2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

2024 IEEE 63rd Conference on Decision and Control (CDC) | 979-8-3503-1633-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/CDC56724.2024.10886857

Decision Boundary Estimation Using Reinforcement Learning for Complex Classification Problems

Josh Netter[1], Kyriakos G. Vamvoudakis[1], Timothy F. Walsh[2], Jaideep Ray[2]

Abstract— **In this paper, we propose a method for quickly training a binary support vector machine (SVM) classifier for recognizing valid input spaces in high-dimensional, highly constrained systems by using reinforcement learning to find inputs along the decision boundary of the classifier while minimizing the number of runs of a simulation representing the system. We find training points by first defining an optimization problem where the action space consists of points to test, and the reward is a function that searches for points that are close to violating the given constraints and are a sufficient distance from one another. After formulating this process, we use a Q-learning framework to find inputs that maximize the reward, and then use these inputs to train the classifier so the decision boundary is quickly well-defined. The efficacy of this approach is then shown in simulations.**

I. Introduction

Currently, much of the development of autonomous vehicles has been driven by the potential of automating certain aspects of flight, such planning in urban air mobility (UAM) scenarios [1], the control of a vehicle in re-entry [2], or even optimizing design of hypersonic components [3]. In each of these use cases, designers must ensure that these autonomous aircraft are safe in a variety of operating conditions, to the point that these craft may be subject to even more rigorous standards than current aircraft [4]. However, these crucial simulated tests are often a major bottleneck in engineering design in recent years.

When designing an aircraft, one must consider and rigorously confirm its valid operating conditions, which are normally dependent on a wide variety of environmental parameters such as velocity, or atmospheric pressure and temperature. The effects that these conditions may have on the operation of a vehicle often fluctuate wildly and are difficult to predict, and the vehicles often have many potential constraints on some *quantities of interest* (QoIs) that must never be violated such as exceeding certain loads or temperatures on various components of the aircraft. This often results in very complex sets of valid flight conditions, that may be non-convex or split into disjoint subsets, and dependent on a high-dimensional input vector. These design challenges can be formulated as a binary "go/no-go" classification problem, where we seek the *decision boundary* between the valid input spaces where all QoIs are below their given thresholds, and the invalid input spaces where one or more constraints are violated.

Training a classifier for a non-convex or disconnected decision boundary is still a significant challenge. It is crucial to select training points intelligently to accurately model the complete decision boundary in these cases, but ironically even selecting points intelligently often requires numerous initial random samples to find good candidate points or some prior knowledge of the system to narrow down an area of interest in the decision space. In addition, because of the complex effects of many environmental conditions on a vehicle, simulating each potential set of parameters may take hours to run to convergence, making the process of training a classifier days or even weeks long. In this work, we propose a method to intelligently find informative training points using reinforcement learning to quickly train a classifier to recognize a complex valid input space with minimal prior information and minimal training data.

First, we consider previous methods of modelling physics-based systems. In [5], the authors propose physics-informed neural networks (PINNs) which proved to be effective at modelling problems constrained by laws of physics defined with partial differential equations, or with finding these PDEs. For our work we do not necessarily require precise QoI values, but instead seek a boundary of when these QoIs are likely to violate constraints. In [6], neural nets are not used to directly learn the behavior of a system, but instead are used to learn the kernel determining the structure of a Gaussian process. This somewhat mitigates the shortcomings of Gaussian processes, which cannot model particularly erratic functions, and may still struggle with less smooth functions.

We must also discuss methods of binary classification, and how effective they are in dealing with irregular design spaces. For example, we first consider Gaussian process classification and Bayesian classification [7], [8], which are effective at not only generating a binary classifier but can additionally provide a probability that the classification is accurate. Support vector machines (SVMs) [9] were also found to be very effective at classifying these systems when provided with sufficient training data in our initial tests. We also examine previous efforts on using learning to train classifiers on the conditions of real systems. For example, in [10], the authors show that physics-informed neural networks combined with a visual classifier can be used to identify bearing fatigue. The authors of [11] alternatively use a learning-based approach for the classification with

