

Active expansion sampling for learning feasible domains in an unbounded input space

Wei Chen¹  · Mark Fuge¹

Received: 17 August 2017 / Revised: 22 November 2017 / Accepted: 25 December 2017 / Published online: 19 January 2018
© Springer-Verlag GmbH Germany, part of Springer Nature 2018

Abstract

Many engineering problems require identifying feasible domains under implicit constraints. One example is finding acceptable car body styling designs based on constraints like aesthetics and functionality. Current active-learning based methods learn feasible domains for bounded input spaces. However, we usually lack prior knowledge about how to set those input variable bounds. Bounds that are too small will fail to cover all feasible domains; while bounds that are too large will waste query budget. To avoid this problem, we introduce Active Expansion Sampling (AES), a method that identifies (possibly disconnected) feasible domains over an unbounded input space. AES progressively expands our knowledge of the input space, and uses successive exploitation and exploration stages to switch between learning the decision boundary and searching for new feasible domains. We show that AES has a misclassification loss guarantee within the explored region, independent of the number of iterations or labeled samples. Thus it can be used for real-time prediction of samples' feasibility within the explored region. We evaluate AES on three test examples and compare AES with two adaptive sampling methods — the *Neighborhood-Voronoi* algorithm and the *straddle* heuristic — that operate over fixed input variable bounds.

Keywords Active learning · Adaptive sampling · Feasible domain identification · Gaussian process · Exploitation-exploration trade-off

1 Introduction

In applications like design space exploration (e.g. Yannou et al. 2005; Devanathan and Ramani 2010; Larson and Mattson 2012) and reliability analysis (e.g. Lee and Jung 2008; Zhuang and Pan 2012), people need to find feasible domains within which solutions are valid. Sometimes the constraints that define those feasible domains are implicit, i.e., they cannot be represented analytically. Examples of these constraints are aesthetics, functionality, or performance requirements, which are usually evaluated by human assessment, experiments, or time-consuming computer simulations. Thus usually it is expensive to detect the feasibility of a given input. In such cases, one would like to use as few samples as possible while still approximating the feasible domain well.

To solve such problems, researchers have used *active learning* (or *adaptive sampling*)¹ to sequentially select the most informative instances and query their feasibility, so that the number of queries can be minimized (Larson and Mattson 2012; Lee and Jung 2008; Zhuang and Pan 2012; Huang and Chan 2010; Ren and Papalambros 2011). These methods require fixed bounds over the input space, and only pick queries inside those bounds. But what if we do not know how wide to set those bounds? If we set the bounds too large, an active learner will require an excessively large budget to explore the input space; whereas if we set the bounds too small, we cannot guarantee that an algorithm will recover all the feasible domains (Chen and Fuge 2017). In this case, we need an active learning method that can gradually expand our knowledge about the input space until we have either discovered all feasible domains or used up our remaining query budget.

This paper proposes a method — which we call *Active Expansion Sampling* (AES) — to solve that problem by

✉ Wei Chen
wchen459@umd.edu

¹ Department of Mechanical Engineering,
University of Maryland, College Park, MD 20742, USA

¹Note that in this paper the terms “active learning” and “adaptive sampling” are interchangeable.

casting the detection of feasible domains as an *unbounded domain estimation problem*. In an unbounded domain estimation problem, given an expensive function $h : \mathcal{X} \in \mathbb{R}^d \rightarrow \{-1, 1\}$ that evaluates any point \mathbf{x} in an *unbounded* input data space \mathcal{X} , we want to find (possibly disconnected) feasible domains in which $h(\mathbf{x}) = 1$. Specifically, h could be costly computation, time-consuming experiments, or human evaluation, so that the problem cannot be solved analytically. By *unbounded*, we mean that we don't manually bound the input space. Thus the input space can be considered as infinite, and theoretically if the query budget allows, our method can keep expanding the explored area of the input space. To use as few function evaluations as necessary to identify feasible domains, AES first fully exploits (up to an accuracy threshold) any feasible domains it knows about and then, budget permitting, searches outward to discover other feasible domains.

The main contributions of this paper are:

1. We introduce the AES method for identifying (possibly disconnected) feasible domains over an unbounded input space.
2. We provide a framework that transfers bounded active learning methods into methods that can operate over unbounded input space.
3. We introduce a dynamic local pool method that efficiently finds near optimal solutions to the global optimization problem (9) for selecting queries.
4. We prove a constant theoretical bound for AES's misclassification error at any iteration inside the explored region.

2 Background and related work

Essentially, the unbounded domain estimation problem breaks down into two tasks explored by past researchers: 1) the active learning task, where we efficiently query the feasibility of inputs; and 2) the classification task, where we estimate decision boundaries (i.e., boundaries of feasible domains) that separates the feasible class and the infeasible class (i.e., feasible regions and infeasible regions). For the first task, we will review relevant past work on active learning. For the second task, we use the Gaussian Process as the classifier in this paper and will introduce basic concepts of Gaussian Processes.

2.1 Feasible domain identification

Past work in design and optimization has proposed ways to identify feasible domains or decision boundaries of expensive functions. Generally those methods were proposed to reduce the number of simulation runs

and improve the accuracy of surrogate models in simulation-based design and reliability assessment (Lee and Jung 2008; Basudhar and Missoum 2010). Also, the problem of feasible domain identification is also equivalent to estimating the level set or the threshold boundaries of a function, where the feasible/infeasible region becomes superlevel/sublevel set (Bryan et al. 2006; Gotovos et al. 2013). Such methods select samples that are expected to best improve the surrogate model's accuracy. A common rule is to sample on the estimated decision boundary, but not close to existing sample points. Existing methods achieve this by (1) explicitly optimizing or constraining the decision function or the distance between the new sample and the existing samples (Basudhar and Missoum 2008, 2010; Singh et al. 2017), or (2) selecting points based on the estimated function values and their confidence at candidate points (Lee and Jung 2008; Bryan et al. 2006; Gotovos et al. 2013; Chen et al. 2014, 2015; Yang et al. 2015a).

2.2 Active learning

Methods for feasible domain identification usually require strategies that sequentially sample points in an input space, such that the sample size is minimized. These strategies fall under the larger category of active learning.

There are three main scenarios of active learning problems: (1) membership query synthesis, (2) stream-based selective sampling, and (3) pool-based sampling (Settles 2010). In the *membership query model*, the learner generates samples de novo for labeling. For classification tasks, researchers have typically applied membership query models to learning finite concept classes (Jackson 1997; Angluin 2004; King et al. 2004; Awasthi et al. 2013) and half-spaces (Alabdulmohsin et al. 2015; Chen et al. 2016). In the *stream-based selective sampling model*, an algorithm draws each unlabeled sample from an incoming data distribution, and then decides whether or not to query that label. This decision can be based on some informativeness measure of the drawn sample (Dagan and Engelson 1995; Freund et al. 1997; Schohn and Cohn 2000; Cavallanti et al. 2009; Cesa-Bianchi et al. 2009; Orabona and Cesa-Bianchi 2011; Dekel et al. 2012; Agarwal 2013), or whether the drawn sample is inside a *region of uncertainty* (Cohn et al. 1994; Dasgupta et al. 2009). In the *pool-based sampling model*, there is a small pool of labeled samples and a large (but finite) pool of unlabeled samples, where the learner selects new queries from the unlabeled pool.

The unbounded domain estimation problem assumes that synthesizing an unlabeled sample from the input space is not expensive (as in the membership query scenario), since otherwise we have to use existing samples and the input space will be bounded. An example that satisfies this assumption is experimental design, where we can form

an experiment by selecting a set of parameters. With this assumption, our proposed method approximates the pool-based sampling setting by synthesizing a pool of unlabeled samples in each iteration.

A pool-based sampling method first trains a classifier using the labeled samples. Then it ranks the unlabeled samples based on their *informativeness* indicated by an *acquisition function*. A query is then selected from the pool of unlabeled samples according to their rankings. After that, we add the selected query into the set of labeled data and repeat the previous process until our query budget is reached. Many of these methods use the informativeness criteria that select queries with the maximum label ambiguity (Lewis and Gale 1994; Settles and Craven 2008; Huang et al. 2010), contributing the highest estimated expected classification error (Campbell et al. 2000; Zhu et al. 2003; Nguyen and Smeulders 2004; Krempel et al. 2015), best reducing the *version space* (Tong and Koller 2001), or where different classifiers disagree the most (McCallum et al. 1998; Argamon-Engelson and Dagan 1999). Such methods are usually good at *exploitation*, since they keep querying points close to the decision boundary, refining our estimate of it.

However, when the input space may have multiple regions of interest (i.e., feasible regions), these methods may not work well if the active learner is not aware of all the regions of interest initially. Note that while some of the methods mentioned above also consider representativeness (McCallum et al. 1998; Zhu et al. 2003; Nguyen and Smeulders 2004; Settles and Craven 2008; Huang et al. 2010), or the diversity of queries (Hoi et al. 2009; Yang et al. 2015b), they don't explicitly explore unknown regions and discover other regions of interests. To address this issue, an active learner also has to allow for *exploration* (i.e., to query in unexplored regions where no labeled sample has been seen yet). A learner must trade-off exploitation and exploration.

To query in an unexplored region, there are methods that (1) take into account the predictive variance at unlabeled samples when selecting new queries (Bryan et al. 2006; Kapoor et al. 2010; Gotovos et al. 2013), (2) naturally balance exploitation/exploration by looking at the expected error (Mac Aodha et al. 2014), or (3) make exploitative and exploratory queries separately using different strategies (Baram et al. 2004; Osugi et al. 2005; Krause and Guestrin 2007; Hoang et al. 2014; Bouneffouf 2016; Hsu and Lin 2015). In previous methods, the exploitation-exploration trade-off was performed in a bounded input space or a fixed sampling pool. However, in the unbounded domain estimation problem, there is no fixed sampling pool and we are usually uncertain about how to set the bounds of the input space for performing active learning. If the bounds are too small, we might miss feasible domains;

while if the bounds are too large, the active learner has to query more samples than necessary to achieve the required accuracy.

In this paper, we introduce a method of using active learning to expand our knowledge about an unbounded input data space, and discover feasible domains in that space. A naïve solution would be to progressively expand a bounded input space, and apply the existing active learning techniques. However, there are two problems with this naïve solution: (1) it is difficult to explicitly specify when and how fast we expand the input space; and (2) the area we need to evaluate increases over time increasing the computational cost. Thus existing active learning techniques cannot apply directly to the unbounded domain estimation problem. To the best of our knowledge, Chen and Fuge (2017) is the first to deal with the active learning problem over an unbounded input space (i.e., the unbounded domain estimation problem). The AES method proposed in this paper improves upon that previous work (as illustrated in Section 3).

2.3 Gaussian process classification (GPC)

Gaussian Processes (GP, also called Kriging) are often used as a classifier in active learning (Bryan et al. 2006; Lee and Jung 2008; Kapoor et al. 2010; Gotovos et al. 2013; Chen et al. 2014, 2015). Compared to other commonly used classifiers such as Support Vector Machines or Logistic Regression, GP naturally models probabilistic predictions. This offers us a way to evaluate a sample's informativeness based on its predictive probability distribution.

The Gaussian process uses a kernel (covariance) function $k(\mathbf{x}, \mathbf{x}')$ to measure the similarity between the two points \mathbf{x} and \mathbf{x}' . It encodes the assumption that “similar inputs should have similar outputs”. Some commonly used kernels are the Gaussian kernel and the exponential kernel (Krause and Guestrin 2007; Ma et al. 2014; Mac Aodha et al. 2014; Kandasamy et al. 2017). In this paper we use the Gaussian kernel:

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2l^2}\right) \quad (1)$$

where l is the length scale.

For binary GP classification, we place a GP prior over the latent function $f(\mathbf{x})$, and then “squash” $f(\mathbf{x})$ through the logistic function to obtain a prior on $\pi(\mathbf{x}) = \sigma(f(\mathbf{x})) = P(y = 1|\mathbf{x})$. In the feasible domain identification setting, we can consider $f : \mathcal{X} \in \mathbb{R}^d \rightarrow \mathbb{R}$ as an estimation of feasibility, thus we can call it *estimated feasibility function*. Under the Laplace approximation, given the labeled data $(\mathbf{X}_L, \mathbf{y})$, the posterior of the latent function $f(\mathbf{x})$ at any $\mathbf{x} \in \mathcal{X}_U$ is a Gaussian distribution: $f(\mathbf{x})|\mathcal{X}_L$,

$\mathbf{y}, \mathbf{x} \sim \mathcal{N}(\bar{f}(\mathbf{x}), V(\mathbf{x}))$ with the mean and the variance expressed as

$$\bar{f}(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T K^{-1} \hat{\mathbf{f}} = \mathbf{k}(\mathbf{x})^T \nabla \log P(\mathbf{y}|\hat{\mathbf{f}}) \quad (2)$$

$$V(\mathbf{x}) = \mathbf{k}(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^T (K + W^{-1})^{-1} \mathbf{k}(\mathbf{x}) \quad (3)$$

where $W = -\nabla \nabla \log P(\mathbf{y}|\mathbf{f})$ is a diagonal matrix with non-negative diagonal elements; \mathbf{f} is the vector of latent function values at X_L , i.e., $f_i = f(\mathbf{x}^{(i)})$ where $\mathbf{x}^{(i)} \in X_L$; K is the covariance matrix of the training samples,

i.e., $K_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$; $\mathbf{k}(\mathbf{x})$ is the vector of covariances between \mathbf{x} and the training samples, i.e., $k_i(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}^{(i)})$; and $\hat{\mathbf{f}} = \arg \max_f P(f|X, \mathbf{y})$. When using the Gaussian kernel shown in (1), $k(\mathbf{x}, \mathbf{x}) = 1$. We refer interested readers to a detailed description by Rasmussen and Williams (2006) about the Laplace approximation for the binary GP classifier.

The decision boundary corresponds to $\bar{f}(\mathbf{x}) = 0$ or $\bar{\pi}(\mathbf{x}) = 0.5$. We predict $y = -1$ when $\bar{f}(\mathbf{x}) < 0$, and $y = 1$ otherwise.

