Exotic Bets: Evolutionary Computing Coupled with Bet Mechanisms for Model Selection

Simon Reichhuber[a] and Sven Tomforde[b]
Intelligent Systems, Kiel University, Hermann-Rodewald-Str. 3, 24118 Kiel, Germany

Keywords: Evolutionary Algorithms, Bet-Based Learning, Model Selection, Gradient-Free Optimisation.

Abstract: This paper presents a framework for the application of an external bet-based evolutionary algorithm to the problem of model selection. In particular, we have defined two new risk functions, called *sample space exoticness* and *configuration space exoticness*. The latter is used to manage the risk of bet placement. Further, we explain how to implement the bet-based approach for model selection in the domain of multi-class classification and experimentally compare the performance of the algorithm to reference derivative-free hyperparameter optimisers (GA and Bayesian Optimisation) on *MNIST*. Finally, we experimentally show that for the classifiers SVM, MLP, and Nearest Neighbors the balanced accuracy can be increased by up to three percentage points.

1 INTRODUCTION

The field of Evolutionary Algorithms (EAs) has intensively been expanded throughout the last decades. Similar to other nature-inspired algorithms such as Neural Networks, the design of these algorithms followed a bottom-up approach, where only the main ideas from nature have been simulated: In Neural Networks the neural structure or in EAs the encoding scheme for bits of information as chromosomes. With the increasing computing capabilities, researchers have extended the basic concept of EAs to solve more complex problems, evolving these methods to one of the standard methods for non-gradient-based optimisation.

At the same time, the underlying methodology of EAs remained the same: The solution of the optimisation problem is evolved by a randomly generated population with the help of the operator's selection, recombination and mutation successively applied over several generations to form a "fitter" population. The term "fitness" indicates how close a chromosome from the population has come to the optimum of the optimisation problem.

Since the idea of "finding the fittest" of a large population is also well-known in the domain of betting, we proposed to augment the concept of the basic EA by introducing another population, the so-called "bet population" (Reichhuber and Tomforde, 2021; Reichhuber and Tomforde, 2022). Both populations are evolved like normal EAs, but the bet population consists of individuals that have insight into the solution space. With this, they can place bets (here represented as Gaussians) to all individuals that are located in the vicinity of the bet position. These individuals benefit from the bets by increasing their fitness. On the other hand, a bet individual can win or lose depending on whether the fitness of the individuals in the vicinity of the bet location has increased or decreased from the last generation.

In this paper, we extend and refine the idea and adapt the concept to the model selection problem. The underlying concept is that this adaptation tackles two major problems of EAs: First, the diversity of the fitness of the population stays healthy, meaning the trade-off between exploring weak solutions and exploiting and refining the so-far best solutions. Second, also the diversity of the locations of the solutions is increased, which is especially interesting for solving optimisation problems with multiple local optima, as is the case in model selection.

For the evaluation of the novel approach, we have chosen the field of model selection for multi-class classification and compared our approach to other model selection algorithms, such as standard Genetic Algorithms (GAs) or Bayesian Optimisation (BO). We propose two novel exoticness metrics that are used for risk management of the betting process and show experimentally the resulting behaviour.

[a] https://orcid.org/0000-0001-8951-8962
[b] https://orcid.org/0000-0002-5825-8915

Reichhuber, S. and Tomforde, S.
Exotic Bets: Evolutionary Computing Coupled with Bet Mechanisms for Model Selection.
DOI: 10.5220/0011698500003393
In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART 2023) - Volume 2*, pages 259-267
ISBN: 978-989-758-623-1; ISSN: 2184-433X

The remainder of the paper is organised as follows. Section 2 discusses related work in the area of Model Selection based on evolutionary algorithms, also covering preliminary work on bet-based evolutionary algorithms. Section 3 explains our approach and Section 4 evaluates the methodology in comparison to other approaches with the goal to evolve suitable hyperparameters[1] for classifiers. Finally, Section 5 summarises the paper and gives an outlook to future work.

2 RELATED WORK

In the year 1988 Peter Asch and Richard Quandt analysed empirical data consisting of horse racetrack bets and found some interesting results (Asch and Quandt, 1988). In the scenario, they found evidence that the total of all bets resulting from individual win probability estimations are good approximates for the objective win probabilities. The winning probability of a single horse has no Markov property in that sense that even if its last run was not successful, it may not have met its win conditions, but normally it is a hidden champion. The trust in such hidden champions may result in so called *long-shot bets* that require the perseverance to lose money over multiple unsuccessful bet placements. The authors found tendencies that a betting bias exists in that favourites are underbet and long-shots overbet. These findings may have inspired the authors to the title 'Exotic' bets. The emergent effect of approximating the win probabilities by the sum of the subjective bets of different betting agents inspired us to simulate the process of betting. For the sake of generality, we replaced the horse riding scenario with an evolutionary environment of an arbitrary fitness function, which shall be optimised (Reichhuber and Tomforde, 2021). In this paper, we focus on Model Selection in the scenario of classification, which is based on the architecture of a Genetic Algorithm (GA).

John Holland published his idea of implementing GAs and refined it to a large extent (Holland, 1975; Booker et al., 1989). Afterwards, several extensions have been proposed. Inspired by evolution, the basic idea was to develop a natural selection process through selection, recombination and mutation. After a certain number of generations, the procedure allowed to create individuals, which are better adapted to the environment.

[1]The terms *hyperparameter* and *model* are used synonymously in this paper. The same holds for *hyperparameter optimisation* and *model selection*.

