Beyond the Known: Detecting Novel Feasible Domains over an Unbounded Design Space

Wei Chen*

Department of Mechanical Engineering

University of Maryland

College Park, MD 20742

Email: wchen459@umd.edu

Mark Fuge

Department of Mechanical Engineering

University of Maryland

College Park, MD 20742

Email: fuge@umd.edu

ABSTRACT

*To solve a design problem, sometimes it is necessary to identify the feasible design space. For design spaces with implicit constraints, sampling methods are usually used. These methods typically bound the design space; that is, limit the range of design variables. But bounds that are too small will fail to cover all possible designs; while bounds that are too large will waste sampling budget. This paper tries to solve the problem of efficiently discovering (possibly disconnected) feasible domains in an unbounded design space. We propose a data-driven adaptive sampling technique—*ε*-margin sampling, which both learns the domain boundary of feasible designs, while also expanding our knowledge of the design space as available budget increases. This technique is data-efficient, in that it makes principled probabilistic trade-offs between refining existing domain boundaries versus expanding the design space.*

We demonstrate that this method can better identify feasible domains on standard test functions compared to both random and active sampling (via uncertainty sampling). However, a fundamental problem when applying adaptive sampling to real world designs is that designs often have high dimensionality and thus require (in the worst case) exponentially more samples per dimension. We show how coupling Design Manifolds with ε*-margin*

*Corresponding author.

sampling allows us to actively expand high-dimensional design spaces without incurring this exponential penalty. We demonstrate this on real-world examples of glassware and bottle design, where our method discovers designs that have different appearance and functionality from its initial design set.

1 Introduction

Identifying the feasible design space is important in design research. For example, design space exploration visualizes a design space and tries to generate feasible designs [1, 2, 3]. Also, reliability-based design optimization (RBDO) requires an accurate boundary of feasible regions in the space [4,5]. The feasible design space is often defined by *constraints*. Constraints expressed via simple mathematical functions are *explicit constraints*; while constraints such as aesthetics, functionality, or performance requirements are often *implicit constraints*. The latter are usually evaluated by human assessment, experiments, or computer simulation.

For implicit constraints, researchers often use sampling methods to identify feasible regions [3, 4, 5, 6, 7]. However, current sampling methods use *bounded* input spaces —they require a predefined bounding box (*i.e.*, the bound for each design variable). For implicit constraints, however, we typically do not know the size of those bounds. For example, consider car body styling design. Even if we have an intuitive sense of how "compact" a car customers are willing to buy, we may not know how to translate those compactness constraints to actual bounds on the design variables; a customer's preference drives an unseen, implicit constraint. If the bounds on the design variables are too small, we might miss feasible designs; while if the bounds are too large, sampling methods will use more (possibly expensive) samples that necessary to identify feasible regions. In such cases we need to sample and expand a possibly unbounded design space—to go beyond known bounds and explore larger design domains (as budget allows).

Note that the feasible domains are possibly disconnected. For example, in a RBDO problem, a probabilistic constraint is defined as $P[g(\boldsymbol{x}) > 0] \leq P_f$, where P_f determines the reliability level and $g(\boldsymbol{x}) \leq 0$ indicates that the design variables $\boldsymbol{x}$ are feasible. The feasible design space identified by $g(\boldsymbol{x}) \leq 0$ shrinks as the reliability level increases (*i.e.*, P_f decreases). When the feasible design space is concave and keeps shrinking, it will eventually become disconnected [6]. Thus given initial designs in only one feasible region, we are also interested in discovering other feasible regions.

In this paper, we address these problems by proposing an adaptive sampling method that picks designs (to simulate, to show to human, or otherwise validate) such that the algorithm both sufficiently *refines* the boundaries for feasible designs that we know about, while also *expanding* the design space to discover new feasible designs (possibly in a disconnected feasible domain) that we may not know about. Compared to past work, this paper differs in that 1) the sampling process is applied on an unbounded design space and discovers multiple disconnected feasible domains; 2) given an increasingly larger budget, our method will handle unbounded design spaces without sacrificing the quality of the boundaries for known feasible designs; and 3) it works for high dimensional real world design spaces by exploiting and expanding along low-dimensional Design Manifolds [8, 9]. This second item, addressed in detail by Sec. 5, is key for applying active domain expansion to practical, real-world designs because active sampling methods in general have an exponential increase in cost as the dimension increases [10].

This active expansion is useful not only for defining boundaries of design spaces, but also for helping discover potentially novel and creative designs. Beyond design, our proposed active learning query strategy—ε-margin sampling—also applies to any *domain expansion problem*, *i.e.*, finding (possibly disconnected) domain boundaries or level sets in an unbounded input data space given an expensive function that evaluates which points are feasible. This expensive function could be costly computation, time-consuming experiments, or human evaluation. The main contributions of this paper are:

1. An active learning query strategy—ε-margin sampling—that balances exploitation and exploration.
2. A method of finding (possibly multiple disconnected) feasible design spaces with implicit constraints (*i.e.*, constraints that cannot be expressed by simple mathematical functions) in an unbounded parameter space.
3. An approach for identifying the feasible design space by exploring low-dimensional embedded design manifolds, rather than in the original high-dimensional space.
4. Demonstrating that domain expansion on real-world design spaces can discover useful and novel designs given a small set of known real-world designs.

2 Background and Related Work

Fundamentally, and without loss of generality, our task in this paper is to detect the boundary ($h(\boldsymbol{x}) = 0$) of an expensive function $h : \mathcal{X} \in \mathbb{R}^d \rightarrow \{-1, 1\}$, where $\boldsymbol{x}$ are design variables, d is the (potentially large) dimensionality of the space, and the output is the validity or feasibility of a given design (with the output $y = 1$ indicates a valid design, and $y = -1$ otherwise). This task has three main questions: 1) how do we probabilistically model level set or classification boundaries; 2) how do we sample new points (queries) such that we refine that boundary efficiently; and 3) how do we tackle boundaries in high dimensional spaces? For each question, we will review past work most closely related this paper's approach: 1) Gaussian Process Classification, 2) Active Learning, and 3) exploration of high dimensional design spaces.

2.1 Gaussian Process Classification

Since we are separating valid designs from invalid ones, we are dealing with a binary classification problem (the label $y \in \{-1, 1\}$ with $y = 1$ indicating valid designs). Unlike other non-linear classification methods such as Support Vector Machines or Neural Networks, Gaussian Processes naturally model probabilistic predictions and can calculate the informativeness of a sample via its posterior probability distribution.

For binary GP classification, we place a GP prior over the latent function $f(\boldsymbol{x}) \in \mathbb{R}$, and then "squash" $f(\boldsymbol{x})$ through the logistic function $\sigma(\cdot)$ to approximate the probability of $y = 1$ at $\boldsymbol{x}$: $\pi(\boldsymbol{x}) = P(y = 1|X, \boldsymbol{y}, \boldsymbol{x}) = \sigma(f(\boldsymbol{x}))$ where X and $\boldsymbol{y}$ are the training data and labels [11]. The distribution of the latent variable f_* for a new sample $\boldsymbol{x}^*$ is

$$P(f_*|X, \boldsymbol{y}, \boldsymbol{x}^*) = \int P(f_*|X, \boldsymbol{x}^*, \boldsymbol{f}) P(\boldsymbol{f}|X, \boldsymbol{y}) d\boldsymbol{f} \tag{1}$$

where $P(\boldsymbol{f}|X,\boldsymbol{y}) = P(\boldsymbol{y}|\boldsymbol{f})P(\boldsymbol{f}|X)/P(\boldsymbol{y}|X)$ is the posterior over the latent variables. Then the probability of $y_* = 1$ at $\boldsymbol{x}^*$ is

$$\bar{\pi}_* = P(y_* = 1|X,\boldsymbol{y},\boldsymbol{x}^*) = \int \sigma(f_*)P(f_*|X,\boldsymbol{y},\boldsymbol{x}^*)df_* \tag{2}$$

The integrals in Eqn. (1) and (2) are analytically intractable. Thus we need either analytic approximations of the integral such as Laplace approximation [12] and expectation propagation methods [13], or Monte Carlo sampling based solutions [14]. In this paper we use Laplace approximation.

The posterior of the latent variable f_* under the Laplace approximation is a Gaussian distribution: $f_*|X,\boldsymbol{y},\boldsymbol{x}^* \sim \mathcal{N}(\bar{f}_*, V_*)$ with the mean and the variance expressed as

$$\bar{f}_* = \boldsymbol{k}_*^T K^{-1}\hat{\boldsymbol{f}} = \boldsymbol{k}_*^T \nabla \log P(\boldsymbol{y}|\hat{\boldsymbol{f}}) \tag{3}$$

$$V_* = k(\boldsymbol{x}^*,\boldsymbol{x}^*) - \boldsymbol{k}_*{}^T (K + W^{-1})^{-1}\boldsymbol{k}_* \tag{4}$$

where $W = -\nabla\nabla \log P(\boldsymbol{y}|\boldsymbol{f})$ is a diagonal matrix with non-negative diagonal elements, K is the covariance matrix of the training samples, $\boldsymbol{k}_*$ is the vector of covariances between $\boldsymbol{x}^*$ and the training samples, $k(\boldsymbol{x}^*,\boldsymbol{x}^*)$ is the prior variance at $\boldsymbol{x}^*$, $\hat{\boldsymbol{f}} = \arg\max_{\boldsymbol{f}} P(\boldsymbol{f}|X,\boldsymbol{y})$. The decision boundary corresponds to $\bar{f}_* = 0$ or $\bar{\pi}_* = 0.5$. We predict $y_* = -1$ when $\bar{f}_* < 0$, and $y_* = 1$ otherwise. Williams and Rasmussen [11] further detail the Laplace approximation for the binary GP classifier.