[1]J. Netter and K. G. Vamvoudakis are with the Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA, 30334 USA (email: jnetter6@gatech.edu, kyriakos@gatech.edu).
[2]T. F. Walsh and J. Ray are with the Sandia National Laboratories, Albuquerque, NM, 87123 USA (email: tfwalsh@sandia.gov, jairay@sandia.gov).
This work was supported in part, by NSF under grant Nos. CAREER CPS-1851588, CPS-2038589, CPS-2227185, by NASA ULI under grant number 80NSSC20M0161, and by Sandia.

costly features (CwCF) problem. In this work, classification is structured as a Markov decision process (MDP) where acquiring each feature of an input has an associated cost, and a Q-learning approach is used to decide which features to acquire and when to classify the input. In each of these works, neural networks are successfully used either as classifiers or to supplement classifiers of complex systems. We seek to expand upon this work by intelligently selecting informative points to train these networks and an SVM classifier with as small of a training set as possible.

The contributions of the present paper are fourfold. We first formulate the problem of finding ideal training points as an optimization problem with a cost we seek to minimize. We then formulate a reinforcement learning method for finding an optimal stochastic policy for a problem without a defined state space. We apply this approach to the problem of finding the decision boundary of a binary classifier across the total design space with no prior information. Lastly, we use these selected points to train a binary classifier and compare the results to other methods of intelligent sampling.

Structure: In Section II, we will formulate the problem of selecting effective points for training a go/no-go classifier. In Section III we will detail the learning-based approach to selecting these points quickly. In Section IV we test this approach in a variety of simulations. In Section V we conclude the paper and discuss possible future directions for this research.

II. Problem Formulation

Consider a function,

$$y = f(x), \tag{1}$$

where $x \in \mathbb{R}^n$ is the current state and/or control input vector of the system, and the output is some number m quantities of interest (QoIs) $y \in \mathbb{R}^m$ we choose to monitor. Each monitored QoI $y_i, 1 \leqslant i \leqslant m$ is constrained by a maximum allowed value $y_{\max,i}$. For a state of the system x to be valid,

$$f_i(x) \leqslant y_{\max,i},\ \forall i \in 1, \ldots, m,$$

where f_i refers to the i-th value in the output vector. There may additionally be minimum constraints $y_{\min,i}$ that require $f_i(x_t) \geqslant y_{\min,i}$, but these can be trivially transformed into maximum constraints by taking the negative of both $f_i(x_t)$ and $y_{\min,i}$. If any constraint is invalid, then the state is considered invalid. This is referred to as a "go/no-go" classification problem. For convenience, we express the total valid input space as $\mathcal{X}^+$, the invalid input space as $\mathcal{X}^-$, and the complete simulation input space $\mathcal{X}$. In this work, we seek to learn the valid input space $\mathcal{X}^+$ in order to accurately determine the valid states of a system. While this can often be done by running an offline physics simulation for all values $x \in \mathcal{X}$ and determining which inputs result in violating the constraints on the output QoIs, these simulations we consider are usually very time-consuming to run. As a result, we seek to learn $\mathcal{X}^+$ with as few simulation runs as possible.

To begin, we assume that $\mathcal{X}^+, \mathcal{X}^- \neq \varnothing$, as otherwise the problem is trivial. To then find the valid input space, we first try to simplify the problem. We assume that the outputs $f_i(x), \forall i \in 1, \ldots, m$ are continuous even if the slope may be very steep or not smooth. We normalize each output by defining $\bar{f}(x)$ such that

$$\bar{f}_i(x) = \frac{f_i(x)}{y_{\max,i}},\ \forall i \in 1, \ldots, m. \tag{2}$$

These normalized outputs allow us to succinctly describe the valid input space $\mathcal{X}^+$ as the set of inputs such that

$$\bar{f}_i(x) \leqslant 1,\ \forall i \in 1, \ldots, m,\ \forall x \in \mathcal{X}^+.$$

With this equation, we then define a combined constrained output,

$$\hat{f}(x) = \min_{i \in 1,\ldots,m} \{1 - \bar{f}_i(x)\}. \tag{3}$$

We use the combined constraints to define the valid input space as the set of inputs such that

$$\hat{f}(x) \geqslant 0,\ \forall x \in \mathcal{X}^+, \tag{4}$$

and conversely define the invalid input space as $\hat{f}(x) < 0,\ \forall x \in \mathcal{X}^-$.