The adaptations of GAs to real numbers and modifications of different mutation or recombination strategies reducing the number of generations, hence the number of fitness function calls, turned the algorithm into a general-purpose optimisation tool for derivative-free multi-variate optimisation. Therefore, many scientists in the domain of Machine Learning (ML) made suggestions on how to adapt GAs to the task of model selection. Another question was to which environments the GAs are applicable to if scientists think about the model selection task in Machine Learning.

In the literature, one can find multiple applications of GAs applied to Model Selection. For example, (Guerbai et al., 2022) have applied one-class support vector machines in the scenario of Novelty Detection and multi-class classification or (Buchtala et al., 2005) evolved radial basis function classifiers for data mining applications. Furthermore, the authors of (Young et al., 2015) applied EAs to deep neural networks with multiple convolutional and fully-connected layers. Similar to our approach, in (de Lacerda et al., 2002) the authors used bootstrapping for modelling the true prediction error but besides this paper, most of the architectures rely on the basic GA as it is presented by Holland (Holland, 1975). However, there are multiple suggested architectures such as the Island/Course-grained Model as it is described in (Gong et al., 2015) or the cellular distributed EAs, which are analysed in (Giacobini et al., 2003; Giacobini et al., 2004) – for example regarding their takeover times, which is the duration that allows a dominant individual to occupy the whole population.

There have also been other types of EAs, besides the encoding of individuals. For example, Thomas Bäck (Bäck and Schwefel, 1993) categorised three types of EAs: *Evolution Strategies*, *Evolutionary Programming*, and *Genetic Algorithms* (Forrest, 1996; Grefenstette, 1993; Bies et al., 2006; Paterlini and Minerva, 2010; Lessmann et al., 2006; Devos et al., 2014; de Lacerda et al., 2002). First bet mechanics have been applied to EAs in the context of a bet mechanism, where an external bet population coexists and is able to place bets on the main population (Reichhuber and Tomforde, 2021; Reichhuber and Tomforde, 2022). Each individual in the bet population represents a certain bet strategy and places bets on the main population. These bets directly influence the fitness of the main population for the sake of high diversity.

Betting is not the only methodology, which steers the evolution of GAs from an external source. In (Guerbai et al., 2022), for instance, the evolutionary process is steered with the help of temperature-based control. Here, the EAs are used to optimise parame-

ters of radial basis networks for novelty detection and multi-class classification (Frazier, 2018; Snoek et al., 2012; Shahriari et al., 2015; Bodnar et al., 2020).

3 METHODOLOGY

In the following, we describe the methodology of model selection for classification with the means of bet-based evolutionary algorithms. Mathematically, we solve an optimisation problem (see Equation 1), in which we want to find the optimal hyperparameter $\theta^* \in \Theta$ among a predefined set of hyperparameters Θ used to train a multi-class classifier $f_{\theta,X_{train},Y_{train}} : \mathcal{X} \to \mathcal{Y}$, which has been trained on the training data (X_{train}, Y_{train}).

Furthermore, a test error function is given $\texttt{test-error} : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ measuring the classification error, i.e. a quantification of the difference between the true labels Y_{test} and the predicted labels $f_\theta(X_{test}) \in \mathcal{Y}$. With these ingredients, we can define the optimisation problem:

$$\theta^* = \arg\min_{\theta \in \Theta} \texttt{test-error}(X_{test}, Y_{test}, f_{\theta,X_{train},Y_{train}}(X_{test})). \quad (1)$$

Due to the size of the parameter space and the computational complexity of the test error, we iteratively find a near-to-optimal solution given a limited computation time. After a rough description of the basic procedure, the novel genetic operators are explained in detail. These are *Initialisation*, *Encoding*, and *fitness calculation*. At the end of the section, we describe the bet placement mechanism and explain the two exoticness metrics used for risk management of betting.

3.1 Procedure of External Bet-Based Evolutionary Algorithms for Model Selection

Finding the optimal D hyperparameters as the optimal hyperparameter configuration $\theta^* \in \mathbb{R}^D$ of Equation 1, we establish a bet-based Evolutionary Algorithm (BEA). By that, we refer to the minimisation of the test error of a machine learning model through evolutionary algorithms steered by an external population. In Figure 1, the basic idea of the two coexisting populations and their interaction during two iterations is visualised. The separation of both populations is taken from (Reichhuber and Tomforde, 2022) with a few adaptations (highlighted in green in Figure 1).

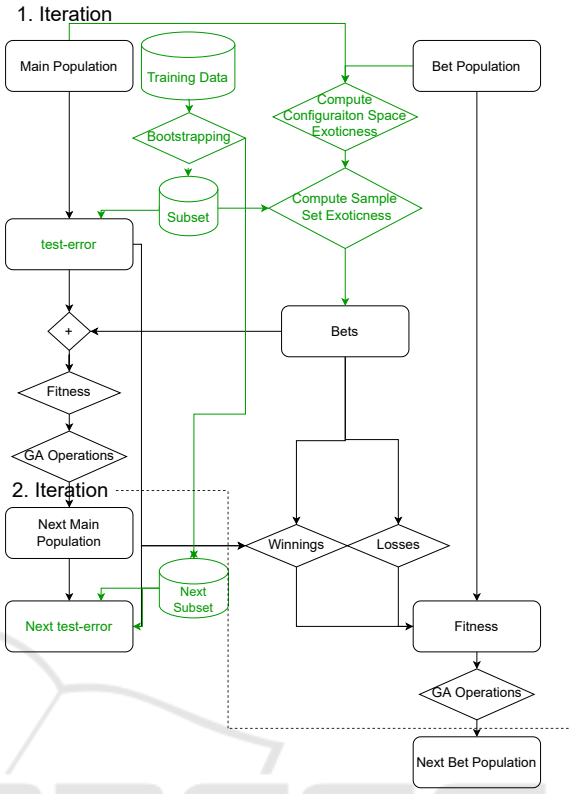


Figure 1: External bet-based Evolutionary Algorithm procedure applied to Model Selection task.

