Struct Multidisc Optim (2017) 55:1425–1438
DOI 10.1007/s00158-016-1584-1

RESEARCH PAPER

A sequential sampling strategy for adaptive classification of computationally expensive data

Prashant Singh[1] · **Joachim van der Herten**[1] · **Dirk Deschrijver**[1] · **Ivo Couckuyt**[1] · **Tom Dhaene**[1]

Received: 20 May 2015 / Revised: 23 August 2016 / Accepted: 3 September 2016 / Published online: 22 September 2016

Abstract Many real-world problems in engineering can be represented and solved as a data-driven classification problem, where the goal is to build a classifier that maps a given set of input parameters onto a corresponding class or label. In some cases, the collection of data samples can be computationally expensive. It is therefore crucial to solve the problem using as little data as possible. To this end, a novel sequential sampling algorithm is proposed that begins with a very small training set and supplements it in each iteration by a small batch of additional (expensive) data points. The outcome is a representative set of data samples that focuses the sampling on those locations in the input space where the class labels are changing more rapidly, while making sure that no class regions are missed.

Keywords Adaptive sampling · Surrogate models · Simulations · Expensive data

1 Introduction

Nowadays, the use of Machine learning techniques is becoming more widespread in engineering. Many problems deal with identifying a group, a category or a *class* to which a given input pattern belongs. Examples in literature include constrained optimization problems (Basudhar et al. 2012; Handoko et al. 2008), finding quasi-optimal regions (QoRs) (Singh et al. 2013b), determining food quality (Cen and He 2007), measuring analog circuit performance (De Bernardinis et al. 2003), detecting faults in aircraft engines (Rausch et al. 2004) and others. Such problems can be solved by fitting a *classifier* to a set of data that consists of a number of *instances* or data points. Each data point has a number of *attribute* values or features and a corresponding *class label*. The classifier can then be used to predict class labels for new, previously unseen, examples.

The data can be taken from databases of precomputed or recorded data. However, in engineering, data typically originates from computer experiments such as simulations which are generated on demand. A potential difficulty is that computer simulations are often computationally expensive. For example, Ford Motor Company reports that the computational cost to perform a single simulation for an automotive crashworthiness test takes on average 98 h to complete. This scale of computational expense would imply a total duration of 12 years to complete the entire analysis (Shan and Gary Wang 2010). In order to alleviate such a computational burden, there is a need to train classification models using as few training instances as possible. Therefore, this paper presents a sequential sampling strategy to collect deterministic data samples that can be used to build classifiers. It starts with an initial small set of training data, and iteratively adds more training points at well-chosen locations in the input space. The sampling algorithm picks additional points in a sequential way based on previously computed data and stops when a predefined stopping criterion is reached (e.g., number of allowed simulations, maximum simulation time,...).

In a post-processing step, the resulting data set can be used to build a classifier that allows an engineer to analyze

✉ Prashant Singh
prashant.singh@intec.ugent.be

[1] Department of Information Technology (INTEC), Ghent University - iMinds, Technologiepark-Zwijnaarde 15, 9052 Ghent, Belgium

e.g. functional dependencies between input variables, perform *what-if* analyses, perform optimization, study uncertainty quantification, etc.

The paper is organised as follows. Section 2 introduces the concept of adaptive classification, while Section 3 describes the related work and state-of-the-art. Section 4 explains the proposed sequential sampling algorithm. The algorithm is demonstrated on analytical examples in Section 5. Section 6 concludes the paper.

2 Adaptive classification

In the context of this work, the term adaptive classification is defined as classifier construction using training data obtained sequentially from an adaptive sampling algorithm. Consider a training set S in some input space $\mathcal{X} \subseteq \mathbb{R}^d$ spanning d attributes, and some output space $\mathcal{Y}$. The output space is $\mathcal{Y} = \{0, 1\}$ for a binary classification problem and $\mathcal{Y} = \{1..K\}$ for a K-class classification problem. The training set is denoted as $\mathsf{S} = (X, Y) \in \mathcal{X} \times \mathcal{Y}$ where X consists of n data points represented as vectors $\{\mathbf{x}_1 ... \mathbf{x}_n\}$ and Y consists of class labels $\{y_1 ... y_n\}$. The classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$ predicts the class label of a given input pattern $\hat{\mathbf{x}}$ as $\hat{y} = h(\hat{\mathbf{x}})$. For details of the classifier training process, the reader is referred to Bousquet et al. (2004).

The flowchart of the adaptive classification process is shown in Fig. 1. The initial training set S is obtained by generating a set X of b points in the input space using a traditional design scheme (e.g., Latin Hypercube Sampling). Then, X is evaluated using the expensive simulator to obtain the corresponding class labels Y.

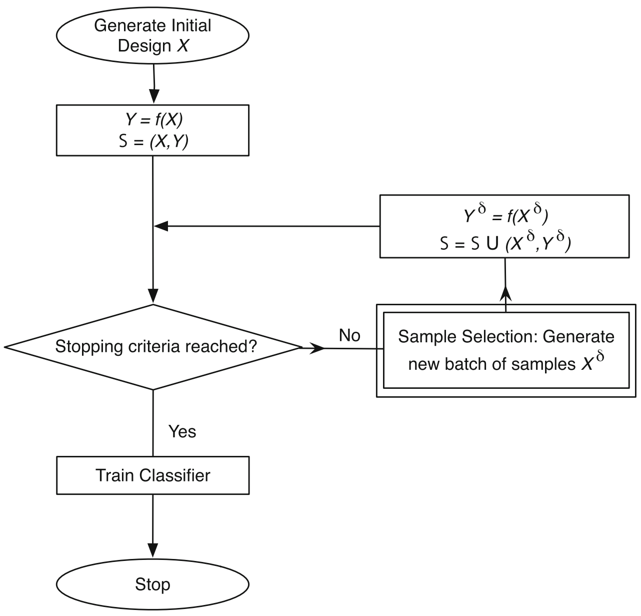

Generate Initial Design X
Y = f(X)
S = (X,Y)
Y^δ = f(X^δ)
S = S U (X^δ,Y^δ)
Stopping criteria reached?
No
Sample Selection: Generate new batch of samples X^δ
Yes
Train Classifier
Stop

Fig. 1 Adaptive classification flowchart

Assuming that the total number of allowed function evaluations is n, the sequential sampling algorithm selects a new batch of informative samples X^δ of size δ at well-chosen locations in the input space. The simulator evaluates X^δ resulting in class labels Y^δ. The training set S is updated as:

$$Y^\delta := f(X^\delta), \tag{1}$$

$$\mathsf{S} := \mathsf{S} \cup (X^\delta, Y^\delta). \tag{2}$$

This sampling process is iterated over $\lfloor \frac{n-b}{\delta} \rfloor$ times until the number of allowed simulations is exceeded, or one of the stopping criteria (if specified) has been reached. Stopping criteria may include exceeding allowed sampling budget, or time duration, etc. The classifier is then constructed using the final training set S.

The focus of this work is only on the sequential sampling process (the outlined box in Fig. 1), with the aim of obtaining an accurate model. The model is assumed not to contribute to the sequential sampling process, while the sampling algorithm aims to sample all the (a priori unknown) class boundaries of the problem at hand.

3 Related work on data sampling

Adaptive sampling is closely related to the field of *active learning* (Cohn et al. 1996; Settles 2012). However, there are subtle differences. Active learning is largely semi-supervised and traditionally assumes a *fixed* unlabeled dataset U, from which the learning algorithm must *subsample* data points to learn from. The learner can only select unlabelled data points $\mathbf{x}_i \in \mathsf{U}$. Often, an active learning algorithm provides a ranking of possible data points (Ailon 2011). The doctoral dissertation of Kevin Jamieson (2014) is an excellent reference for a mathematical treatment thereof. Active learning is also used in reinforcement learning (e.g. optimal learning for multi-armed bandits (Carpentier and Valko 2015)). The focus of this paper is on data sampling in a supervised learning context, where data samples are not taken from a database U, but instead they are queried from an oracle (e.g. a simulator) that provides a class label given a data point $\mathbf{x}_i$.

Adaptive sampling algorithms can be input-based, output-based, model-based, or a combination of the three

Table 1 Types of adaptive sampling algorithms: a combination of the three types is also possible

Input-based	Output-based	Model-based
Random	Neighborhood-Voronoi (Classification) (Singh et al. 2013b)	Probability of Feasibility (Forrester and Keane 2009)
Low discrepancy sequences (Hickernell 1998; Jin et al. 2005; Niederreiter 1978)		Model error sampling (Hendrickx and Dhaene 2005)
Latin Hypercube Sampling (Van Dam et al. 2007; Husslage et al. 2006; Qian 2009)		EDSD (Basudhar et al. 2012; Basudhar and Missoum 2008; 2010; Basudhar et al. 2008)
Monte-carlo/Optimization-based (Crombecq et al. 2009)		VSVM (Song 2013)
Voronoi-based (Crombecq et al. 2011a)		

depending on the information utilised in the sampling process. Table 1 lists the different type of sampling algorithms.