Algorithm 1 The Active Expansion Sampling algorithm

```

1: Inputs:
   Query budget  $T$ 
   Initial point  $\mathbf{x}^{(0)}$  and its label  $y_0$ 
    $d$ -dimensional evaluation function  $h(\cdot)$ 
   Hyperparameters  $\epsilon$  and  $\tau$ 
2: Initialize:
    $X_L \leftarrow \{\mathbf{x}^{(0)}\}$ ,  $Y_L \leftarrow \{y_0\}$ ,  $INIT \leftarrow True$ 
3: for  $t = 1, 2, \dots, T$  do
4:   if  $INIT$  is  $True$  then
5:     if  $X_L$  consists of only one class (all feasible or all infeasible) then
6:        $c \leftarrow \mathbf{x}^{(0)}$ 
7:     else
8:        $INIT \leftarrow False$ 
9:        $c \leftarrow$  centroid of positive samples in  $X_L$ 
10:    end if
11:   end if
12:   Train the GP classifier using  $X_L$ 
13:   Compute  $\delta_{exploit}$  using (12)
14:    $X_U \leftarrow$  uniform samples inside the  $(d - 1)$ -sphere  $\mathcal{C}(\mathbf{x}^{(t-1)}, \delta_{exploit})$ 
15:   Compute  $\bar{f}(\mathbf{x})$ ,  $V(\mathbf{x})$ , and  $p_\epsilon(\mathbf{x})$  for  $\mathbf{x} \in X_U$  using (2), (3), and (4)
16:   if there are both  $\bar{f}(\mathbf{x}) < 0$  and  $\bar{f}(\mathbf{x}) > 0$  for  $\{\mathbf{x} \in X_U | p_\epsilon(\mathbf{x}) > \tau\}$  then           ▷ Exploitation stage
17:     Select a new query  $\mathbf{x}^{(t)}$  from  $X_U$  based on (9)
18:   else                                         ▷ Exploration stage
19:     Compute  $\delta_{explore}$  using (11)
20:     if previous iteration is in exploitation stage then
21:        $\hat{\mathbf{x}} \leftarrow \operatorname{argmax}_{\mathbf{x} \in X_L} \|\mathbf{x} - c\|$ 
22:        $X_U \leftarrow$  uniform samples inside the  $(d - 1)$ -sphere  $\mathcal{C}(\hat{\mathbf{x}}, \delta_{explore})$ 
23:     else
24:        $X_U \leftarrow$  uniform samples inside the  $(d - 1)$ -sphere  $\mathcal{C}(\mathbf{x}^{(t-1)}, \delta_{explore})$ 
25:     end if
26:     Compute  $\bar{f}(\mathbf{x})$ ,  $V(\mathbf{x})$ , and  $p_\epsilon(\mathbf{x})$  for  $\mathbf{x} \in X_U$  using (2), (3), and (4)
27:     Select a new query  $\mathbf{x}^{(t)}$  from  $X_U$  based on (9)
28:   end if
29:    $y_t \leftarrow h(\mathbf{x}^{(t)})$ 
30:    $X_L \leftarrow X_L \cup \{\mathbf{x}^{(t)}\}$ ,  $Y_L \leftarrow Y_L \cup \{y_t\}$ 
31: end for

```

3 Active expansion sampling (AES)

Algorithm 1 summarizes our proposed Active Expansion Sampling method. Overall, the method consists of the following steps:

1. Select an initial sample $\mathbf{x}^{(0)}$ to label.
2. In each subsequent iteration,
 - (a) check the exploitation/exploration status (Section 4.4),
 - (b) generate a pool of candidate samples X_U based on the exploitation/exploration status and previous queries (Sections 4.2 and 4.3),
 - (c) train a GP classifier using the labeled set X_L to evaluate the informativeness of candidate samples in X_U ,
 - (d) select a sample from X_U based on its informativeness and its distance from \mathbf{c} (Section 3.1),
 - (e) label the new sample and put it into X_L .
3. Exit when the query budget is reached.

This AES method improves upon our previous domain expansion method (Chen and Fuge 2017) in several ways. For example, the previous method generates a pool X_U that expands with the explored region each iteration. So its pool size and hence the computational cost increase significantly over time if using a constant sample density. To avoid this problem, this paper proposes a dynamic local pool method (Section 4). Another major difference is that AES provides a verifiable way to distinguish between exploitation and exploration (Section 4.4); while the previous method uses a heuristic based on the labels of last few queries (which is more likely to make mistakes). In this section and Section 6, we show comprehensive theoretical analysis and experiments to prove favorable properties of our new method.

3.1 ϵ -margin probability

We train a GP classification model to evaluate the informativeness of candidate samples based on the ϵ -margin probability (Fig. 1):

$$\begin{aligned} p_\epsilon(\mathbf{x}) &= \begin{cases} P(f(\mathbf{x}) < -\epsilon | \mathbf{x}), & \text{if } \hat{y} = 1 \\ P(f(\mathbf{x}) > \epsilon | \mathbf{x}), & \text{if } \hat{y} = -1 \end{cases} \\ &= P(\hat{y} f(\mathbf{x}) < -\epsilon | \mathbf{x}) \\ &= \Phi\left(-\frac{|\bar{f}(\mathbf{x})| + \epsilon}{\sqrt{V(\mathbf{x})}}\right) \end{aligned} \quad (4)$$

where \hat{y} is the estimated label of \mathbf{x} , the margin $\epsilon > 0$, and $\Phi(\cdot)$ is the cumulative distribution function of standard Gaussian distribution $\mathcal{N}(0, 1)$. The ϵ -margin probability represents the probability of \mathbf{x} being misclassified with

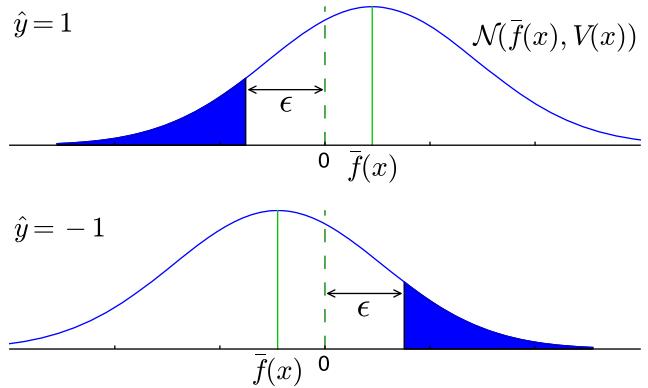


Fig. 1 The probability density function of the latent function $f(\mathbf{x})$ (Chen and Fuge 2017). The blue areas represent the ϵ -margin probability $p_\epsilon(\mathbf{x})$

some degree of certainty (controlled by the margin ϵ). Let the misclassification loss be

$$L(\mathbf{x}) = \begin{cases} \max\{0, -f(\mathbf{x})\}, & \text{if } y = 1 \\ \max\{0, f(\mathbf{x})\}, & \text{if } y = -1 \end{cases} \quad (5)$$

where y is the true label of \mathbf{x} . $L(\mathbf{x})$ measures the deviation of the estimated feasibility function value $f(\mathbf{x})$ from 0 when the class prediction is wrong. Then, based on (4) and (5), $p_\epsilon(\mathbf{x}) = P(L(\mathbf{x}) > \epsilon)$, which is the probability that the expected misclassification loss exceeds ϵ . A high $p_\epsilon(\mathbf{x})$ indicates that \mathbf{x} is very likely to be misclassified, and requires further evaluation. Thus we use this probability to measure informativeness.

3.2 Exploitation and exploration

Since our input space is unbounded, naïvely maximizing the ϵ -margin probability (informativeness) will always query points infinitely far away from previous queries.² To avoid this issue, one solution is to query informative samples that are close to previously labeled samples. This allows the active learner to progressively expand its knowledge as the queries cover an increasingly large area of the input space. When a new decision boundary is discovered during expansion, we want a query strategy that continues querying points on that decision boundary, such that the new feasible region can be identified as quickly as possible. Therefore, to enable continuous exploitation of the decision boundary, we propose the following query strategy

$$\begin{aligned} \min_{\mathbf{x} \in X_U} & V(\mathbf{x}) \\ \text{s.t.} & p_\epsilon(\mathbf{x}) \geq \tau \end{aligned} \quad (6)$$

²A point infinitely far away from previous queries has the $\bar{f}(\mathbf{x})$ close to 0 and the maximum $V(\mathbf{x})$, thus the highest $p_\epsilon(\mathbf{x})$.

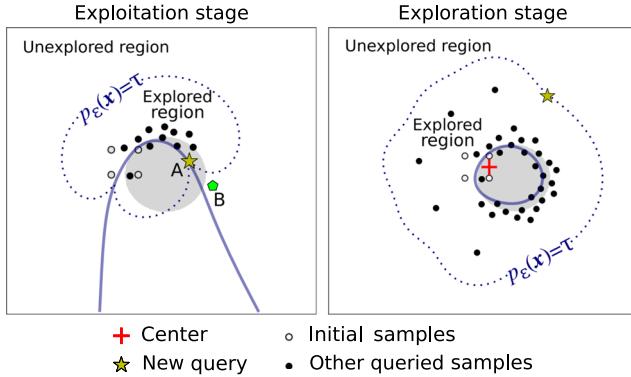


Fig. 2 Queries at the exploitation stage (left) and the exploration stage (right). The gray area is the ground truth of the feasible domain. The solid line is the decision boundary estimated by the GP classifier; and the dotted line is the isocontour of $p_\epsilon(\mathbf{x})$. At the exploitation stage (left), the center \mathbf{c} is the previous query, which makes the next query stay along the decision boundary. At the exploration stage (right), \mathbf{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

where $V(\mathbf{x})$ is the predictive variance at \mathbf{x} , and τ is a threshold of the informativeness measure $p_\epsilon(\mathbf{x})$.

Theorem 1 *The solution to (6) will lie at the intersection of the estimated decision boundary ($\bar{f}(\mathbf{x}) = 0$) and the isocontour of $p_\epsilon(\mathbf{x}) = \tau$ (Point A in Fig. 2), if that intersection A exists.*

Proof In the following proof, we denote $\bar{f}_P = \bar{f}(\mathbf{x}_P)$, and $V_P = V(\mathbf{x}_P)$. For a sample \mathbf{x}_A at the intersection of $\bar{f}(\mathbf{x}) = 0$ and $p_\epsilon(\mathbf{x}) = \tau$, we have $\bar{f}_A = 0$ and $p_\epsilon(\mathbf{x}_A) = \Phi(-\epsilon/\sqrt{V_A}) = \tau$ (Point A in Fig. 2); and for a sample \mathbf{x}_B that is any feasible solution to (6), we have $p_\epsilon(\mathbf{x}_B) = \Phi(-(|\bar{f}_B| + \epsilon)/\sqrt{V_B}) \geq \tau$ (Point B in Fig. 2). Thus we get $\epsilon/\sqrt{V_B} \leq (|\bar{f}_B| + \epsilon)/\sqrt{V_B} \leq \epsilon/\sqrt{V_A}$. Therefore, $V_A \leq V_B$. The equality holds when $|\bar{f}_B| = 0$ and $p_\epsilon(\mathbf{x}_B) = \tau$, i.e., \mathbf{x}_B is also at the intersection of $\bar{f}(\mathbf{x}) = 0$ and $p_\epsilon(\mathbf{x}) = \tau$. Thus we proved the intersection has the minimal predictive variance among feasible solutions to (6), and hence it is the optimal solution. \square

Theorem 1 indicates that when applying the query strategy shown in (6), the active learner will only query points at the estimated decision boundary³ as long as the estimated decision boundary and the isocontour of $p_\epsilon(\mathbf{x}) = \tau$ intersect. The fact that this intersection exists indicates that there are points on the decision boundary that are informative to some extent (i.e., with $p_\epsilon(\mathbf{x}) \geq \tau$). We

call this stage the *exploitation stage*—at this stage the active learner exploits the decision boundary. Equation (6) ensures that the queries are always on the estimated decision boundary until the exploitation stage ends (i.e., there are no longer informative points on the decision boundary).

If the estimated decision boundary and the isocontour of $p_\epsilon(\mathbf{x}) = \tau$ do not intersect, then the algorithm has fully exploited any informative points on the estimated decision boundary (i.e., for all the points on the estimated decision boundary, we have $p_\epsilon(\mathbf{x}) < \tau$). We call this stage the *exploration stage* since the active learner starts to search for another decision boundary (Fig. 2). In this stage, we want the new query to be both informative and close to where we started, since we don't want the new query to deviate too far from where we start. Therefore, the query strategy at the exploration stage is

$$\begin{aligned} \min_{\mathbf{x} \in X_U} \quad & \|\mathbf{x} - \mathbf{c}\| \\ \text{s.t.} \quad & p_\epsilon(\mathbf{x}) \geq \tau \end{aligned} \quad (7)$$

where the objective function is the Euclidean distance between \mathbf{x} and a center \mathbf{c} . This objective keeps the new query selected by (7) close to \mathbf{c} . In practice, initially when there are only samples from one class, we set \mathbf{c} as the initial point $\mathbf{x}^{(0)}$ to keep new queries close to where we start; once there are both positive and negative samples, we set \mathbf{c} as the centroid of these initial positive samples, since we want to keep new queries close to the initial feasible region.

Theorem 2 *Given \mathbf{x}^* as the solution to (7), we have $p_\epsilon(\mathbf{x}^*) = \tau$, if $p_\epsilon(\mathbf{c}) < \tau$.*

Proof Since $p_\epsilon(\mathbf{c}) < \tau$, \mathbf{c} itself is not the solution of (9). Thus $\|\mathbf{x}^* - \mathbf{c}\| > 0$. Then we have $p_\epsilon(\mathbf{x}) < \tau$ at any point within a $(d-1)$ -sphere centered at \mathbf{c} with radius $\|\mathbf{x}^* - \mathbf{c}\|$, because otherwise the query will be inside the sphere. Thus on that sphere we have $p_\epsilon(\mathbf{x}) \leq \tau$. So $p_\epsilon(\mathbf{x}^*) \leq \tau$, since \mathbf{x}^* is on that sphere. Because \mathbf{x}^* is a feasible solution to (9), we also have $p_\epsilon(\mathbf{x}^*) \geq \tau$ at \mathbf{x}^* . Therefore $p_\epsilon(\mathbf{x}^*) = \tau$. \square

Theorem 2 shows that in each iteration, the optimal query \mathbf{x}^* selected by (7) is on the isocontour of $p_\epsilon(\mathbf{x}) = \tau$.