For example, in car body styling design $f(\boldsymbol{x})$ may represent the Gaussian process classifier's belief customer satisfaction with a car body design represented by $\boldsymbol{x}$ (roof length, wheelbase, *etc.*). This belief is stochastic and has a Gaussian distribution: $f(\boldsymbol{x}) \sim \mathcal{N}(\bar{f}, V)$. Specifically, $\bar{f} = 0$ represents the boundary of acceptable car designs (the feasible design domain); one of our goals is to best refine our estimate of $\bar{f}_*$ over time. In contrast, V represents how uncertain we are about a car's acceptability; another, competing goal is to explore designs about which we are uncertain.

The covariance K, $\boldsymbol{k}_*$, and $k(\boldsymbol{x}^*,\boldsymbol{x}^*)$ are specified by a kernel function. In this paper we use the Gaussian kernel:

$$k(\boldsymbol{x},\boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2l^2}\right) \tag{5}$$

where l is the length scale. In our context, the kernel function $k(\boldsymbol{x},\boldsymbol{x}')$ measures the similarity between the two designs $\boldsymbol{x}$ and $\boldsymbol{x}'$. The kernel encodes the assumption on f that "similar inputs $\boldsymbol{x}$ should have similar outputs $f(\boldsymbol{x})$" (*i.e.*, similar designs should have similar feasibility). The specific choice of kernel is not central to the paper's contributions. In practice, one may use any positive semi-definite kernel.

2.2 Sequential Sampling and Active Learning

Sequential sampling methods are one means to identify feasible designs with implicit constraints. The typical procedures works as follows: 1) select initial samples via methods like Latin hypercube sampling; 2) learn a surrogate model (*e.g.*, Gaussian Process regression model) for the implicit constraint using those samples; 3) compute a score using the learned model for any set of design variables $\boldsymbol{x}$; 4) select a new sample based on those scores; and 5) repeat steps 2-4. This technique, although applied in design research [3,4,5,7], is also heavily researched in an area called *Active Learning*.

Active learning is a semi-supervised machine learning approach that interactively queries labels for data. In our case, the labels are the validity of designs. There are three main scenarios of active learning: membership query synthesis, stream-based selective sampling, and pool-based sampling [15]. Here we apply pool-based sampling (where queries are selected from a large pool of unlabeled data X_U). In each iteration, pool-based sampling uses the labeled samples from all the previous iterations to train a classifier, updates the measure of informativeness based on the classifier's prediction, and picks the most informative sample to next query and label. The pool is often generated via random sampling in the input space using a fixed-size pool, though we show later in the paper (Sec. 4.1) that intelligently modifying this pool can improve performance.

To achieve higher accuracy with fewer queried labels, active learning algorithms follow various *query strategies*. *Random sampling* is the simplest, where a learner randomly picks queries from the set of unlabeled samples X_U. *Uncertainty sampling* is the most commonly used strategy in which a learner queries an instance whose label is the least certain under the current model (*i.e.*, is least certain about the label). For a probabilistic binary classification model, this approach is straightforward [16,17]:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}\in X_U}{\arg\max}\ 1 - P_\theta(\hat{y}|\boldsymbol{x}) \tag{6}$$

where the estimated label $\hat{y} = \arg\max_y\ P_\theta(y|\boldsymbol{x})$ is the most probable label (in our case, valid/invalid) at $\boldsymbol{x}$ under the model θ. Initially, the model θ is trained with a seed set of labeled samples $X_{L_0} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(n_0)}\}$. Uncertainty sampling will query an instance closest to the decision boundary (*e.g.*, Point 1 or 3 in Fig. 2).

Query strategies like uncertainty sampling exploit the most informative sample for prediction by refining the decision boundary. But in cases where there are multiple disconnected feasible domains, exploring areas away from the current decision boundary is necessary to discover other decision boundaries. Therefore, we have to trade off *exploiting* the decision boundary with *exploring* unknown regions.

Past research studied balancing exploitation and exploration in active learning. One common way is to query based on the sample's output distribution obtained by a probabilistic model (*e.g.*, a Gaussian process). To determine the informativeness of unlabeled data, such approaches use the model's predictive mean and variance to either model the uncertainty of unlabeled data [18, 19, 20, 21, 22, 23] or estimate the expected model change affected by unlabeled data [24, 25]. Instead of using a probabilistic model, others use deterministic models (*e.g.*, support vector machines) and combine the classification results with some geometric rules for selecting queries [26, 7, 27, 28]. Another body of work models active learning

as a multi-armed bandit problem, automatically selecting queries to balance exploitation and exploration via a performance estimate [29, 30].

In previous work, exploitation and exploration trade-off occured on a bounded domains or within fixed sampling pools. However, there are cases where we can synthesize samples at any point in the input data space and we are uncertain about the bounds of that space. For example, we may not know the parameter bounds for a design, but by specifying any set of parameters, we can generate a design instance (either feasible or infeasible). In this paper, we introduce a method of using active learning to expand the initial sample region, and learn the feasible or infeasible regions in an unbounded design space. A naïve solution would progressively expand a bounded input space, and apply the existing active learning techniques. However, a input space expansion rule and a query strategy should ensure that the selected samples are optimal given the query strategy regardless of the imposed space boundary (*i.e.*, the imposed boundary should not interfere with the query strategy's decision). Thus designing such a method is not trivial.

2.3 Design Space Exploration and Design Synthesis

To identify feasible design regions or find novel designs, one needs to explore the design space. However, one problem with exploring design spaces is that they are usually high-dimensional. This creates two issues: 1) the active learner must query more exponentially more samples due to the curse of dimensionality [31] while most designs in that design space are often unrealistic or nonfunctional; and 2) Gaussian processes perform poorly over a high-dimensional space. Past research has tackled these problems by using a low-dimensional representation of a design space, such that one can explore that low-dimensional space and synthesis designs. One way to get this low-dimensional representation is to use manifold learning such as Multi-Dimensional Scaling (MDS) to map the design space to a low-dimensional embedding space, and reconstruct the original shapes or synthesize new shapes based on how shapes vary [32]. Instead of learning a one-way mapping via methods like MDS, one can also directly learn a two-way mapping between the design space and a low-dimensional space by using methods like principal component analysis (PCA) or autoencoders [33, 34, 8, 9]. Another way is to associate designs with a few semantic attributes (*e.g.*, compactness/luxuriousness of a car) by crowd-sourcing, and then learn how those attributes map to the design variables, such that new designs can be synthesized by editing these semantic attributes [35].

3 ε-Margin Sampling

ε-margin sampling queries samples based on the probability of $\boldsymbol{x}$ being labeled as $-\hat{y}$ with some degree of certainty (ε):

$$\begin{aligned} p_\varepsilon &= \begin{cases} P(f < -\varepsilon|\boldsymbol{x}), & \text{if } \hat{y} = 1 \\ P(f > \varepsilon|\boldsymbol{x}), & \text{if } \hat{y} = -1 \end{cases} \\ &= P(\hat{y}f < -\varepsilon|\boldsymbol{x}) \end{aligned} \tag{7}$$

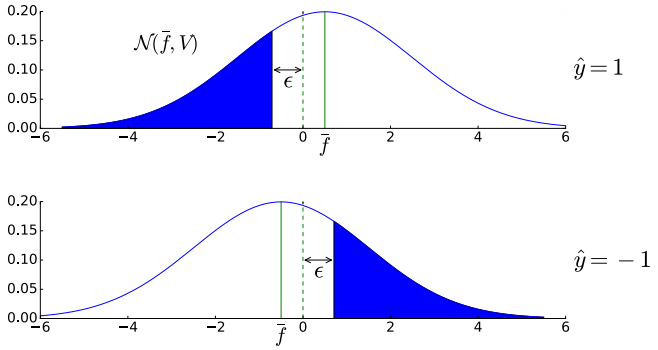


Fig. 1: The probability density function of the latent function f. The shaded areas represent the probability of a sample being labeled as the opposite label with some degree of certainty (controlled by the margin ε). When the predicted label $\hat{y}=1$, the probability is $P(f<-\varepsilon)$; and when $\hat{y}=-1$, the probability is $P(f>\varepsilon)$.