In the first iteration, a population is initialised, which we denote the *main population* $\mathcal{P}_{main}$. Additionally, we create the coexisting population denoted *bet population* $\mathcal{P}_{bet}$. Both populations are uniform-randomly drawn from the hyperparameter ranges, $\boldsymbol{\theta}_{min}, \boldsymbol{\theta}_{max} \in \mathbb{R}^D$ and the sigma step ranges $\boldsymbol{\sigma}_{max}, \boldsymbol{\sigma}_{max} \in \mathbb{R}^D$ are given *a-priori*. Afterwards, a new, small subset (X_b, Y_b) is drawn from the training data (X_{train}, Y_{train}), which we call *bootstrapped set*. Then, each individual of the main population is evaluated according to the test error measuring the difference between the test data X_{test}, Y_{test} and the predictions from a trained classifier. The latter is equipped with parameters from the individual and was trained on the bootstrapped set.

Extracting the risk of the current bet-placement, which we explain later, each individual $j = 1, \ldots, N_{bet}$ of the bet population places bets on the parameter space represented by a Gaussian with diagonal covariance, i.e. $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_j, \boldsymbol{\sigma}_j), \boldsymbol{\mu}_j, \boldsymbol{\sigma}_j \in \mathbb{R}^D$. Individuals of the main population benefit from a fitness increase when they are located in the vicinity of bets of the main population. The exact calculation can be seen in Algorithm 1.

Algorithm 1: Procedure of exotic bet-based model selection.

function EXOTIC BET-BASED MODEL SELECTION($f, \boldsymbol{\theta}_{min}, \boldsymbol{\theta}_{max}, \boldsymbol{\sigma}_{min}, \boldsymbol{\sigma}_{max}$, $X_{train}, Y_{train}, X_{test}, Y_{test}$)
▷ Normalise the data
  $\boldsymbol{X}_{train} \leftarrow \text{norm}(\boldsymbol{X}_{train}, \mu_{train}, \sigma_{train})$
  $\boldsymbol{X}_{test} \leftarrow \text{norm}(\boldsymbol{X}_{test}, \mu_{train}, \sigma_{train})$
▷ Initialise main and bet population
  $\mathcal{P}^{(0)}_{main} \leftarrow \{\boldsymbol{x}_i = (x_{i,d})^D_{d=1})\}^{N_{main}}_{i=1}$,
    $x_{i,d} \sim U([\theta_{min,d}, \theta_{max,d}])$
  $\mathcal{P}^{(0)}_{bet} \leftarrow \{\boldsymbol{b}_j = (\mu_{j,d}, \sigma_{j,d})\}^{N_{bet}}_{j=1}$,
    $\mu_{j,d} \sim U([\theta_{min,d}, \theta_{max,d}])$
    $\sigma_{j,d} \sim U([\sigma_{min,d}, \sigma_{max,d}])$
  for each $g = 1, \ldots, \#Generations$ **do**
▷ Draw new bootstrap subset
    $X_b, Y_b \leftarrow \texttt{draw}(X_{train}, Y_{train})$
▷ Calculate Sample Space Exoticness
    $SSE \leftarrow 1 - \texttt{LDOVL}(X_{train}, X_b)$
    for each $i = 1, \ldots, N_{main}$ **do**
▷ Calculate fitness of main population
      $F_{main,i} \leftarrow$
        $-\texttt{test-error}$
          $(X_{test}, Y_{test}, f_{\boldsymbol{x}_i, X_b, Y_b}(X_{test}))$
▷ Calc. Configuration Space Exoticness
      $CSE_i \leftarrow \texttt{CSE} = 1 - \mathcal{N}(\boldsymbol{x}|\mathcal{P}_{main})$
▷ Calculate the individual risk
      $risk_i \leftarrow (1 - SSE) * CSE_i$
▷ Calculate bets
      $bet_i \leftarrow risk_i * \frac{1}{N_{bet}} \sum_j^{N_{bet}} \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\sigma}_j)$
▷ Add individual bet to the fitness
      $F_{main,i} \leftarrow F_{main,i} + \texttt{bet\_influence} * bet_i$
    end for
    for each $j = 1, \ldots, N_{bet}$ **do**
▷ Calculate fitness of bet population
      $F_{bet,j} \leftarrow \frac{1}{N_{main}} \sum_{i=1}^{N_{main}} \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\sigma}_j)$
    end for
▷ Apply GA Operations:
▷ Selection, Recombination, Mutation
    $\mathcal{P}^{(g)}_{main} \leftarrow \texttt{GA-Operation}(P^{(g-1)}_{main}, F_{main})$
    $\mathcal{P}^{(g)}_{bet} \leftarrow \texttt{GA-Operation}(P^{(g-1)}_{bet}, F_{bet})$
  end for
  $\boldsymbol{\theta}^* \leftarrow \arg\max_{\boldsymbol{x} \in P^{(g)}_{main}} -\texttt{test-error}(X_{test}, Y_{test}, f_{\boldsymbol{x}, X_{train}, Y_{train}}(X_{test}))$
  return $\boldsymbol{\theta}^*$
end function