For both (6) and (7), the feasible solutions are in the region of $p_\epsilon(\mathbf{x}) \geq \tau$. Intuitively this means that we only query samples with at least some level of informativeness. We call the region where $p_\epsilon(\mathbf{x}) \geq \tau$ the *unexplored region*, since it contains informative samples (feasible solutions) that our query strategy cares about; while we call the rest of the input space ($p_\epsilon(\mathbf{x}) \leq \tau$) the *explored region* (Fig. 2).

The upper bound of $p_\epsilon(\mathbf{x})$ is $\Phi(-\epsilon/\sup_{\mathbf{x}} V(\mathbf{x}))$, and it lies infinitely far away from the labeled samples. In (3), $K + W^{-1}$ is positive semidefinite, thus $\mathbf{k}(\mathbf{x})^T(K + W^{-1})^{-1}\mathbf{k}(\mathbf{x}) \geq 0$ and $V(\mathbf{x}) \leq \mathbf{k}(\mathbf{x}, \mathbf{x})$. For a kernel $k(\cdot)$ with $k(\mathbf{x}, \mathbf{x}) = 1$ (e.g., the Gaussian or the exponential

³In Section 3, we assume that the queried point is the exact solution to the query strategy. However since we approximate the exact solution by using a pool-based sampling setting, the query may be deviate from the exact solution slightly.

kernel), we have $V(\mathbf{x}) \leq 1$. Thus $p_\epsilon(\mathbf{x}) \leq \Phi(-\epsilon)$. To ensure that (9) has a feasible solution, we have to set $\tau \leq \Phi(-\epsilon)$. Therefore, we can set $\tau = \Phi(-\eta\epsilon)$, where $\eta \geq 1.0$. Then the constraint in (6) and (7) can be expressed as

$$\Phi\left(-\frac{|\bar{f}(\mathbf{x})| + \epsilon}{\sqrt{V(\mathbf{x})}}\right) \geq \Phi(-\eta\epsilon)$$

which can be written as

$$\eta\epsilon\sqrt{V(\mathbf{x})} - |\bar{f}(\mathbf{x})| \geq \epsilon \quad (8)$$

The left-hand side of (8) is identical to the acquisition function of the *straddle heuristic* when $\eta\epsilon = 1.96$ (Bryan et al. 2006). The straddle heuristic queries the sample with the largest value of the acquisition function. This acquisition function accounts for the *ambiguity* of samples in terms of their confidence intervals (Gotovos et al. 2013):

$$a(\mathbf{x}) = \min\{-\min Q(\mathbf{x}), \max Q(\mathbf{x})\} \\ = 1.96\sqrt{V(\mathbf{x})} - |\bar{f}(\mathbf{x})|$$

where $Q(\mathbf{x})$ is the 95% confidence interval of \mathbf{x} .

Substituting (8) for the constraint in (6) and (7), and combining the exploitation and exploration stages, our overall query strategy becomes

$$\begin{aligned} \min_{\mathbf{x} \in X_U} & V(\mathbf{x})^\alpha \|\mathbf{x} - \mathbf{c}\|^{1-\alpha} \\ \text{s.t.} & \eta\epsilon\sqrt{V(\mathbf{x})} - |\bar{f}(\mathbf{x})| \geq \epsilon \end{aligned} \quad (9)$$

where the indicator α is 1 at the exploitation stage, and 0 otherwise. Section 4.4 introduces how to set α (i.e., when to exploit vs explore).

In general, the unbounded domain estimation problem can be solved using a family of query strategies with the following form

$$\begin{aligned} \min_{\mathbf{x} \in X_U} & D(\mathbf{x}) \\ \text{s.t.} & I(\mathbf{x}) \geq \tau \end{aligned}$$

where $D(\mathbf{x})$ is a function that increases as \mathbf{x} moves away from the labeled samples, and $I(\mathbf{x})$ is the informativeness measure that is used in any bounded active learning methods. Our query strategies of (6) and (7) all have this form. Comparatively, for bounded active learning methods, the query strategies are usually in the form of $\max_{\mathbf{x} \in X_U} I(\mathbf{x})$.

4 Dynamic local pool generation

We cast our problem as pool-based sampling by generating a pool of unlabeled instances de novo in each iteration. A naïve way to generate this pool is to try to sample points anywhere near the $p_\epsilon(\mathbf{x}) = \tau$ isocontour. However, intuitively, as the algorithm searches progressively larger volumes of the input space, the pool volume will likewise expand. This expansion means that the size of the pool will increase dramatically over time (assuming we want a

constant sample density). This increase, however, makes the computation of (2) and (3) expensive during later expansion stages.

To bypass this problem, we propose a *dynamic local pool* method that generates the pool of candidate samples only at a certain location in each iteration, rather than sampling the entire domain.⁴ The key insight behind our local pooling method is that while the optimal solution to (9) can, in principle, occur anywhere on the $p_\epsilon(\mathbf{x}) = \tau$ isocontour, in practice, multiple points on the isocontour are equally optimal. All we need to do is sample points around any one of those optima. Below, we derive guarantees for how to sample volumes near one of those optima, thus only needing to sample a small fraction of the total domain volume.

4.1 Scope of an optimal query

Theorem 3 Let δ be the distance between an optimal query⁵ and its nearest labeled sample. We have

$$\delta < \beta l \quad (10)$$

where β is a coefficient depends on ϵ , η , and the GP model.

We include the proof of Theorem 3 and the way of computing β in the appendix (Section A1). Theorem 3 indicates that if we set the pool boundary by extending the current labeled sample range by βl , then that pool is guaranteed to contain all solutions to (9); that is, extending the overall pool boundary further will not increase the chances of sampling near $p_\epsilon(\mathbf{x}) = \tau$, and will only decrease the sample density (given a fixed pool size) or increase the evaluated samples (given a fixed sample density). However, if we generate the pool based solely on this principle (i.e., extending the current labeled sample range by βl), the pool size will still increase over time as the domain size grows. The next two sections show how, for the exploration and exploitation stages respectively, we can further reduce the sample boundary to only a local hyper-sphere.

4.2 Pool for the exploration stage

Theorem 4 During the exploration stage of Active Expansion Sampling, the distance between an optimal query and its nearest labeled sample is

$$\delta < \delta_{explore} = \beta l \quad (11)$$

⁴Sampling methods like random sampling or Poisson-disc sampling (Bridson 2007) can be used to generate the pool. We use random sampling here thereby for simplicity. The specific choice of the sampling method within the local pool is not central to the overall method.

⁵The optimal query means the exact solution to the AES query strategy shown in (6), (7), or (9).

Theorem 4 is derived from (10). The nearest labeled sample of the optimal query could be any border point (a sample lying on the periphery of the labeled set). There are multiple local optima that are equally useful for expanding the explored region (Fig. 3). Thus we just sample near one of those optima. Specifically, we approximate the nearest labeled sample as the previous query. With this approximation, incorporating Theorem 4, the optimal query will be inside $\mathcal{C}(\mathbf{x}^{(t-1)}, \delta_{explore})$, the $(d-1)$ -sphere with a radius of $\delta_{explore}$, centered at the previous query $\mathbf{x}^{(t-1)}$. Thus during the exploration stage, we set the pool boundary to be that sphere (Fig. 3).

Sometimes when AES switches from exploitation to exploration, the previous query may not lie on the periphery of the labeled samples. This causes samples around the previous query to have low values of $p_\epsilon(\mathbf{x})$. In this case, there might not be feasible solution to (9). Thus, every time AES switches from exploitation to exploration, we center the pool around the farthest labeled sample from the centroid of the initial positive samples (i.e., $\text{argmax}_{\mathbf{x} \in X_L} \|\mathbf{x} - \mathbf{c}\|$). This ensures that AES generates pool samples near the periphery of the labeled samples.

4.3 Pool for the exploitation stage

Theorem 5 During the exploitation stage of Active Expansion Sampling, the distance between an optimal query and its nearest labeled sample is

$$\delta < \delta_{exploit} = \gamma l \quad (12)$$

where γ is a coefficient depends on ϵ , η , and the GP model.

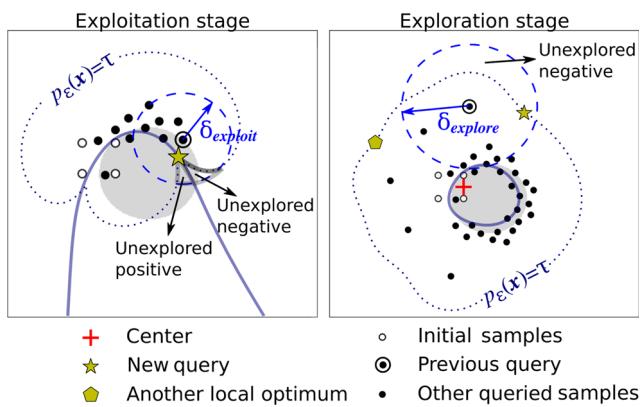


Fig. 3 Dynamic local pools (dashed circles) at the exploitation stage (left) and the exploration stage (right). During the exploitation stage, the estimated decision boundary divided the unexplored region into two subregions: unexplored negative $\mathcal{R}_1 = \{\mathbf{x} | p_\epsilon(\mathbf{x}) > \tau, \bar{f}(\mathbf{x}) < 0\}$ and unexplored positive $\mathcal{R}_2 = \{\mathbf{x} | p_\epsilon(\mathbf{x}) > \tau, \bar{f}(\mathbf{x}) > 0\}$; while during the exploration stage, there will be at most one of \mathcal{R}_1 and \mathcal{R}_2 in the unexplored region. This property can be used to distinguish between the exploitation/exploration stages

We include the proof of Theorem 5 and the way of computing γ in the appendix (Section A2). Similar to the exploration stage, based on Theorem 5, we define the pool boundary during the exploitation stage as $\mathcal{C}(\mathbf{x}^{(t-1)}, \delta_{exploit})$, a $(d-1)$ -sphere with a radius of $\delta_{exploit}$, centered at the previous query $\mathbf{x}^{(t-1)}$ (Fig. 3).

4.4 Choosing when to exploit versus explore

Since we use different rules to generate the pool at the exploitation and exploration stage, we need to distinguish between the two stages at the beginning of each iteration. In the exploitation stage, according to Theorem 5, the optimal query lies within the $(d-1)$ -sphere $\mathcal{C}(\mathbf{x}^{(t-1)}, \delta_{exploit})$ centered at the previous query. While, according to Theorem 1, that same query must lie where the estimated decision boundary and the isocontour of $p_\epsilon(\mathbf{x}) = \tau$ intersect. Thus, the decision boundary and the isocontour divide the sphere \mathcal{C} into four regions (Fig. 3):

$$\text{unexplored negative } \mathcal{R}_1 = \{\mathbf{x} | p_\epsilon(\mathbf{x}) > \tau, \bar{f}(\mathbf{x}) < 0\};$$

$$\text{unexplored positive } \mathcal{R}_2 = \{\mathbf{x} | p_\epsilon(\mathbf{x}) > \tau, \bar{f}(\mathbf{x}) > 0\};$$

$$\text{explored negative } \mathcal{R}_3 = \{\mathbf{x} | p_\epsilon(\mathbf{x}) < \tau, \bar{f}(\mathbf{x}) < 0\}; \text{ and}$$

$$\text{explored positive } \mathcal{R}_4 = \{\mathbf{x} | p_\epsilon(\mathbf{x}) < \tau, \bar{f}(\mathbf{x}) > 0\}.$$

In contrast, during exploration the estimated decision boundary and the $p_\epsilon(\mathbf{x}) = \tau$ isocontour do not intersect—meaning, unlike exploitation, there exist only two of the four regions (either \mathcal{R}_1 & \mathcal{R}_3 or \mathcal{R}_2 & \mathcal{R}_4). In particular, within the unexplored region, $\bar{f}(\mathbf{x})$ will be either all positive or all negative, i.e., \mathcal{R}_1 and \mathcal{R}_2 cannot exist simultaneously (Fig. 3).

We use this property to detect exploitation or exploration by generating a pool (a set of uniformly distributed samples) within the boundary $\mathcal{C}(\mathbf{x}^{(t-1)}, \delta_{exploit})$ and checking if, for samples with $p_\epsilon(\mathbf{x}) > \tau$, samples differ in $\bar{f}(\mathbf{x}) > 0$ and $\bar{f}(\mathbf{x}) < 0$. If so, AES is in the exploitation stage; otherwise it is in the exploration stage.

5 Theoretical analysis

In this section, we derive a theoretical accuracy bound for AES with respect to its hyperparameters. We further discuss the influence of those hyperparameters on the classification accuracy, the query density, and the exploration speed. The results of this section guides the selection of proper hyperparameters given an accuracy or budget requirement.

5.1 Accuracy analysis

It is impossible to discuss the function accuracy across the entire input space, since the input space is unbounded.

However, we can consider ways to bound the accuracy within bounded explored regions at any time step.

As mentioned in Section 3.1, $p_\epsilon(\mathbf{x}) = P(L(\mathbf{x}) > \epsilon)$, where $L(\mathbf{x})$ is the misclassification loss at \mathbf{x} defined in (5). Thus within the explored region, we have

$$P(L(\mathbf{x}) \geq \epsilon) \leq \tau \quad \forall \mathbf{x} \in \{\mathbf{x} | p_\epsilon(\mathbf{x}) \leq \tau\}$$

or

$$P(L(\mathbf{x}) \leq \epsilon) \geq 1 - \tau \quad \forall \mathbf{x} \in \{\mathbf{x} | p_\epsilon(\mathbf{x}) \leq \tau\} \quad (13)$$

This shows that at any location within the explored region of the input space, the proposed method guarantees an upper bound ϵ of misclassification loss with a probability of at least $1 - \tau$ at any given point. Since, in the exploration stage, the estimated decision boundary lies inside the $p_\epsilon(\mathbf{x}) \leq \tau$ region (as discussed in Section 3.2), we have

$$P(L(\mathbf{x}) \leq \epsilon) \geq 1 - \tau \quad \forall \mathbf{x} \in \{\mathbf{x} | \bar{f}(\mathbf{x}) = 0\}$$

This means that in the exploration stage, the estimated decision boundary $\bar{f}(\mathbf{x}) = 0$ lies in between the isocontours of $f(\mathbf{x}) = \pm\epsilon$ with a probability of at least $1 - \tau$, where f is the true latent function.