With our valid and invalid input spaces defined, we define the decision boundary $\mathcal{X}^b$ such that for an input $x^b \in \mathcal{X}^b$,

$$\hat{f}(x^b) = 0, \tag{5}$$

We can now simplify the problem of finding the valid input space $\mathcal{X}^+$ to finding the decision boundary $\mathcal{X}^b$.

III. Point Selection With Reinforcement Learning

In this section, we present a formulation for an optimal stochastic policy for selecting actions in problems without a well-defined state, as well as an actor-critic method for finding this optimal policy. First, we require a method of quickly and intelligently finding these optimal inputs. For this, we formulate a learning approach that can learn an optimal policy efficiently with a relatively small number of samples and little to no prior knowledge of the problem. Our approach also must be effective at finding the decision boundary even in a continuous domain and with a non-convex decision boundary or multiple disjoint decision boundaries. With these priorities in mind, we consider a soft actor-critic (SAC) approach to learn an optimal policy [12]. This approach utilizes a stochastic actor-critic approach and has been shown to quickly and reliably find an optimal policy even for complex functions across continuous domains.

The SAC algorithm seeks to both maximize a reward function as well as favor stochastic policies by adding a term for the expected entropy of a policy π,

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{s_t,a_t}[r(s_t, a_t) - \log \pi(\cdot|s_t)], \tag{6}$$

where $s_t \in \mathcal{S}$ represents a state in the state space $\mathcal{S}$, and $a_t \in \mathcal{A}$ represents a potential action in the action space $\mathcal{A}$. For this work, we define the action space as the input space of a simulation, $\mathcal{A} = \mathcal{X}$. The "state" of the system, however, is not well-defined. It can be thought of as the constant parameter of every test (such as, for example, the shape of the body to analyze or the medium it is travelling through). As a consequence, we augment (6) to a simplified stateless form,

$$J(\pi) = \mathbb{E}_{a\sim\pi}[r(a) - \log \pi(\cdot)]. \tag{7}$$

We then consider how to find the optimal policy $\pi^\star$ such that $Q^{\pi^\star}(a) \geqslant Q^{\pi}(a)\, \forall \pi \in \Pi,\, a \in \mathcal{A}$. We begin by finding the soft Q-value of the policy, defined in [13] to reward a higher entropy policy,

$$Q_{\text{soft}}(s_t, a_t) = r_t + \mathbb{E}_{(s_{t+1},\ldots)}\Bigg[\sum_{l=1}^{\infty} \gamma^l (r_{t+l} - \log \pi(\cdot|s_{t+l}))\Bigg].$$

We consider any function $Q : \mathcal{A} \to \mathbb{R}$, and then define a similar Bellman operator to that in [12],

$$\mathcal{T}^\pi Q(a_t) = r(a_t) + \gamma \mathbb{E}_{a_{t+1}\sim\pi}[Q(a_{t+1}) - \log \pi(a_{t+1})], \tag{8}$$

and define $Q^{k+1} = \mathcal{T}^\pi Q^k$. We then define the *entropy-augmented reward* as $r_\pi(a) = r(a) + \mathbb{E}(\pi(\cdot))$, and subsequently rewrite the update rule as

$$Q(a_t) \leftarrow r_\pi(a_t) + \gamma \mathbb{E}_{a_{t+1}\sim\pi}[Q(a_{t+1})]. \tag{9}$$

By using the policy evaluation convergence method described in [14] provided $\mathcal{A} < \infty$, the Bellman equation (8) and update rule (9) show that Q^k approaches the soft Q-value of the policy π as $k \to \infty$. The next step for finding the optimal policy is to iteratively improve the policy. In the original proposed SAC algorithm in [12], this is done by minimizing the Kullback-Leibler divergence in each step. We adopt a similar approach, updating the policy in each step as

$$\pi_{\text{new}} = \min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}}\big(\pi'(\cdot) || \exp(Q^{\pi_{\text{old}}}(\cdot))\big). \tag{10}$$

The proof that this converges to an optimal policy is omitted for the journal paper.