To calculate the risk of betting, the bootstrapped subset is analysed by each individual of the bet population to estimate an individual value of exoticness called *Sample Space Exoticness*. Additionally, given the deviation of the bet individual's own guess to all other guesses, (also known as *Configuration Space Exoticness*), each bet individual is able to calculate its own risk of the bet placement as follows:

$$bet_i = risk_i * \frac{1}{N_{bet}} \sum_{j}^{N_{bet}} \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\sigma}_j) \tag{2}$$

where the risk is a combination of *SSE* and *CSE*:

$$risk_i = (1 - SSE) * CSE_i. \tag{3}$$

The calculated bets will then be added to the main population's fitness. After this correctly predicted wins or losses of fitness in the main population are used as fitness for the bet population. After the evaluation of the fitness of both populations, the Genetic Operations will be applied, which are listed below and explained in the following.

1) Selection of the parents for example by using universal stochastic sampling
2) Recombination of the parents to obtain the children population called offspring, e.g. with the help of uniform crossover
3) Mutations on all individuals beside a small fraction representing the elite individuals, e.g. with one-step mutations
4) Declare the parental population and the offspring as next generation

3.2 Genetic Operators

In the following, we point out the differences between our approach compared to the EA presented in (Reichhuber and Tomforde, 2021). Following the general procedure of an EA, we describe each step in detail:

3.2.1 Initialisation

In the first step, the individuals of both populations are uniformly drawn from the configuration space $\boldsymbol{x}_i \sim U[\boldsymbol{x}_{\min}, \boldsymbol{x}_{max}] \subset \mathbb{R}^D$. Here, the limits of the hyperparameters have to be defined in advance. Extensions to a dynamically increasing feature space are possible but have not been considered further in this paper.

3.2.2 Encoding

In the main population, each individual represents a specific hyperparameter configuration. Depending on the classifier we have to choose from, which requires D hyperparameters, we encode the individual

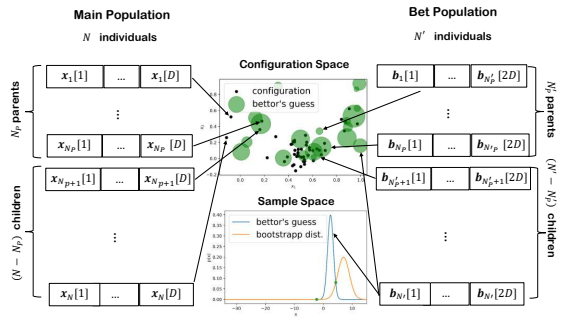


Figure 2: Encoding and interactions of the main population and the bet population.

$i \in \{1,\ldots,N_{main}\}$ of the main population of size N_{main} as:

$$\boldsymbol{x}_i = (\boldsymbol{x}_i[1],\ldots,\boldsymbol{x}_i[D])$$

On the other hand, we encode the individual of the bet population $j \in N_{bet}$ such that each bet individual consists of two components: The estimated guess of the optimum $\boldsymbol{\mu} \in \mathbb{R}^D$ and the certainty about this guess $\boldsymbol{\sigma} \in \mathbb{R}^D$. Equipped with these parameters, the bet individual is able to calculate a so-called *sample space exoticness (SSE)* and *configuration space exoticness (CSE)*, which are explained in more detail in Section 3.3.

In Figure 2, the encoding and interactions of both populations are depicted. The bet-based parameters are abbreviated as:

$$\boldsymbol{b}_j = (\boldsymbol{b}_j[1],\ldots,\boldsymbol{b}_j[2D])$$

3.2.3 Fitness Calculation: Main Population

The fitness function aims at an estimation of the test classification balanced accuracy of the classification problem. For most of the models, it is computationally infeasible to train and test each individual on the whole given sample data, i.e. the matrix inversion in finding the optimal hyperplane in support vector machines has time complexity $O(n^3)$ and cannot deal with magnitudes of millions. Therefore, we test each individual on a small subset. The problem of approximating the test error from a small subset was pointed out by (Lessmann et al., 2006). They argue that the problem of finding the correct hyperparameter for support vector machines is quite hard when no validation set is available. To make the most out of the subset, we use k-fold cross-validation as in (de Lacerda et al., 2002). Each individual's hyperparameter choice is evaluated on a small subset of the training data, which is uniformly and randomly drawn with replacement. The training data represents a standard classification problem consisting of N measurements $x_i \in X$ and their target values $y_i \in Y$. In the case of classification, we suggest the balanced accuracy value as a validation metric.

Since the computing time for the fitness calculation crucially influences the total computing time (the fitness has to be calculated in each iteration for each individual), we consider the balanced accuracy on a bootstrapped set as a computational low-cost estimation of the test error. Since the calculation of the fitness value has a sensitive share in the total computing time of the algorithm, it should take as little time as possible. To avoid overfitting to a specific bootstrap set, we repeatedly draw a novel set in each iteration. Using bootstrapping we draw $N_b << N_{train}$ samples from the training data. These evaluate the balanced accuracy score for each configuration. In addition to the negative test error the fitness is refined by the bets:

$$F_{main,i} \leftarrow F_{main,i} + \texttt{bet\_influence} * bet_i, \quad (4)$$

where $\texttt{bet\_influence} \in \mathbb{R}_0^+$ is a parameter that controls the intensity of betting.