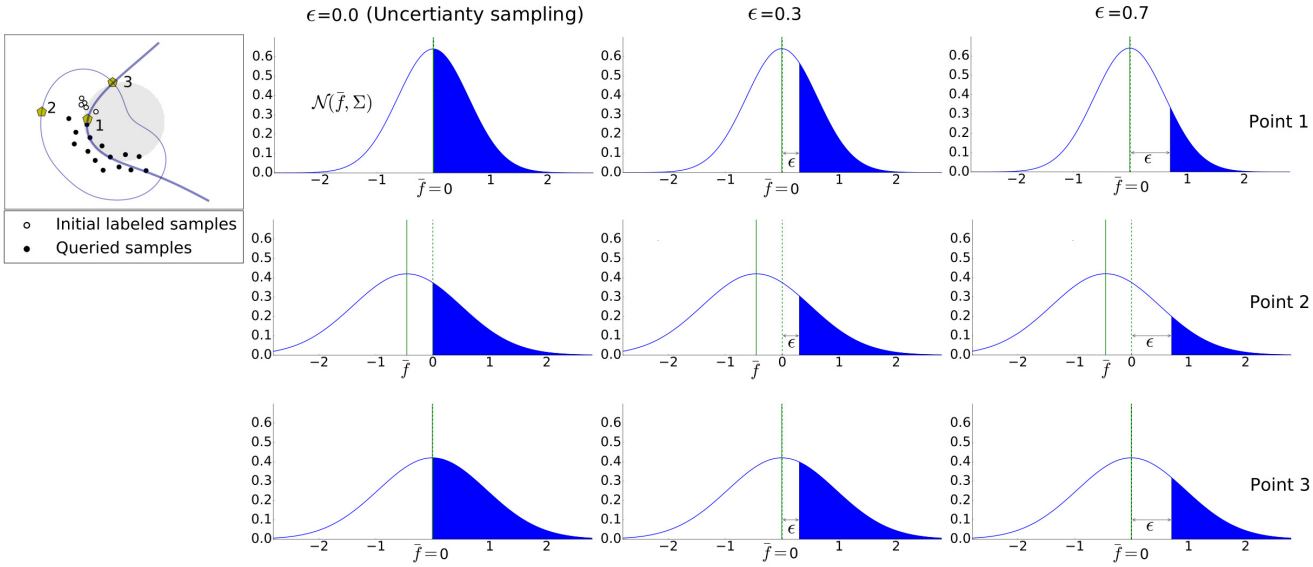


Fig. 2: The value of p_ε under different ε. On the left plot, the gray area is the ground truth of the feasible domain; the thicker line is the decision boundary obtained by the GP classifier; the thinner lines are isocontours of V; and the circle points are training samples. When $\varepsilon=0$, $p_\varepsilon=0.5$ for all the points on the decision boundary, thus in this case ε-margin sampling is equivalent to uncertainty sampling. As ε increases, ε-margin sampling starts to take the variance into consideration, *i.e.*, given two points (*e.g.*, Points 1 and 3) on the decision boundary it will pick the one with a higher variance. Whereas for points having the same variance (*e.g.*, Points 2 and 3), it always prefers the one on the decision boundary.

where $\hat{y}$ is the estimated label of $\boldsymbol{x}$ (Fig. 1). The intuition is that the sample which has the highest probability of having an opposite label is the most informative. We can express p_ε using the cumulative distribution function of a standard Gaussian distribution:

$$p_\varepsilon = P(\hat{y}f < -\varepsilon|\boldsymbol{x}) = \Phi\left(-\frac{|\bar{f}|+\varepsilon}{\sqrt{V}}\right) \tag{8}$$

where $\Phi(\cdot)$ is the cumulative distribution function of standard Gaussian distribution $\mathcal{N}(0,1)$. The ε-margin sampling picks a sample that has the largest p_ε:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}\in X_U}{\arg\max}\ \Phi\left(-\frac{|\bar{f}|+\varepsilon}{\sqrt{V}}\right) \tag{9}$$

Since $\Phi(\cdot)$ is a monotonically increasing function, p_ε decreases with $(|\bar{f}|+\varepsilon)/\sqrt{V}$. Thus finding the maximum of p_ε is equivalent to finding the minimum of $(|\bar{f}|+\varepsilon)/\sqrt{V}$. Figure 2 shows the value of p_ε under different ε. When $\varepsilon = 0$, Eqn. (7) becomes

$$\begin{aligned} p_0 &= \begin{cases} P(f<0|\boldsymbol{x}), & \text{if } \hat{y}=1 \\ P(f>0|\boldsymbol{x}), & \text{if } \hat{y}=-1 \end{cases} \\ &= \begin{cases} 1-P(f>0|\boldsymbol{x}), & \text{if } \hat{y}=1 \\ 1-P(f<0|\boldsymbol{x}), & \text{if } \hat{y}=-1 \end{cases} \\ &= 1-P(\hat{y}|\boldsymbol{x}) \end{aligned}$$

Thus, for $\varepsilon = 0$, ε-margin sampling is equivalent to uncertainty sampling (Eqn. (6)). As ε increases, the value of $(|\bar{f}|+\varepsilon)/\sqrt{V}$ is affected less by $|\bar{f}|$ and more by V. Thus ε-margin sampling will take the variance into consideration, *i.e.*, given two points (*e.g.*, Points 1 and 3) on the decision boundary it will pick the one with a higher variance. For points with the same variance (*e.g.*, Points 2 and 3), it always prefers one closer to the decision boundary.

When an active learning algorithm queries a point on the decision boundary, it is refining the boundary (exploitation); while when the algorithm queries a point with a high variance, it is exploring unknown areas (exploration).Therefore, increasing the margin $\varepsilon > 0$ trades off exploitation for exploration.

4 Domain Expansion with Active learning

This paper focuses on the active learning in an unbounded space. To do so, we introduce a method that progressively expands the currently known region and discovers new feasible domains. In a *domain expansion problem*, given an expensive function $h : \mathcal{X} \in \mathbb{R}^d \to \{-1,1\}$ that evaluates each point $\boldsymbol{x}$ in an unbounded input data space $\mathcal{X}$, we want to find the feasible domain $\{\boldsymbol{x} \in \mathcal{X} | h(\boldsymbol{x}) = 1\}$. The domain boundaries may be disconnected, in which case there are multiple feasible domains separated by regions of infeasible space. Our domain expansion problem differs from other active learning problems in that we have to sample from an unbounded space. Obviously, we cannot discover *all* the feasible domains because the input space is infinitely large. Thus our proposed method progressively expands our knowledge of that space and its feasible domains by first refining domains that we know about and then, budget permitting, searching outward to discover other domains.

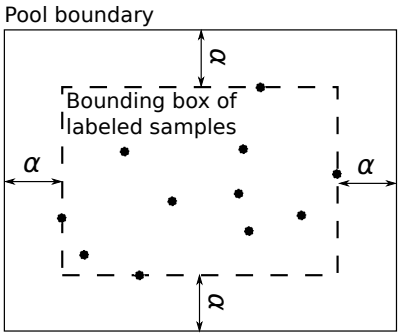


Fig. 3: The flexible pool boundary. In each active learning iteration, we set the pool boundary by expanding the bounding box of the current labeled samples by a constant α.

We can also express the domain expansion problem as $h' : \mathcal{X} \in \mathbb{R}^d \rightarrow \mathbb{R}$, in which case the problem relates to level set estimation [21], where we want to find the superlevel set $\{\boldsymbol{x} \in \mathcal{X} | h'(\boldsymbol{x}) > \tau\}$ or the sublevel set $\{\boldsymbol{x} \in \mathcal{X} | h'(\boldsymbol{x}) \leq \tau\}$ in an unbounded space $\mathcal{X}$ given the function h' and a threshold τ.

Our proposed active learning method first generates a bounded sampling pool nearby training samples. Second, it uses ε-margin sampling to select and label a sample from that pool. Lastly, it progressively expands the sampling pool. These steps repeat until the sampling budget is exhausted. Interested readers can review the full source code to fully reproduce the below steps.[1]

4.1 Expand the Pool

A pool of unlabeled samples X_U can be generated by randomly sampling points in the parameter space. Although we assume an unbounded parameter space, obviously we cannot generate the pool from an unbounded space. Thus we define a flexible pool boundary that expands progressively as we explore the parameter space. Specifically, we set the range of the j-th dimension of the parameter space as $[\min_{\boldsymbol{x}^{(i)} \in X_L} x_j^{(i)} - \alpha,\ \max_{\boldsymbol{x}^{(i)} \in X_L} x_j^{(i)} + \alpha]$, where X_L is the set of labeled samples, $x_j^{(i)}$ is the j-th component of the i-th sample $\boldsymbol{x}^{(i)}$, and α is the *expansion rate* that controls how much the pool boundary expands based on the current bounding box of labeled samples (Fig. 3). The next section discusses how to set α such that the sampling pool is optimal for the query strategy, regardless of the imposed pool boundary (*i.e.*, the pool boundary should not bias with the query strategy's decision).

4.2 Query Strategy

ε-margin sampling performs both exploitation and exploration by balancing proximity to the decision boundary and the size of the variance. However, this creates a problem with an expanding domain; nïave querying will attempt to pick points infinitely far away from current training samples. Therefore, we add another criteria—the Euclidean distance between the new sample $\boldsymbol{x}^*$ and some center $\boldsymbol{c}$—that keeps the active learner from querying points too distant from current training

[1] https://github.com/IDEALLab/domain_expansion_jmd_2017

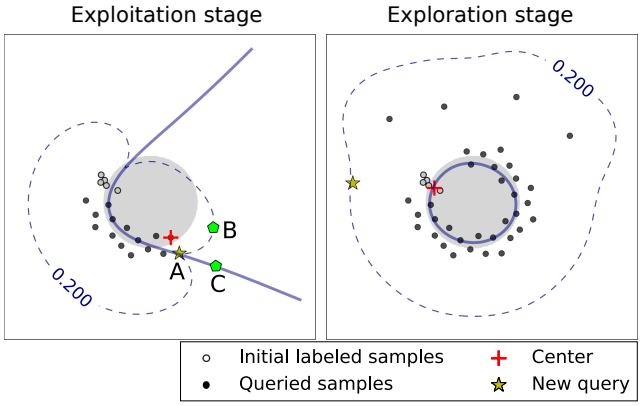


Fig. 4: Queries at the exploitation stage (left) and the exploration stage (right). The gray area is the ground truth of the feasible domain. The solid line is the decision boundary obtained by the GP classifier; and the dashed line is the isocontour of p_ε. At the exploitation stage, the center $\boldsymbol{c}$ is the last queried positive sample, which makes the next query stay along the decision boundary. At the exploration stage, $\boldsymbol{c}$ is the centroid of the initial positive samples, which keeps the queries centered around the existing (real-world) samples rather than biasing towards some direction.

samples. This criteria can be expressed as

$$\begin{aligned} &\min_{\boldsymbol{x}\in X_U} \ \|\boldsymbol{x}-\boldsymbol{c}\|_2 \\ &\text{s.t. } p_\varepsilon = \Phi\left(-\frac{|\bar{f}|+\varepsilon}{\sqrt{V}}\right) \geq t \end{aligned} \tag{10}$$

where $t \in (0, 0.5)$ is a threshold that determines the expansion speed. Since $\Phi(-(|\bar{f}|+\varepsilon)/\sqrt{V})$ reaches its upper bound when $\bar{f}=0$ and $V = V_{max}$ (which corresponds to the prior of the GP or the posterior far away from the training samples), we have $t \leq \Phi(-\varepsilon/V_{max})$, where V_{max} is the maximum predictive variance. In Eqn. (4), $K+W^{-1}$ is positive semidefinite, thus $\boldsymbol{k}_*{}^T(K+W^{-1})^{-1}\boldsymbol{k}_* \geq 0$ and $V \leq k(\boldsymbol{x}^*,\boldsymbol{x}^*) = 1$ if we use a Gaussian kernel. Thus $t \leq \Phi(-\varepsilon)$. We can set $t = \eta\Phi(-\varepsilon)$, where η is a coefficient slightly smaller than 1.0. In practice, we set $\eta = 0.8$.