Note that (13) shows that AES's accuracy bound within the explored region is independent of the number of iterations or labeled samples. One advantage of keeping a constant accuracy bound for AES is that the accuracy in the explored region meets our requirements⁶ whenever AES stops. This also means that the estimation within the explored region is reliable at any iteration (although this is not true if one includes the unexplored region). In contrast, bounded active learning methods usually only achieve required accuracy after a certain number of iterations, before which the estimation may not be reliable. Therefore, AES can be used for real-time prediction of samples' feasibility in the explored region.

5.2 Query density

In Gaussian Processes, given a fixed homoscedastic Gaussian or exponential kernel, we can measure the query density by looking at the predictive variance at queried points. According to (3), $V(\mathbf{x})$ only depends on $\mathbf{k}(\mathbf{x})$, which is affected by the distances between \mathbf{x} and other queries. A smaller variance at a query indicates that it is closer to other queries, and hence a higher query density; and vice versa.

Theorem 6 *The predictive variance of an optimal query in the exploitation and exploration stage is*

$$V(\mathbf{x}_{\text{exploit}}) = \frac{1}{\eta^2} \quad (14)$$

⁶We can set ϵ and τ such that the accuracy bound is as required. Details about how to set hyperparameters are in Section 5.3.

and

$$V(\mathbf{x}_{\text{explore}}) = \frac{1}{\eta^2} \left(1 + \frac{|\bar{f}(\mathbf{x}_{\text{explore}})|}{\epsilon} \right)^2 \quad (15)$$

where $\mathbf{x}_{\text{exploit}}$ and $\mathbf{x}_{\text{explore}}$ are optimal queries at the exploitation stage and exploration stage, respectively.

The proof of Theorem 6 is in the appendix (Section A3). This theorem indicates that the predictive variances of queries at the exploitation stage are always smaller than those at the exploration stage (as $|\bar{f}(\mathbf{x}_{\text{explore}})| > 0$). Thus the query density at the exploitation stage is always higher than that at the exploration stage. The property of having a denser set of points along the decision boundary (queried during the exploitation stage) and a sparser set of points at other regions (queried during the exploration stage) is desirable because we want to save our query budget for refining the decision boundary rather than other regions of the input space.

Equation (13) and (14) also reflect the trade-off between the accuracy and the running time. When the query density near the decision boundary is high (small $V(\mathbf{x}_{\text{exploit}})$ in (14)), η is large, thus τ in (13) is small, which means our model will have a higher probability of having a misclassification loss less than ϵ . However, as the query density gets higher, we need more queries to cover a certain region, thus the running time increases.

5.3 Influence of hyperparameters

There are four hyperparameters that control Active Expansion Sampling — the initial point $\mathbf{x}^{(0)}$, ϵ and η in the exploitation/exploration stage, and the length scale l of the GP kernel. The choice of the kernel function and length scale depends on assumptions regarding the nature and smoothness of the underlying feasibility function. Such kernel choices have been covered extensively in prior research and we refer interested readers to Rasmussen and Williams (2006) for multiple methods of choosing l . Note that it is difficult to optimize the length scale at each iteration, since the length scale will eventually be pushed to extremes. In the exploitation stage, for example, once the length scale is smaller than the previous iteration, the distance between the new query and its nearest query will also be smaller (due to Eq. 12). Then the maximum marginal likelihood estimation will result in a smaller length scale, as the estimated function is steeper. This process will repeat and eventually cause the optimal length scale to converge to 0. The initial point $\mathbf{x}^{(0)}$ can be any point not too far away from the boundary of feasible regions, since otherwise it will take a large budget to just search for a sample from the opposite class. Here we

focus on the analysis of the other two hyperparameters — ϵ and η .

According to (13), ϵ and τ affect the classification accuracy in a probabilistic way. When $\tau = \Phi(-\eta\epsilon)$, we have $P(L(\mathbf{x}) \leq \epsilon) \geq 1 - \Phi(-\eta\epsilon)$ in the explored region. This offers us a guideline for setting ϵ and η with respect to a given accuracy requirement.

According to (14) and (15), η controls the density of queries in both exploitation and exploration stages. Specifically, as we increase η , $V_{exploit}$ and $V_{explore}$ decreases, increasing the query density and essentially placing labeled points closer together.

In contrast, ϵ only controls the distances between queries in the exploration stage.⁷ Increasing ϵ decreases $V_{explore}$ and hence increases the density of queries in the exploration stage. This density of queries affects (1) how fast we can expand the explored region, and (2) how likely we are to capture small feasible regions. When η or ϵ increases, we expand the explored region slower, making it more likely that we will discover smaller feasible regions. Likewise, we also slow down the expansion in exploitation stages, making the classifier more likely to capture a sudden change along domain boundaries.

Note that when $\epsilon = 0$, the constraint of $p_\epsilon(\mathbf{x}) \geq \tau$ in (9) is equivalent to $\bar{f}(\mathbf{x}) = 0$, thus theoretically all queries should lie near the estimated decision boundary. In this case, the Active Expansion Sampling acts like *Uncertainty Sampling* (Lewis and Catlett 1994; Lewis and Gale 1994). In practice, however, AES will be unable to find a feasible solution when $\epsilon = 0$ since no candidate sample will be exactly on the decision boundary under the pool-based sampling setting.

6 Experimental evaluation

We evaluate the performance of AES in capturing feasible domains using both synthesized and real-world examples. The performance is measured by the F1 score, which 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

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

⁷Technically, due to sampling error introduced when generating the pool, the exploitation stage will be influenced by ϵ (since $\bar{f}(\mathbf{x}^*)$ is only ≈ 0). But this effect is negligible compared to ϵ 's influence on the exploration stage.

We compare AES with two conventional bounded adaptive sampling methods — the Neighborhood-Voronoi (NV) algorithm (Singh et al. 2017) and the straddle heuristic (Bryan et al. 2006). We also investigate the effects of noise and dimensionality on AES.

We use the same pool size (500 candidate samples⁸) in all the experiments. In Fig. 7–10 and 13, the F1 scores are averaged over 100 runs. We run all 2-dimensional experiments on a Dell Precision Tower 5810 with 16 GB RAM, a 3.5 GHz Intel Xeon CPU E5-1620 v3 processor, and a Ubuntu 16.04 operating system. We run all higher-dimensional experiments on a Dell Precision Tower 7810 with 32 GB RAM, a 2.4 GHz Intel Xeon CPU E5-2620 v3 processor, and a Red Hat Enterprise Linux Workstation 7.2 operating system. The Python code needed to reproduce our AES algorithm, our baseline implementations of NV and Straddle, and all of our below experiments is available at <https://github.com/IDEALLab/Active-Expansion-Sampling>.

6.1 Effect of hyperparameters

We first use two 2-dimensional test functions — the Branin function and Hosaki function, respectively — as indicator functions to evaluate whether an input is inside the feasible domain. Both examples construct an input space with multiple disconnected feasible regions, which makes the feasible domain identification task challenging.

The Branin function is

$$g(\mathbf{x}) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos x_1 + 10$$

We define the label $y = 1$ if $\mathbf{x} \in \{\mathbf{x} | g(\mathbf{x}) \leq 8, -9 < x_1 < 14, -7 < x_2 < 17\}$; and $y = -1$ otherwise. The resulting feasible domains resemble three isolated feasible regions (Fig. 4). The initial point $\mathbf{x}^{(0)} = (3, 3)$. For the Gaussian process, we use a Gaussian kernel (1). We set the kernel length scale $l = 0.9$. To compute the F1 scores, we generate samples along a 100×100 grid as the test set in the region where $x_1 \in [-13, 18]$ and $x_2 \in [-8, 23]$.

This section mainly describes the Branin example — as both the Branin and Hosaki examples show similar results — however we direct interested readers to the appendix (Section B1) where we describe the Hosaki example in detail and show its experimental results.

For both examples, we use three levels of ϵ (0.1, 0.3, 0.5) and η (1.2, 1.3, 1.4) to demonstrate their effects on AES's performance.

Figures 4 and 5 show the sequence of queries selected by AES and the two bounded adaptive sampling methods,

⁸For NV algorithm, its pool size refers to the test samples generated for the Monte Carlo simulation.

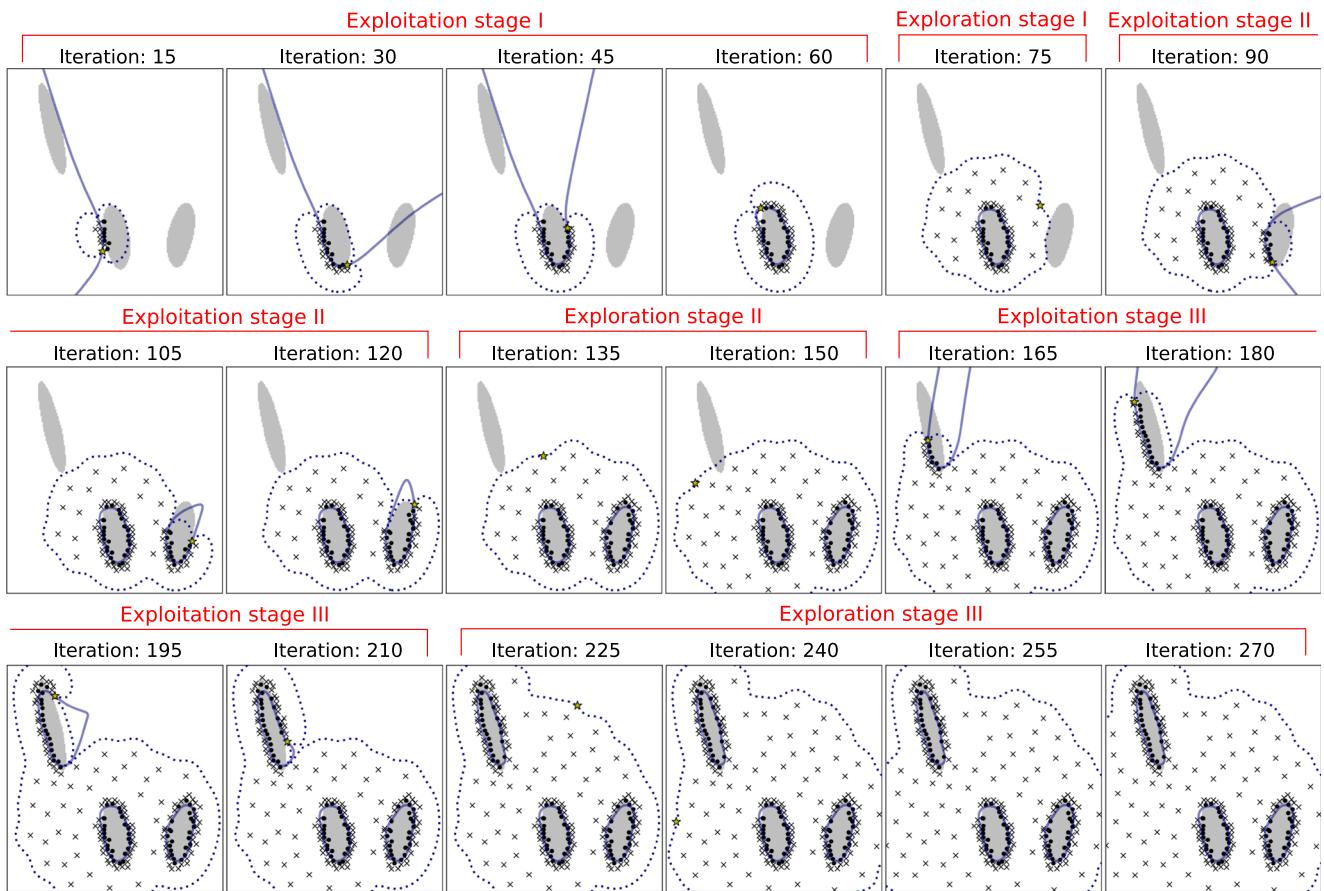


Fig. 4 Querying sequence for Active Expansion Sampling ($\epsilon = 0.5$ and $\eta = 1.3$). The solid lines are estimated decision boundaries, and the dotted lines are the isocontour $p_\epsilon(\mathbf{x}) = \tau$. The gray areas are actual feasible regions

respectively, applied on the Branin example. For AES, there are three exploitation stages, as there are three disconnected feasible domains. AES starts by querying samples along the initial estimated decision boundary, and then expands queries outward to discover other feasible regions. In contrast, the straddle heuristic simultaneously explores the whole bounded input space, and refines all three decision boundaries. Fig. 6 shows the corresponding F1 scores of the experiment in Fig. 4. During exploitation stages, AES's F1 score non-monotonically increases as part of the estimated decision boundary is outside the explored region

(where AES has confidence on the accuracy); while in the exploration stage, the current decision boundaries are inside the explored region and remain unchanged, thus the F1 score stabilizes.

Figure 7a and b demonstrate the effects of hyperparameters ϵ and η , respectively, on AES's performance. Increasing ϵ or η leads to a slower expansion of the explored region and a higher F1 score. This means that using a higher ϵ or η enables accuracy improvement but requires larger query budget. In both examples, the F1 score is more sensitive to η than ϵ .