3.2.4 Fitness Calculation: Bet Population

The fitness of the bet population is then defined as:

$$F_{bet}(\boldsymbol{b}_j) = \frac{1}{N_{main}} \sum_{i=1}^{N_{main}} \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_j,\boldsymbol{\sigma}_j) \quad (5)$$

3.3 Sample Space Exoticness and Configuration Space Exoticness

Each bet individual places bets according to risk management based on the *Configuration Space Exoticness* and the *Sample Space Exoticness*. The *Configuration Space Exoticness* expresses the deviation of a hyperparameter from the main population and the *Sample Space Exoticness* is referred to the deviation of the bootstrapped set to the training set.

3.3.1 Configuration Space Exoticness

Based on the hyperparameter configuration $\boldsymbol{x}_i$, represented by an individual of the main population, the bet individual calculates the *Configuration Space Exoticness* (CSE) value for each sample. This value represents the deviation of the hyperparameter to all other hyperparameters of the main population and is measured as the normal distribution, which can be seen in Equation 6

$$\texttt{CSE} = 1 - \mathcal{N}(\boldsymbol{x}|\mathcal{P}_{main}) = 1 - \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{\mathcal{P}_{main}},\boldsymbol{\sigma}_{\mathcal{P}_{main}}) \quad (6)$$

, where means and variances of the main population are defined as follows:

$$\boldsymbol{\mu}_{\mathcal{P}_{main}} = \frac{1}{N_{main}} \sum_{\boldsymbol{x}\in\mathcal{P}_{main}} \boldsymbol{x},$$

$$\boldsymbol{\sigma}_{\mathcal{P}_{main}} = \frac{1}{\mathcal{P}_{main}} \sum_{\boldsymbol{x} \in \mathcal{P}_{main}} (\boldsymbol{x} - \boldsymbol{\mu}_{\mathcal{P}_{main}})^2.$$

3.3.2 Sample Space Exoticness

On the other hand, for each bootstrapped subset X_b drawn at the beginning of a generation, a bet individual can calculate its *Sample Space Exoticness* value $\texttt{SSE}(X_b)$ as seen in Equation 7.

$$\texttt{SSE}(X^{(t)}) = 1 - \texttt{LDOVL}(X_{train}, X_b) \tag{7}$$

The function LDOVL is a special metric for the comparison of two distributions, which we designed for the requirements of exotic BEAs. When calculating the similarity of distributions in the configuration space, a major problem arises: On the one hand, hyperparameters can be sensitive to slight changes, on the other hand, local maxima can be far from each other. This means that the similarity measure has to be sensible to overlapping Gaussians in the vicinity of each other (like it holds for the joint probability or the overlapping coefficient (OVL) (Inman and Jr, 1989)) and simultaneously has to be capable of measuring far-distant Gaussians (like the Euclidean distance of the means or the RMSE of the means and the standard deviations) without the risk of vanishing similarity due to numerical instabilities.

Therefore, we have developed a so-called *long-distant overlapping coefficient* (*LDOVL*).

LDOVL. To measure slight differences of two distributions, which we assume in the case of bootstrapping, we base our approach on the overlapping coefficient (OVL) (Inman and Jr, 1989). Given two normal distributions $\mathcal{N}(\cdot|\mu_1,\sigma_1)$ and $\mathcal{N}(x|\mu_2,\sigma_2)$ the overlapping coefficient is calculated as following:

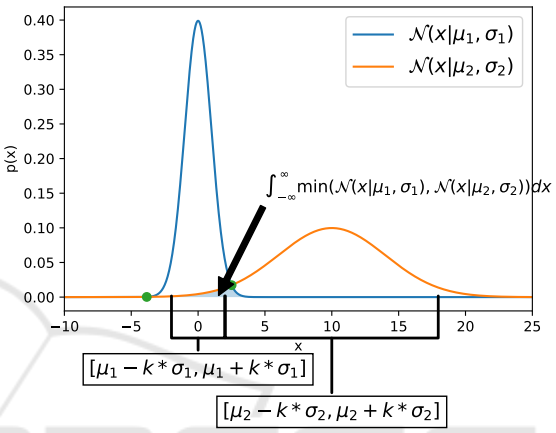
$$OVL(\mu_1,\sigma_1,\mu_2,\sigma_2) = \int_{-\infty}^{\infty} \min\left(\mathcal{N}(x|\mu_1,\sigma_1), \mathcal{N}(x|\mu_2,\sigma_2)\right) dx. \tag{8}$$

One drawback of the OVL is that it vanishes for long-distant distributions, where $|\mu_1 - \mu_2| >> \sigma_1 + \sigma_2$. To get around this problem, we introduce two areas: The *touching area* and the *long-distant area*. The *touching area* is defined for distributions where $|\mu_1 - \mu_2| \leq k(\sigma_1 + \sigma_2)$ with k is a multiplier for the standard deviations. Within this range, we refer to touching distributions and derive a normalised version of the overlapping coefficient that is not vanishingly small. The normalised version of the *OVL* is derived as follows:

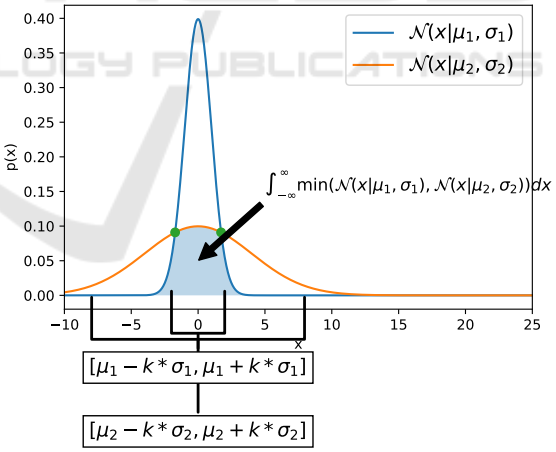
$$OVL'(\mu_1,\sigma_1,\mu_2,\sigma_2) = \frac{OVL(\mu_1,\sigma_1,\mu_2 OVL_{min}(\sigma_1,\sigma_2)}{OVL_{max}(\sigma_1,\sigma_2) - OVL_{min}(\sigma_1,\sigma_2)} \tag{9}$$

where $OVL_{min}(\sigma_1,\sigma_2) = OVL(0,\sigma_1,0+k*(\sigma_1+\sigma_2),\sigma_2)$
and $OVL_{max}(\sigma_1,\sigma_2) = OVL(0,\sigma_1,0,\sigma_2)$

