$$\gamma_i = \frac{\hat{y}_i \left(\boldsymbol{w}^T \boldsymbol{x} + w_0\right)}{\|\boldsymbol{w}\|} \tag{2.4}$$

Notice that there are infinitely many planes that can separate the data perfectly. The task behind SVMs is to find a particular plane: the one that creates the widest margin. Towards this goal, we first introduce the geometric margin as

$$\gamma^* = \min_i \gamma_i \tag{2.5}$$

That is, the geometric margin is given by the distance of the closest point to the plane. Now, let $\boldsymbol{x}_g$ be a point that is on the geometric margin. Then,

$$\gamma^* = \frac{\hat{y}_g \left(\boldsymbol{w}^T \boldsymbol{x}_g + w_0\right)}{\|\boldsymbol{w}\|} \tag{2.6}$$

A plane that maximizes the geometric margin is defined by solving the optimization problem

$$\boldsymbol{w}^*, w_0^* = \arg\max_{\boldsymbol{w}, w_0} \frac{\hat{y}_g \left(\boldsymbol{w}^T \boldsymbol{x}_g + w_0\right)}{\|\boldsymbol{w}\|} \tag{2.7}$$

However, this optimization problem is not convex. We can make progress by noticing that the distance from any point to the plane does not change if we multiply $\boldsymbol{w}$ and w_0 by a constant. Hence, we can write

$$\hat{y}_g \left(\hat{\boldsymbol{w}}^T \boldsymbol{x}_g + \hat{w}_0\right) = 1 \tag{2.8}$$

where we have introduced $\hat{\boldsymbol{w}} = \alpha \boldsymbol{w}$ and $\hat{w}_0 = \alpha w_0$ with $\alpha = \boldsymbol{w}^T \boldsymbol{x}_g + w_0$. Then, since $\boldsymbol{x}_g$ has the smallest distance to the boundary, notice that for any point $\boldsymbol{x}_i$, we have

$$\hat{y}_i \left(\hat{\boldsymbol{w}}^T \boldsymbol{x}_i + \hat{w}_0 \right) \geq 1 \tag{2.9}$$

It is customary in the literature to just use $\boldsymbol{w}$ and $\boldsymbol{w}_0$ to represent the scaled weights. We will adopt this notation in the sequel. We can recast the optimization problem in (2.7) as

$$\boldsymbol{w}^*, w_0^* = \arg\max_{\boldsymbol{w}, w_0} \frac{1}{\|\boldsymbol{w}\|} \quad \text{subject to} \quad \hat{y}_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + w_0 \right) \geq 1 \tag{2.10}$$

Equivalently, we can solve the problem

$$\boldsymbol{w}^*, w_0^* = \arg\min_{\boldsymbol{w}, w_0} \frac{1}{2} \|\boldsymbol{w}\|^2 \quad \text{subject to} \quad \hat{y}_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + w_0 \right) \geq 1 \tag{2.11}$$

which is a convex optimization problem that can be readily solved with conventional techniques.

2.2.1. Nonlinear Separability

Most applications encompass data that is not linearly separable in input space as shown in Figure 2.2-3. We can handle this case by introducing basis functions $\{\phi_j\}, j = 1, \ldots, d$ that map

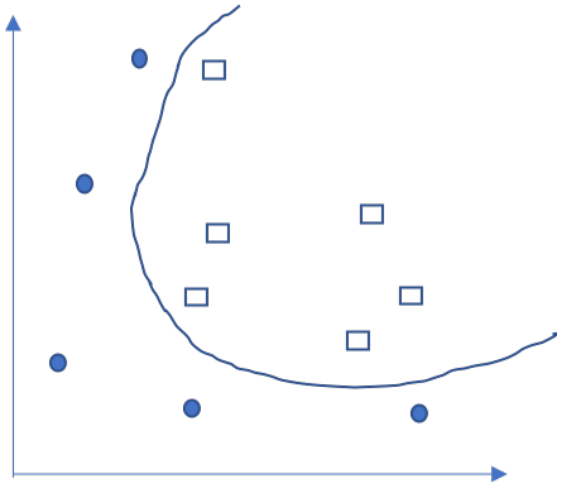


Figure 2.2-3. Non-linearly Separable Data.

the input space to another (usually of higher dimension) space. To this end, the decision function, (2.2) is written as

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) + w_0 \tag{2.12}$$

Then, the optimization problem becomes

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}, w_0} \frac{1}{2} \|\boldsymbol{w}\|^2 \quad \text{subject to} \quad \hat{y}_i \left(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) + w_0 \right) \geq 1 \tag{2.13}$$

