



FACULDADE DE
ECONOMIA E GESTÃO
UNIVERSIDADE DO PORTO

AutoML for Data Streams

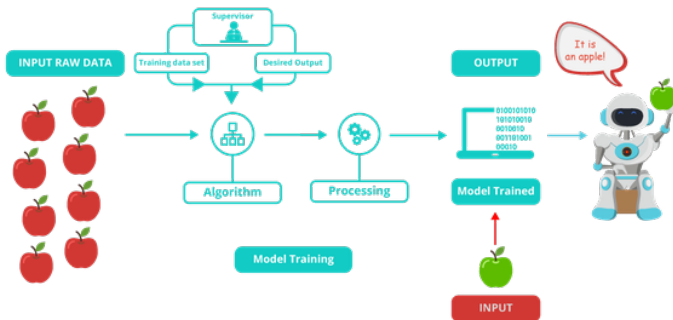
Bruno Veloso

FEP - School of Economics and Management and INESC TEC

2026/2027

Definition: Supervised Learning

- Supervised ML addresses the problem of inferring a model from labelled training data that is then used to predict data from the same underlying distribution with minimal error.

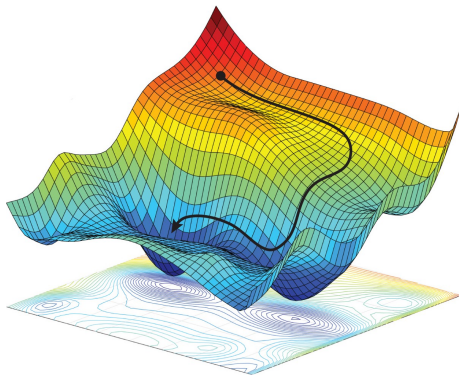


Definition: Hyper Parameter Optimization

- Machine learning (ML) algorithms are highly configurable by their hyperparameters
- These parameters often substantially influence the complexity, behavior, speed as well as other aspects of the learner
- Their values must be selected with care to achieve optimal performance.

Definition: Hyper Parameter Optimization

- Machine learning (ML) algorithms are highly configurable by their hyperparameters
- These parameters often substantially influence the complexity, behavior, speed as well as other aspects of the learner
- Their values must be selected with care to achieve optimal performance.

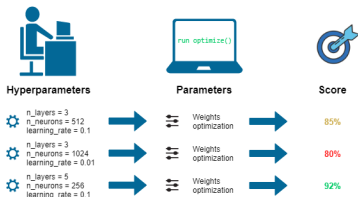


Definition: Hyper Parameter Optimization

- Human trial-and-error to select these values is time-consuming, often somewhat biased, error-prone and computationally irreproducible.
- As the mathematical formalization of hyperparameter optimization is essentially black-box optimization, often in a higher-dimensional space, this is better delegated to appropriate algorithms and machines to increase efficiency and ensure reproducibility.

Definition: Hyper Parameter Optimization

- Human trial-and-error to select these values is time-consuming, often somewhat biased, error-prone and computationally irreproducible.
- As the mathematical formalization of hyperparameter optimization is essentially black-box optimization, often in a higher-dimensional space, this is better delegated to appropriate algorithms and machines to increase efficiency and ensure reproducibility.



Real-World Applications & Economic Impact

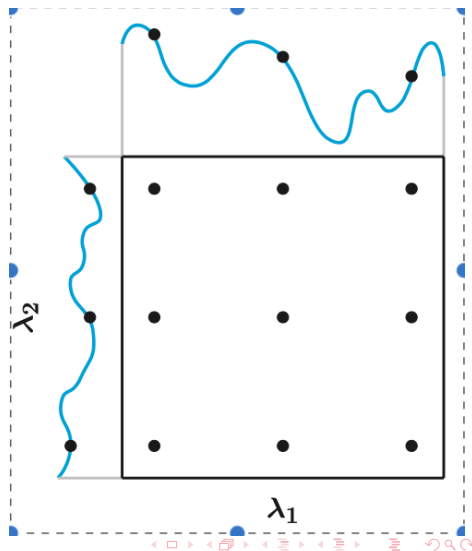
- **Financial Fraud Detection:** Models adapt to new fraud patterns in milliseconds without human intervention.
- **E-commerce & Recommendation:** Instant adaptation to sudden consumer trends (e.g., Black Friday).
- **Economic Impact:** Reduction of computational costs and elimination of downtime during model recalibration.