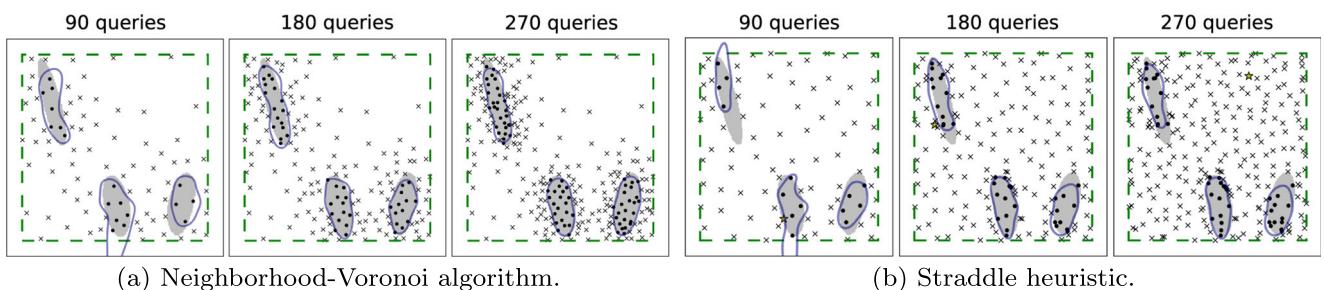


Fig. 5 Querying sequence for bounded adaptive sampling methods. The dashed lines are pool boundaries

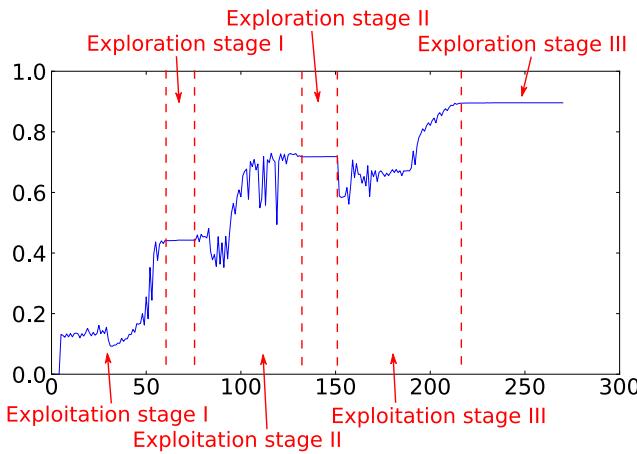


Fig. 6 F1 score plot for Fig. 4. During exploitation stages, the F1 score increases stochastically as the decision boundary changes; while in the exploration stage, the current decision boundaries have been exploited and do not change, thus the F1 score also does not change

6.2 Unbounded versus bounded

We use the NV algorithm and the straddle heuristic as examples of bounded adaptive sampling methods. Because these two methods do not progressively expand the region (as in AES), but rather assumes a fixed region, we create a “bounding box” in the input space, and generate queries inside this box.

When comparing AES with the bounded methods, we use $\epsilon = 0.3$ and $\eta = 1.3$ for AES. In each experiment, we change the size of the input space bounds to evaluate

the effect of bound size on these methods. Specifically, we simulate the cases where we set *tight*, *loose*, and *insufficient* bounds, as shown in Table 1 and Fig. 8. “Tight” means the bounds cover the entire feasible domain while being no larger than needed (in practice we use bounds slightly larger than this to ensure the feasible domain boundary is inside the tight bounds); “loose” means the bounds cover the entire feasible domain but are larger than the tight bounds; “insufficient” means the variable bounds do not cover the entire feasible domain.

As shown in Fig. 8, the NV algorithm makes fast accuracy improvement at early stages, and slows down after some iterations. The F1 score of NV is almost monotonically increasing; while AES’s score fluctuates because it focuses first on refining the domains it knows about during exploitation (at the expense of accuracy on domains it has not seen yet). This causes AES to have a lower F1 score early on. For the NV algorithm, when the input variable bounds are set properly, both AES and NV achieve similar final F1 scores. However, NV requires more iterations to achieve a similar final accuracy to AES, especially when the bounds are set too large, where NV exhausts its query budget exploring unknown regions. When the bounds are set too small to cover certain feasible regions, NV stops improving the F1 score when it begins to over-sample the space and is unable to reach similar accuracy as AES. Note that in this case, we purposefully set the bounds such that they cover the vast majority of the feasible region, leaving only a small feasible area outside of those bounds. Our explicit purpose here is to demonstrate how

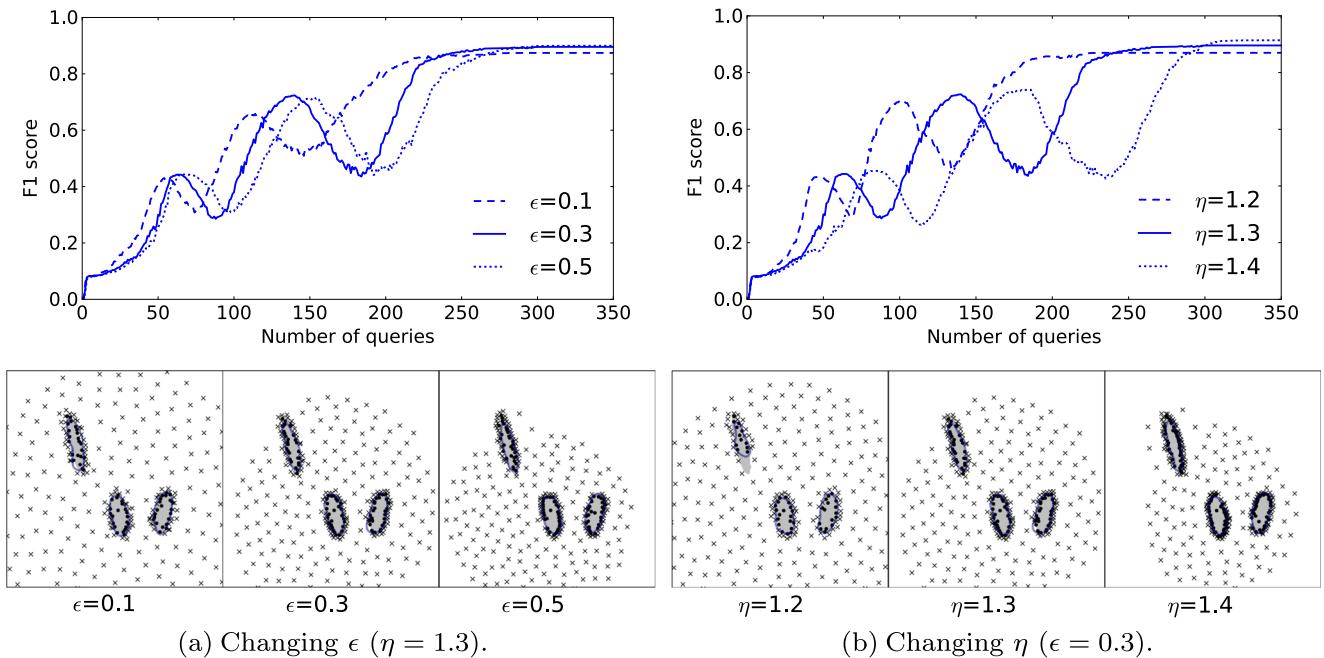


Fig. 7 AES with different ϵ and η on the Branin example. The upper plots show their F1 scores averaged over 100 runs. The lower plots show queried points during one of the 100 runs

Table 1 Input space bounds for the NV algorithm and the straddle heuristic (Branin example)

	Tight	Loose	Insufficient
Branin	$x_1 \in [-9, 14], x_2 \in [-7, 17]$	$x_1 \in [-14, 19], x_2 \in [-12, 22]$	$x_1 \in [-4, 9], x_2 \in [-2, 12]$

sensitive such bounded heuristics can be when their bounds are misspecified (even by small amounts). The performance of bounded methods degrades rapidly as their bound sizes decrease further.

Although AES shows slow accuracy improvement over the entire test region, it keeps a constant accuracy bound within the explored region (as discussed in Section 5.1). Fig. 9 shows the F1 scores within the $p_\epsilon(\mathbf{x}) < \tau$ region, which is AES's explored region. Specifically, we set $\epsilon = 0.3$, $\eta = 1.3$, and $\tau = \Phi(-\eta\epsilon)$. For the NV algorithm, we use the tight input space bounds from the previous experiments. By just considering the explored region, AES's F1 scores are quite stable throughout the sampling sequence; while NV's F1 scores are low at the beginning, and then increase until stable.⁹ Since AES's accuracy inside the explored region is invariant of the number of iterations, it can be used for real-time prediction of samples' feasibility in the explored region.

Table 2 shows the final F1 scores and wall-clock running time of AES, NV, and the straddle heuristic. Note that the confidence interval for NV's averaged F1 scores are much larger than AES. This is because during some runs NV fails to discover all the three feasible regions (Fig. 8 for example).

6.3 Effect of noise

Label noise is usually inevitable in active learning tasks. The noise comes from, for example, simulation/experimental error or human annotators' mistakes. We test the cases where the labels are under (1) uniform noise and (2) Gaussian noise centered at the decision boundary.

We simulate the first case by randomly flipping the labels. The noisy label is set as $y' = (-1)^\lambda y$, where $\lambda \sim \text{Bernoulli}(p)$, p is the parameter of the Bernoulli distribution that indicates the noise level, and y is the true label.

The second case is probably more common in practice, since it is usually harder to decide the labels near the decision boundary. To simulate this case, we add Gaussian noise to the test functions: $g'(\mathbf{x}) = g(\mathbf{x}) + e$, where $g(\mathbf{x})$ is the Branin or Hosaki function, and $e \sim s \cdot \mathcal{N}(0, 1)$.

In each case we compare the performance of AES ($\epsilon = 0.3$, $\eta = 1.3$) and NV (with tight bounds) under two noise levels. As expected, adding noise to the labels decreases

the accuracy of both methods (Fig. 10a and b). However, in both cases (Bernoulli noise and Gaussian noise), the noise appears to influence NV more than AES. As shown in Fig. 11, when adding noise to the labels, NV has high error mostly along the input space boundaries, where it cannot query samples outside to further investigate those apparent feasible regions. In contrast, AES tries to exploit those rogue points to try to find new feasible regions, realizing after a few new samples that they are noise.

6.4 Effect of dimensionality

To test the effects of dimensionality on AES's performance, we apply both AES and NV on higher-dimensional examples where the feasible domains are inside two $(d-1)$ -spheres of radius 1 centered at \mathbf{a} and \mathbf{b} respectively. Here $\mathbf{a} = \mathbf{0}$ and $\mathbf{b} = (3, 0, \dots, 0)$. Fig. 12 shows the input space of the 3-dimensional double-sphere example. The initial point $\mathbf{x}^{(0)} = \mathbf{0}$. For the Gaussian process, we use a Gaussian kernel with a length scale of 0.5. We set $\epsilon = 0.3$ and $\eta = 1.3$. To compute the F1 scores, we randomly generate 10,000 samples uniformly within the region where $x_1 \in [-2, 5]$ and $x_k \in [-2, 2], k = 2, \dots, d$. The input space bounds for the NV algorithm are $x_1 \in [-1.5, 4.5]$ and $x_k \in [-1.5, 1.5], k = 2, \dots, d$. We get the F1 scores and running time after querying 1,000 points.

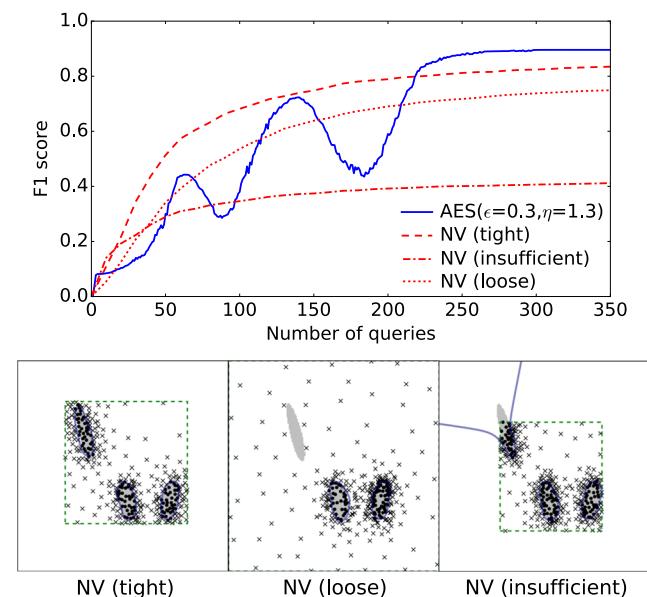


Fig. 8 AES and NV (with different input variable bounds) on the Branin example. The pool boundaries set in the Neighborhood-Voronoi algorithm are shown as dashed lines

⁹This difference is because NV's explored region covers more area than AES at the beginning.