The expansion rate α should be large enough such that the pool boundary will not interfere with the query strategy—as if each query was chosen from an infinite space by the query strategy. However, as α increases, we have to evaluate a larger population of samples at each iteration (assuming a fixed sample pool density), which means the computational cost increases. We can set α based on the Gaussian kernel's length scale l. The length scale l controls the range of the training samples' effects on the posterior mean and variance, and hence p_ε. In general, when a point has a distance of l away from the training samples, it will not be affected by those training samples. Then for that point we will have $p_\varepsilon = \Phi(-\varepsilon)$. In our case, if we set $\alpha = l$ then all the samples outside the pool boundary will have $p_\varepsilon = \Phi(-\varepsilon)$. And if the threshold $t = \eta\Phi(-\varepsilon)$ where $\eta < 1$, it is guaranteed that the inequality in Eqn. (10) will have a feasible set inside the pool boundary. Thus the optimal solution of Eqn. (10) will be inside the pool boundary regardless of its existence. Therefore the pool boundary does not interfere with the query strategy's decision.

4.3 Relocate Center

When a learnr discovers a boundary that it has not completely exploited, the learner should first exploit that boundary, after which it should find other unknown boundaries by expanding domain. Thus the domain expansion process consists of two stages: 1) exploitation, where a learner queries along the decision boundary; and 2) exploration, where a learner queries away from current decision boundaries (Fig. 4). These two stages may oscillate if there are many disconnected feasible regions. For each stage, we use the following heuristics to set the center $\boldsymbol{c}$:

1. If both positive and negative samples occur within the last M queries, set $\boldsymbol{c} = \boldsymbol{x}^+$, where $\boldsymbol{x}^+$ is the most recent positive sample.
2. Otherwise, set $\boldsymbol{c}$ as the centroid of the initial positive samples: $\boldsymbol{c} = \sum_{i=1}^{n_0} \boldsymbol{x}^{(i)}(y_i+1)/\sum_{i=1}^{n_0}(y_i+1)$, where n_0 is the number of initial labeled samples.

When both positive and negative samples appear within the latest queries, there is at least one unexploited decision boundary (*i.e.*, the exploitation stage), and the active learner should query along that boundary.

Theorem 1. *Under* ε*-Margin Sampling, a new query* $\boldsymbol{x}^*$ *will lie at the intersection of the decision boundary and the iso-contour of* $\Phi(-(|\bar{f}|+\varepsilon)/\sqrt{V}) = t$ *(Point A in Fig. 4), if there are feasible solutions to Eqn (10) on the decision boundary, and the center* $\boldsymbol{c} \in X_L$ *is close to* $\boldsymbol{x}^*$.

Proof. For Point A we have $\bar{f}_A = 0$ and $\Phi(-(|\bar{f}_A|+\varepsilon)/\sqrt{V_A}) = t$. Suppose there is another unlabeled sample $\boldsymbol{x}_B$ (Point B in Fig. 4) with $|\bar{f}_B| > 0$ and $\Phi(-(|\bar{f}_B|+\varepsilon)/\sqrt{V_B}) = t$. In that case, we have $V_B > V_A$. Based on Eqn. (4), the predictive variance V_* solely depends on $\boldsymbol{k}_*$ given the training samples (*i.e.*, fixed positive semidefinite matrix $K + W^{-1}$, and $k(\boldsymbol{x}^*, \boldsymbol{x}^*) = 1$). Suppose the sample $\boldsymbol{x}^*$ is very close to $\boldsymbol{c}$ and $\boldsymbol{c} \in X_L$, such that $\boldsymbol{k}_*$ is mainly affected by the covariance between $\boldsymbol{x}^*$ and $\boldsymbol{c}$, and hence the distance $\|\boldsymbol{x}^* - \boldsymbol{c}\|$. Then V_* will increase with $\|\boldsymbol{x}^* - \boldsymbol{c}\|$. Therefore we have $\|\boldsymbol{x}_B - \boldsymbol{c}\| > \|\boldsymbol{x}_A - \boldsymbol{c}\|$. Thus the active learner prefers $\boldsymbol{x}_A$ to $\boldsymbol{x}_B$ due to the objective function in Eqn (10). For the same reason, the active learner prefers $\boldsymbol{x}_A$ to a sample $\boldsymbol{x}_C$ (Point C in Fig. 4) with $\bar{f}_C = 0$ and $\Phi(-(|\bar{f}_C|+\varepsilon)/\sqrt{V_C}) > t$. Thus the active learner will query a point at the intersection of the decision boundary and the isocontour of $p_\varepsilon = t$.

In practice, we set the center to be the last positive sample because this forces new queries to stay close to positive samples (which we are interested in). When the latest queries contain only a single class of samples, it indicates that the active learner is no longer querying samples on the decision boundary since $\Phi(-(|\bar{f}|+\varepsilon)/\sqrt{V}) < t$ (Fig. 4). The active learner starts to explore and look for other decision boundaries. The second rule ensures that the exploration is centered around the existing (real-world) samples rather than biasing towards some direction.

4.4 Hyperparameters for ε-Margin Sampling

The ε-margin sampling method uses two hyper-parameters—the margin ε and the length scale l.

The margin ε controls the query strategy's emphasis on exploration. When $\varepsilon = 0$, the proposed method is equivalent to uncertainty sampling, which only samples points on the decision boundary ($\bar{f} = 0$), *i.e.*, exploitation. As ε increases, the

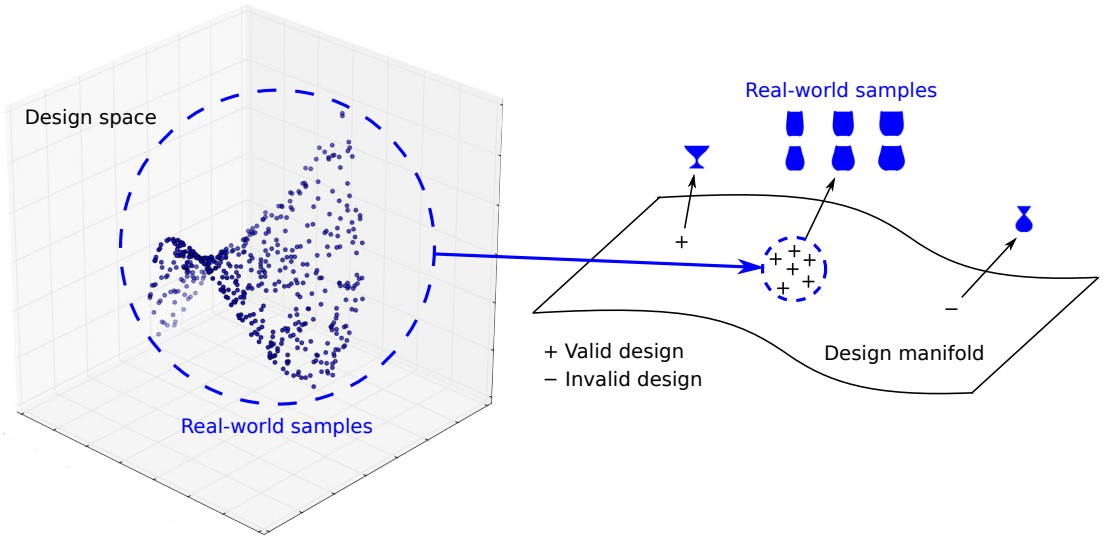


Fig. 5: 3D visualization of high dimensional design space showing that design variables actually lie on a 2-dimensional manifold [8, 9]. At a point away from the real-world stemless glass samples, the glass contours are self-intersecting; at another point, the shape becomes a stem glass.

algorithm prefers to sample points with higher variance V, *i.e.*, exploration (as illustrated in Fig. 2). Thus, as ε increases, the learner prioritizes expanding the domain faster versus refining the boundary accuracy.

The length scale l controls the influence that a point has on a Gaussian process. When l is larger, the function $p_\varepsilon = \Phi(-(|\bar{f}|+\varepsilon)/\sqrt{V})$ is smoother and increases slower, thus the next query will be farther away from the previous queries. This means the distances between queried points increase. The upside of this is that the domain expands faster, while the downside is that we may overshoot small feasible domains (see Sec. 6.1.2).

In general, we recommend $\varepsilon < 1$, because when $\varepsilon \gg \bar{f} \in [0,1]$, the the algorithm neglects the effect of $\bar{f}$ in $p_\varepsilon = \Phi(-(|\bar{f}|+\varepsilon)/\sqrt{V})$ and will bias queries awau from the decision boundary during exploitation. If we find that the domain expansion is too slow, we can increase l; or if we are concerned about missing small feasible domains, we can decrease l. In practice, we have found an initial guess of $\varepsilon = 0.7$ and $l = 1.0$ worked well in our below experiments, with adjustments to ε. Intuitively, ε and l together control the desired accuracy of the decision boundary, though this precise relationship is a topic for future work.