Input-based sampling algorithms like Latin Hypercube Sampling and Voronoi-based sampling aim at selecting points in a space-filling manner, so as to cover as much of the design (input) space as possible. Similarly, Low discrepancy sequences and Monte-carlo techniques distribute points as uniformly as possible.

Model-based sampling algorithms make use of intermediate models to guide the sample selection process. Typically, criteria such as Probability of Feasibility, model error, classifier boundary characteristics, etc. are used to guide sample selection. Support Vector Machine (SVM) classifiers have been used in literature to solve constrained optimization problems and failure domain identification using sequential sampling (e.g., Explicit Design Space Decomposition (EDSD) algorithm) (Basudhar et al. 2008, 2012; Basudhar and Missoum 2008, 2010).

EDSD uses SVMs to construct an explicit decision function that models a given constraint (for example). The algorithm works for multi and single-response problems with possible discontinuities. The classification approach enables better handling of discontinuities and potential non-smoothness in the problem. A convergence criterion, or sampling budget controls the number of iterations of the algorithm.

Although the EDSD algorithm is very effective for quickly and accurately refining the constraint function, it does not account for statistical distribution of the variables. New samples are selected along the decision boundary by maximizing the minimum distance from existing samples. Since the joint distribution of the variables is not accounted for, samples may be selected in regions of low probabilistic content (Lacaze and Missoum 2014). This poses a problem for applications dealing with expensive-to-evaluate objective functions. The generalized max-min sampling scheme (Lacaze and Missoum 2014) is a popular algorithm for solution of Reliability Based Design Optimization (RBDO) problems that takes the distribution of variables into consideration. This is crucial for problems where the design variables are not uniformly distributed.

Virtual SVMs (VSVM) (Song 2013) have been used to improve the accuracy of SVM classifiers for RBDO problems. A VSVM (Schölkopf et al. 1996) constructs a decision function by sampling near the class boundary. The sampling algorithm selects additional *virtual* samples in order to incorporate invariances (e.g., for image classification problems, transformations such as translation are often used) in the problem. The hope is that the enlarged training set incorporating virtual samples will lead to gains in accuracy over the original training set.

A detailed discussion on input and model-based sampling algorithms is out of scope of this work, and the interested reader can refer to Crombecq et al. (2011a, b), van der Herten et al. (2015), Forrester and Keane (2009), Hendrickx and Dhaene (2005).

In this paper, an input-output-based algorithm is proposed that uses the class labels of previously computed data points to narrow down the selection of new samples to interesting regions. The algorithms identifies local changes in the class labels and focuses the selection of samples in those areas. This kind of *exploitation* is merged with a space-filling *exploration* component to make sure that no regions are missed.

A key advantage of this method is that no intermediate classifiers (like SVM's) need to be built, which can lead to substantial savings in terms of computation time. While model-based methods entail the potential of exploiting model-specific information to better select new samples, they also run the risk of being misled by the model. For instance, in the initial stages of the sampling process, the model might be inaccurate and might drive the search towards non-optimal regions. This can result in interesting regions not being covered by the algorithm. Input or output-based methods are independent of the model, and therefore are less prone to such pitfalls.

4 Neighborhood-Voronoi sequential sampling algorithm

In this section, a new approach for sequential sampling in a classification context is proposed. The term sequential implies that the sampling algorithm is dynamic. The goal is to collect as much information as possible about the different class regions present, while using as few data samples as possible. The algorithm presented in this work is solely data driven. The data are collected, analysed, and new data points are chosen in a sequential manner. No intermediate (classification) models are required during the sampling process. Intermediate classifiers can be constructed if the user desires (to test accuracy as stopping criterion, for example) but is not required by the algorithm. Thus, the proposed algorithm is independent of any particular classifier.

The Neighborhood-Voronoi algorithm is based on the LOLA-Voronoi algorithm proposed by Crombecq et al. (2011a), with modifications made to handle classification problems instead of regression. The algorithm aims to balance *exploration* of the input space and *exploitation* to identify separating boundaries of the different class labels. In the following subsections, the Neighborhood-Voronoi sampling algorithm is explained by separately discussing the Neighborhood (exploitation) and Voronoi (exploration) components.

4.1 Exploitation

The exploitation component makes sure that samples are chosen more densely in the interesting regions, i.e., regions where a transition of class labels is present. A local neighborhood N of size m is computed for each instance $\mathbf{x}_i$, $\forall i \in 1, ..., n$ as:

$$N(\mathbf{x}_i) = \{\mathbf{x}_{i1}, \mathbf{x}_{i2}, ..., \mathbf{x}_{im}\} \subset X_r = \{\mathbf{x}_{ij}\}_{j=1}^{m}, \tag{3}$$

where $X_r = X \setminus \{\mathbf{x}_i\}$, with $\setminus$ being the set difference operator. To ensure that all directions around the instance $\mathbf{x}_i$ are covered uniformly, N is chosen according to optimal *adhesion* and *cohesion*. The terms adhesion and cohesion used in this work are defined below, and are unrelated in meaning to the use of the terms in biology, chemistry and materials science.

– Cohesion makes sure that the neighbors are as close to $\mathbf{x}_i$ as possible. It is defined as the average minimum distance of neighboring points from $\mathbf{x}_i$. The cohesion of a neighborhood N with respect to the fixed instance $\mathbf{x}_i$ is defined as:

$$C(N(\mathbf{x}_i)) = \frac{1}{m}\sum_{j=1}^{m} \|\mathbf{x}_{ij} - \mathbf{x}_i\|_2. \tag{4}$$

– Adhesion ensures that the neighbors are as far away from each other as possible. It is defined as the average minimum distance of neighbors from each other. The adhesion of a neighborhood N with respect to the fixed instance $\mathbf{x}_i$ is defined as:

$$A(N(\mathbf{x}_i)) = \frac{1}{m}\sum_{j=1}^{m} \min_{l\neq j} \|\mathbf{x}_{ij} - \mathbf{x}_{il}\|_2. \tag{5}$$

Ideally, a neighborhood N should have a low value of cohesion $C(N(\mathbf{x}_i))$ and a high value of adhesion $A(N(\mathbf{x}_i))$. Finding such a neighborhood becomes a multi-objective optimization problem involving minimising $C(N(\mathbf{x}_i))$ and maximising $A(N(\mathbf{x}_i))$ simultaneously, given a discrete set of candidate neighborhoods. In order to solve the multi-objective optimization problem efficiently, a simple approach is to combine the different objectives into a single aggregate objective function. The pre-requisite for such a solution would be to know the scale of both objectives, so that they can be combined into a formula with each objective having equal weight. The following text explains the method proposed to combine adhesion and cohesion into a single quantity $S(N(\mathbf{x}_i))$.

In an ideal scenario, the neighbors of the reference point $\mathbf{x}_i$ would be chosen such that they have equal cohesion contribution and form a $m-$sided regular polygon. The problem is extended to placing m points in an ideal configuration on a d-dimensional hyper-sphere such that the adhesion value $A(N(\mathbf{x}_i))$ of the reference point $\mathbf{x}_i$ is maximized. This is an open problem in mathematics (Croft et al. 1991).

Since there is no optimal solution to the problem of placing m points on a d-dimensional hypersphere (Saff and Kuijlaars 1997), a subproblem with a known solution is considered. This concerns the special case when $m = 2d$. Intuitively, for a one-dimensional case, $m = 2$ and the configuration will involve placing one point on either side of the reference point $\mathbf{x}$. In the two-dimensional case, $m = 4$ and the points will form a square around the reference point.

For $d-$dimensions, the optimal configuration is a d-cross-polytope (Cohn and Kumar 2007) which contains all points obtained by permuting the d coordinates:

$$\begin{gathered}(\pm 1, 0, 0, ..., 0)\\(0, \pm 1, 0, ..., 0)\\\vdots\\(0, 0, 0, ..., \pm 1).\end{gathered}$$

The cross-polytope configuration maximizes adhesion (Cohn and Kumar 2007).