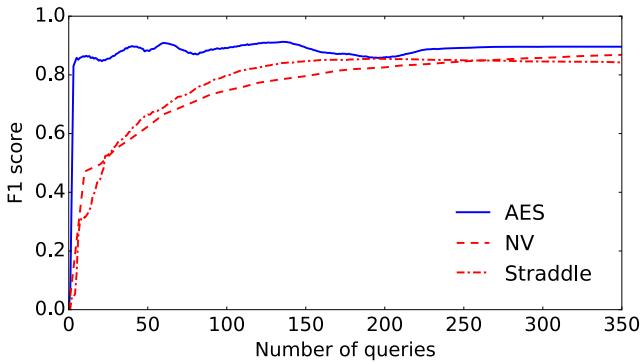


Fig. 9 F1 scores of AES ($\epsilon = 0.3$ and $\eta = 1.3$), NV, and Straddle (with tight bounds) on the Branin example within the explored region (i.e., the $p_\epsilon(\mathbf{x}) < \tau$ region, where $\tau = \Phi(-\eta\epsilon)$, $\epsilon = 0.3$, and $\eta = 1.3$)

Table 2 Final F1 scores and running time (Branin example)

		F1 score	Time (s)
Branin (350 queries)	AES ($\epsilon = 0.3, \eta = 1.3$)	0.90 ± 0.004	92.34 ± 0.62
	AES ($\epsilon = 0.1, \eta = 1.3$)	0.87 ± 0.008	95.71 ± 0.37
	AES ($\epsilon = 0.5, \eta = 1.3$)	0.90 ± 0.002	89.71 ± 0.38
	AES ($\epsilon = 0.3, \eta = 1.2$)	0.87 ± 0.006	96.73 ± 0.26
	AES ($\epsilon = 0.3, \eta = 1.4$)	0.91 ± 0.002	80.70 ± 0.33
	NV (tight)	0.83 ± 0.021	64.40 ± 0.09
	NV (loose)	0.75 ± 0.030	63.68 ± 0.06
	NV (insufficient)	0.41 ± 0.028	63.83 ± 0.06
	Straddle (tight)	0.82 ± 0.012	43.72 ± 0.22
	Straddle (loose)	0.71 ± 0.014	41.72 ± 0.22
	Straddle (insufficient)	0.34 ± 0.009	54.44 ± 0.21

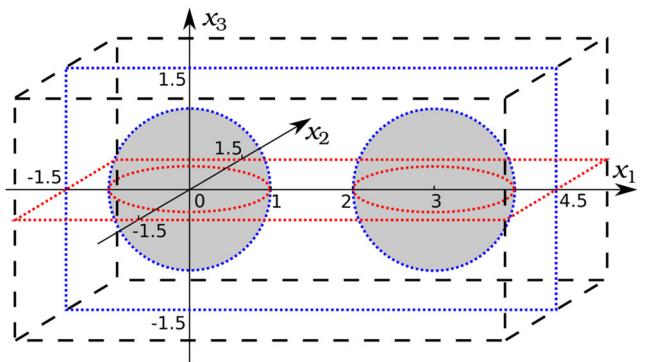
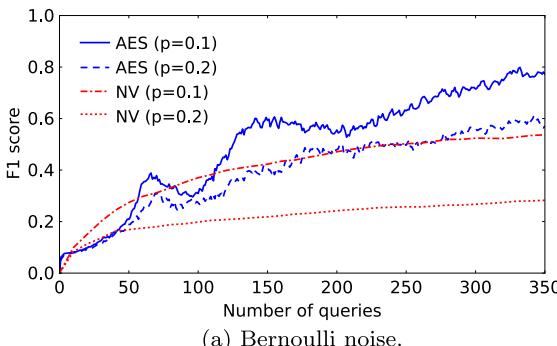
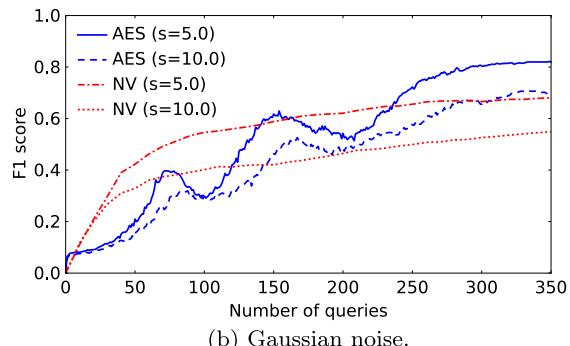


Fig. 12 The 3-dimensional double-sphere example. The gray regions are the feasible domains. The dashed boxes are the input space bounds for the NV algorithm



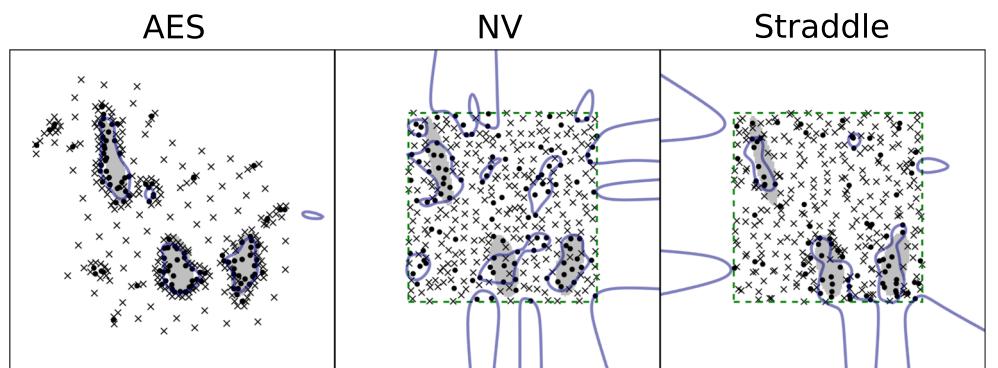
(a) Bernoulli noise.



(b) Gaussian noise.

Fig. 10 AES and NV on the Branin example using noisy labels

Fig. 11 Queried points under uniform label noise ($p = 0.2$)



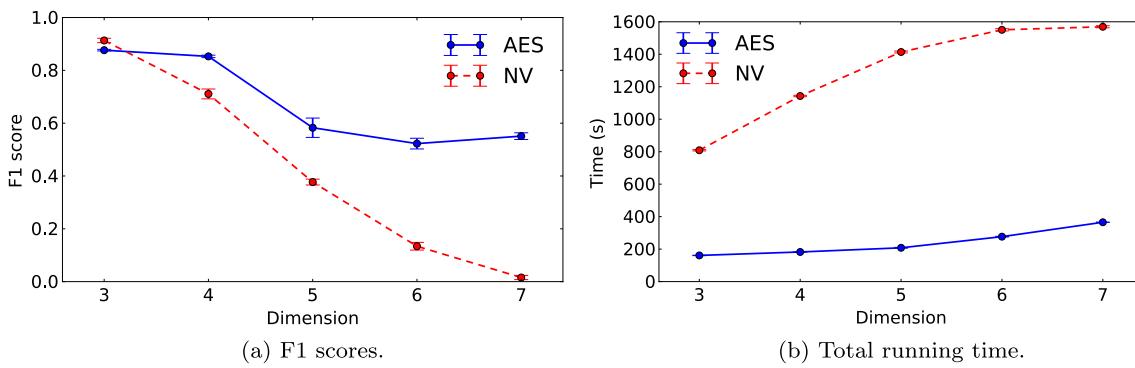


Fig. 13 AES and NV on high-dimensional double-sphere examples

As shown in Fig. 13, both AES and NV shows an accuracy drop and running time increase as the problem's dimensionality increases. This is expected, since based on the curse of dimensionality (Bellman 1957), the number of queries needed to achieve the same accuracy increases with the input space dimensionality. The curse of dimensionality is inevitable in machine learning problems. However, since AES explores the input space only when necessary (i.e., only after it has seen the entire decision boundary of the discovered feasible domain), its queries do not need to fill up the large volume of the high-dimensional space. Therefore, AES's accuracy drop with problem dimensionality is not as severe as bounded methods like NV. For particularly high-dimensional design problems, another complementary approach is to construct explicit lower-dimensional design manifolds upon which to run AES (Chen et al. 2017; Chen and Fuge 2017).

6.5 Nowacki beam example

To test AES's performance in a real-world scenario, we consider the Nowacki beam problem (Nowacki 1980). The original Nowacki beam problem is a design optimization problem where we minimize the cross-section area A of a cantilever beam of length l with concentrated load F at its end. The design variables are the beam's breadth b and height h . We turn this problem into a feasible domain identification problem by replacing the objective with a constraint $A = bh \leq 0.0025\text{m}^2$. Other constraints are (1) the maximum tip deflection $\delta = Fl^3/(3EI_Y) \leq 5\text{mm}$, (2) the maximum bending stress $\sigma_B = 6Fl/(bh^2) \leq \sigma_Y$, (3) the maximum shear stress $\tau = 1.5F/(bh) \leq \sigma_Y/2$, (4) the ratio $h/b \leq 10$, and (5) the failure force of buckling $F_{crit} = (4/l^2)\sqrt{(GI_T)(EI_Z)/(1-\nu^2)} \geq fF$, where $I_Y = bh^3/12$, $I_Z = b^3h/12$, $I_T = I_Y + I_Z$, and f is the safety factor. And σ_Y , E , ν , and G are the yield stress, Young's modulus, Poisson's ratio, and shear modulus of the beam's material, respectively. We use the settings from (Singh et al. 2017), where $l = 0.5\text{m}$, $F = 5\text{kN}$,

$f = 2$, $\sigma_Y = 240\text{MPa}$, $E = 216.62\text{GPa}$, $\nu = 0.27$, and $G = 86.65\text{GPa}$. As shown in Fig. 14, the feasible domain is a crescent-shaped region. Given only these constraints, it is unclear what appropriately tight bounds on the design variables should be.

In this experiment, we set the Gaussian kernel's length scale as 0.005, $\epsilon = 0.3$ and $\eta = 1.3$. The initial point $x^{(0)} = (b_0, h_0) = (0.05, 0.05)$. The test samples are generated along a 100×100 grid in the region where $b \in [0, 0.02]$ and $h \in [0.1, 0.16]$.

After 242 iterations, the F1 score of AES reaches 0.933 and remains constant. Note that mostly the estimation error comes from the two sharp ends of the crescent-shaped feasible region (Fig. 14). This is because the kernel's assumption on function smoothness (i.e., similar inputs should have similar outputs) causes the GP to have bad performance where the labels shift frequently. The similar

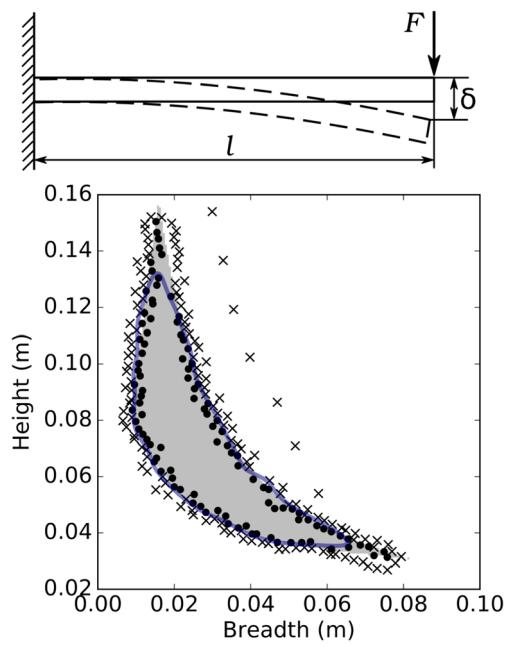


Fig. 14 AES on the Nowacki beam example

problem also exists when using other classifiers like SVM, where a kernel is also used to enforce similar outputs between similar inputs. This problem can be alleviated by using a smaller kernel length scale.

7 Conclusion

We presented a pool-based sampling method, AES, for identifying (possibly disconnected) feasible domains over an unbounded input space. Unlike conventional methods that sample inside a fixed boundary, AES progressively expands our knowledge of the input space under an accuracy guarantee. We showed that AES uses successive exploitation and exploration stages to switch between learning the decision boundary and searching for new feasible domains. To avoid increasing the pool size and hence the computation cost as the explored area grows, we proposed a dynamic local pool generation method that samples the pool locally at a certain location in each iteration. We showed that at any point within the explored region, AES guarantees an upper bound ϵ of misclassification loss with a probability of at least $1 - \tau$, regardless of the number of iterations or labeled samples. This means that AES can be used for real-time prediction of samples' feasibility inside the explored region. We also demonstrated that, compared to existing methods, AES can achieve comparable or higher accuracy without needing to set exact bounds on the input space.

Note that AES cannot be applied on input spaces where synthesizing a useful sample is difficult. For example, in an image classification task, we cannot directly synthesize an image by arbitrarily setting its pixels, since most of the synthesized images may be unrealistic and hence useless. Usually in such cases, we use real-world samples as the pool and apply bounded active learning methods (since we know the bounds of real-world samples). Or instead, we first embed the original inputs onto a lower-dimensional space, such that given the low-dimensional representation, we can synthesize realistic samples. We can then apply AES on that embedded space. This approach can be used for discovering novel feasible domains (i.e., finding feasible inputs that are nonexistent in the real-world). We refer interested readers to a detailed introduction of this approach by Chen and Fuge (2017).

One limitation of AES is that the accuracy improves slowly at the early stage compared to bounded active learning methods. This is because AES focuses on only the explored region (which is small at the beginning), while bounded active learning methods usually do space-filling at first. In the situation where we want fast accuracy improvement at the beginning, one possible way of tackling this problem is by dynamically setting AES's

hyperparameters. Specifically, since the expansion speed increases as ϵ or η decreases, we can accelerate AES's accuracy improvement at earlier stages by setting small values of ϵ and η , so that queries quickly fill up a larger region. Then to achieve high final accuracy, we can increase ϵ and η to meet the accuracy requirement.

Acknowledgments The authors thank the anonymous reviewers whose efforts improved the manuscript. This work was funded through a University of Maryland Minta Martin Grant.