5 Using Design Manifolds to Synthesize Novel Designs

It is difficult to directly explore a real-world design space since it is usually high-dimensional, and most points in that space will be unrealistic or nonfunctional. To handle real-world design spaces, our proposed method assumes that design variables original expressed in a high-dimensional design space $\mathcal{X}$ usually lie on a lower-dimensional *design manifold* (Fig. 5) [8, 9, 36, 37]. We can thus project designs on that lower-dimensional manifold $\mathcal{F}$ by a mapping $g : \mathcal{X} \to \mathcal{F}$. Then given any point in that space, we can synthesize new designs by a reverse mapping $g' : \mathcal{F} \to \mathcal{X}$. This reduces the problem of exploring the high-dimensional design space $\mathcal{X}$ to exploring a corresponding embedding space $\mathcal{F}$.

Just like in the original design space, there are boundaries for feasible designs in the embedded space. The function

that evaluates feasible domains can thus be expressed as $h : \mathcal{F} \to \{-1, 1\}$. Figure 5 shows an example of a glassware design manifold, where the synthesized contours are self-intersecting at a point away from the real-world samples. We call these *invalid designs*—designs that are unrealistic or nonfunctional in the real-world. Since real-world samples are all valid designs, normally designs lying between any two real-world samples (*i.e.*, inside the convex hull of all the real-world samples in the embedding space) will also be valid [8,9]. However, this assumption may not hold for designs that lie *on the manifold* but *beyond* the real-world samples. Since they are on the manifold, they obey similar rules of variation and deformation learned from the real-world samples, but because they are away from the real-world samples, there is no guarantee that those designs are functional or aesthetically valid.

On a manifold that preserves pairwise distances between samples in the design space, designs far away from the real-world samples will look different from them [8,9]. If these designs remain valid, despite being far away from the real-world samples, then we are discovering innovative designs. The methods presented in this paper are one way to achieve this creative exploration, even in high dimensional design spaces. Note that this method assumes that the dimensionality of the design space can be reduced. In cases where this assumption does not hold, we have to directly explore the original high-dimensional design space.

6 Experimental Evaluation and Discussion

We conducted experiments on benchmark functions and real-world design examples to demonstrate how well our proposed method identifies feasible domains and discovers novel designs. To generate the pool for each experiment, we synthesized 400 samples per a 1×1 area from a uniform distribution inside the initial pool boundary. We benchmark the ε-margin sampling with random sampling and uncertainty sampling, as well as characterize some of the properties of our method (such as the role of the kernel length scale l and the role of the margin ε).

6.1 Feasible Domain Identification

For the benchmark function examples, we can manually define feasible domains as the ground truth. We compare each query strategy's exploitation and exploration behavior and compute the classification's F1 score. The F1 score is a measure of accuracy usually used in binary classification. It is expressed as $F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$ where $\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$ and $\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$.

6.1.1 Synthetic Example: Branin Function

Since we may expect disconnected feasible domains in a parameter space, we created a 2-dimensional test case where feasible domains form separate "islands". Specifically, we use the Branin function as an indicator of whether a sample is inside the feasible domain. The Branin function is $g(\boldsymbol{x}) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$. We define the label $y = 1$ if $g(\boldsymbol{x}) < 8$ and $-5 < x_1 < 12, -5 < x_2 < 20$, and $y = -1$ otherwise. The resulting feasible domains resemble three isolated "islands" (Fig. 6).

The initial samples are randomly generated inside the region where $x_1 \in [1.0, 2.0]$ and $x_2 \in [3.0, 4.0]$. For the Gaussian

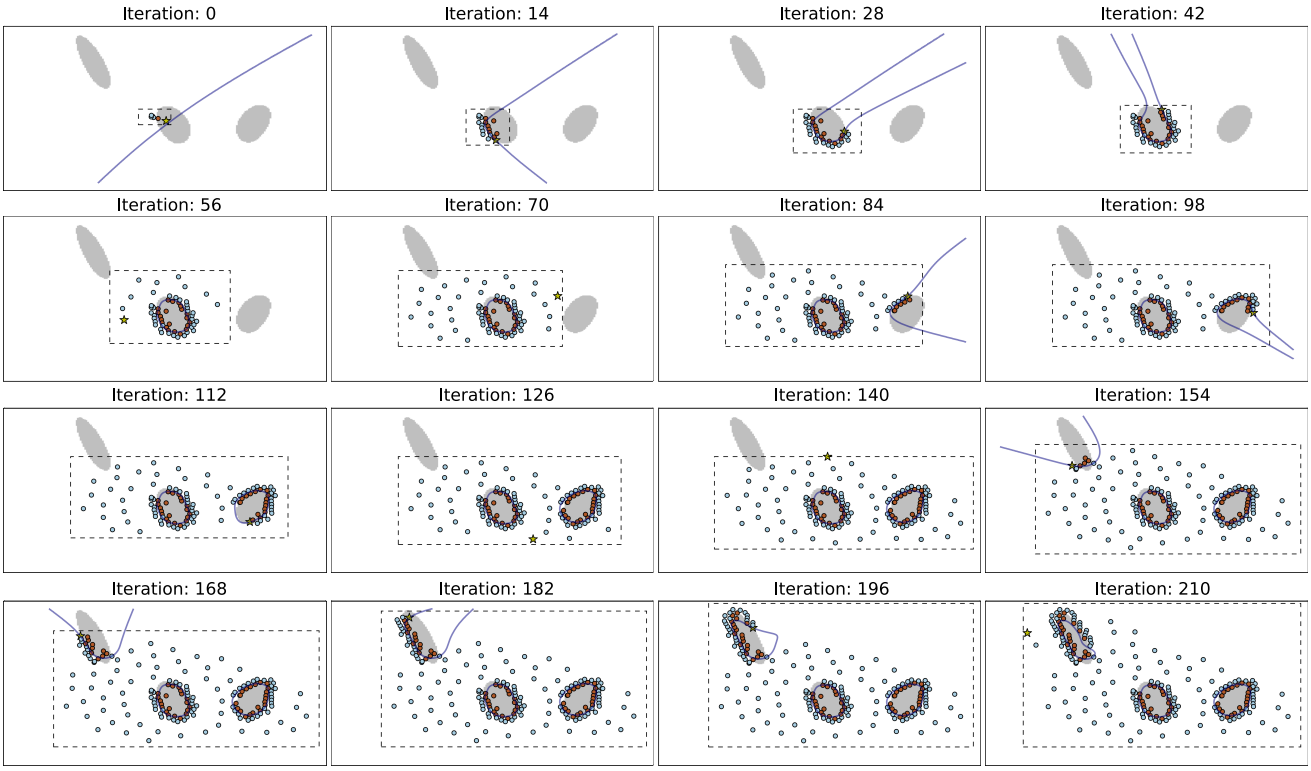


Fig. 6: Domain expansion process for the Branin example. The points are queried samples before the current iteration; and the stars are current queries. The solid lines are decision boundaries obtained by the GP classifier; and the dashed lines are the pool boundaries.

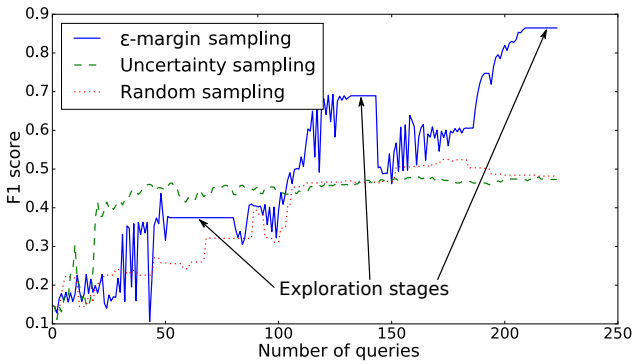
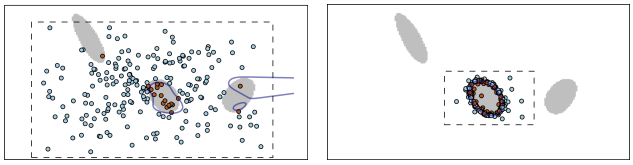


Fig. 7: F1 scores for the Branin example. For ε-margin sampling, during exploration stages, the exploited decision boundaries do not change, so does the F1 score; while during rest of the time (exploitation stages), the F1 score shows a fluctuant increase as the decision boundary changes.

process, we used a Gaussian kernel (Eqn. (5)). We set the kernel length scale $l = 1.0$, and the margin $\varepsilon = 0.7$.

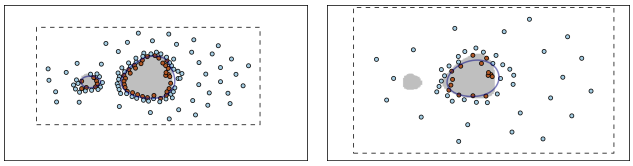
As shown in Fig. 6, the algorithm oscillates between exploitation and the exploration stages (*i.e.*, exploit $\rightarrow$ explore $\rightarrow$ exploit $\rightarrow \ldots$). It exploits whenever it discovers a new decision boundary via exploration, and keep exploiting until the uncertainty about the boundary becomes small (as determined by ε). This behavior allows the learner to exploit a discovered decision boundary as quickly as possible.

To measure our method's performance, we randomly generated 10 test samples per a 1×1 area, and computed the test F1 scores for domain expansion using ε-margin sampling, uncertainty sampling, and random sampling. For uncertainty



(a) Random sampling. (b) Uncertainty sampling.

Fig. 8: Queried samples for random sampling and uncertainty sampling within 210 iterations. The solid lines are decision boundaries obtained by the GP classifier; and the dashed lines are the pool boundaries.



(a) Length scale $l = 0.5$. (b) Length scale $l = 1.0$.