Background

- **Grid search**
an exhaustive search through a manually specified subset of the hyper-parameter space of a learning algorithm.
- **Random search**
replaces the exhaustive enumeration of all combinations by selecting them randomly.
- **Evolutionary optimization**
uses evolutionary algorithms to search the space of hyper-parameters for a given algorithm.
- **Bayesian optimization**
builds a probabilistic model of the function mapping from hyperparameter values to the objective evaluated on a validation set.

Grid Search

- Grid search is the process of discretizing the range of each HP and exhaustively evaluating every combination of values.
- Numeric and integer HP values are usually equidistantly spaced in their box constraints.
- The number of distinct values per HP is called the resolution of the grid.
- For categorical HPs, a subset or all possible values are considered.



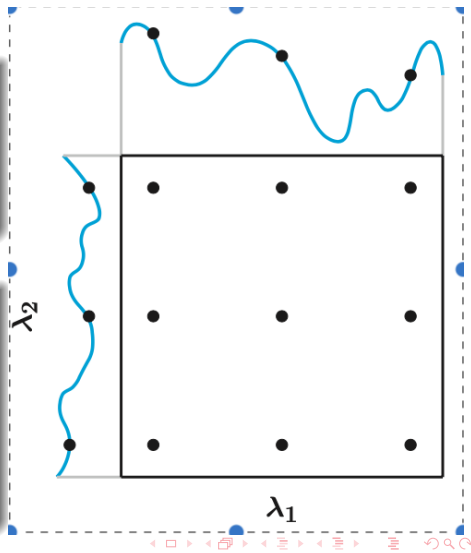
Grid Search

Pros:

- Exhaustive and guaranteed to find the best combination within the specified grid; very simple to implement.

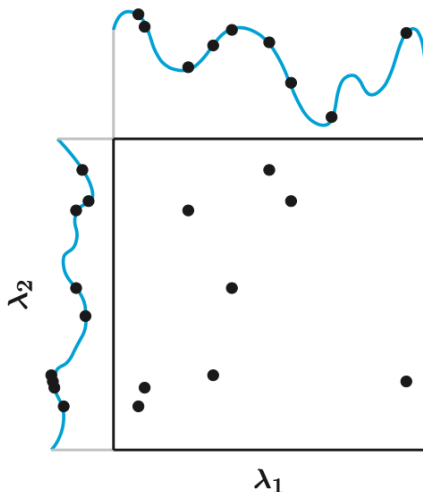
Cons:

- Suffers severely from the "curse of dimensionality" computationally unfeasible for a large number of parameters, making it poorly suited for fast data streams.



Random Search

- Random search in its simplest form, values for each HP are drawn independently of each other and from a pre-specified (often uniform) distribution, which works for (box-constrained) numeric, integer, or categorical parameters
- Due to their simplicity, both grid and random search can handle hierarchical search spaces.



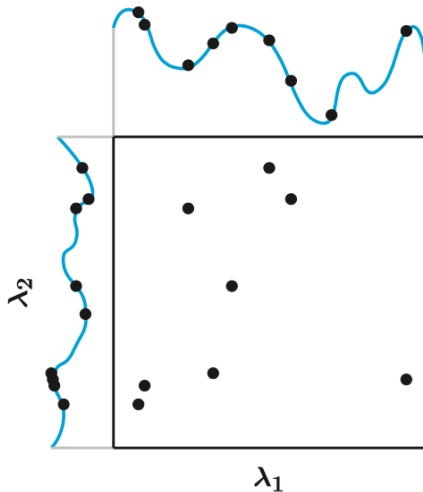
Random Search

Pros:

- Highly parallelizable and often finds better models faster than grid search in high-dimensional spaces.

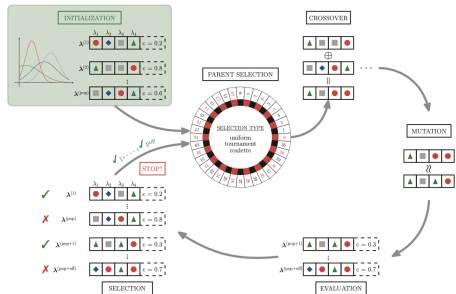
Cons:

- Blindly searches the space without learning from past evaluations; can waste computational resources evaluating poor regions.



Evolutionary Optimization

- Evolution strategies (ES) are a class of stochastic population-based optimization methods inspired by the concepts of biological evolution
- belonging to the larger class of evolutionary algorithms
- They do not require gradients, making them generally applicable in black-box settings such as HPO.