Practical implementations of SVMs use the dual formulation of this problem. We can derive the dual problem by first introducing a Lagrangian

$$\mathscr{L}(\boldsymbol{w}, w_0, \boldsymbol{\lambda}) := \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_i \lambda_i \hat{y}_i \left(\left(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) + w_0 \right) - 1 \right) \tag{2.14}$$

where $\lambda_i, i = 1, ..., N$ are lagrange multipliers. Then, the dual function is defined as $q(\boldsymbol{\lambda}) := \min_{\boldsymbol{w}, w_0} \mathscr{L}(\boldsymbol{w}, w_0, \boldsymbol{\lambda})$. Taking partial derivatives of the Lagrangian with respect to $\boldsymbol{w}, w_0$, setting to zero, and solving in terms of $\boldsymbol{\lambda}$, we get

$$\boldsymbol{w} = \frac{1}{2} \sum_i \lambda_i \hat{y}_i \boldsymbol{\phi}(\boldsymbol{x}_i) \tag{2.15}$$

and

$$\sum_i \lambda_i \hat{y}_i = 0 \tag{2.16}$$

Using these expressions with the Lagrangian, we arrive at the dual problem

$$\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}} q(\boldsymbol{\lambda}) := -\frac{1}{2}\boldsymbol{\lambda}^T Q \boldsymbol{\lambda} + \mathbf{1}^T \boldsymbol{\lambda} \quad \text{subject to} \quad \sum_i \hat{y}_i \lambda_i = 0, \lambda_i \geq 0 \tag{2.17}$$

where Q is a matrix with components $Q_{ij} = \hat{y}_i \hat{y}_j \boldsymbol{\phi}(\boldsymbol{x}_i)^T \boldsymbol{\phi}(\boldsymbol{x}_j)$.

Once we have solved the above optimization problem, we can compute $\boldsymbol{w}$ and w_0. Using (2.15) in (2.12), we get

$$y(\boldsymbol{x}) = \sum_i \lambda_i \hat{y}_i \boldsymbol{\phi}(\boldsymbol{x}_i)^T \boldsymbol{\phi}(\boldsymbol{x}) \tag{2.18}$$

We will refer to (2.18) as the SVM decision function, which plays a crucial role in our adaptive algorithm for constructing decision boundaries. Given a new point $\boldsymbol{x}_b$, we can predict the corresponding class by evaluating $\text{sgn}(y(\boldsymbol{x}_b))$. The interested reader can consult [8] for a simple procedure on how to calculate w_0^*.

We can now introduce the kernel trick [12]. The inner products that appear in (2.18) can be replaced by the evaluation of a kernel as $\mathscr{K}(\boldsymbol{x}, \boldsymbol{x}') \equiv \boldsymbol{\phi}(x)^T \boldsymbol{\phi}(\boldsymbol{x}')$. Then, we obtain

$$y(\boldsymbol{x}) = \sum_i \lambda_i \hat{y}_i \mathscr{K}(\boldsymbol{x}_i, \boldsymbol{x}) \tag{2.19}$$

The use of kernels as substitutes for inner products has its roots in the theory of Reproducing Kernel Hilbert Spaces. Common kernels used in practice include polynomials and radial basis functions (i.e. Gaussian) . For instance, a Gaussian kernel will be used in the work presented herein and is defined as

$$\mathscr{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) := \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2} \right)) \tag{2.20}$$

where σ is a user-defined parameter that describes the spread. The interested reader can consult [9] for an in depth exposition on the theory and applications of Kernels to machine learning.

2.2.2. Support Vectors

Recall that the complementarity condition requires

$$\lambda_i \hat{y}_i \left(\left(\boldsymbol{\phi}^T(\boldsymbol{x}_i)\boldsymbol{w}^* + w_0^* \right) - 1 \right) = 0, \, i = 1, ..., N \tag{2.21}$$

Then, notice that $\lambda_i > 0$ only for the points that satisfy $\boldsymbol{\phi}^T(\boldsymbol{x}_i)\boldsymbol{w}^* + w_0^* = 1$. These points lie

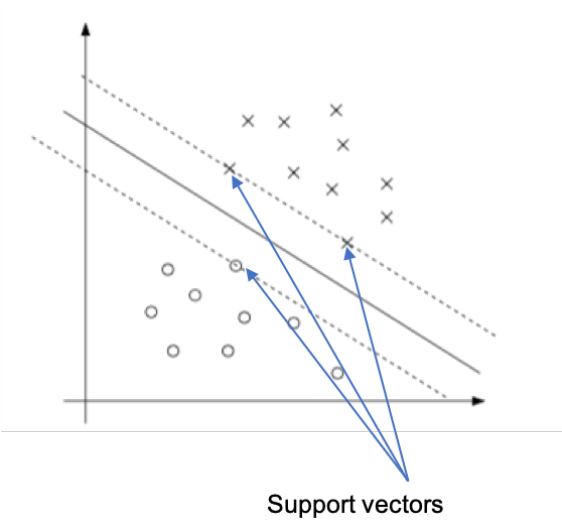


Figure 2.2-4. Non-linearly Separable Data.