Fig. 9: Queried samples by ε-margin sampling with different GP kernel length scales. A large length scale accelerates exploration but may miss small feasible domains.

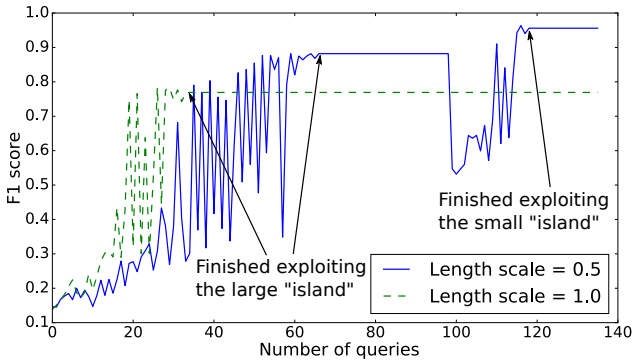


Fig. 10: F1 scores for the Hosaki example. With a larger length scale, the F1 score increases faster, but may stop increasing because it cannot discover small feasible domains.

sampling and random sampling, we used the same pool expansion rule as mentioned in Sec. 4.1. As shown in Fig. 7, the F1 scores of ε-margin sampling fluctuated during the exploitation stage (due to the changing decision boundary), and then stabilized during the exploration stage. For ε-margin sampling, the F1 score is around 0.85; whereas for random sampling and uncertainty sampling, the F1 scores are less than 0.5 at the last iteration. The problem with random sampling is that it has no strategy for balancing exploitation and exploration, thus a large portion of its query budget is wasted on unimportant samples (Fig. 8a). The problem with uncertainty sampling is that it only focuses on the current decision boundary and does not explore other regions (Fig. 8b).

6.1.2 Synthetic Example: Hosaki Function

What if the feasible domains are different in length scale (*i.e.*, one large island and one small island)? To test such a case, we use the Hosaki function: $g(\boldsymbol{x}) = \left(1 - 8x_1 + 7x_1^2 - \frac{7}{3}x_1^3 + \frac{1}{4}x_1^4\right) x_2^2 e^{-x_2}$. We define the label $y = 1$ if $g(\boldsymbol{x}) < -1$ and

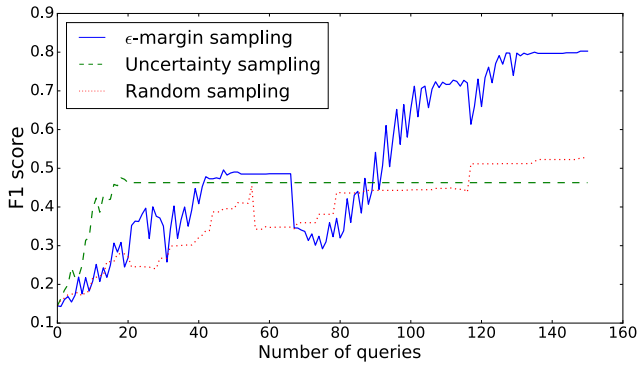


Fig. 11: F1 scores for the two-sphere example.

$0 < x_1, x_2 < 5$, and $y = -1$ otherwise. The resulting feasible domains consists of two isolated "islands"—one large and one small (Fig. 9). In this experiment, the initial samples are randomly generated inside the region where $x_1 \in [4.0, 5.0]$ and $x_2 \in [2.0, 3.0]$. We set the margin $\varepsilon = 0.7$.

When the length scale of the Gaussian kernel is $l = 0.5$, the active learner successfully detected two feasible domains (Fig. 9a). However, when l is increased to 1.0, the active learner expands the domain more aggressively, missing the small "island" and less accurately learns the the large island's decision boundary (Fig. 10). Intuitively, the length scale implicitly encodes our belief about the minimum size of possible islands. When this choice over-estimates the actual length scale, exploration may be too aggressive. If the length scale is too small, the exploration will become too conservative and take longer to discover new areas. Fruitful areas for future research include adaptive ways of choosing l or even considering non-stationary kernel spaces that could model different length scales in different portions of the space.

6.1.3 Synthetic Example: Two Sphere Feasible Domains

To test how the proposed domain expansion method performs in higher-dimensional input spaces, we build a test example with two spherical feasible domains. Specifically, the label $y = 1$ if and only if $\min\{\|\boldsymbol{x} - \boldsymbol{a}\|, \|\boldsymbol{x} - \boldsymbol{b}\|\} \leq 1$, which creates two sphere feasible domains of radius 1 centered at $\boldsymbol{a}$ and $\boldsymbol{b}$ respectively. To test over a 3-dimensional input space, we set $\boldsymbol{a} = (0,0,0)$ and $\boldsymbol{b} = (3,0,0)$. The initial samples are generated within the region where $x_1, x_2, x_3 \in [0,1]$.

We computed the F1 scores under ε-margin sampling, uncertainty sampling, and random sampling (Fig. 11). The latter two sampling methods still perform poorly in this case. For ε-margin sampling, we have to set a lower margin ($\varepsilon = 0.5$) and a smaller length scale ($l = 0.5$) to achieve a reasonable classification accuracy, compared with the 2-dimensional input space (*e.g.*, the Branin example), because the number of samples needed to learn an accurate model grows exponentially with the dimensionality (the curse of dimensionality [31]). By using a lower l, we increase the overall sample density; and by reducing ε, we increase samples along the decision boundary (the surfaces of the spheres).

As the dimensionality increases, the design space becomes impossible to sample or explore (with bounded variance) using less than exponential sample size as dimension increases for any method [10]. Thus, all sampling methods will suffer from high computational cost. However, if a high dimensional design space has points or solutions that *actually* vary on a low-dimensional (possibly non-linear) subspace (a design manifold) within that high dimensional space (*i.e.*, it has low

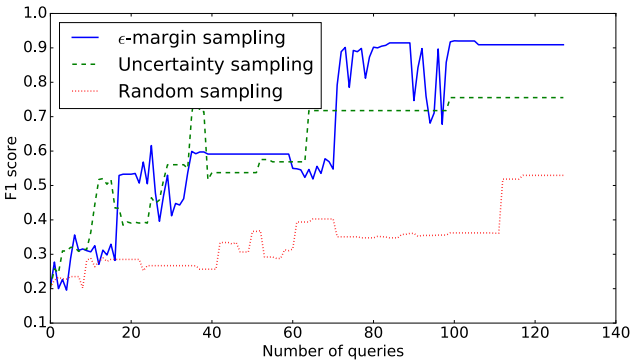


Fig. 12: F1 scores for the airfoil example.

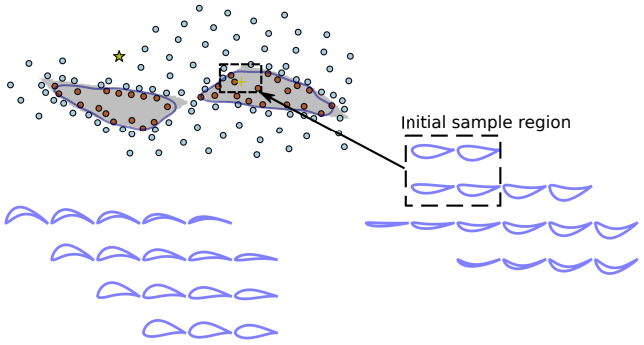


Fig. 13: The estimated feasible domains and corresponding valid airfoil designs. The top left figure shows the initial and queried samples and the estimated feasible domains in the embedding space $\mathcal{F}$. The solid dots represent valid designs, while the hollow dots represent invalid ones. The bottom figure shows the airfoil designs in the estimated feasible domain.

intrinsic dimension), we can sample or explore only the low-dimensional subspace, and hence reduce the computational cost. We demonstrate this approach next.

6.1.4 Example: Airfoil

When the feasible solutions of a high-dimensional design problem only lie on a low-dimensional subspace, it is almost impossible to identify that feasible solution by exploring the original high-dimensional space because the feasible domain only occupies a very small slice of that original space. In this case, we can explore only the subspace on which design solutions vary. This example uses airfoil shapes to demonstrate the approach of performing feasible domain identification in a subspace. An airfoil is the cross-section shape of a wing or blade (of a propeller, rotor, or turbine).

Specifically, we represent an airfoil with 100 Cartesian coordinates from its 2D outline. Our airfoil samples are from the UIUC Airfoil Coordinates Database, which provides the Cartesian coordinates for nearly 1,600 airfoils.[2] We use PCA to learn a two-way mapping between the original design space $\mathcal{X} \in \mathbb{R}^{200}$ and the subspace (embedding space) $\mathcal{F} \in \mathbb{R}^2$. Thus given any point in $\mathcal{F}$, we can get its corresponding representation in $\mathcal{X}$, and synthesize an airfoil design. More details on this embedding can be found in [8]. By incorporating this design embedding, we reduce the problem of exploring the high-dimensional design space $\mathcal{X}$ to exploring a corresponding low-dimensional space $\mathcal{F}$.

[2] http://m-selig.ae.illinois.edu/ads/coord_database.html

Fig. 14: Some of the initial designs used in the stemless glassware example.

To define a feasible domain, we set two simple constraints just for the purpose of demonstrating of our proposed method: (1) the upper and lower curves should not intersect; and (2) the aspect ratio is lower than 0.4. Theses geometric constraints allow easy feasibility checks, and enable us to visually verify the results of our method. To get the ground-truth label (feasibility) of samples for computing F1 scores, we generate test samples from $\mathcal{F}$, get their high-dimensional representations in $\mathcal{X}$ and then check if the two constraints are satisfied using those high-dimensional representations.