We then formulate how to approximate the Q-function and policy. We consider a Q-function $Q_\theta(s_t, a_t)$ and policy $\pi_\phi(a_t|s_t)$ parameterized by θ and ϕ, respectively. In this case, we model the Q-function parameters using a neural network, and the policy as a Gaussian random vector with means and covariances given by a neural network. The parameters θ and ϕ are thus the weights of these networks. We train both of these approximations to estimate the Q-value and optimal policy using gradient descent. We train the Q-function approximation to minimize the residual error,

$$\begin{aligned} J_Q(\theta) = \mathbb{E}_{a_t}[&\tfrac{1}{2}(Q_\theta(a_t) \\ &- (r(a_t) + \gamma \mathbb{E}_{a_{t+1}}[Q_\theta(a_{t+1}) - \log_{\pi_\phi}(a_{t+1})]))^2], \end{aligned}$$

with the gradient found to be,

$$\begin{aligned} \hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(a_t)(&Q_\theta(a_t) \\ &- (r(a_t) + \gamma \mathbb{E}_{a_{t+1}}[Q_\theta(a_{t+1}) - \log_{\pi_\phi}(a_{t+1})]). \end{aligned} \tag{11}$$

To approximate the policy, we define the cost to minimize as Equation (10), which can be simplified as,

$$J_\pi(\phi) = \mathbb{E}_{\epsilon_t}\left[\log \pi_\phi(f_\phi(\epsilon_t)) - Q_\theta(f_\phi(\epsilon_t))\right], \tag{12}$$

where $f_\phi(\epsilon_t)$ is a transformation of the policy including an input noise vector ϵ_t to allow for differentiation of the cost. This allows us to define the gradient simply as,

$$\begin{aligned} \hat{\nabla}_\phi J_\pi(\phi) = &\nabla_\phi \log \pi_\phi(a_t) \\ &+ (\nabla_{a_t} \log \pi_\phi(a_t) - \nabla_{a_t} Q(a_t)) \nabla_\phi f_\phi(\epsilon_t). \end{aligned} \tag{13}$$

We use the gradients (11) and (13) to adjust the values of θ and ϕ to find the Q-function and the optimal policy.

We now define a reward function $r(a)$ to effectively search for the boundary inputs using the proposed actor-critic method. For training a classifier, we want to prioritize points close to the decision boundary regardless of if they are valid or invalid. As a result, we base our reward on the square of the combined constrained output defined in Equation (3),

$$r(a) = -\hat{f}(a)^2. \tag{14}$$

It can be trivially seen that given the reward function (14), the highest possible is $r(a) = 0$, and an input $a^\star$ is optimal if $\hat{f}(a^\star) = 0$. In other words, the reward increases as a point a approaches the decision boundary, and is maximized when $a^\star$ represents a point on the decision boundary. In turn, the reward of the policy as defined by Equation (7) is maximized when the policy both approaches the decision boundary while also maximizing expected entropy. With this optimal policy, we prioritize sampling a large input space where the inputs are relatively close to the decision boundary, allowing us to intelligently sample the input regions of most interest for efficiently training a binary classifier.

To create the go/no-go classifier, we first initialize the actor and critic and use the actor to select inputs along a series of steps. After each step, the model weights of both the actor and critic are updated to improve the policy. With each input returned by the modeled policy, we calculate the reward using a run of the simulation. Once a simulation is run, both the input and its reward are saved for future use if the policy once again returns the same input. This continues until enough points have been simulated to train the classifier. This complete approach is shown in Algorithm 1.

IV. Simulations

The proposed method of selecting training points was tested in a variety of binary classification problems of varying complexity to measure the effectiveness of the approach. In each of these problems, the selected training points are used to train an SVM to determine if a certain constraint is violated. In each example problem, a value b_i is chosen based on the range of potential values of the i-th value of the input a_i such that $(b_i(a_{\max,i} - a_{\min,i}))^2 = 1$, where $a_{\max,i}$ and $a_{\min,i}$ are the maximum and minimum values of the input respectively. For example, for an input with a range $[20, 30]$, we take b_i to equal $\frac{1}{10}$. With this, we normalize the effect each value of the input vector has on the measured "distance"