Appendix A: Theorem proofs

A1 Proof of theorem 3

According to (2), given an optimal query \mathbf{x}^* , we have

$$\begin{aligned} |\bar{f}(\mathbf{x}^*)| &= |\mathbf{k}(\mathbf{x}^*)^T \nabla \log p(\mathbf{y}|\hat{\mathbf{f}})| \\ &= \left| \mathbf{k}(\mathbf{x}^*)^T \nabla \log \begin{bmatrix} \Phi(y_1 f_1) \\ \vdots \\ \Phi(y_{t-1} f_{t-1}) \end{bmatrix} \right| \\ &= \left| \mathbf{k}(\mathbf{x}^*)^T \begin{bmatrix} y_1 \mathcal{N}(f_1)/\Phi(y_1 f_1) \\ \vdots \\ y_{t-1} \mathcal{N}(f_{t-1})/\Phi(y_{t-1} f_{t-1}) \end{bmatrix} \right| \\ &= \left| \sum_{i=1}^{t-1} k(\mathbf{x}^*, \mathbf{x}^{(i)}) y_i \frac{\mathcal{N}(f_i)}{\Phi(y_i f_i)} \right| \\ &\leq \sum_{i=1}^{t-1} \left| k(\mathbf{x}^*, \mathbf{x}^{(i)}) y_i \frac{\mathcal{N}(f_i)}{\Phi(y_i f_i)} \right| \\ &< \sum_{i=1}^{t-1} k_m \text{sign}(y_i) y_i \frac{\mathcal{N}(f_i)}{\Phi(y_i f_i)} \\ &= k_m \text{sign}(\mathbf{y})^T \begin{bmatrix} y_1 \mathcal{N}(f_1)/\Phi(y_1 f_1) \\ \vdots \\ y_{t-1} \mathcal{N}(f_{t-1})/\Phi(y_{t-1} f_{t-1}) \end{bmatrix} \\ &= k_m \mu \end{aligned}$$

where

$$\begin{aligned} k_m &= \max_{\mathbf{x}^{(i)} \in X_L} k(\mathbf{x}^*, \mathbf{x}^{(i)}) \\ &= \exp \left(-\frac{\min_{\mathbf{x}^{(i)} \in X_L} \|\mathbf{x}^* - \mathbf{x}^{(i)}\|^2}{2l^2} \right) \\ &= e^{-\delta^2/(2l^2)} \end{aligned} \tag{A1}$$

and

$$\mu = \text{sign}(\mathbf{y})^T \nabla \log p(\mathbf{y}|\hat{\mathbf{f}}) \tag{A2}$$

Similarly,

$$\begin{aligned} V(\mathbf{x}^*) &= 1 - \mathbf{k}(\mathbf{x}^*)^T (K + W^{-1})^{-1} \mathbf{k}(\mathbf{x}^*) \\ &> 1 - (k_m \mathbf{1})^T (K + W^{-1})^{-1} (k_m \mathbf{1}) \\ &= 1 - k_m^2 \mathbf{1}^T (K + W^{-1})^{-1} \mathbf{1} \\ &= 1 - k_m^2 v \end{aligned} \tag{A3}$$

where

$$v = \mathbf{1}^T (K + W^{-1})^{-1} \mathbf{1} \tag{A4}$$

Therefore for the optimal query \mathbf{x}^* we have

$$p_\epsilon(\mathbf{x}^*) = \Phi\left(-\frac{|\bar{f}(\mathbf{x}^*)| + \epsilon}{\sqrt{V(\mathbf{x}^*)}}\right) > \Phi\left(-\frac{k_m \mu + \epsilon}{\sqrt{1 - k_m^2 v}}\right)$$

Both Theorem 1 and 2 state that $p_\epsilon(\mathbf{x}^*) = \tau$, thus

$$\Phi\left(-\frac{k_m \mu + \epsilon}{\sqrt{1 - k_m^2 v}}\right) < \tau$$

When $\tau = \Phi(-\eta\epsilon)$, we have

$$\frac{k_m \mu + \epsilon}{\sqrt{1 - k_m^2 v}} > \eta\epsilon \quad (\text{A5})$$

Plugging (A1) into (A5) and solving for the distance δ , we get

$$\delta < \beta l$$

where

$$\beta = \sqrt{2 \log \frac{\mu^2 + \eta^2 \epsilon^2 v}{\eta \epsilon \sqrt{\mu^2 + (\eta^2 - 1) \epsilon^2 v} - \epsilon \mu}} \quad (\text{A6})$$

A2 Proof of theorem 5

Theorem 1 states that the optimal query in the exploitation stage lies at the intersection of $\bar{f}(\mathbf{x}) = 0$ and $p_\epsilon(\mathbf{x}) = \tau$. By substituting $\Phi(-\eta\epsilon)$ for τ , we have

$$V(\mathbf{x}^*) = \frac{1}{\eta^2} \quad (\text{A7})$$

According to (A3), we have $V(\mathbf{x}^*) > 1 - k_m^2 v$. Combining (A1), (A4), and (A7), we get

$$\delta < \delta_{\text{exploit}} = \gamma l$$

where

$$\gamma = \sqrt{\log \frac{\eta^2 v}{\eta^2 - 1}} \quad (\text{A8})$$

A3 Proof of theorem 6

According to (A7), the predictive variance of an optimal query $\mathbf{x}_{\text{exploit}}$ in the exploitation stage is

$$V(\mathbf{x}_{\text{exploit}}) = \frac{1}{\eta^2}$$

While in the exploration stage, we have $p_\epsilon(\mathbf{x}_{\text{explore}}) = \tau$ at the optimal query $\mathbf{x}_{\text{explore}}$ (Theorem 2). And by applying (4) and setting $\tau = \Phi(-\eta\epsilon)$, we have

$$V(\mathbf{x}_{\text{explore}}) = \frac{1}{\eta^2} \left(1 + \frac{|\bar{f}(\mathbf{x}_{\text{explore}})|}{\epsilon}\right)^2$$

Appendix B: Additional experimental results

B1 Hosaki example

We use the Hosaki example as an additional 2-dimensional example to demonstrate the performance of our proposed method. Different from the Branin example, the Hosaki example has feasible domains of different scales. Its feasible domains resemble two isolated feasible regions — a large “island” and a small one (Fig. 15a). The Hosaki function is

$$g(\mathbf{x}) = \left(1 - 8x_1 + 7x_1^2 - \frac{7}{3}x_1^3 + \frac{1}{4}x_1^4\right)x_2^2 e^{-x_2}$$

We define the label $y = 1$ if $\mathbf{x} \in \{\mathbf{x} | g(\mathbf{x}) \leq -1, 0 < x_1, x_2 < 5\}$; and $y = -1$ otherwise.

For AES, we set the initial point $\mathbf{x}^{(0)} = (3, 3)$. We use a Gaussian kernel with a length scale $l = 0.4$. The test set to compute F1 scores is generated along a 100×100 grid in the region where $x_1 \in [-3, 9]$ and $x_2 \in [-3.5, 8.5]$. For NV and straddle, the input space bounds are shown in Table 3.

Table 4 shows the final F1 scores and running time of AES, NV, and the straddle heuristic. Fig. 15 shows the F1 scores and queries under different ϵ and η . Fig. 16 compares the performance of AES and NV with different boundary sizes. Fig. 17 shows the performance of AES and NV under Bernoulli and Gaussian noise.

B2 Results of straddle heuristic

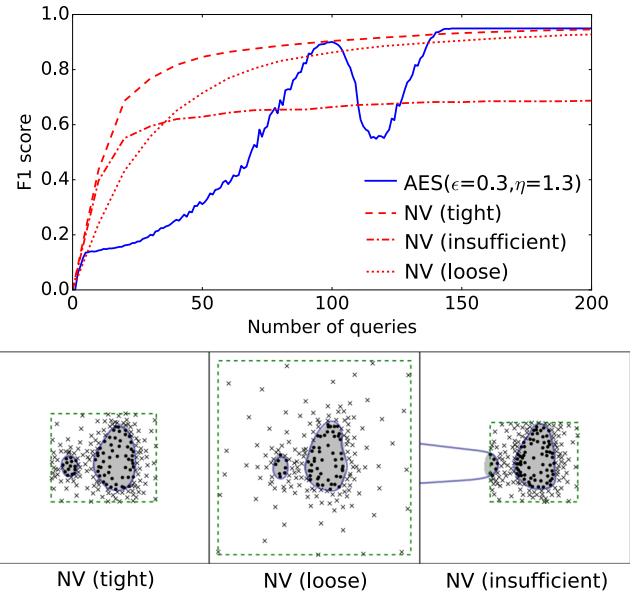
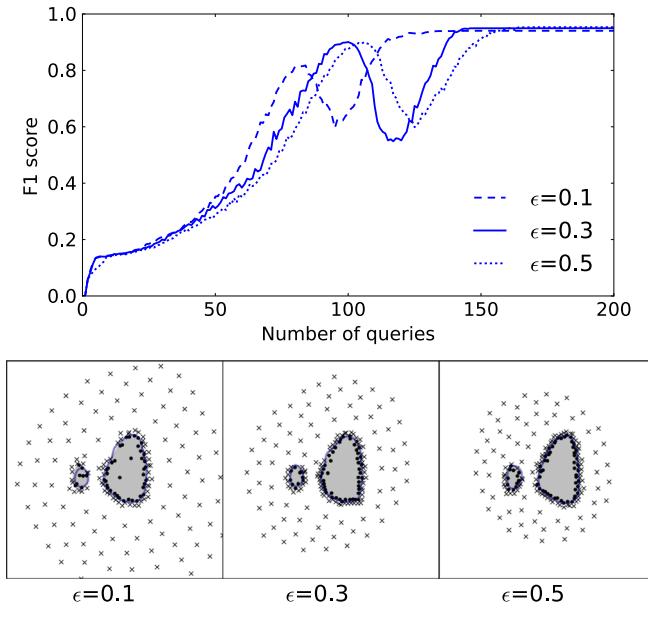
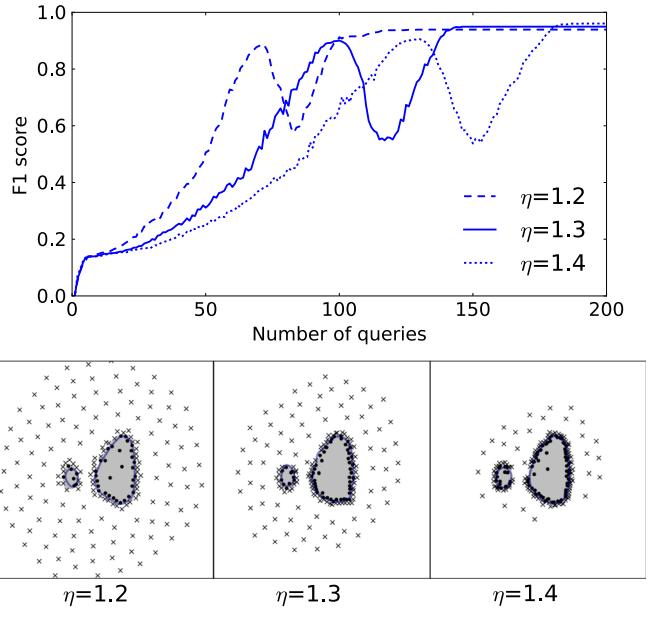
In this section we list experimental results related to the straddle heuristic. Specifically, Fig. 18 shows straddle’s F1 scores and queries using different sizes of input variable bounds, and the comparison with AES. Fig. 19 shows the comparison of AES and straddle under noisy labels.

Table 3 Input space bounds for the NV algorithm and the straddle heuristic (Hosaki example)

	Tight	Loose	Insufficient
Hosaki	$x_1 \in [0, 6], x_2 \in [0, 5]$	$x_1 \in [-2.5, 8.5], x_2 \in [-3, 8]$	$x_1 \in [1, 6], x_2 \in [0, 4.5]$

Table 4 Final F1 scores and running time (Hosaki example)

		F1 score	Time (s)
Hosaki (200 queries)	AES ($\epsilon = 0.3, \eta = 1.3$)	0.95 ± 0.003	28.25 ± 0.25
	AES ($\epsilon = 0.1, \eta = 1.3$)	0.94 ± 0.004	30.86 ± 0.19
	AES ($\epsilon = 0.5, \eta = 1.3$)	0.95 ± 0.002	28.32 ± 0.33
	AES ($\epsilon = 0.3, \eta = 1.2$)	0.94 ± 0.003	31.69 ± 0.45
	AES ($\epsilon = 0.3, \eta = 1.4$)	0.96 ± 0.002	26.39 ± 0.38
	NV (tight)	0.95 ± 0.003	22.58 ± 0.03
	NV (loose)	0.93 ± 0.004	22.28 ± 0.03
	NV (insufficient)	0.69 ± 0.010	22.27 ± 0.03
	Straddle (tight)	0.95 ± 0.002	16.20 ± 0.19
	Straddle (loose)	0.88 ± 0.005	14.00 ± 0.14
	Straddle (insufficient)	0.69 ± 0.010	16.92 ± 0.25

**Fig. 16** AES and NV (with different input variable bounds) on the Hosaki example(a) Changing ϵ ($\eta = 1.3$).(b) Changing η ($\epsilon = 0.3$).**Fig. 15** AES with different ϵ and η on the Hosaki example

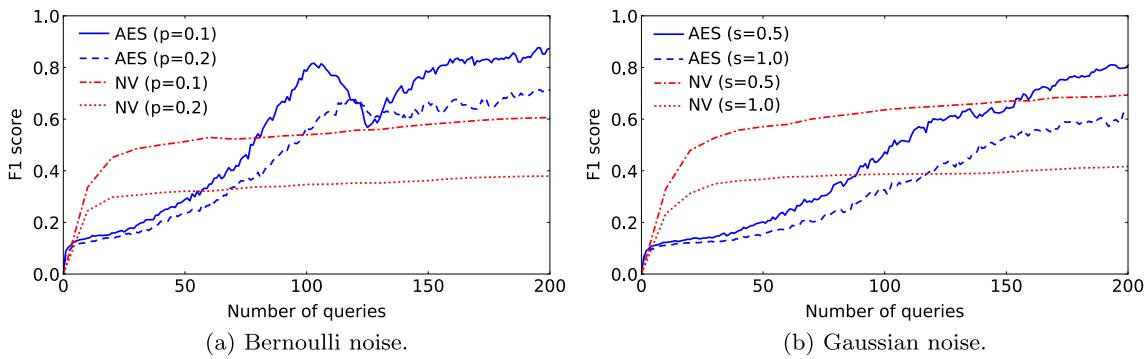


Fig. 17 AES and NV on the Hosaki example using noisy labels

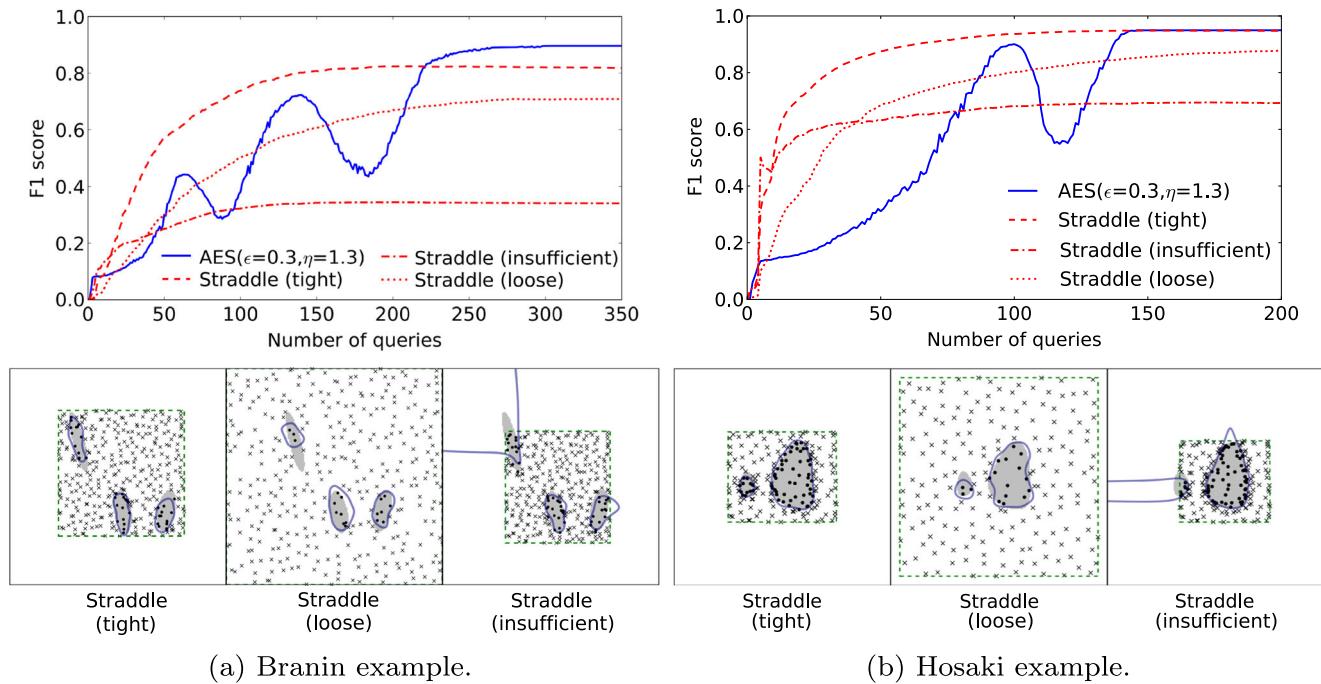


Fig. 18 AES and straddle (with different input variable bounds)