right on the geometric margin and are called Support Vectors. This result has strong practical implications in the evaluation of the decision function (2.19). Notice the summation can be carried out over the support vectors only, which are generally far fewer than the number of training points. To this end, define an index set as

$$\mathscr{S}_v := \{ j : \boldsymbol{\phi}^T(\boldsymbol{x}_j)\boldsymbol{w}^* + w_0^* = 1 \} \tag{2.22}$$

Then, the decision function can be evaluated as

$$y(\boldsymbol{x}) = \sum_{i \in \mathscr{S}_v} \lambda_i \hat{y}_i \mathscr{K}(\boldsymbol{x}_i, \boldsymbol{x}) \tag{2.23}$$

2.2.3. Overlapping Classes

We now present a brief exposition on how overlapping classes are treated in the theory of SVMs. In the case where classes cannot be separated exactly, we modify the optimization problem in (2.13) as

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}, w_0}{\arg\min} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i \quad \text{subject to} \quad \hat{y}_i \left(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) + w_0 \right) \geq 1 - \xi_i, \; \xi_i \geq 0 \tag{2.24}$$

Here we introduced the slack variables ξ_i in the constraints, which allows for data points to be misclassified by a margin ξ_i. The second term in the objective forces the slack variables to be close to zero. The coefficient C is a regularization parameter. As $C \to \infty$, we recover perfect class separability (of course, if possible). The decision function for this problem is exactly the same as (2.23), but the support vectors now include points that are on or inside the margin or are misclassified.

Theoretically, the decision boundary problem leads to class separability. Therefore, in practice we will select C to be as large as possible. We do not discuss the non-separable class problem any further in this report.

2.3. ALGORITHM FOR DECISION BOUNDARY IDENTIFICATION

We now present our complete algorithm for decision boundary estimation. First, we introduce some notation that will be useful in the sequel. Let $\Xi_b \subset X$ be a set of background parameter samples. This set will have a large number of points and is the main set from which we will draw training samples without replacement. Also, let $\Xi_t \subseteq \Xi_b$ be a set of training samples on which we perform expensive numerical simulations. That is, evaluating the true decision function (2.1) entails the execution of large scale simulations. Hence, the training set $\{x_i, \hat{y}_i\}$ should be judiciously constructed to ameliorate computational cost. This issue has received wide attention in the decision boundary estimation literature [11, 2, 13, 1]. The common element in existing work is an adaptive strategy for generating training samples. To this end, we can exploit the fact that SVMs produce an explicit representation of the boundary through its decision function (i.e. Eq. (2.23)).

Intuitively, samples close to the decision surface are more important that samples away from the decision boundary. So, assuming that we have an initial data set that provides a reasonable approximation to the decision boundary, we could adopt a greedy strategy in which we perform expensive numerical evaluations only on samples that are close to the decision boundary. In this early version of our work, we use a simple, greedy approach that has produced excellent results in our preliminary investigations.

Assume that we have updated our SVM at iteration k. Next, we evaluate the SVM decision function (2.19) over the background set. Then, we group samples on either side of the boundary in sets $\mathscr{A}_k^+ := \{\boldsymbol{x} \in \Xi_b : y_k(\boldsymbol{x}) > 0\}$ and $\mathscr{A}_k^- := \{x \in \Xi_b : y_k(\boldsymbol{x}) < 0\}$. Here, a subscript or superscript k denotes a quantity that depends on the SVM at iteration k. For instance, $y_k(x) \equiv y(\boldsymbol{x}; \boldsymbol{w}^k, w_0^k)$. Next, we select the next best two samples where to evaluate our simulation model as

$$\boldsymbol{x}_k^+ = \underset{\boldsymbol{x} \in \mathscr{A}_k^+}{\arg\min} |y_k(\boldsymbol{x})|, \ \boldsymbol{x}_k^- = \underset{\boldsymbol{x} \in \mathscr{A}_k^-}{\arg\min} |y_k(\boldsymbol{x})| \tag{2.25}$$

Notice that the above optimization problems are very simple to solve. We want to point out that the greedy strategy put forward herein can miss part of the decision boundary in problems when

$\mathscr{A}^+$ and/or $\mathscr{A}^-$ are small in parameter space. Also, some authors [11] have reported stagnation (i.e. very slow change in the boundary during iterations) in greedy strategies. This issue was addressed in [2, 1], but with much more involved algorithms than the one given in this report. One idea to circumvent this problem is to introduce an exploration step that allows for the sampling of regions away from the current decision boundary estimate. More robust exploration-exploitation strategies are being developed and will be added to our algorithm in the near future. Algorithm 1 shows the steps implemented for the current work.