Algorithm 1 Learned Sampling

Input: $b_1, \ldots, b_n$ - input distance constants; ϕ, θ - SAC parameters; $y_{\max,1}, \ldots, y_{\max,n}$ - Output constraints

Output: SVM - trained support vector machine classifier

```
1:  while $\pi$ not converged do
2:      $a_{\text{new}} \leftarrow \texttt{Actor}(\pi_\phi)$
3:      if ($a_{\text{new}} \in \mathcal{A}_{\text{train}}$) then
4:          $r(a_{\text{new}}) \leftarrow r(a_{\text{train}})$
5:      else
6:          $y \leftarrow \texttt{SimulationRun}(a_{\text{new}})$
7:          $r(a_{\text{new}}) \leftarrow \texttt{Reward}(y, y_{\max,1}, \ldots, y_{\max,n})$
8:          $\mathcal{A}_{\text{train}} \leftarrow \mathcal{A}_{\text{train}} \bigcup a_{\text{new}}$
9:      end if
10:     $\theta \leftarrow \theta - \nabla_\theta J_Q(\theta)$ (11)
11:     $\phi \leftarrow \phi - \nabla_\phi J_\pi(\phi)$ (13)
12: end while
13: SVM $\leftarrow \texttt{TrainClassifier}(\mathcal{A}_{\text{train}})$
14: return SVM
```

between inputs. For each simulation, we also use a greedy sampling approach as a baseline that, on each iteration k, adds two points x_k^+, x_k^- to the training data used by the classifier. These points are defined as

$$x_k^+ = \min_{x \in \mathcal{X}_k^+} |y_k(x)|, x_k^- = \min_{x \in \mathcal{X}_k^-} |y_k(x)|,$$

where y_k represents the SVM decision function at iteration k, and $\mathcal{X}_k^+, \mathcal{X}_k^-$ represent the believed valid and invalid input spaces at iteration k, respectively. This approach attempts to choose two points that are both close to the decision boundary, and on opposite sides of the decision boundary. This approach is detailed in [15] and is shown to be effective at selecting an informative training set. A key difference between these two approaches is that our SAC approach is informed by the actual simulation, whereas the greedy approach is informed by the classifier. This means that the greedy approach does not know, quantitatively, how close constraints are to being violated. This may lead to worse performance along non-convex decision boundaries which require more sampling in certain areas close to the constraints.

For the first problem, we consider a bar of unit area made of N materials, each of with a fixed length $l_1, l_2, \ldots, l_N$ and a Young's modulus $E_1, E_2, \ldots, E_N$. We then apply a point force F at one end along the length of the bar, displacing the bar by a distance $u = F \sum_{i=1}^{N} \frac{l_i}{E_i}$. We seek Young's moduli values of each material where the bar displacement does not exceed a given threshold u_0. This example produces a relatively simple convex valid input space. In our simulations we use a bar with 2 segments, or $N = 2$, as well as one where $N = 10$ to test the effectiveness of the actor-critic approach in problems of higher dimensionality. The misclassification rate as a function of number of samples for both the actor-critic method as well as the previous greedy method is shown in Figure 1 for $N = 2$, and Figure 2 for $N = 2$. The model learned using SAC and the "true" model for $N = 2$ is shown in Figure 3 to illustrate their similarity. Figure 3 also shows how the selected training points follow closely along the decision boundary across much of the input space.

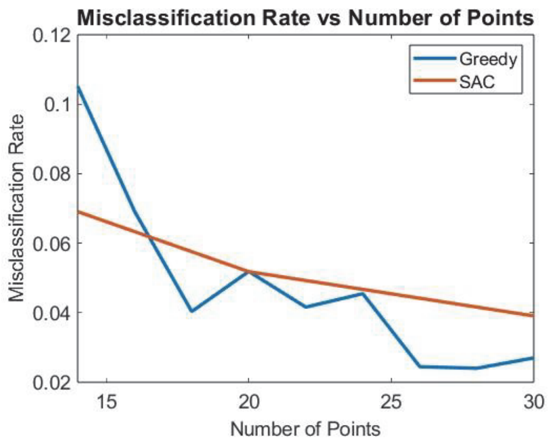

Fig. 1: Comparison of the average misclassification rate of the greedy and learned sampling point selection methods on the 2-dimensional uniaxial bar problem with a small sample size. The misclassification rates of the approaches are comparable despite the greedy approach being supplied with initial training samples.