In the formula above, we take the min-max normalisation based on the minimum *OVL* value (cf. Figure 3a) and the maximum *OVL* value (cf. Figure 3b) that is possible when shifting the second distribution over the other without changing the standard deviations.



(a) Minimum overlapping.



(b) Maximum overlapping.

Figure 3.

In contrast to the *touching area*, the *long-distant area* is only used for long-distant distributions, where $|\mu_1 - \mu_2| > k*(\sigma_1 + \sigma_2)$. Here, we use the proportion of the distance of both means $|\mu_1 - \mu_2|$ to the maximum feature range $f_{max} - f_{min}$ and normalise this value with the min/max normalisation.

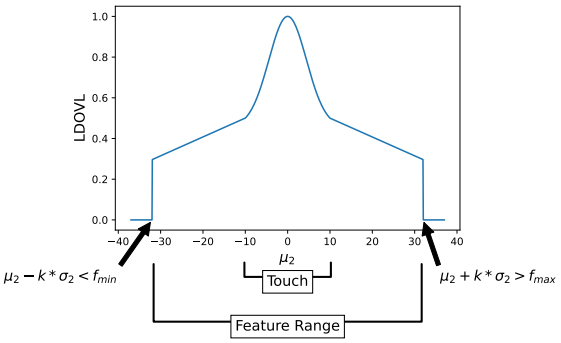


Figure 4: Long-distant-overlapping coefficient (LDOVL) of two Normal distributions $\mathcal{N}(\mu_1 = 0, \sigma_1 = 1)$ and $\mathcal{N}(\mu_2, \sigma_2 = 4)$, $\mu_2 \in [-40, 40]$. Maximum *LDOVL*-value is observed where the means of the two distributions are equal, i.e. $\mu_1 = \mu_2 = 0$.

$$LD'(\mu_1, \sigma_1, \mu_2, \sigma_2) = 1 - \frac{|\mu_1 - \mu_2| - k*(\sigma_1 + \sigma_2)}{f_{max} - f_{min} - k*(\sigma_1 + \sigma_2)} \tag{10}$$

Finally, to take care of a given feature range defined within a minimum/maximum feature value, i.e. $[f_{min}, f_{max}]$, we cap distances, where one distribution is outside the feature range to zero.

Combining all cases into one formula and introducing a weight to shift between the proportion of touch distance and long-distance, we summarise the long distant overlapping coefficient (LDOVL):

$$LDOVL(\mu_1, \sigma_1, \mu_2, \sigma_2) = \begin{cases} 0 & \text{if } I \text{ holds} \\ (1-\lambda) + \lambda * OVL'(\mu_1, \sigma_1, \mu_2, \sigma_2), & \text{if } II \text{ holds} \\ (1-\lambda) * LD'(\mu_1, \sigma_1, \mu_2, \sigma_2), & \text{if } III \text{ holds} \end{cases},$$

where

$$I : \mu_1 \notin [f_{min}, f_{max}] \text{ or } \mu_2 \notin [f_{min}, f_{max}]$$
$$II : |\mu_1 - \mu_2| \leq k*(\sigma_1 + \sigma_2)$$
$$III : k*(\sigma_1 + \sigma_2) < |\mu_1 - \mu_2| \leq f_{max} - f_{min}$$
$$\text{and } 0 \leq \lambda \leq 1 \tag{11}$$

For a better understanding, we plotted the different LDOVL values of two distributions (see Figure 4). Here, the first distribution remains constant ($\mathcal{N}(\mu_1 = 0, \sigma_1 = 1)$) and the mean value of the second distribution is shifted over the range -40 to 40 ($\mathcal{N}(\mu_2, \sigma_2 = 4)$). We set the standard deviation multiplier to 2, which means that the touching area is defined between $-k*(\sigma_1 + \sigma_2) = -2*(1+4) = -10$ and $+k*(\sigma_1 + \sigma_2) = 2*(1+4) = 10$.

One-Dimensional Box Embedding. As the metric *LDOVL* is only applicable for one dimensional point distributions, we applied a one-dimensional box embedding as visualised in Figure 5.

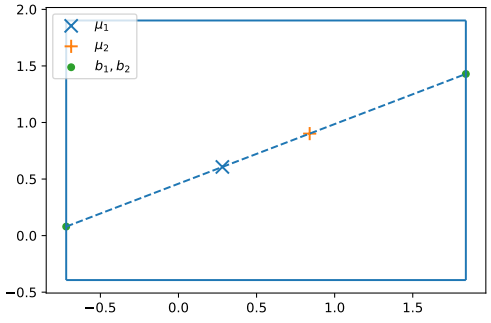


Figure 5: One-dimensional box embedding.