2.3.1. Error Metric

In our current implementation, we use the error definition reported in [11] to monitor convergence. We first create a test set Ξ_c independent of Ξ_b and Ξ_t. Let Δ_k denote the number samples that change labels (i.e. signs) from iteration $k-1$ to iteration k. Then, we can define an error metric as

$$\varepsilon_k := \frac{\Delta_k}{|\Xi_b|} \tag{2.26}$$

This error is monitored from iteration to iteration and the algorithm is stopped when the error falls below a user-defined tolerance.

Algorithm 1: Adaptive sequential construction of the decision boundary

Initialization:

- Draw M uniformly distributed samples and store in Ξ_b.
- Draw N_1 samples from Ξ_b without replacement and store in Ξ_t.
- Obtain the initial training labels $\{\hat{y}\}$ by evaluating samples in Ξ_t using (2.1) (numerical model).
- Add elements in $\{\hat{y}\}$ to corresponding $\mathscr{A}^+$ or $\mathscr{A}^-$.

// Ensure that there are samples on both sides of the boundary.

while $|\mathscr{A}^+| = 0$ or $|\mathscr{A}^-| = 0$ and $|\Xi_b| \neq 0$ **do**

  Draw a sample without replacement from Ξ_b to Ξ_t;
  Evaluate new sample x_k using (2.1) to obtain label $\hat{y}_k$;
  Update $\mathscr{A}^+$ and $\mathscr{A}^-$ accordingly;

end

Specify the SVM kernel, C, and other hyperparameters ;

while not converged and maximum iterations not exceeded **do**

  Train the SVM using $\Xi_r, \{\hat{y}_j, j = 1, ..., |\Xi_t|\}$;
  Evaluate (2.19) over the background set Ξ_b to compute the distance from each sample to the boundary;
  Select the sample closest to the boundary from each of the sets $\mathscr{A}^+$ and $\mathscr{A}^-$. These samples denoted as x_k^+ and x_k^-;
  Evaluate the computer model at x_k^+ and x_k^- and compute their labels using (2.1) ;
  Add x_k^+ and x_k^- to Ξ_t and remove from Ξ_b;
  Compute error;

end

3. EXAMPLES

We present three examples to demonstrate the performance of the decision boundary estimation approach. The first example consists of a uni-axial bar made out of two materials, loaded on one end, and fixed at the other. This is a 2D parameter space and the decision boundary isolates regions where a threshold displacement is exceeded. The second example considers a conical structure under a harmonic pressure loading. The pressure is parameterized using a low dimensional representation to make the problem tractable. The quantity of interest in this problem is the resultant acceleration at a given point and seek the decision boundary that separates regions where a threshold acceleration is exceeded. The last example is a heat conduction problem involving lithium-ion batteries in thermal runaway. This simplified problem removes chemical reactions and focuses on heat transfer from a failed cell to an adjacent "target" cell through an inert material. Given three design parameters, the boundary between thermal runaway being mitigated and propagating to the target cell is found using a temperature criteria.

We implemented Algorithm 1 in Python 3.9.7 and used Scikit-Learn 1.02 for all the SVM operations. A support vector machine with a Gaussian kernel was used for all the examples in this report with a constant $C = 1000$. The spread parameter, σ, was selected automatically using the default option in Scikit-Learn. Our code is available through Gitlab upon request.

3.1. BAR UNDER UNIAXIAL LOAD

Consider a bar of unit area and made of two materials. The bar is loaded with a point force at one end and fixed at the other as shown in Figure 3.1-1 Assuming that the bar is divided into two

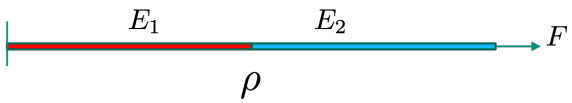


Figure 3.1-1. Bar example with two materials.