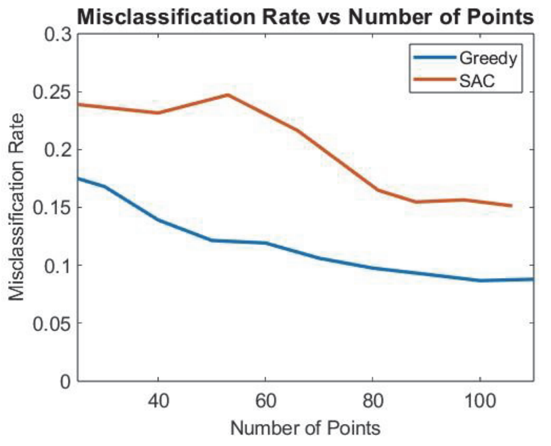

Fig. 2: Comparison of the average misclassification rate of the greedy and learned sampling point selection methods on the 10-dimensional uniaxial bar problem. Despite no initial training data across a very large input space, the SAC method remains competitive with the greedy approach.

For the second problem, we consider a function with a two-dimensional input $f(x) = x_2 - |\tan(\frac{x_1}{2} + 2)|$ with a constraint that $f(x) < 3$. This results in an irregular, non-convex valid input space. The misclassification rates for this problem are shown in Figure 4. The model trained with a small training set supplied by our actor-critic approach, as well as the true decision boundary, is shown in Figure 5. Unlike the uniaxial bar problem tested previously, the SAC sampling method outperforms the greedy method used in previous work.

In our final experiment, we considered a basic aerodynamic simulation using the Sandia Parallel Aerodynamics Reentry Code (SPARC) [16]. We model the flight of the Sandia Test Vehicle (STV), and take velocity in meters per

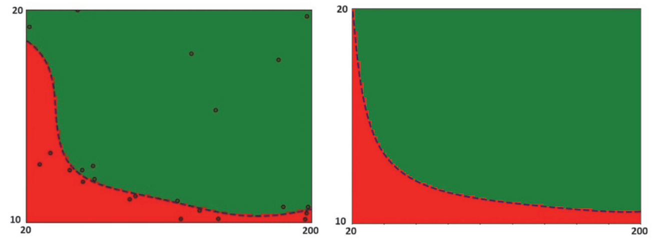

Fig. 3: The left image shows the 2-dimensional uniaxial bar classifier formed using learned sampling with a shown small sample size, and the right shows the decision boundary learned with significantly more samples. As the problem is rather simple, both the greedy and learned sampling approach quickly approach a >95% successful classification rate.

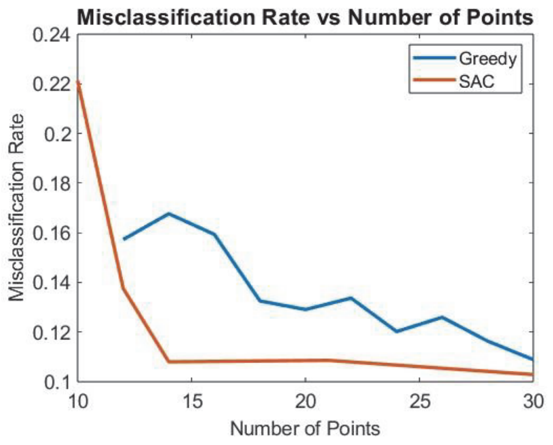

Fig. 4: Comparison of the average misclassification rate of the greedy and learned sampling point selection methods on a non-convex classification problem with a small sample size. On this problem, our learned sampling approach outperforms the previous greedy approach even without initial training samples.