Table 1: Hyperparameter ranges for polynomial support vector machines (Poly SVM), radial basis function support vector machines (RBF SVM), multi-layer perceptrons (MLP), and nearest neighbours (NN). The Poly SVM is equipped with the kernel $k(\boldsymbol{x}, \boldsymbol{x}') = (\gamma\langle \boldsymbol{x}, \boldsymbol{x}' + r\rangle)^d$ and the RBF SVM with the kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma||\boldsymbol{x} - \boldsymbol{x}'||^2)$.

Classifier	Hyperparameters	Parameter grid
Poly SVM	C	$Log([e-3, e3])$
	r	$U([0, 100])$
	d	$[1, \ldots, 6]$
	γ	$Log([e-3, e3])$
RBF SVM	C	$Log([-3, 3])$
	d	$[1, \ldots, 4]$
	γ	$Log([e-3, e3])$
	pca_comp	$[5, \ldots, 200]$
MLP	*layers*	$\{(300), (150, 150), (100, 200), (200, 100), (100, 100, 100), (75, 75, 75, 75), (50, 100, 100, 50), (100, 50, 50, 100)\}$
	α	$Log([e-6, e-1])$
	batch_size	$\{2^1, 2^2, \ldots, 2^6\}$
	learning_rate	$\{'constant', 'adaptive'\}$
	learning_rate_init	$Log([e-4, e-1])$
	activation_function	$\{'identity', 'logistic', 'tanh', 'relu'\}$
NN	# neighbours	$[1, 2, 3]$
	weights	$['\texttt{uniform}', '\texttt{distance}']$
	p-norm	$U[1, 3]$

4 EXPERIMENTAL EVALUATION

We conducted a showcase on the dataset mnist (see Table 2) of using exotic bet-based Evolutionary Algorithms (exotic BEA) for the hyperparameter optimisation using the classifier from Table 1 with their most important parameters and selected a hyperparameter range with specific scale. The classifiers are taken from the python library *sklearn* (version 1.1.1).

All classifiers besides NN are preceded by a normalisation consisting of a zero-mean-unit-standard-deviation transformation and a min-max scaling to the range $[-1, 1]$.

Table 2: Properties of used dataset mnist.

Dataset	# Features	(train size, test size)
`mnist` (LeCun et al., 2010)	(28, 28) gray images	$(60k, 10k)$

Table 3: Hyperparameter Optimisation (HPO) comparison of the algorithms Genetic Algorithms (GA), Exotic bet-based Evolutionary algorithms Exotic BEA, and Bayesian Optimisation (BO) on the mnist dataset.

Classifier	HPO	balanced accuracy
Poly SVM	GA	0.864
	Exotic BEA (bet_infl. = 5e4)	**0.898**
	Exotic BEA (bet_infl. = 8e4)	0.889
	Bayesian Optimisation	0.844
RBF SVM	GA	0.925
	Exotic BEA (bet_infl. = 5e4)	**0.948**
	Exotic BEA (bet_infl. = 8e4)	0.925
	Bayesian Optimisation	0.902
NN	GA	0.910
	Exotic BEA (bet_infl. = 5e4)	0.904
	Exotic BEA (bet_infl. = 8e4)	**0.931**
	Bayesian Optimisation	0.901
MLP	GA	0.903
	Exotic BEA (bet_infl. = 5e4)	0.922
	Exotic BEA (bet_infl. = 8e4)	**0.945**
	Bayesian Optimisation	0.892

Table 4: Parameters used for the evolutionary algorithm.

Parameter	Description	Value
N_P	Population size	100
N_G	Number of generations	100
r_p	Parents ratio	50 %
r_e	Elites ratio	1 %
$p_{\boldsymbol{\mu}}$	Mutation probability for means	1 %
$p_{\boldsymbol{\sigma}}$	Mutation probability for covariances	1 %

For comparisons, we investigated the following hyperparameter optimisers:

- Genetic Algorithms
- Bayesian optimisation (Nogueira, 14)
- Exotic BEA with bet_influence = 50000
- Exotic BEA with bet_influence = 80000

For the sake of comparability, all the hyperparameter optimisers have been called the test-error function 1000 times. For example, they have been evaluated 100 main individuals over a period of 100 generations. Here, the bootstrap size was also set to 1000. The EA- parameters from Table 4 have been used for the main and the bet population.

In the case of *mnist*, the history of the maximum test-error values of the evaluation of all EA algorithms can be seen in Figure 7. This also indicates that the test-error of the bootstrap set is only a rough estimation of the test error on the whole training set. In Figure 6, the effects of the betting process on the main population's fitness over two generations is visualised.

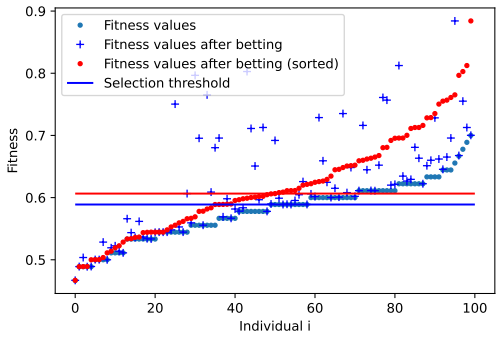


Figure 6: Influence of betting on the fitness distribution of the main population (bet_influence= 50000).