regions of lengths L_1 and L_2, the displacement at the loaded end is given as

$$u = F\left(\frac{L_1}{E_1} + \frac{L_2}{E_2}\right) \tag{3.1}$$

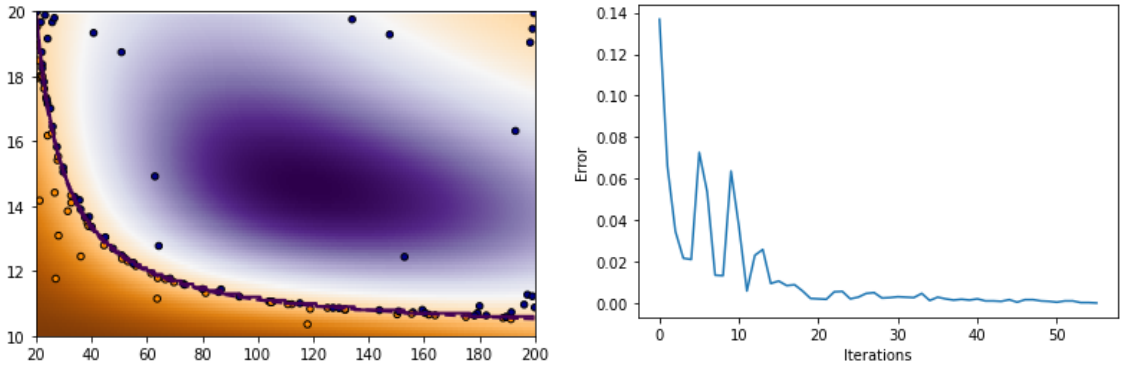
We are interested in finding the values of E_1 and E_2 for which the displacement exceeds a certain threshold u_0. The decision function for this case is

$$g(E_1, E_2) = F\left(\frac{L_1}{E_1} + \frac{L_2}{E_2}\right) - u_0 \tag{3.2}$$

The acceptance or decision boundary is given by the values of E_1 and E_2 that result in $g(E_1, E_2) = 0$, which can be described analytically in the current case.

We used Algorithm 1 to approximate the decision boundary in this problem. To that end, we used 5000 samples in the background set, 5000 samples in the test set to compute the error according to (2.26), and initialized the algorithm with 5 training samples. Recall that we evaluate our numerical model only at the training samples. In the current case, the latter amounts to evaluating the sign of (3.2).

Figure 3.1-2 shows the reconstructed decision boundary and the error versus number of iterations. As we can see, the algorithm was able to identify the decision boundary with high accuracy. For this example, we needed 117 training samples (i.e. function evaluations) and the error was zero at the end of 55 iterations. Recall that a zero error just means that no sample in the test set changed labels from one iteration to the other.



(a) Decision boundary. **(b) Error versus iterations.**

Figure 3.1-2. Results from Bar Example. Samples shown belong to the training set. The dashed line corresponds to the estimated boundary, while the solid line is the true boundary. They are practically indistinguishable in this example.

3.2. CONE STRUCTURE UNDER SPATIALLY-VARYING PRESSURE LOAD

Now we consider the cone structure shown in Figure 3.2-1. The structure is excited with a harmonic pressure whose spatial distribution is given as

$$p(r,\theta,x) = A_1 \cos(\gamma\theta) + A_2 \sin\left(2\pi V \frac{x}{H}\right)$$
$$r = \sqrt{y^2+z^2},\ \theta = \cos^{-1}\left(\frac{y}{r}\right)$$

where the parameters A_1, A_2, γ, V are user-defined, and H is the height of the structure. For the current example, we will fix $A_1 = A_2 = 10^5$ and search for a decision boundary using $V \in [0,1]$ and $\gamma \in [1,3]$.

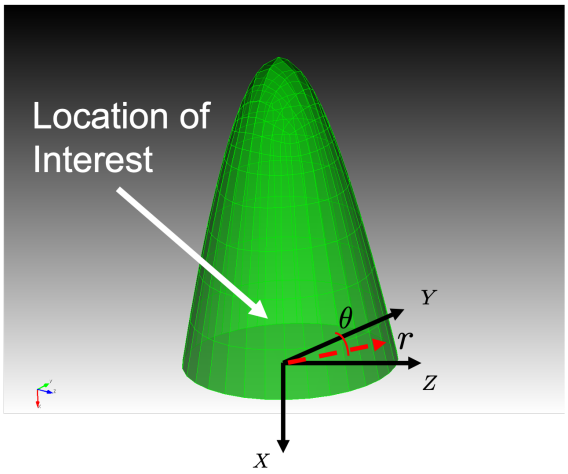
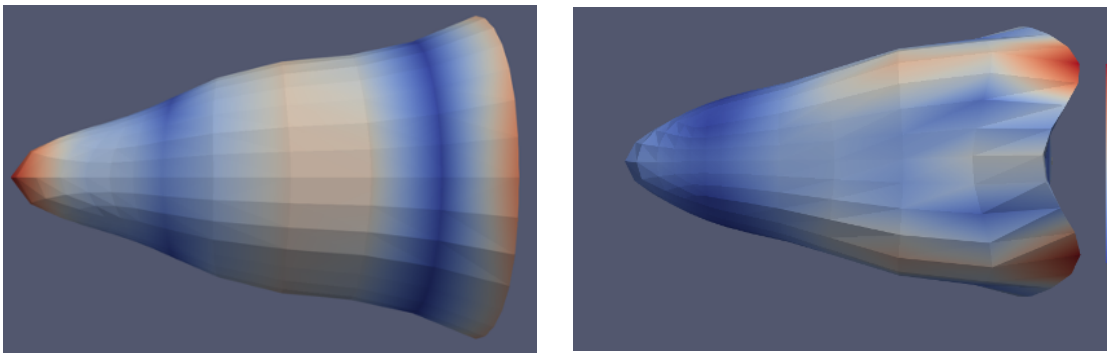