The cross-polytope ratio Having established that for points lying on a hyper-sphere, the cross-polytope is the optimal configuration which maximizes adhesion, it can be inferred that any given neighborhood with cohesion $C(N(\mathbf{x}_i))$ must always have an adhesion value $A(N(\mathbf{x}_i))$ lower than that of the cross-polytope with radius $C(N(\mathbf{x}_i))$. For a cross-polytope, the distance between points is $\sqrt{2}$ times the distance from the origin (the reference point) for any dimension higher than 1. This implies that $\sqrt{2}C(N(\mathbf{x}_i))$ is the absolute upper bound for adhesion value of any neighborhood with cohesion $C(N(\mathbf{x}_i))$. Therefore, the following measure $R(N(\mathbf{x}_i))$ can be used to gauge how closely a neighborhood resembles a cross-polytope:

$$R(N(\mathbf{x}_i)) = \begin{cases} \frac{A(N(\mathbf{x}_i))}{\sqrt{2}C(N(\mathbf{x}_i))} & , d > 1 \\ 1 - \frac{|x_{i1}+x_{i2}|}{|x_{i1}|+|x_{i2}|+|x_{i1}-x_{i2}|} & , d = 1. \end{cases} \tag{6}$$

The exception for the one-dimensional case is due to the fact that the distance of the two points from each other is twice the distance from the reference point (Crombecq et al. 2011a).

A neighborhood score that combines adhesion and cohesion can be used to assign scores to neighborhoods:

$$S(N(\mathbf{x}_i)) = \frac{R(N(\mathbf{x}_i))}{C(N(\mathbf{x}_i))}. \tag{7}$$

This measure will prefer neighborhoods that lie close to the reference point $\mathbf{x}_i$ and resemble a cross-polytope. S can be used as a criterion to choose N for all instances. The neighborhood score thus is a single quantity which captures the desired balance of adhesion and cohesion mentioned above.

After such a neighborhood is constructed, the *class disagreement* χ corresponding to the sample $\mathbf{x}_i$ belonging to the neighborhood N is calculated according to the formula:

$$\chi(\mathbf{x}_i) = \begin{cases} 1, & \alpha > 1, \\ 0, & \alpha = 1. \end{cases} \tag{8}$$

where $(1 \leq \alpha \leq K)$ is the number of unique class labels in N. An observation with a higher value of χ is surrounded by samples having differing class labels, and needs to be sampled more intensely as it is located along the class boundaries.

Algorithm 1 Pseudocode for the exploitation component of the Neighborhood-Voronoi sequential sampling algorithm. X consists of all points processed by the algorithm previously. X_δ is the set of points selected by the algorithm in the previous iteration which are yet to be processed. δ is the number of new samples to be selected by the algorithm.

```
for all $\mathbf{x}_\delta \in X_\delta$ do
    for all $\mathbf{x} \in X$ do
        Evaluate membership of $\mathbf{x}_\delta$ for neighborhood $N(\mathbf{x})$ of $\mathbf{x}$
        Evaluate membership of $\mathbf{x}$ for neighborhood $N(\mathbf{x}_\delta)$
        Update class disagreement information for $\mathbf{x}$ and $\mathbf{x}_\delta$
    end for
    $X \leftarrow X \cup \mathbf{x}_\delta$
end for
for all $\mathbf{x} \in X$ do
    Calculate class disagreement score for $\mathbf{x}$
end for
Identify neighborhoods corresponding to $\delta$ highest ranked samples in $X$
Select new samples in these neighborhoods
```

Algorithm 1 describes the pseudocode of the exploitation component of the Neighborhood-Voronoi algorithm. The algorithm begins by updating the state of the samples selected by the algorithm in the previous iteration. Each new sample $\mathbf{x}_\delta$ is considered as a candidate neighbor for each processed sample $\mathbf{x}$ and vice-versa. The class disagreement scores for these samples are then updated according to Eq. 8. After processing all previously unprocessed samples, the metric χ is calculated for each sample in X which reflects the exploitation score of the sample in question. Finally, each of the neighborhoods corresponding to the top δ samples ranked according to χ are chosen to generate a new sample.

4.2 Exploration

The exploration component identifies regions in the input space that are prone to under-sampling, or under-representation. Such regions have a low density of points and a mechanism to identify these regions is required.

A Voronoi tessellation is a well-known way to partition a space based on density (Aurenhammer 1991). Assuming that our training set $X \subset \mathcal{X}$ in Euclidean space, the *Voronoi cell* $C_i \subset \mathcal{X}$ of the point $\mathbf{x}_i$ contains all points in $\mathcal{X}$ which lie closer to $\mathbf{x}_i$ than any other point in X. The Voronoi tessellation corresponding to X consists of all Voronoi cells $\{C_1, C_2, ..., C_n\}$ which tessellate the complete space $\mathcal{X}$. To define Voronoi cells formally, the notion of dominance (Aurenhammer 1991; Crombecq et al. 2011a) is used.

Dominance Given two distinct instances $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, the dominance of the instance $\mathbf{x}_i$ over the instance $\mathbf{x}_j$ is defined as the subset of the plane being at least as close to $\mathbf{x}_i$ as it is to $\mathbf{x}_j$ (Crombecq et al. 2011a):

$$dom(\mathbf{x}_i, \mathbf{x}_j) = \{\mathbf{x} \in \mathcal{X} \mid \|\mathbf{x} - \mathbf{x}_i\|_2 \le \|\mathbf{x} - \mathbf{x}_j\|_2\}. \qquad (9)$$

The plane $dom(\mathbf{x}_i, \mathbf{x}_j)$ is half-closed, bounded by the perpendicular bisector of $\mathbf{x}_i$ and $\mathbf{x}_j$. The bisector is called the *separator* of $\mathbf{x}_i$ and $\mathbf{x}_j$ which separates all points in $\mathcal{X}$ closer to $\mathbf{x}_i$ as opposed to $\mathbf{x}_j$. The Voronoi cell C_i corresponding to the instance $\mathbf{x}_i$ is the part of the design space $\mathcal{X}$ with is dominated by $\mathbf{x}_i$ over all other instances in X:

$$C_i = \bigcap_{\mathbf{x}_j \in X \setminus \{\mathbf{x}_i\}} dom(\mathbf{x}_i, \mathbf{x}_j). \qquad (10)$$

Figure 2 shows the Voronoi tessellation of a set $\{\mathbf{x}_i\}_{i=1}^{10}$ of randomly generated instances. The test instance $\mathbf{p}$ is closer to $\mathbf{x}_4$, and so are all points in $\mathcal{X}$ in the Voronoi cell corresponding to $\mathbf{x}_4$. It is also apparent from Fig. 2 that larger Voronoi cells correspond to regions in the design space that are sampled more sparsely. To fully explore the design space $\mathcal{X}$, new samples should be chosen in Voronoi cells with a large volume. For example, generating a new sample point or instance in the Voronoi cell corresponding to $\mathbf{x}_3$ will be more beneficial in terms of space-fillingness as compared to sampling the Voronoi cell corresponding to the instance $\mathbf{x}_8$. Therefore, a way to compute the hypervolume of Voronoi cells is required in order to compare them.

Voronoi tessellations are geometric duals of Delaunay triangulations. The Voronoi tessellation of a set of points X can be obtained from the Delaunay triangulation of X in O(n) time (Aurenhammer 1991). Computing the volume of Voronoi cells is harder, since the Voronoi cells near the border of $\mathcal{X}$ are unbounded. These Voronoi cells will therefore have infinite volume. Hence, the border-lying Voronoi cells must first be bounded before their volume can be computed.

Algorithm 2 Pseudocode for the exploration component of the Neighborhood-Voronoi sequential sampling algorithm. X consists of all points that have to be ranked by the algorithm according to their respective Voronoi cell size.

```
$T \leftarrow L$ random points $\in \mathcal{X}$
$V \leftarrow [0, 0, ..., 0]$
for all t $\in T$ do
    $d \leftarrow \infty$
    for all x $\in X$ do
        if $\|\mathbf{x} - \mathbf{t}\| < d$ then
            $\mathbf{x}_{closest} \leftarrow \mathbf{x}$
            $d \leftarrow \|\mathbf{x} - \mathbf{t}\|$
        end if
    end for
    $V[\mathbf{x}_{closest}] \leftarrow V[\mathbf{x}_{closest}] + \frac{1}{L}$
end for
```

As this is complex, the volume of Voronoi cells is approximated using a Monte Carlo approach described in Algorithm 2, since only the relative differences in volume of the Voronoi cells are important, and computing the exact volume is computationally very expensive. Additionally, exact computation of Voronoi volumes becomes infeasible