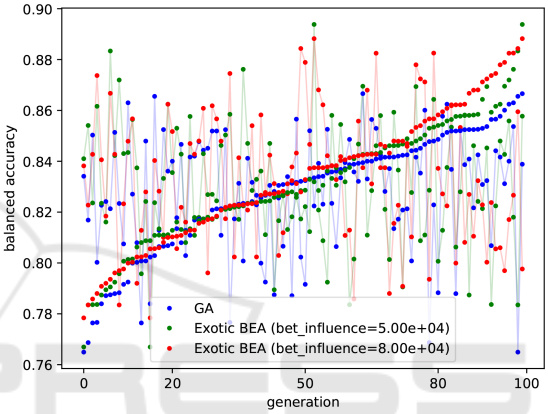


Figure 7: History of maximum fitness values of 100 generations over 10 runs of Genetic Algorithms and exotic BEA algorithms with various bet_influence parameters. The classifier Poly SVM was trained on mnist. The maximum fitness values are sorted by fitness over all generations.

5 CONCLUSION

The paper has shown a framework of how to exploit bet-based Evolutionary Algorithms to solve the Model Selection task. In detail, we have defined two new risk functions, called *sample space exoticness* and *configuration space exoticness*. The latter is used to manage the risk of bet placement. We also compared the new model selector on different experimental scenarios and compared it to normal Genetic Algorithms and Bayesian Optimisation, which showed a slight advantage in terms of balanced accuracy: For RBF SVMs evaluated on mnist the Exotic BEA acquired a balanced accuracy of 0.948 in comparison to Bayesian Optimisation (0.844) or normal Genetic Algorithms (0.925).

REFERENCES

Asch, P. and Quandt, R. E. (1988). Betting bias in 'exotic'bets. *Economics Letters*, 28(3):215–219.

Bäck, T. and Schwefel, H.-P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1):1–23.

Bies, R. R., Muldoon, M. F., Pollock, B. G., Manuck, S., Smith, G., and Sale, M. E. (2006). A genetic algorithm-based, hybrid machine learning approach to model selection. *Journal of pharmacokinetics and pharmacodynamics*, 33(2):195.

Bodnar, C., Day, B., and Lió, P. (2020). Proximal distilled evolutionary reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3283–3290.

Booker, L. B., Goldberg, D. E., and Holland, J. H. (1989). Classifier systems and genetic algorithms. *Artificial intelligence*, 40(1-3):235–282.

Buchtala, O., Klimek, M., and Sick, B. (2005). Evolutionary optimization of radial basis function classifiers for data mining applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(5):928–947.

de Lacerda, E., de Carvalho, A., and Ludermir, T. (2002). A study of cross-validation and bootstrap as objective functions for genetic algorithms. In *VII Brazilian Symposium on Neural Networks, 2002. SBRN 2002. Proceedings.*, pages 118–123.

Devos, O., Downey, G., and Duponchel, L. (2014). Simultaneous data pre-processing and svm classification model selection based on a parallel genetic algorithm applied to spectroscopic data of olive oils. *Food chemistry*, 148:124–130.

Forrest, S. (1996). Genetic algorithms. *ACM Computing Surveys (CSUR)*, 28(1):77–80.

Frazier, P. I. (2018). Bayesian optimization. In *Recent advances in optimization and modeling of contemporary problems*, pages 255–278. Informs.

Giacobini, M., Alba, E., Tettamanzi, A., and Tomassini, M. (2004). Modeling selection intensity for toroidal cellular evolutionary algorithms. In *Genetic and Evolutionary Computation Conference*, pages 1138–1149. Springer.

Giacobini, M., Tomassini, M., and Tettamanzi, A. (2003). Modeling selection intensity for linear cellular evolutionary algorithms. In *International Conference on Artificial Evolution (Evolution Artificielle)*, pages 345–356. Springer.

Gong, Y.-J., Chen, W.-N., Zhan, Z.-H., Zhang, J., Li, Y., Zhang, Q., and Li, J.-J. (2015). Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing*, 34:286–300.

Grefenstette, J. J. (1993). Genetic algorithms and machine learning. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 3–4.

Guerbai, Y., Chibani, Y., and Meraihi, Y. (2022). Techniques for selecting the optimal parameters of one-class support vector machine classifier for reduced samples. *International Journal of Applied Metaheuristic Computing (IJAMC)*, 13(1):1–15.

Holland, J. (1975). Adaptation in natural and artificial systems, univ. of mich. press. *Ann Arbor*.

Inman, H. F. and Jr, E. L. B. (1989). The overlapping coefficient as a measure of agreement between probability distributions and point estimation of the overlap of two normal densities. *Communications in Statistics - Theory and Methods*, 18(10):3851–3874.

LeCun, Y., Cortes, C., and Burges, C. (2010). Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2.

Lessmann, S., Stahlbock, R., and Crone, S. F. (2006). Genetic algorithms for support vector machine model selection. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 3063–3069. IEEE.

Nogueira, F. (2014–). Bayesian Optimization: Open source constrained global optimization tool for Python.

Paterlini, S. and Minerva, T. (2010). Regression model selection using genetic algorithms. In *Proceedings of the 11th WSEAS international conference on nural networks and 11th WSEAS international conference on evolutionary computing and 11th WSEAS international conference on Fuzzy systems*, pages 19–27. World Scientific and Engineering Academy and Society (WSEAS).

Reichhuber, S. and Tomforde, S. (2021). Bet-based evolutionary algorithms: Self-improving dynamics in offspring generation. In *ICAART (2)*, pages 1192–1199.

Reichhuber, S. and Tomforde, S. (2022). Evolving gaussian mixture models for classification. In *ICAART (3)*, pages 964–974.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. (2015). Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25.

Young, S. R., Rose, D. C., Karnowski, T. P., Lim, S.-H., and Patton, R. M. (2015). Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Proceedings of the workshop on machine learning in high-performance computing environments*, pages 1–5.