Figure 3.2-1. Cone structure.

We used SIERRA/SD to construct the transfer matrix, T, for the cone structure. Then, the acceleration resultant at the node of interest i was computed as

$$a_i = \|Q(i)T\hat{p}\|$$

where $Q(i)$ is an observation matrix that extracts the acceleration components at Node i and $\hat{p}$ is a vector of input pressures. Figure 3.2-2 shows displacement fields corresponding to two representative points in the $V-\gamma$ space.

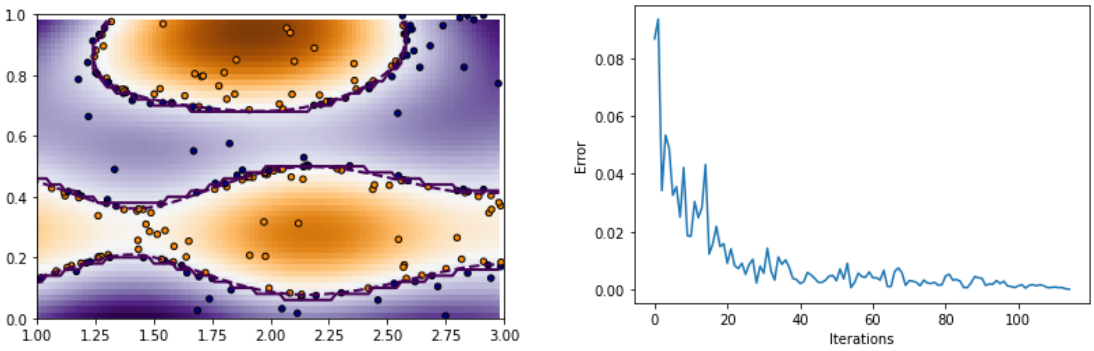


(a) $\gamma = 1, V = 2$. **(b)** $\gamma = 4, V = 1$.

Figure 3.2-2. Representative deformation for cone structure under different parameters.

We initialized the training set using 25 initial points and the background set had 5000 samples. We also used 5000 test samples to monitor convergence. Figure 3.2-3 shows the decision boundary and the error versus iterations corresponding to a threshold $a_{th} = 0.15$. Notice that the decision boundary was accurately estimated. The total number of model (simulation) evaluations for this example was 255 and the error tolerance (10^{-10}) was satisfied after 114 iterations.

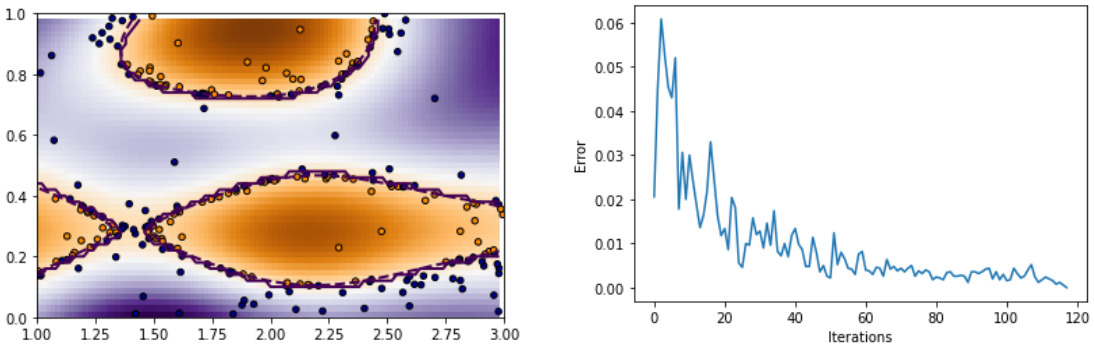
We solved the same problem, but now using a threshold $a_{th} = 0.17$. The latter resulted in three disconnected regions as shown in Figure 3.2-4. In this more challenging case, we were also able



(a) Decision boundary **(b) Error versus iterations.**

Figure 3.2-3. Results from Cone Model Example. Samples shown belong to the training set. Threshold used: $a_{th} = 0.15$. The dashed line corresponds to the estimated boundary, while the solid line is the true boundary.

to identify the decision boundary accurately. A total of 261 model evaluations were performed in 117 iterations. The algorithm stopped when the error tolerance was satisfied.