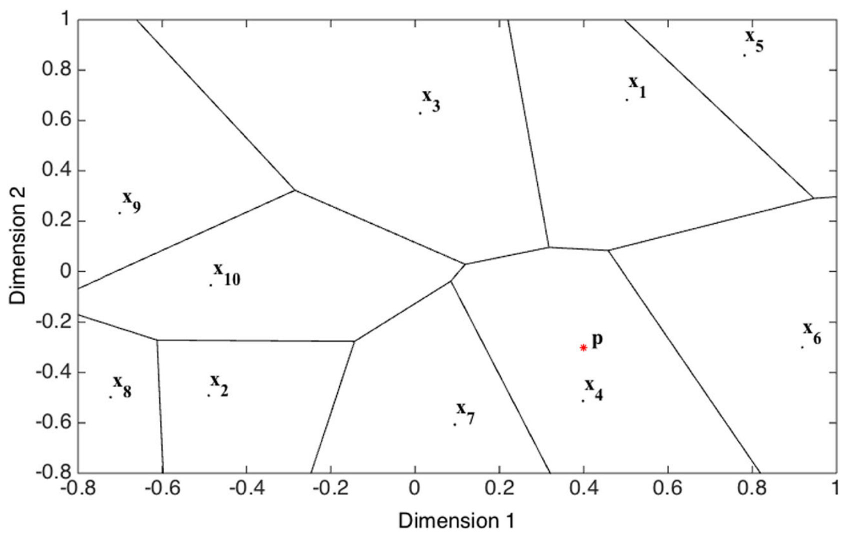


Fig. 2 The bounded Voronoi tessellation of a set of points $\{\mathbf{x}_i\}_{i=1}^{10}$. The test point $\mathbf{p}$ lying in the Voronoi cell corresponding to $\mathbf{x_4}$ lies closer to $\mathbf{x_4}$ than any other point

above 6 dimensions (Crombecq et al. 2009). A large number of random uniformly distributed test samples $T = \{\mathbf{t}_l\}_{l=1}^{L}$ are generated in $\mathcal{X}$. The minimum distance between each test point $\mathbf{t}_l$ and existing instance $\mathbf{x}_i$ is calculated. The test point is then assigned to the instance closest to it. By having enough test points, it is possible to estimate the volume of each Voronoi cell. The reader is referred to Crombecq et al. (2011a) for details of the algorithm to approximate the hypervolume of each Voronoi cell. Although distance computation will be adversely affected by the effect of *distance concentration* in high-dimensions, the Neighborhood-Voronoi algorithm is limited to 5–6 dimensional problems where these affects are not as strong (Beyer et al. 1999; Kabán 2012).

The exploration metric ψ of an instance $\mathbf{x}_i$ is defined as the ratio of the estimated volume of Voronoi cell C_i containing $\mathbf{x}_i$ with respect to the combined volume of all Voronoi cells in the design space $\mathcal{X}$:

$$\psi(\mathbf{x}_i) = \frac{\text{Vol}(C_i)}{\text{Vol}(C_1) + \text{Vol}(C_2) + \ldots + \text{Vol}(C_n)}. \tag{11}$$

A higher value of $\psi(\mathbf{x}_i)$ implies that the corresponding Voronoi cell C_i is large, whereas a smaller value of $\psi(\mathbf{x}_i)$ implies that C_i is smaller. The sampling algorithm should focus on cells with a higher value of ψ since they might be under-sampled.

4.3 Combining exploitation and exploration score

Algorithm 3 Pseudocode for the Neighborhood-Voronoi sequential sampling algorithm. δ is the number of new samples to be selected by the algorithm

```
for all $\mathbf{x} \in X$ do
    Compute $\chi(\mathbf{x})$
    Compute $\psi(\mathbf{x})$
    Compute final ranking $\Lambda(\mathbf{x}) = \chi(\mathbf{x}) + \psi(\mathbf{x})$
end for
Sort $X$ according to $\Lambda$
for $i = 1$ to $\delta$ do
    $\mathbf{x}_\delta \leftarrow$ generate a sample near $\mathbf{x}_i$ farthest from other samples
    $X_\delta \leftarrow X_\delta \cup \mathbf{x}_\delta$
end for
```

After obtaining the two metrics χ and ψ for exploitation and exploration respectively, the algorithm (Algorithm 3) assigns a combined score Λ for each existing sample $\mathbf{x} \in X$ as:

$$\Lambda(\mathbf{x}) = \chi(\mathbf{x}) + \psi(\mathbf{x}). \tag{12}$$

The algorithm ranks all samples in X in order of how well each sample ranks in exploitation and exploration according to the criterion Λ. The top δ samples in X are then selected and a new point is generated near each of these samples such that the generated point is as far away from other existing samples as possible (maximizing the minimum distance to other existing samples).

Although the combination scheme described above assigns equal weights to exploration and exploitation, it is possible to vary the contribution of each depending upon the characteristics of the problem at hand. Possible balancing schemes that can be used are $\epsilon-greedy$ and $\epsilon-decreasing$ as proposed in Singh et al. (2013a).

In the ϵ-greedy scheme, a user-specified tuning parameter $\epsilon \in [0, 1]$ decides the proportion of purely exploration-based sampling iterations. The remaining proportion of $1-\epsilon$ sampling iterations is purely exploitation-based. In each iteration, a random number α is generated according to a uniform distribution. If $\alpha < \epsilon$, then the current sampling iteration consists of pure exploration. If $\alpha \geq \epsilon$, then the current sampling iteration consists of pure exploitation.

The ϵ-decreasing variant is similar to ϵ-greedy strategy, but for the choice of the parameter ϵ. The initial value of ϵ can be user defined (or a default of 1), and decreases over proceeding sampling iterations. Therefore, it is possible to start with only exploration, which progressively decreases and makes way for increasing exploitation. This is intuitive since it is desirable to perform more exploration up-front when little is known about the design space. With time, as more information is obtained, performing more exploitation may be beneficial.

5 Examples

5.1 Example: non-linearly separable classification problem

A Gaussian function centered at $(x_1', x_2') = (0, 0)$ having a standard deviation $\sigma = \sqrt{5}$ is defined as:

$$f(\mathbf{x}) = \exp^{-\left(\frac{(x_1-x_1')^2+(x_2-x_2')^2}{\sigma^2}\right)},$$
$$dom(f(\mathbf{x})) = \{x_1, x_2 \in [-5, 5]\},$$

where $\mathbf{x} = \{x_1, x_2\}$. The problem involves finding the region in the input space which corresponds to function values within 50 % of the highest possible function value ($f_{max} = 1$). The classification problem is defined as:

$$y_i = \begin{cases} 1, & f(\mathbf{x}_i) \in [0.5, \infty), \\ 0, & f(\mathbf{x}_i) \in (-\infty, 0.5). \end{cases}$$

A classifier is trained over instances obtained according to a Latin Hypercube design of $b = 15$ points (including the corner points of the design space). The Neighborhood-Voronoi sequential sampling algorithm is used to select

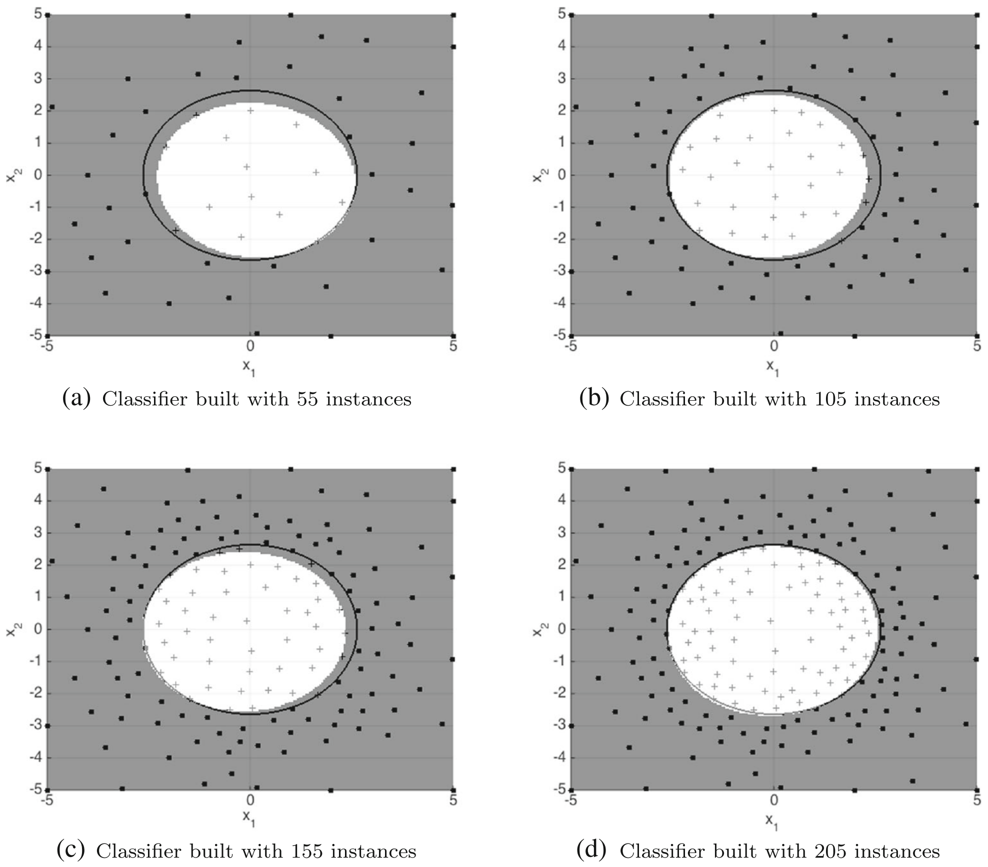


Fig. 3 Non-Linearly Separable Classification Problem: The sampling performed by the Neighborhood-Voronoi algorithm for the Gaussian function. The *black circle* is the true class boundary. The learned positive class is represented by the white region, while the learned negative class is represented by the *grey region*. The *dots* are the instances in the training set for that particular iteration

additional samples iteratively in batches of $\delta = 10$ each. The total number of function evaluations allowed is $n = 205$. For the sake of visualization, a Support Vector Machine (SVM) classifier is built based on the outcome of the proposed sampling strategy. All experiments have been performed using the SUrrogate MOdeling (SUMO) toolbox (Gorissen et al. 2010) for MATLAB, running on a MacBook Pro machine with 16 GB RAM and a 2.4 GHz Intel Core i5 processor. The operating system is OS X El Capitan. The SUMO toolbox is freely available for personal academic use at http://www.sumo.intec.ugent.be.