We set the length scale of the Gaussian kernel as $l = 0.2$, and the margin $\varepsilon = 0.7$. The F1 scores have similar characteristics to previous examples (Fig. 12). The ε-margin sampling method achieves an F1 score above 0.9, which is much higher than the other two methods. As shown in Fig. 13, started with a few initial samples in a smaller region in $\mathcal{F}$, we uncovered two larger feasible regions which are very close to the ground-truth feasible domains.

6.2 Novelty Discovery

We use two other real-world design examples to show how our proposed method discovers novel designs. Similar to the airfoil example, we represent each design with 100 Cartesian coordinates from their 2D outlines, and use the inverse mapping $g : \mathcal{F} \in \mathbb{R}^2 \to \mathcal{X} \in \mathbb{R}^{200}$ generated by PCA to synthesize samples.

Different from previous examples, we use human judgment as the feasibility criterion. Although we can use some rules to determine the feasibility (*e.g.*, whether the outlines of a design are self-intersecting), a design's actual feasibility may depend on more complicated factors (*e.g.*, functionality and aesthetics). Therefore, initially we start with limited number and styles of real-world designs and do not have labels (*i.e.*, valid or invalid) for synthesized designs outside the real-world design domain until human assessment.

In the following two experiments, we select a sample in $\mathcal{F}$, and synthesize a design $\boldsymbol{x} \in \mathcal{X}$, where $\boldsymbol{x}$ consists of the Cartesian coordinates of that design's outline [8]. Then we use an interface that shows a human oracle the picture of that design, and then the oracle labels the design valid or invalid based on whether the designed shape is aesthetically pleasing. This process forms the function $h : \mathcal{F} \to \{-1, 1\}$. The specific choice of human oracle and interface is not central to the contributions of this paper, serving only as an illustrative real-world example. In practice, we need to take human annotators' mistakes and disagreements into consideration. For example, we can model human disagreement by adding a white noise kernel to the Gaussian process kernel function. The white noise kernel is expressed as $k(\boldsymbol{x}, \boldsymbol{x}') = r$ if $\boldsymbol{x} = \boldsymbol{x}'$ and 0 otherwise, where r is the noise level. In the future, we will look into ways of dealing with the problem of input-dependent noise level (*i.e.*, different degree of human mistakes and disagreements for different design instances) [38, 39].

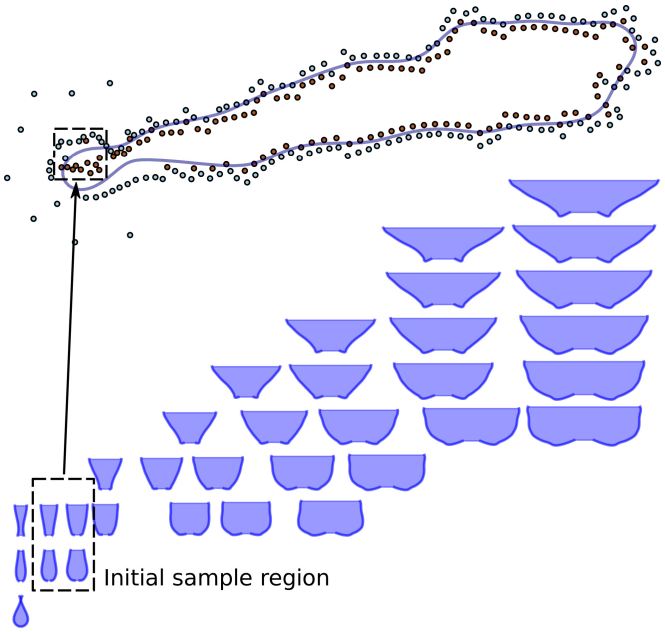


Fig. 15: The discovered feasible domain and valid designs. The top figure shows the initial and queried samples and the estimated feasible domain in the embedding space $\mathcal{F}$ The solid dots represent valid designs, while the hollow dots represent invalid ones. Started with the stemless glasses shown in Fig. 14, the proposed method discovered other types of revolved objects such as vases and bowls.

Fig. 16: Some of the initial designs used in the bottle example.

6.2.1 Example: Stemless Glasses as Initial Samples.

In this example, we used only stemless glasses as initial designs (Fig. 14). We set the length scale of the Gaussian kernel as $l = 2.4$, and the margin $\varepsilon = 1.5$. As shown in Fig. 15, on the given design manifold, our proposed method discovered a feasible domain with other revolved objects such as vases and bowls. The new designs increasingly differ as they get farther away from the initial design samples.

6.2.2 Example: Bottles as Initial Samples.

Similar to the previous example, we used only bottles as initial designs (Fig. 16). We set the length scale of the GP classifier $l = 1.3$, and the margin $\varepsilon = 0.7$. As shown in Fig. 17, our proposed method discovered two feasible domains. In the left feasible domain, there are designs that look like bowling balls and flasks; and in the right feasible domain, there are designs that still looks like bottles, but with a larger aspect ratio. Between these two feasible domains, designs have self-intersecting contours and thus are invalid.

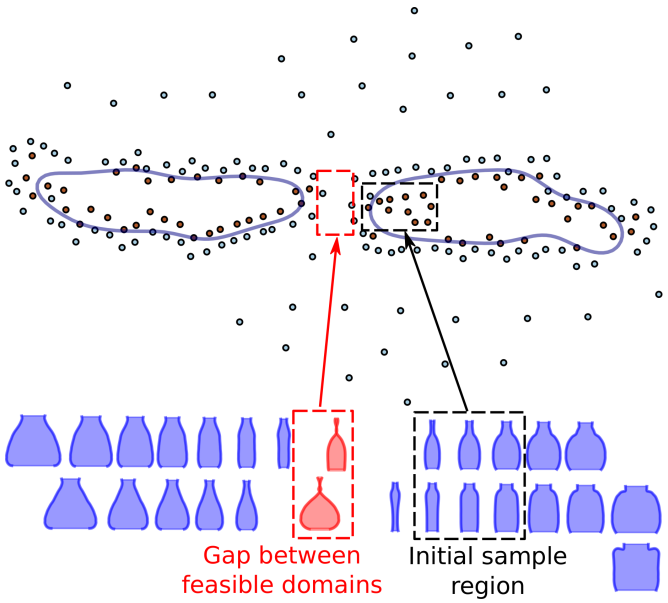


Fig. 17: The discovered feasible domains and valid designs. The solid dots represent valid designs, while the hollow dots represent invalid ones. Started with the bottles shown in Fig. 16, the proposed method discovered two feasible domains, between which there are designs with self-intersecting contours.

7 Conclusion

We proposed an active learning approach to solve the problem of identifying (possibly disconnected) feasible domains in an unbounded design space. This method can be used for identifying the feasible design space with implicit constraints. The method arranges exploitation and exploration stages in a way where it rapidly exploits a discovered decision boundary. One limitation of this method is that we have to tune two hyper-parameters: we set the length scale l according to how precisely one wishes to recover the exact boundary versus the domain expansion speed; and we set the margin ε based on our (problem specific) emphasis on exploitation versus exploration. In the future, we will seek ways to adaptively select l and ε given the query budget and the expected minimum accuracy.

By incorporating the concept of a design manifold, we applied the proposed method to real-world design problems for learning feasible design domains and discovering novel designs. The method discovered designs that have different appearance and functionality from the initial designs. The resulting novel designs depends on the given design manifold. Since PCA is a linear embedding, there is less shape variation among the synthesized designs on the manifold obtained by PCA. If the manifold were instead obtained by using non-linear embeddings, such as Kernel PCA, there is more allowable shape variation, but this may come at the cost of finding fewer valid designs (on account of the difficult in fitting such non-linear subspaces). Thus there is a trade-off between the chance of discovering novel designs and the risk of querying invalid designs. A clear limitation of using a lower-dimensional embedding to resolve the exponential cost of high-dimensional design space exploration is that the Design Manifold may not initially capture important sub-spaces. Part of our future work to resolve this problem includes probabilistic models of not just the domain, but also of the manifold itself with the goal of uncovering both the “right” manifold and design domain via active expansion.

References

[1] Yannou, B., Moreno, F., Thevenot, H. J., and Simpson, T. W., 2005. "Faster generation of feasible design points". In ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 355–363.

[2] Devanathan, S., and Ramani, K., 2010. "Creating polytope representations of design spaces for visual exploration using consistency techniques". *Journal of Mechanical Design,* **132**(8), p. 081011.

[3] Larson, B. J., and Mattson, C. A., 2012. "Design space exploration for quantifying a system models feasible domain". *Journal of Mechanical Design,* **134**(4), p. 041010.

[4] Lee, T. H., and Jung, J. J., 2008. "A sampling technique enhancing accuracy and efficiency of metamodel-based rbdo: Constraint boundary sampling". *Computers & Structures,* **86**(13), pp. 1463–1476.

[5] Zhuang, X., and Pan, R., 2012. "A sequential sampling strategy to improve reliability-based design optimization with implicit constraint functions". *Journal of Mechanical Design,* **134**(2), p. 021002.

[6] Huang, Y.-C., and Chan, K.-Y., 2010. "A modified efficient global optimization algorithm for maximal reliability in a probabilistic constrained space". *Journal of Mechanical Design,* **132**(6), p. 061002.

[7] Ren, Y., and Papalambros, P. Y., 2011. "A design preference elicitation query as an optimization process". *Journal of Mechanical Design,* **133**(11), p. 111004.

[8] Chen, W., Fuge, M., and Chazan, J., 2017. "Design manifolds capture the intrinsic complexity and dimension of design spaces". *Journal of Mechanical Design,* **139**(5), p. 051102.

[9] Chen, W., Chazan, J., and Fuge, M., 2016. "How designs differ: Non-linear embeddings illuminate intrinsic design complexity". In ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, p. V02AT03A014.

[10] Regier, J. C., and Stark, P. B., 2015. "Mini-minimax uncertainty quantification for emulators". *SIAM/ASA Journal on Uncertainty Quantification,* **3**(1), pp. 686–708.