(a) Decision boundary **(b) Error versus iterations.**

Figure 3.2-4. Results from Cone Model Example. Samples shown belong to the training set. Threshold used: $a_{th} = 0.17$. The dashed line corresponds to the estimated boundary, while the solid line is the true boundary.

3.3. PROPAGATING THERMAL RUNAWAY BETWEEN LI-ION BATTERIES

When designing assemblies of Li-ion batteries, the thermal properties of the system can be manipulated to improve system safety in a thermal runaway event. Figure 3.3-1 depicts the a simplified scenario where the trigger cell is at an elevated temperature due to thermal runaway,

and the adjacent cells are beginning to heat up. Typically, Li-ion batteries will enter thermal runaway between 150°C and 200°C. In this scenario, heating of the adjacent "target" cells can be slowed down by increasing external convection, increasing thermal contact resistance, or the addition of inert materials between the cells (such as a plastic case).

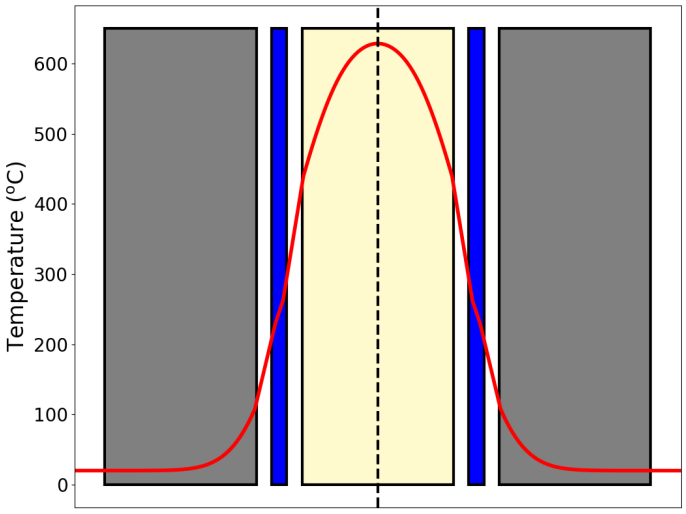
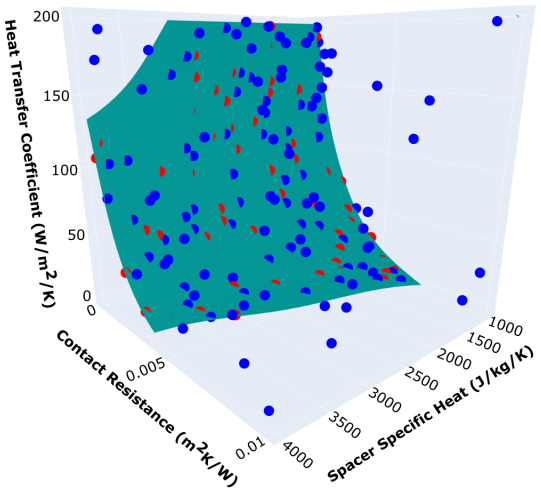


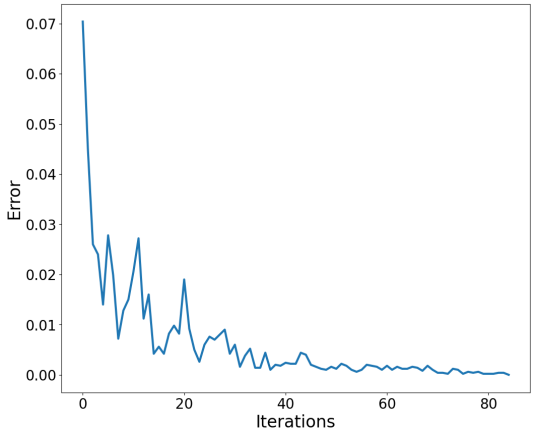
Figure 3.3-1. Temperature at some time after thermal runaway in center (tan) cell. Heat flows through the spacers (blue) to the adjacent cells (gray).