The well-known LIBSVM implementation (Chang and Lin 2011) is used for all experiments in this paper. The radial basis function (RBF) kernel is chosen for its overall performance and the hyperparameter are optimized using the DIRECT (DIviding RECTangles) (Jones 2001) algorithm.

The results of applying the Neighborhood-Voronoi algorithm can be seen in Fig. 3. There is a large discrepancy between true and learned class boundaries in the initial iterations. In subsequent iterations, the classifier boundary is refined by selecting samples near the boundary. The accuracy of the classifier over 200 randomly generated test points was 98 % with Precision and Recall being 1 and 0.98 respectively. The evolution of classifier accuracy with increasing number of training instances over a static set of test instances can be seen in Fig. 4. The accuracy rises rapidly between 35 and 65 training samples, after which it begins to stabilise. Figure 4 also shows a comparison with random sampling. It is observed that random

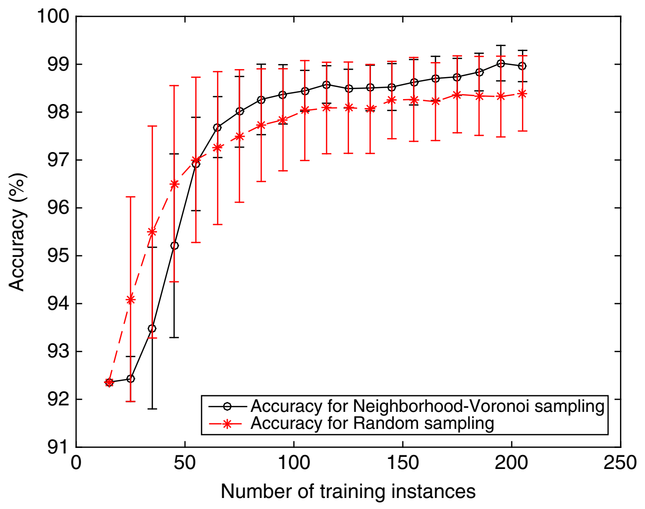


Fig. 4 Non-Linearly Separable Classification Problem: The evolution of classifier accuracy with respect to number of training instances. The results are averaged over 50 separate runs. The bars correspond to confidence intervals with each bar being $2 * stdev(accuracy)$ long

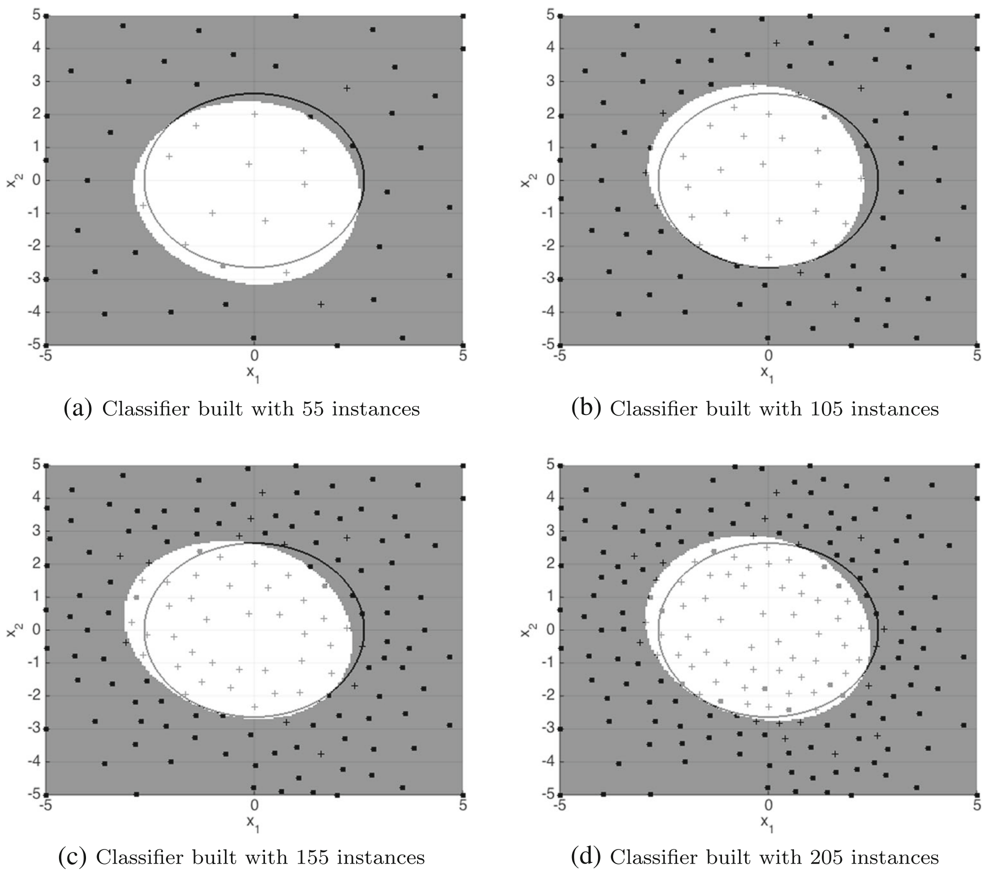


Fig. 5 Non-Linearly Separable Classification Problem with noise: The sampling performed by the Neighborhood-Voronoi algorithm for the Gaussian function with added noise. The black circle is the true class boundary. The learned positive class is represented by the white region, while the learned negative class is represented by the grey region. The dots are the instances in the training set for that particular iteration

(a) Classifier built with 55 instances

(b) Classifier built with 105 instances

(c) Classifier built with 155 instances

(d) Classifier built with 205 instances

sampling climbs in accuracy quickly, but the balanced sampling properties of the Neighborhood-Voronoi algorithm make sure it outperforms random sampling consistently. The initial lethargy can be attributed to too few samples being near the actual boundary in the initial iterations. As sampling progresses, the uncertainty near the boundary decreases and accuracy of trained classifier improves. This is reflected in tighter confidence intervals corresponding to the Neighborhood-Voronoi algorithm in Fig. 4 towards the end. The expected accuracy of the classifier trained using the Neighborhood-Voronoi algorithm is higher, and the variance of the accuracy over several runs is smaller than that corresponding values associated with random sampling.

5.2 Effect of noise

In case of stochastic computer experiments, the effect of noise must be taken into consideration. In order to study how noise affects the algorithm, random Gaussian noise with zero-mean and standard deviation of 0.2 is added to the previous example. It can be seen in Fig. 5 that the nature of the sampling is unaffected and robust, although the noise will inevitably lead to accuracy loss when the data is used to build a classifier. The accuracy of the resulting classifier over 200 randomly generated test points was 94.35 % with Precision and Recall being 0.9522 and 0.9891 respectively. From the sampling behavior depicted in Fig. 5 it can be inferred that the dip in accuracy, Precision and Recall is due to the noise in the data rather than inefficacy of the sampling algorithm. The algorithm avoids zooming in on noisy areas, as it causes corresponding Voronoi cells to become

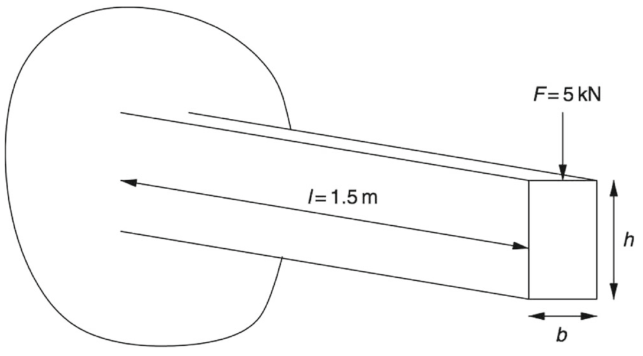


Fig. 6 The Nowacki Beam Problem

Table 2 Nowacki Beam Problem: Problem Definition

$\min_{b,h}$	A, σ_B	s.t.	$\delta \leq 5$mm
			$\sigma_B \leq \sigma_Y$
for	20 mm $< h <$ 250 mm		$\tau \leq \sigma_Y/2$
	10 mm $< b <$ 50 mm		$h/b \leq 10$
			$F_{CRIT} \geq f \times F$

increasingly smaller, leading to lower $\psi(\mathbf{x})$ and $\chi(\mathbf{x})$ scores in Eqs. 11 and 12.