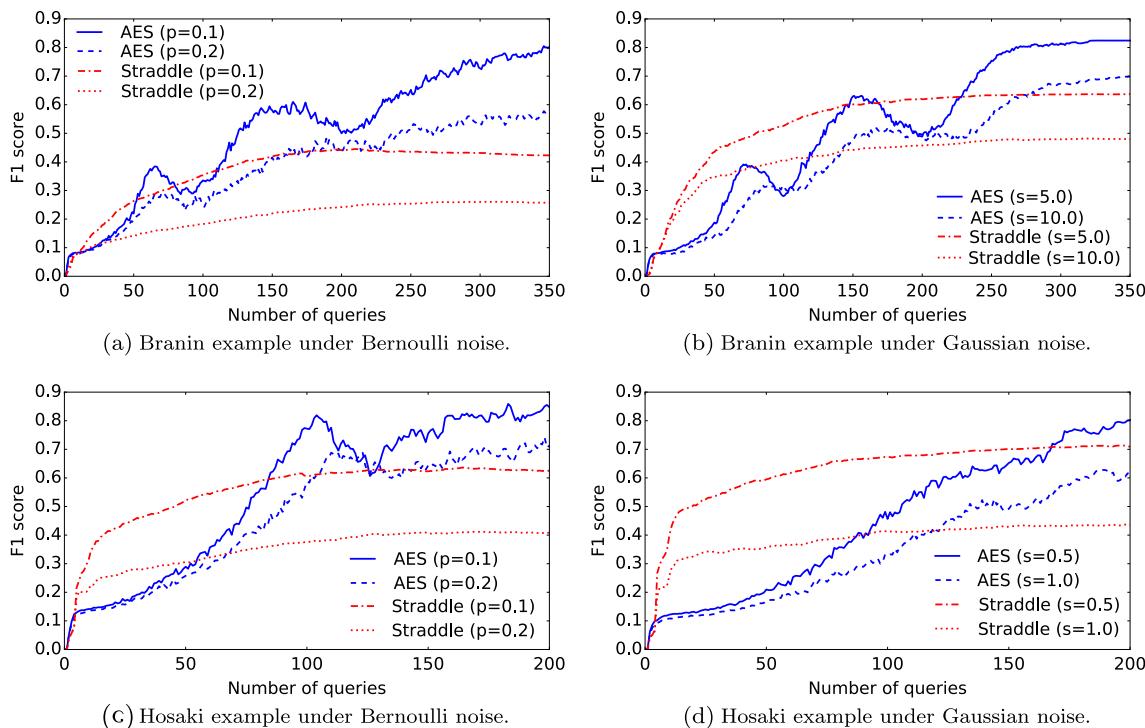


Fig. 19 AES and straddle under noisy labels

References

- Agarwal A (2013) Selective sampling algorithms for cost-sensitive multiclass prediction. *ICML* (3) 28:1220–1228
- Alabdulmohsin I, Gao X, Zhang X (2015) Efficient active learning of halfspaces via query synthesis. In: Proceedings of the Twenty-Ninth AAAI conference on artificial intelligence. AAAI Press, pp 2483–2489
- Angluin D (2004) Queries revisited. *Theor Comput Sci* 313(2):175–194
- Argamon-Engelson S, Dagan I (1999) Committee-based sample selection for probabilistic classifiers. *J Artif Intell Res(JAIR)* 11:335–360
- Awasthi P, Feldman V, Kanade V (2013) Learning using local membership queries Shalev-Shwartz S, Steinwart I (eds), vol 30, *Proceedings of Machine Learning Research*, Princeton
- Baram Y, Yaniv RE, Luz K (2004) Online choice of active learning algorithms. *J Mach Learn Res* 5:255–291
- Basudhar A, Missoum S (2008) Adaptive explicit decision functions for probabilistic design and optimization using support vector machines. *Comput Struct* 86(19):1904–1917
- Basudhar A, Missoum S (2010) An improved adaptive sampling scheme for the construction of explicit boundaries. *Struct Multidiscip Optim* 42(4):517–529
- Bellman R (1957) *Dynamic programming*. Princeton University Press, Princeton
- Bouneffouf D (2016) Exponentiated gradient exploration for active learning. *Computers* 5(1):1
- Bridson R (2007) Fast poisson disk sampling in arbitrary dimensions. In: ACM SIGGRAPH 2007 sketches SIGGRAPH '07. ACM, New York, <https://doi.org/10.1145/1278780.1278807>
- Bryan B, Nichol RC, Genovese CR, Schneider J, Miller CJ, Wasserman L (2006) Active learning for identifying function threshold boundaries. In: Advances in neural information processing systems, pp 163–170
- Campbell C, Cristianini N, Smola AJ (2000) Query learning with large margin classifiers. In: Proceedings of the seventeenth international conference on machine learning. Morgan Kaufmann Publishers Inc., pp 111–118
- Cavallanti G, Cesa-Bianchi N, Gentile C (2009) Linear classification and selective sampling under low noise conditions. In: Advances in neural information processing systems, pp 249–256
- Cesa-Bianchi N, Gentile C, Orabona F (2009) Robust bounds for classification via selective sampling. In: Proceedings of the 26th annual international conference on machine learning. ACM, pp 121–128
- Chen W, Fuge M (2017) Beyond the known: detecting novel feasible domains over an unbounded design space. *J Mech Des* 139(11):111,405
- Chen Z, Qiu H, Gao L, Li X, Li P (2014) A local adaptive sampling method for reliability-based design optimization using kriging model. *Struct Multidiscip Optim* 49(3):401–416
- Chen Z, Peng S, Li X, Qiu H, Xiong H, Gao L, Li P (2015) An important boundary sampling method for reliability-based design optimization using kriging model. *Struct Multidiscip Optim* 52(1):55–70
- Chen L, Hassani H, Karbasi A (2016) Near-optimal active learning of halfspaces via query synthesis in the noisy setting. arXiv:[1603.0515](https://arxiv.org/abs/1603.0515)
- Chen W, Fuge M, Chazan J (2017) Design manifolds capture the intrinsic complexity and dimension of design spaces. *J Mech Des* 139(5):051,102. <https://doi.org/10.1115/1.4036134>
- Cohn D, Atlas L, Ladner R (1994) Improving generalization with active learning. *Mach Learn* 15(2):201–221
- Dagan I, Engelson SP (1995) Committee-based sampling for training probabilistic classifiers. In: Proceedings of the twelfth international conference on machine learning

- Dasgupta S, Kalai AT, Monteleoni C (2009) Analysis of perceptron-based active learning. *J Mach Learn Res* 10:281–299
- Dekel O, Gentile C, Sridharan K (2012) Selective sampling and active learning from single and multiple teachers. *J Mach Learn Res* 13(Sep):2655–2697
- Devanathan S, Ramani K (2010) Creating polytope representations of design spaces for visual exploration using consistency techniques. *J Mech Des* 132(8):081,011
- Freund Y, Seung HS, Shamir E, Tishby N (1997) Selective sampling using the query by committee algorithm. *Mach Learn* 28(2):133–168
- Gotovos A, Casati N, Hitz G, Krause A (2013) Active learning for level set estimation. In: Proceedings of the twenty-third international joint conference on artificial intelligence. AAAI Press, pp 1344–1350
- Hoang TN, Low BKH, Jaillet P, Kankanhalli M (2014) Nonmyopic ϵ -bayes-optimal active learning of gaussian processes. In: Xing EP, Jebara T (eds) Proceedings of the 31st international conference on machine learning, PMLR, vol 32. Proceedings of Machine Learning Research, Beijing, pp 739–747
- Hoi SC, Jin R, Zhu J, Lyu MR (2009) Semisupervised svm batch mode active learning with applications to image retrieval. *ACM Trans Inf Syst (TOIS)* 27(3):16
- Hsu WN, Lin HT (2015) Active learning by learning. In: Twenty-Ninth AAAI conference on artificial intelligence
- Huang YC, Chan KY (2010) A modified efficient global optimization algorithm for maximal reliability in a probabilistic constrained space. *J Mech Des* 132(6):061,002
- Huang SJ, Jin R, Zhou ZH (2010) Active learning by querying informative and representative examples. In: Advances in neural information processing systems, pp 892–900
- Jackson JC (1997) An efficient membership-query algorithm for learning dnf with respect to the uniform distribution. *J Comput Syst Sci* 55(3):414–440
- Kandasamy K, Schneider J, Póczos B (2017) Query efficient posterior estimation in scientific experiments via bayesian active learning. *Artif Intell* 243:45–56
- Kapoor A, Grauman K, Urtasun R, Darrell T (2010) Gaussian processes for object categorization. *Int J Comput Vis* 88(2):169–188
- King RD, Whelan KE, Jones FM, Reiser PG, Bryant CH, Muggleton SH, Kell DB, Oliver SG (2004) Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature* 427(6971):247–252
- Krause A, Guestrin C (2007) Nonmyopic active learning of gaussian processes: an exploration-exploitation approach. In: Proceedings of the 24th international conference on machine learning. ACM, pp 449–456
- Kreml G, Kottke D, Lemaire V (2015) Optimised probabilistic active learning (opal). *Mach Learn* 100(2-3):449–476
- Larson BJ, Mattson CA (2012) Design space exploration for quantifying a system model's feasible domain. *J Mech Des* 134(4):041,010
- Lee TH, Jung JJ (2008) A sampling technique enhancing accuracy and efficiency of metamodel-based rbdo: constraint boundary sampling. *Comput Struct* 86(13):1463–1476
- Lewis DD, Catlett J (1994) Heterogeneous uncertainty sampling for supervised learning. In: Proceedings of the eleventh international conference on machine learning, pp 148–156
- Lewis DD, Gale WA (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., New York, pp 3–12
- Ma Y, Garnett R, Schneider J (2014) Active area search via bayesian quadrature. In: Artificial intelligence and statistics, pp 595–603
- Mac Aodha O, Campbell ND, Kautz J, Brostow GJ (2014) Hierarchical subquery evaluation for active learning on a graph. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 564–571
- McCallum A, Nigam K et al (1998) Employing em and pool-based active learning for text classification. In: ICML, vol 98, pp 359–367
- Nguyen HT, Smeulders A (2004) Active learning using pre-clustering. In: Proceedings of the twenty-first international conference on machine learning ICML '04. ACM, New York, p 79, <https://doi.org/10.1145/1015330.1015349>
- Nowacki H (1980) Modelling of design decisions for cad. In: Computer aided design modelling, systems engineering, CAD-Systems. Springer, pp 177–223
- Orabona F, Cesa-Bianchi N (2011) Better algorithms for selective sampling. In: Proceedings of the 28th international conference on machine learning (ICML-11), pp 433–440
- Osugi T, Kim D, Scott S (2005) Balancing exploration and exploitation: a new algorithm for active machine learning. In: Fifth IEEE international conference on data mining. IEEE
- Rasmussen C, Williams C (2006) Gaussian processes for machine learning. The MIT Press
- Ren Y, Papalambros PY (2011) A design preference elicitation query as an optimization process. *J Mech Des* 133(11):111,004
- Schohn G, Cohn D (2000) Less is more: active learning with support vector machines. In: ICML, pp 839–846
- Settles B (2010) Active learning literature survey. *Univ Wiscons Madison* 52(55–66):11
- Settles B, Craven M (2008) An analysis of active learning strategies for sequence labeling tasks. In: Proceedings of the conference on empirical methods in natural language processing. Association for Computational Linguistics, pp 1070–1079
- Singh P, Van Der Herten J, Deschrijver D, Couckuyt I, Dhaene T (2017) A sequential sampling strategy for adaptive classification of computationally expensive data. *Struct Multidiscip Optim* 55(4):1425–1438
- Tong S, Koller D (2001) Support vector machine active learning with applications to text classification. *J Mach Learn Res* 2:45–66
- Yang X, Liu Y, Gao Y, Zhang Y, Gao Z (2015a) An active learning kriging model for hybrid reliability analysis with both random and interval variables. *Struct Multidiscip Optim* 51(5):1003–1016
- Yang Y, Ma Z, Nie F, Chang X, Hauptmann AG (2015b) Multi-class active learning by uncertainty sampling with diversity maximization. *Int J Comput Vis* 113(2):113–127
- Yannou B, Moreno F, Thevenot HJ, Simpson TW (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
- Zhu X, Lafferty J, Ghahramani Z (2003) Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In: ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining, vol 3
- Zhuang X, Pan R (2012) A sequential sampling strategy to improve reliability-based design optimization with implicit constraint functions. *J Mech Des* 134(2):021,002