For this demonstration, the system is treated as symmetric along the center of the trigger cell, and the simulations domain includes half of the trigger cell, an inert plastic spacer, and the target cell. The system is modeled in the open-source thermal runaway software LIM1TR [14]. The simulation is initiated with the trigger cell at 650°C and the rest of the system at 25°C. If the target cell exceeds 200°C, it fails, otherwise thermal runaway is assumed to have been mitigated. The algorithm maps out this decision boundary in the space of three key heat transfer parameters that could be manipulated by a system designer: external heat transfer coefficient, thermal contact resistance, and the heat capacity of the inert material.

The decision boundary and error versus iterations are shown in Figure 3.3-2. We expect there to be a single boundary, and the algorithm efficiently captures it in 195 evaluations with a background sample size of 500,000. In Figure 3.3-2a, red dots are cases where thermal runaway propagated to the target cell, and blue dots represent cases where thermal runaway was mitigated.



(a) Decision boundary



(b) Error versus iterations.

Figure 3.3-2. Results from the battery thermal runaway example. The surface between propagation and mitigation is shown in green, where red and blue dots are model evaluations (a).

4. CONCLUDING REMARKS AND FUTURE DIRECTIONS

We presented an approach that allows for the construction of decision boundaries that is amenable to large scale compute simulations. We used support vector machines to adaptively construct a decision boundary, while resorting to a limited number of expensive simulations. We demonstrated the performance of the main algorithm through three examples from the areas of structural mechanics and heat transport. In all cases, we were able to identify with high accuracy the decision boundaries with a limited number of function evaluations (at most 120). We expect that this type of technology will pave the way towards the integration of simulation and testing in the decision making process across many disciplines.

The approach presented herein is largely based on the work reported in [1] and [2]. Although we obtained satisfactory results in our preliminary investigations, we foresee that improvements will be needed to deploy this problems to realistic applications. For instance, although the adaptive algorithm can decrease the demand for costly simulations, the number of function evaluations need may still be excessive for problems such as those involving turbulent flows and fluid structure interaction where one simulation can take in the order of weeks. To this end, we foresee a strong need for the integration of reduced order modeling in the process of constructing the decision boundary. This would be a fertile research direction.

Another area of improvement would be the formal treatment of uncertainty in the decision boundary construction. Uncertainty is always present in the form of, for instance, model and measurement errors. The latter would render the class labels in the training set as random. That is, the acceptance and rejection sets would become fuzzy.

BIBLIOGRAPHY

[1] Michael E Cholette, Pietro Borghesani, Egidio Di Gialleonardo, and Francesco Braghin. Using support vector machines for the computationally efficient identification of acceptable design parameters in computer-aided engineering applications. Expert Systems with Applications, 81:39–52, 2017.

[2] Anirban Basudhar and Samy Missoum. An improved adaptive sampling scheme for the construction of explicit boundaries. Structural and Multidisciplinary Optimization, 42(4):517–529, 2010.

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. Machine Learning, 20:273–297, 1995.

[4] Achmad Widodo and Bo-Suk Yang. Support vector machine in machine condition monitoring and fault diagnosis. Mechanical Systems and Signal Processing, 21(6):2560–2574, 2007.

[5] Marcus Musselman and Dragan Djurdjanovic. Time-frequency distributions in the classification of epilepsy from eeg signals. Expert Systems with Applications, 39(13):11413–11422, 2012.

[6] Colin Campbell and Yiming Ying. Learning with support vector machines. Synthesis lectures on artificial intelligence and machine learning, 5(1):1–95, 2011.

[7] Shigeo Abe. Support vector machines for pattern classification, volume 2. Springer, 2005.

[8] Christopher M Bishop. Pattern recognition. Machine learning, 128(9), 2006.

[9] Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press, 2002.

[10] Jorge E Hurtado. An examination of methods for approximating implicit limit state functions from the viewpoint of statistical learning theory. Structural Safety, 26(3):271–293, 2004.

[11] Anirban Basudhar and Samy Missoum. Adaptive explicit decision functions for probabilistic design and optimization using support vector machines. Computers & Structures, 86(19-20):1904–1917, 2008.

[12] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, pages 144–152, 1992.

[13] Geritt Kampmann, Nataša Kieft, and Oliver Nelles. Support vector machines for design space exploration. In Proceedings of the World Congress on Engineering and Computer Science, volume 2, pages 1116–1121, 2012.

[14] Andrew Kurzawski and Randy Shurtz. SAND2021-12281, LIM1TR: Lithium-ion Modeling with 1-D Thermal Runaway v1.0. Report, October 2021.

DISTRIBUTION

Hardcopy—Internal

Number of Copies	Name	Org.	Mailstop
2	Technical Library	9616	0899

This page intentionally left blank.

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.