5.3 Example: Nowacki beam problem

A constrained multi-objective optimization problem described by Nowacki (1980) is now considered. The aim is to design a tip-loaded encastre cantilever beam (Fig. 6) minimizing the cross-sectional area and bending stress subject to certain constraints. In order to achieve the goal, the problem of finding regions of feasibility must be solved first. The rectangular beam has length $l = 0.5$ m and is subjected to a tip-load $F = 5$ kN. The design variables are the height h and breadth b of the beam. The optimization problem can be formulated as described in Table 2, with $A = b \times h$ being the cross-sectional area of the beam, $\sigma_B = 6Fl/(bh^2)$ the bending stress, $\delta = Fl^3/(3EI_Y)$ the maximum tip deflection, σ_Y the yield stress of the material, $\tau = 3F/(2bh)$ the maximum allowable shear stress, h/b the height-to-breadth ratio, and $F_{CRIT} = (4/l^2)\sqrt{GI_T EI_Z/(1-\nu^2)}$ the failure force of buckling. Here, $I_T = (b^3h + bh^3)/12$, $I_Z = b^3h/12$, $I_Y = bh^3/12$, and f is a safety factor of two. The material under consideration is mild steel with yield stress $\sigma_Y = 240$ MPa, Young's modulus $E = 216.62$ GPa, $\nu = 0.27$ and shear modulus $G = 86.65$ GPa.

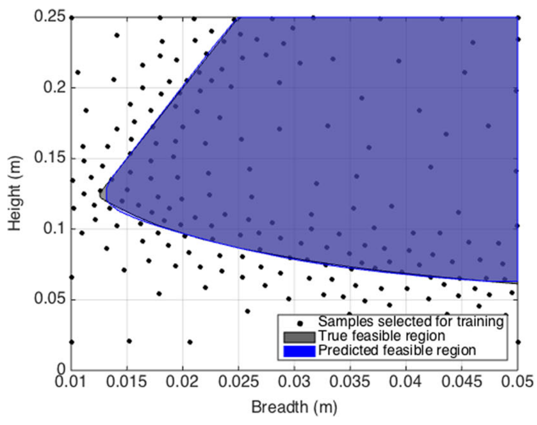
Instead of finding the optima, the problem of finding the *region of feasibility* in the design space meeting all constraints is considered. This can be also seen as an inverse problem of finding a region (quasi-optimal region) in the design space corresponding to desired (known) output. For complex problems, a practitioner might find it useful to find a small region in the design space containing possible solutions first, and concentrating future efforts in only that region. This kind of *domain reduction* can be very useful (Spaans and Luus 1992) while solving expensive constrained optimization problems. Finding the feasible region efficiently will save the practitioner a lot of time and effort.

The problem of finding the feasible region is solved using adaptive classification. The problem can be cast as a classification problem with the class label y_i assigned to instance $\mathbf{x}_i = (b, h)$ as:

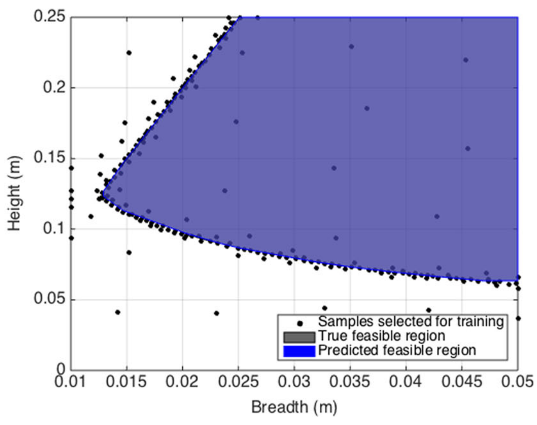
$$y_i = \begin{cases} 1, & \delta \leq 5\text{mm}; \sigma_B \leq \sigma_Y; \tau \leq \sigma_Y/2; \\ & h/b \leq 10; F_{CRIT} \geq f \times F, \\ 0, & \text{otherwise.} \end{cases}$$

An Artificial Neural Network (ANN) classifier available from the WEKA data mining software (Hall et al. 2009), and a SVM classifier were used to model the constrained problem. The initial design was a Latin Hypercube of 20 instances. The Neighborhood-Voronoi sequential sampling algorithm was used to select 10 new samples in each iteration and the total number of allowed function evaluations was 200.

The result can be seen in Fig. 7a. It is observed that samples have been selected densely along the edge of the feasible region, which is desirable (Schoenauer and Michalewicz 1996). Also, the algorithm spreads exploitation samples evenly across the boundary, which is prudent since nothing can be assumed about how well the model is



(a) The sampling performed by the Neighborhood-Voronoi algorithm using an SVM classifier



(b) The sampling performed by the EDSD algorithm

Fig. 7 Nowacki Beam Problem: The comparison of sampling performed by the Neighborhood-Voronoi and EDSD algorithms. The sampling budget is set to 200 points

Table 3 Nowacki beam problem: Classifier test performance

Run	Algorithm	Classifier	# samples	Precision	Recall	Accuracy (%)		Time
						Test Set	5-fold CV	
1	Neighborhood-Voronoi	ANN	200	0.9859	0.9861	98.65	98.6	120.02
2	Neighborhood-Voronoi	SVM	200	0.9962	0.9954	99.60	98	361.15
3	EDSD	SVM	200	0.9993	0.9994	99.93	84	2515.82

approximating the boundary. In the ideal scenario, the algorithm should assign more samples to regions where the class labels are changing more rapidly, i.e., the leftmost tip of the gray shaded region in Fig. 7a. The final classifier built using 200 samples has an *accuracy* of 99.6 %, *precision* of 0.9962 and *recall* of 0.9954.

As a comparison, the state-of-the-art EDSD algorithm is also applied to obtain the feasible region of the Nowacki beam problem. The implementation used is from the CODES toolbox[1] (Lacaze and Missoum 2015). The initial design was a Centroidal Voronoi Tessellation (CVT) of 20 points matching the size of LHD used in case of the Neighborhood-Voronoi algorithm. The sampling budget was also set to 200 points to match the experimental settings described above. All other parameters of the algorithm were left at their default values.

Table 3 compares the results obtained using the Neighborhood-Voronoi algorithm and the EDSD algorithm on a separate test set of 4900 samples in addition to 5-fold cross-validation. The cross-validation accuracy of EDSD is lower than Neighborhood-Voronoi owing to the distribution of selected samples. A vast majority of samples are selected very near (or at) the decision boundary, where the classifier is more prone to misclassify test samples. Using cross-validation runs the risk of the estimated accuracy being prone to the distribution of samples. The excellent performance of EDSD on a separate validation set (uniformly distributed) reaffirms this notion and demonstrates the good global performance of the model.

It can be seen that both EDSD and Neighborhood-Voronoi algorithms lead to models with comparable validation accuracy. The Neighborhood-Voronoi algorithm provides faster sampling, but marginally less accurate models. The difference in accuracy can be attributed in part to the presence of the purpose-designed exploration component in the sampling process, while the EDSD algorithm relies predominantly on the initial design for exploration. EDSD exhibits very aggressive exploitation (Fig. 7b) that leads to a very accurate characterization of the decision boundary. Since a part of the sampling budget of Neighborhood-Voronoi goes towards exploration, the model accuracy improvement is comparatively slower. Since both algorithms were run with their default settings, the running times mentioned in Table 3 reflect the time taken in a typical run. Different hyperparameter combinations might yield faster or slower running times.

Although the exploration component of the Neighborhood-Voronoi algorithm may lead to slower improvement in accuracy, it ensures that unknown feasible regions will be found if given enough sampling budget. The following example illustrates the importance of exploration.

5.4 Example: disconnected feasible regions

The problem of finding feasible regions becomes challenging when the area occupied by feasible regions is very small in comparison to the entire design space. Problems are compounded if there are multiple disjoint feasible regions forming *islands* in the design space.