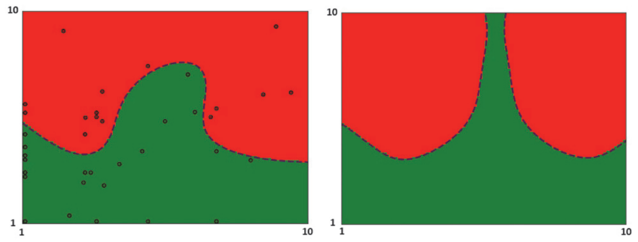

Fig. 5: The left image shows the non-convex classifier formed using learned sampling with a shown small sample size, and the right shows the decision boundary of the problem learned with significantly more samples. Even with a small sample size, many of the points on the left image clearly show rapid convergence to the location of the later decision boundary.

second, angle of attack in degrees, and air temperature in Kelvin as our independent variables. We then set a single constraint on the maximum tolerated air pressure on the surface of the vehicle and classify any inputs resulting in a pressure exceeding this constraint as invalid. We show the resultant misclassification rates for this problem in Figure 6, and the initial and final leaned models in Figures 7 and 8, respectively. In this experiment, the SAC approach went through several iterations without gathering meaningful training points, resulting in a higher misclassification rate at first. This is likely due to a volatile reward function at first, where small changes in policy parameter weights drastically affect the reward. However, after a few iterations when the policy begins to converge, the misclassification rate sharply drops and approaches the misclassification rate of the greedy approach, which saw a much steadier, slower decrease.

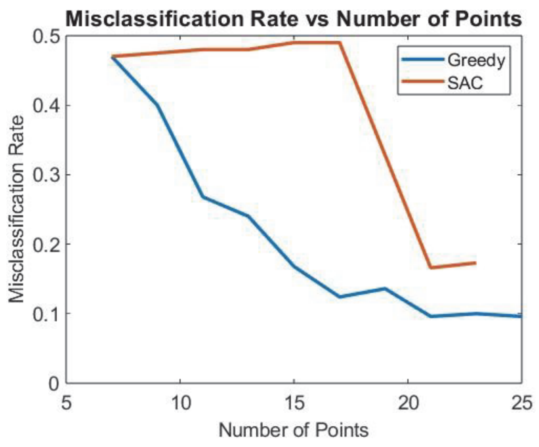

Fig. 6: Comparison of the misclassification rate of the greedy and learned sampling point selection methods on the STV pressure classification problem with a small sample size. Both classifiers start off very inaccurate, but the STV takes some additional iterations to converge.

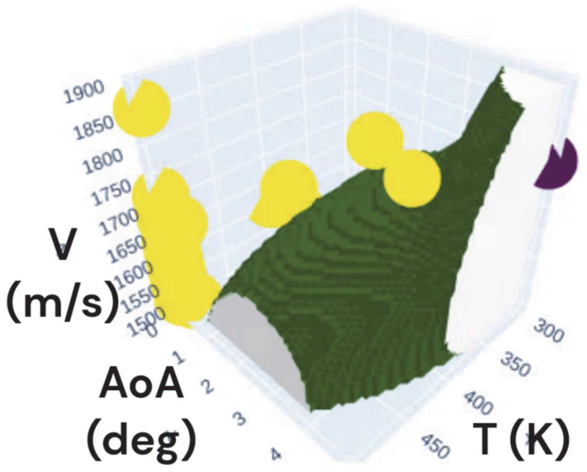

Fig. 7: The STV pressure classifier formed using learned sampling with a shown small sample size. Yellow and purple dots signify positive and negative samples, respectively. The image shows the samples clustering at edges of the simulation from initial sampling, following by converging towards the true boundary in the center.

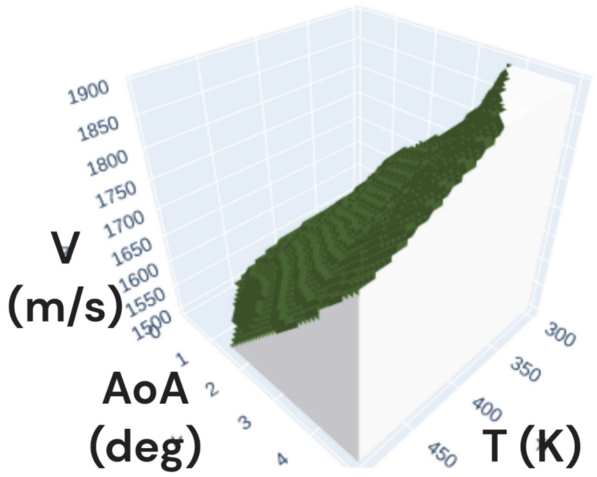

Fig. 8: The decision boundary of the STV pressure learned with significantly more samples.