[11] Rasmussen, C., and Williams, C., 2006. *Gaussian Processes for Machine Learning*. the MIT Press.

[12] Williams, C. K., and Barber, D., 1998. "Bayesian classification with gaussian processes". *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **20**(12), pp. 1342–1351.

[13] Minka, T. P., 2001. "A family of algorithms for approximate bayesian inference". PhD thesis, Massachusetts Institute of Technology.

[14] Hastings, W. K., 1970. "Monte carlo sampling methods using markov chains and their applications". *Biometrika,* **57**(1), pp. 97–109.

[15] Settles, B. "Active learning literature survey". *University of Wisconsin, Madison,* **52**(55-66), p. 11.

[16] Lewis, D. D., and Catlett, J., 1994. "Heterogeneous uncertainty sampling for supervised learning". In Proceedings of the eleventh international conference on machine learning, pp. 148–156.

[17] Lewis, D. D., and Gale, W. A., 1994. "A sequential algorithm for training text classifiers". In Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, Springer-Verlag

New York, Inc., pp. 3–12.

[18] Bryan, B., Schneider, J., Nichol, R., Miller, C., Genovese, C. R., and Wasserman, L., 2005. "Active learning for identifying function threshold boundaries". In NIPS, pp. 163–170.

[19] Kapoor, A., Grauman, K., Urtasun, R., and Darrell, T., 2007. "Active learning with gaussian processes for object categorization". In Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, IEEE, pp. 1–8.

[20] Kapoor, A., Grauman, K., Urtasun, R., and Darrell, T., 2010. "Gaussian processes for object categorization". *International journal of computer vision,* **88**(2), pp. 169–188.

[21] Gotovos, A., Casati, N., Hitz, G., and Krause, A., 2013. "Active learning for level set estimation". In Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI '13, AAAI Press, pp. 1344–1350.

[22] Hoang, T. N., Low, B., Jaillet, P., and Kankanhalli, M., 2014. "Nonmyopic ε-bayes-optimal active learning of gaussian processes". In Proceedings of the 31st International Conference on Machine Learning (ICML-14), JMLR Workshop and Conference Proceedings, pp. 739–747.

[23] Schreiter, J., Nguyen-Tuong, D., Eberts, M., Bischoff, B., Markert, H., and Toussaint, M., 2015. "Safe exploration for active learning with gaussian processes". In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, pp. 133–149.

[24] Freytag, A., Rodner, E., Bodesheim, P., and Denzler, J., 2013. "Labeling examples that matter: Relevance-based active learning with gaussian processes". In German Conference on Pattern Recognition, Springer, pp. 282–291.

[25] Freytag, A., Rodner, E., and Denzler, J., 2014. "Selecting influential examples: Active learning with expected model output changes". In European Conference on Computer Vision, Springer, pp. 562–577.

[26] Basudhar, A., and Missoum, S., 2010. "An improved adaptive sampling scheme for the construction of explicit boundaries". *Structural and Multidisciplinary Optimization,* **42**(4), pp. 517–529.

[27] Basudhar, A., Dribusch, C., Lacaze, S., and Missoum, S., 2012. "Constrained efficient global optimization with support vector machines". *Structural and Multidisciplinary Optimization,* **46**(2), pp. 201–221.

[28] Singh, P., van der Herten, J., Deschrijver, D., Couckuyt, I., and Dhaene, T., 2017. "A sequential sampling strategy for adaptive classification of computationally expensive data". *Structural and Multidisciplinary Optimization,* **55**(4), pp. 1425–1438.

[29] Bouneffouf, D., Laroche, R., Urvoy, T., Féraud, R., and Allesiardo, R., 2014. "Contextual bandit for active learning: Active thompson sampling". In International Conference on Neural Information Processing, Springer, pp. 405–412.

[30] Hsu, W.-N., and Lin, H.-T., 2015. "Active learning by learning". In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI'15, AAAI Press, pp. 2659–2665.

[31] Bellman, R., 1957. "Dynamic programming". *Princeton University Press*.

[32] Averkiou, M., Kim, V. G., Zheng, Y., and Mitra, N. J., 2014. "Shapesynth: Parameterizing model collections for coupled shape exploration and synthesis". In Computer Graphics Forum, Vol. 33, Wiley Online Library, pp. 125–134.

[33] Yumer, M. E., Asente, P., Mech, R., and Kara, L. B., 2015. "Procedural modeling using autoencoder networks". In Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology, UIST '15, ACM,

pp. 109–118.

[34] Burnap, A., Liu, Y., Pan, Y., Lee, H., Gonzalez, R., and Papalambros, P. Y., 2016. "Estimating and exploring the product form design space using deep generative models". In ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, p. V02AT03A013.

[35] Yumer, M. E., Chaudhuri, S., Hodgins, J. K., and Kara, L. B., 2015. "Semantic shape editing using deformation handles". *ACM Trans. Graph.,* **34**(4), July, pp. 86:1–86:12.

[36] Van Der Maaten, L., Postma, E., and Van den Herik, J., 2009. "Dimensionality reduction: a comparative". pp. 66–71.

[37] Bengio, Y., Courville, A., and Vincent, P., 2013. "Representation learning: A review and new perspectives". *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* **35**(8), Aug, pp. 1798–1828.

[38] Rodrigues, F., Pereira, F., and Ribeiro, B., 2014. "Gaussian process classification and active learning with multiple annotators". In Proceedings of the 31st International Conference on Machine Learning (ICML-14), T. Jebara and E. P. Xing, eds., JMLR Workshop and Conference Proceedings, pp. 433–441.

[39] Sharmanska, V., Hernández-Lobato, D., Miguel Hernandez-Lobato, J., and Quadrianto, N., 2016. "Ambiguity helps: Classification with disagreements in crowdsourced annotations". In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2194–2202.

List of Figures

Fig. 1: The probability density function of the latent function f. The shaded areas represent the probability of a sample being labeled as the opposite label with some degree of certainty (controlled by the margin ε). When the predicted label $\hat{y} = 1$, the probability is $P(f < -\varepsilon)$; and when $\hat{y} = -1$, the probability is $P(f > \varepsilon)$.

Fig. 2: The value of p_ε under different ε. On the left plot, the gray area is the ground truth of the feasible domain; the thicker line is the decision boundary obtained by the GP classifier; the thinner lines are isocontours of V; and the circle points are training samples. When $\varepsilon = 0$, $p_\varepsilon = 0.5$ for all the points on the decision boundary, thus in this case ε-margin sampling is equivalent to uncertainty sampling. As ε increases, ε-margin sampling starts to take the variance into consideration, *i.e.*, given two points (*e.g.*, Points 1 and 3) on the decision boundary it will pick the one with a higher variance. Whereas for points having the same variance (*e.g.*, Points 2 and 3), it always prefers the one on the decision boundary.

Fig. 3: The flexible pool boundary. In each active learning iteration, we set the pool boundary by expanding the bounding box of the current labeled samples by a constant α.

Fig. 4: Queries at the exploitation stage (left) and the exploration stage (right). The gray area is the ground truth of the feasible domain. The solid line is the decision boundary obtained by the GP classifier; and the dashed line is the isocontour of p_ε. At the exploitation stage, the center $\boldsymbol{c}$ is the last queried positive sample, which makes the next query stay along the decision boundary. At the exploration stage, $\boldsymbol{c}$ is the centroid of the initial positive samples, which keeps the queries centered around the existing (real-world) samples rather than biasing towards some direction.

Fig. 5: 3D visualization of high dimensional design space showing that design variables actually lie on a 2-dimensional manifold [8, 9]. At a point away from the real-world stemless glass samples, the glass contours are self-intersecting; at another point, the shape becomes a stem glass.

Fig. 6: Domain expansion process for the Branin example. The points are queried samples before the current iteration; and the stars are current queries. The solid lines are decision boundaries obtained by the GP classifier; and the dashed lines are the pool boundaries.

Fig. 7: F1 scores for the Branin example. For ε-margin sampling, during exploration stages, the exploited decision boundaries do not change, so does the F1 score; while during rest of the time (exploitation stages), the F1 score shows a fluctuant increase as the decision boundary changes.

Fig. 8: Queried samples for random sampling and uncertainty sampling within 210 iterations. The solid lines are decision boundaries obtained by the GP classifier; and the dashed lines are the pool boundaries.

Fig. 9: Queried samples by ε-margin sampling with different GP kernel length scales. A large length scale accelerates exploration but may miss small feasible domains.

Fig. 10: F1 scores for the Hosaki example. With a larger length scale, the F1 score increases faster, but may stop increasing because it cannot discover small feasible domains.

Fig. 11: F1 scores for the two-sphere example.

Fig. 12: F1 scores for the airfoil example.

Fig. 13: The estimated feasible domains and corresponding valid airfoil designs. The top left figure shows the initial and

queried samples and the estimated feasible domains in the embedding space $\mathcal{F}$. The solid dots represent valid designs, while the hollow dots represent invalid ones. The bottom figure shows the airfoil designs in the estimated feasible domain.

Fig. 14: Some of the initial designs used in the stemless glassware example.

Fig. 15: The discovered feasible domain and valid designs. The top figure shows the initial and queried samples and the estimated feasible domain in the embedding space $\mathcal{F}$ The solid dots represent valid designs, while the hollow dots represent invalid ones. Started with the stemless glasses shown in Fig. 14, the proposed method discovered other types of revolved objects such as vases and bowls.

Fig. 16: Some of the initial designs used in the bottle example.

Fig. 17: The discovered feasible domains and valid designs. The solid dots represent valid designs, while the hollow dots represent invalid ones. Started with the bottles shown in Fig. 16, the proposed method discovered two feasible domains, between which there are designs with self-intersecting contours.