Consider the modified Branin function (Sasena 2002) of the form:

$$\begin{aligned}\text{Min } f(\mathbf{x}) &= -(x_1-10)^2-(x_2-15)^2,\\ \text{s.t. } g(\mathbf{x}) &= \left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2\\ &\quad +10\left(1-\frac{1}{8\pi}\right)\cos x_1+10\le 5.\end{aligned}$$

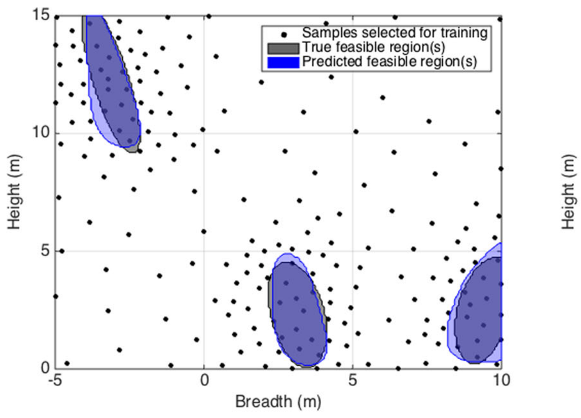
The problem translated to the following classification problem with the class label y_i assigned to instance $\mathbf{x}_i$ as:

$$y_i=\begin{cases}1, & g(\mathbf{x}_i)\le 5,\\ 0, & \text{otherwise.}\end{cases}$$

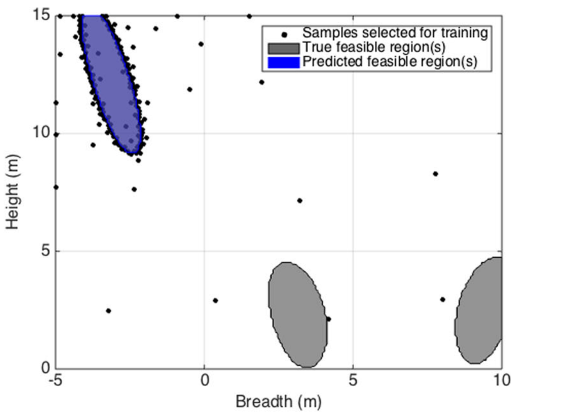
The Neighborhood-Voronoi and EDSD algorithms are used to solve for the constrained design space represented by $g(\mathbf{x})$. In order to illustrate the need for exploration, a small initial design of 10 points in the form of a CVT is used. The same initial design is used with both algorithms to ensure a fair start for the sampling process. The sampling budget is set to a total of 200 points.

The results are shown in Fig. 8. Since the initial design missed two of the three feasible regions, the EDSD algorithm had no means to reach the two distant islands. The EDSD algorithm incorporates local exploration on and around the decision boundary but lacks a global exploration component. The Neighborhood-Voronoi algorithm was able

[1] http://codes.arizona.edu/toolbox/



(a) The sampling performed by the Neighborhood-Voronoi algorithm using an SVM classifier



(b) The sampling performed by the EDSD algorithm

Fig. 8 Modified Branin Function: The comparison of sampling performed by the EDSD and Neighborhood-Voronoi algorithms. The initial design common to both algorithms consisted of 10 points that did not cover the two feasible regions at the bottom. The sampling budget is set to 200 points

to identify all three feasible regions owing to Voronoi-based global exploration, even though the initial design had missed two regions. This can be critical in problems where the feasible regions occupy a small area of the design space, and the initial design is not large enough to cover all feasible regions.

Indeed, the exploration component eliminates the need to carefully choose the size of the initial design and allows for automatic sequential coverage of the design space. The EDSD algorithm works very well in quickly refining SVM classifier boundaries, but will struggle in such scenarios and can also benefit from incorporation of a global exploration component.

5.5 Example: optimization of a GPS antenna

Finally, a 5-dimensional classification problem is considered. Consider a textile microstrip probe-fed compressible GPS patch antenna (Vallozzi et al. 2009) shown in Fig. 9. The antenna consists of a square patch with two truncated corners glued on a flexible closed-cell expanded rubber protective foam substrate. The patch is fed in the top right corner by a coaxial probe, exciting a right hand circular polarization. The nominal characteristics of the substrate are relative permittivity ϵ_r equal to 1.56, loss tangent $tan\delta$ equal to 0.012 and thickness h equal to 3.94 mm.

The optimization of the design of such a GPS antenna is a nontrivial task, as multiple constraints have to be satisfied. First, the antenna has to comply with the requirements of the GPS-L1 standard. Therefore, its return loss $|S_{11}|$ has to be lower than -10 dB and its axial ratio AR (defined as the ratio between the amplitudes of the orthogonal components composing the circularly polarized field) has to be smaller than 3 dB in the [1.56342,1.58742] GHz frequency band. Second, the fulfilment of these criteria has to be achieved without sacrificing the directive gain of the antenna, which is of paramount importance for its correct operation. Moreover, since the antenna is simulated by means of the Keysight's ADS Momentum 2012–08 full-wave solver, the whole process is expected to be very time consuming. Each simulation takes approximately one minute on an Intel Core i5 machine with 4 GB RAM.

Therefore, the Neighborhood-Voronoi algorithm is applied to find the feasible region of the considered design with respect to specified constraints over the objectives

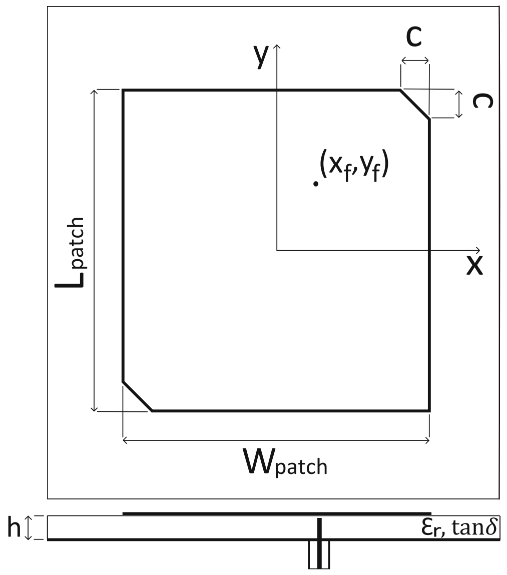


Fig. 9 GPS antenna: topview and cross-section of textile microstrip probe-fed GPS patch antenna

$|S_{11}|$, boresight AR and boresight Gain in the GPS-L1 frequency band. More specifically, the objectives of the optimization are minimizing $|S_{11}|_{\max}$ and $\text{AR}_{\max}$, and maximizing Gain. The constraint satisfaction problem is formulated as the following classification problem with the class label y_i assigned to instance $\mathbf{x}_i$ as:

$$y_i = \begin{cases} 1, & (\text{AR}_{\max} < \text{AR}_{\text{lim}}, |S_{11}|_{\max} < |S_{11}|_{\text{lim}}), \\ 0, & \text{otherwise}, \end{cases} \tag{13}$$

where the limits AR_{lim} and $|S_{11}|_{\text{lim}}$ are dictated by the GPS-L1 standard, being 3 dB and -10 dB, respectively. $\text{AR}_{\max}$, $|S_{11}|_{\max}$ and $\text{Gain}_{\min}$ are the maximum and the minimum values, respectively, at operating frequencies 1.56342 GHz, 1.57542 GHz and 1.58742 GHz. Each point is a 5-dimensional vector $\mathbf{x}_i = \{L^i, W^i, c^i, x_f^i, y_f^i\}$ corresponding to a realization of the GPS antenna under study, and is simulated in Keysight's ADS Momentum 2012–08 to obtain the values of $|S_{11}|$, boresight AR and boresight Gain. Consequently, the class label y_i is assigned to $\mathbf{x}_i$ based on the simulated values (13). The geometric parameters are varied within the following ranges:

$$\begin{aligned} 72.6\text{ mm} &< L_{patch} < 75.2\text{ mm}, \\ 69.2\text{ mm} &< W_{patch} < 71.5\text{ mm}, \\ 6.5\text{ mm} &< x_f < 9.7\text{ mm}, \\ 13.8\text{ mm} &< y_f < 16.4\text{ mm, and} \\ 3\text{ mm} &< c < 6\text{ mm}. \end{aligned}$$

The initial design is a Latin Hypercube of 300 points in 5 dimensions, in addition to the 32 corner points of the design space. The Neighborhood-Voronoi algorithm selects 5 new points in each iteration until a simulation budget of 500 simulations is exhausted.

The resulting feasible region in the output space can be seen in Fig. 10. The Neighborhood-Voronoi algorithm sampled the 5D input space in order to characterize the feasible region. The algorithm selected 28 points within the feasible region, and the rest outside. The total running time of 7 h and 4 min included approximately 6 h and 45 min spent performing simulations.