To summarize our results, the two approaches perform comparably well on each test problem. We found that the traditional greedy sampling method was more effective on a high-dimensionality problem with a convex decision boundary, whereas learned sampling was more successful at finding points along a more complicated non-convex decision boundary. Additionally, we note the repeated success of the learned sampling approach even though it does not require any prior knowledge of the decision boundary, which is required for the greedy approach. This means that randomly selected initial training samples can be kept to a minimum to reduce the amount of necessary simulation runs.

V. Conclusion

In this work we introduce a learning-based method of efficiently selecting training points for classifying the valid input space of computationally intensive simulations of complex dynamic systems such as flight vehicles. We compared this approach to a greedy method of selecting training points and found it to be competitive in a variety of example problems as well as a flight simulation despite not being supplied with an initial training set. In our future work, we will experiment with normalizing both input parameters and the expected reward of a problem to see if this can make results more consistent, rather than unpredictable sharp improvements like those seen in the STV test. Future work will also include testing this approach with more complex systems with a larger input space and multiple constraints to determine the valid input space, which we anticipate will create difficulties in the greedy approach from previous work. Additionally, as the learned sampling approach requires no initial data, we also consider the possibility of supplementing the classifier with data gathered online during flight to learn the valid input space more accurately in real-time. This unique advantage will also allow us to measure the effectiveness of entirely relearning a valid input classifier in the case of unforeseen interference on the system, such as inclement weather or hardware failure.

Acknowledgements

Sandia National Laboratories is a multi-mission 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.

References

[1] A. Straubinger, R. Rothfeld, M. Shamiyeh, K.-D. Büchter, J. Kaiser, and K. O. Plötner, "An overview of current research and developments in urban air mobility–setting the scene for UAM introduction," *Journal of Air Transport Management*, vol. 87, p. 101852, 2020.

[2] W. R. Johnson, P. Lu, and S. Stachowiak, "Automated re-entry system using fnpeg," in *AIAA guidance, navigation, and control conference*, 2017, p. 1899.

[3] V. Shukla, A. Gelsey, M. Schwabacher, D. Smith, and D. D. Knight, "Automated design optimization for hypersonic inlets," *AIAA Journal of Aircraft*, 1996.

[4] M. Vrdoljak, O. Halbe, T. Mehling, and M. Hajek, "Flight guidance concepts to mitigate flight control system degradation in urban air mobility scenarios," *IEEE Aerospace and Electronic Systems Magazine*, vol. 38, no. 5, pp. 18–33, 2023.

[5] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.

[6] I. R. Goumiri, B. W. Priest, and M. D. Schneider, "Reinforcement learning via gaussian processes with neural network dual kernels," in *2020 IEEE Conference on Games (CoG)*, 2020, pp. 1–8.

[7] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[8] C. K. Williams and D. Barber, "Bayesian classification with gaussian processes," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 20, no. 12, pp. 1342–1351, 1998.

[9] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, pp. 273–297, 1995.

[10] Y. A. Yucesan and F. A. Viana, "Hybrid physics-informed neural networks for main bearing fatigue prognosis with visual grease inspection," *Computers in Industry*, vol. 125, p. 103386, 2021.

[11] J. Janisch, T. Pevnỳ, and V. Lisỳ, "Classification with costly features using deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3959–3966.

[12] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018.

[13] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *International conference on machine learning*. PMLR, 2017, pp. 1352–1361.

[14] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[15] T. Walsh, W. Aquino, A. Kurzawski, C. McCormick, C. Sanders, C. Smith, and B. Treweek, "Support vector machines for estimating decision boundaries with numerical simulations." Sandia National Lab.(SNL-NM), Albuquerque, NM (United States); Sandia . . . , Tech. Rep., 2022.

[16] J. Smith, "Sandia parallel aerodynamics reentry code (sparc)–the future of production and research aerodynamics [brief]," Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), Tech. Rep., 2020.