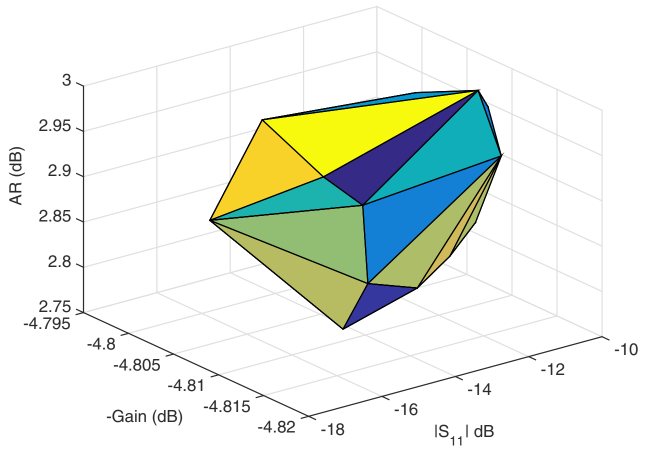


Fig. 10 GPS antenna: feasible region in the output space of the GPS antenna design problem

The resulting model can be used to quickly test if a given combination of input parameters satisfies the requirements of the L1-band GPS standard almost instantaneously. This is a gain of an order of magnitude over performing a simulation ($<$ 1 s versus 45 s). Such models aid the practitioner in performing design space exploration and expedite the design process.

6 Conclusion and future work

Many design and optimization problems in engineering involve training of a classification model based on computationally expensive simulation data. A novel sequential sampling strategy for training classification models is presented in this paper that minimizes the number of training points needed to obtain an accurate classifier. The novel sequential sampling algorithm is compared to state-of-the-art algorithms and illustrated on several non-linear analytical examples and on a structural design problem. Although only binary classification problems are discussed as examples for the purpose of exposition, the proposed algorithm functions as described for multi-class classification problems as well. The algorithm is scalable till approximately 5–6 dimensions, beyond which the running time prolongs considerably (van der Herten et al. 2015). Future work involves exploring fuzzy theory-based approaches to extend the algorithm towards handling problems having upto 10 input dimensions.

Acknowledgments This work was supported by the Research Foundation Flanders (FWO-Vlaanderen). Ivo Couckuyt is a Postdoctoral Researcher of FWO-Vlaanderen. The authors would like to thank Prof. Hendrik Rogier and Marco Rossi from the Department of Information Technology, Ghent University for providing the GPS antenna design problem.

References

Ailon N (2011) Active learning ranking from pairwise preferences with almost optimal query complexity. In: Advances in Neural Information Processing Systems, pp 810–818

Aurenhammer F (1991) Voronoi diagrams a survey of a fundamental geometric data structure. ACM Comput Surv (CSUR) 23(3):345–405

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

Basudhar A, Missoum S, Sanchez AH (2008) Limit state function identification using support vector machines for discontinuous responses and disjoint failure domains. Probab Eng Mech 23(1):1–11

Basudhar A, Dribusch C, Lacaze S, Missoum S (2012) Constrained efficient global optimization with support vector machines. Struct Multidiscip Optim 46(2):201–221

Beyer K, Goldstein J, Ramakrishnan R, Shaft U (1999) When is nearest neighbor meaningful? In: Database theory ICDT 99. Springer, pp 217–235

Bousquet O, Boucheron S, Lugosi G (2004) Introduction to statistical learning theory. In: Advanced lectures on machine learning. Springer, pp 169–207

Carpentier A, Valko M (2015) Simple regret for infinitely many armed bandits. arXiv preprint arXiv:1505.04627

Cen Haiyan, He Yong (2007) Theory and application of near infrared reflectance spectroscopy in determination of food quality. Trends Food Sci Technol 18(2):72–83

Chang C-C, Lin C-J (2011) LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol 2:27:1–27:27. Software available at, http://www.csie.ntu.edu.tw/cjlin/libsvm

Cohn DA, Ghahramani Z, Jordan MI (1996) Active learning with statistical models. J Artif Intell Res 4:129–145

Cohn H, Kumar A (2007) Universally optimal distribution of points on spheres. J Am Math Soc 20(1):99–148

Croft HT, Falconer KJ, Guy RK (1991) Unsolved problems in geometry. Springer, Berlin

Crombecq K, Couckuyt I, Gorissen D, Dhaene T (2009) Space-filling sequential design strategies for adaptive surrogate modelling. In: The first international conference on soft computing technology in civil, structural and environmental engineering

Crombecq K, Gorissen D, Deschrijver D, Dhaene T (2011a) A novel hybrid sequential design strategy for global surrogate modeling of computer experiments. SIAM J Sci Comput 33(4):1948–1974

Crombecq K, Laermans E, Dhaene T (2011b) Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling. Eur J Oper Res 214(3):683–696

De Bernardinis F, Jordan MI, SangiovanniVincentelli A (2003) Support vector machines for analog circuit performance representation. In: Design automation conference, 2003. Proceedings, pages 964–969. IEEE

Forrester AIJ, Keane AJ (2009) Recent advances in surrogate-based optimization. Prog Aerosp Sci 45(1):50–79

Gorissen D, Couckuyt I, Demeester P, Dhaene T, Crombecq K (2010) A surrogate modeling and adaptive sampling toolbox for computer based design. J Mach Learn Res 11:2051–2055

Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: an update. ACM SIGKDD Explorations Newsl 11(1):10–18

Handoko SD, Keong KC, Soon OY (2008) Using classification for constrained memetic algorithm: a new paradigm. In: IEEE international conference on systems, man and cybernetics, 2008. SMC 2008. IEEE, pages 547–552

Hendrickx W, Dhaene T (2005) Sequential design and rational metamodelling. In: Proceedings of the 37th conference on Winter simulation. Winter Simulation Conference, pp 290–298

Hickernell F (1998) A generalized discrepancy and quadrature error bound. Math Comput Am Math Soc 67(221):299–322

Husslage BGM et al. (2006) Maximin designs for computer experiments. Technical report, Tilburg University

Kevin Jamieson (2014) The analysis of adaptive data collection methods for machine learning. PhD thesis, UW-Madison

Jin R, Chen W, Sudjianto A (2005) An efficient algorithm for constructing optimal design of computer experiments. J Stat Plan Inference 134(1):268–287

Jones DR (2001) Direct global optimization algorithmdirect global optimization algorithm. In: Encyclopedia of Optimization. Springer, pp 431–440

Kabán A (2012) Non-parametric detection of meaningless distances in high dimensional data. Stat Comput 22(2):375–385

Lacaze S, Missoum S (2014) A generalized max-min sample for surrogate update. Struct Multidiscip Optim 49(4):683–687

Lacaze S, Missoum S (2015) CODES: a toolbox for computational design. sVersion 1.0. www.codes.arizona.edu/toolbox

Niederreiter H (1978) Quasi-monte carlo methods and pseudo-random numbers. Bull Am Math Soc 84(6):957–1041

Nowacki H (1980) Modelling of design decisions for cad. In: Computer Aided Design Modelling, Systems Engineering, CAD-Systems. Springer, pp 177–223

Qian PZG (2009) Nested latin hypercube designs. Biometrika page asp045

Rausch R, Viassolo DE, Kumar A, Goebel K, Eklund N, Brunell B, Bonanni P (2004) Towards in-flight detection and accommodation of faults in aircraft engines. In: AIAA 1st Intelligent Systems Technical Conference, Chicago, IL, pp 20–22

Saff EB, Kuijlaars ABJ (1997) Distributing many points on a sphere. Math Intell 19(1):5–11

Sasena MJ (2002) Flexibility and efficiency enhancements for constrained global design optimization with kriging approximations. PhD thesis, General Motors

Schoenauer M, Michalewicz Z (1996) Evolutionary computation at the edge of feasibility. In: Parallel Problem Solving from Nature PPSN IV. Springer, pp 245–254

Schölkopf B, Burges C, Vapnik V (1996) Incorporating invariances in support vector learning machines. In: Artificial Neural Networks ICANN 96. Springer, pp 47–52

Settles B (2012) Active learning. Synth Lect Artif Intell Mach Learn 6(1):1–114

Shan S, Gary Wang G (2010) Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. Struct Multidiscip Optim 41(2):219–241

Singh P, Deschrijver D, Dhaene T (2013a) A balanced sequential design strategy for global surrogate modeling. In: Simulation conference (WSC), 2013 Winter. IEEE, pp 2172–2179

Singh P, Deschrijver D, Pissoort D, Dhaene T (2013b) Adaptive classification algorithm for emc-compliance testing of electronic devices. Electron Lett 49(24):1526–1528

Song H (2013) Efficient sampling-based rbdo by using virtual support vector machine and improving the accuracy of the kriging method

Spaans R, Luus R (1992) Importance of search-domain reduction in random optimization. J Optim Theory Appl 75(3):635–638

Vallozzi L, Vandendriessche W, Rogier H, Hertleer C, Scarpello M (2009) Design of a protective garment gps antenna. Microw Opt Technol Lett 51(6):1504–1508

Van Dam ER, Husslage B, Hertog DD, Melissen H (2007) Maximin latin hypercube designs in two dimensions. Oper Res 55(1):158–169

van der Herten J, Couckuyt I, Deschrijver D, Dhaene T (2015) A fuzzy hybrid sequential design strategy for global surrogate modeling of high-dimensional computer experiments. SIAM J Sci Comput 37(2):A1020–A1039