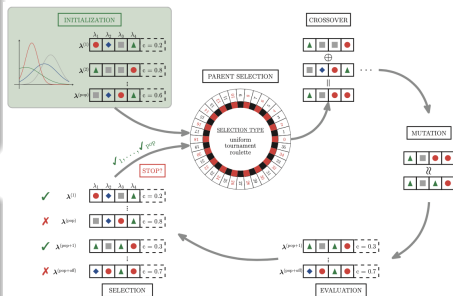
Evolutionary Optimization

Pros:

- Gradient-free and excellent at navigating complex, non-linear search spaces; easily adapts to multi-objective optimization.

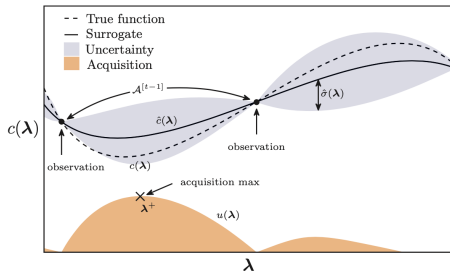
Cons:

- Can be slow to converge and requires tuning its own hyper parameters (like mutation rate or population size).



Bayesian Optimization

- BO is an iterative algorithm whose key strategy is to model based on observed performance values found in archive A via (non-linear) regression.
- This approximating model is called a surrogate model, for which a Gaussian process or a random forest are typically used.
- BO starts on an archive A filled with evaluated configurations, typically sampled randomly



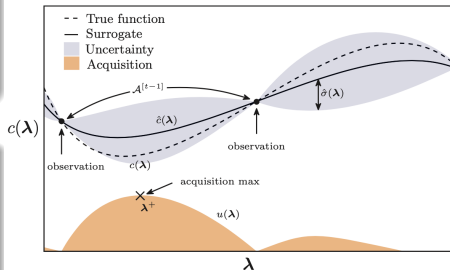
Bayesian Optimization

Pros:

- Highly sample-efficient; uses a surrogate model to make intelligent, informed guesses about where to search next.

Cons:

- The surrogate model itself (like Gaussian Processes) can become a computational bottleneck as the number of observations grows, which is tricky for continuous streams.



HPO Tools

- R:
 - MLR
 - Caret
 - H2O
- Python:
 - Optuna
 - SMAC
 - Hyperopt

HPO Idea

- **Traditional AutoML says:**

HPO Idea

- **Traditional AutoML says:** "Give me your static dataset,

HPO Idea

- **Traditional AutoML says:** "Give me your static dataset, and I'll find the best model"

HPO Idea

- **Traditional AutoML says:** "Give me your static dataset, and I'll find the best model"
- **Online AutoML says:**

HPO Idea

- **Traditional AutoML says:** "Give me your static dataset, and I'll find the best model"
- **Online AutoML says:** "Give me a continuous stream of data,

HPO Idea

- **Traditional AutoML says:** "Give me your static dataset, and I'll find the best model"
- **Online AutoML says:** "Give me a continuous stream of data, and I'll keep updating the best possible model on the fly"

Motivation

- Stream-based algorithms have several parameters that requires a tuning process
- Typically, these algorithms are tuned using a initial training step to adjust the model parameters

Relevance of problem

- **Problem:** In dynamic environments, working conditions change over time
- **Goal:** Develop online optimization methods for hyper-parameter tuning
- **Responsible design:** taking into account the ethical and technological concerns

Goal and Assumptions

- Our goal is to tune these parameters on the fly
- The set of *optimal* hyper-parameters can evolve over time.
- Maintaining a set of parameters “optimal” with the current state of the process generating data

The Research Journey



- Step 1 (2021): SPT (Nelder-Mead) - Continuous heuristic adaptation.
Published in Information Fusion
- Step 2 (2024): MESSPT (Micro-Evolutionary) - Greater exploration capacity to escape local minima.
Published in Data Mining and Knowledge Discovery
- Step 3 (2026): FSS-SPT (Swarm Intelligence) - Organic adaptation to concept drift without external detectors.
Published in Data Mining and Knowledge Discovery

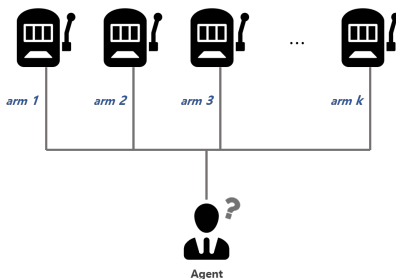
Self Hyper-Parameter Tuning (SPT)

Reinforcement Learning

- The multi-armed bandit is a problem in which a fixed limited set of resources must be allocated between competing choices in a way that maximizes their expected gain
- Classic reinforcement learning problem that exemplifies the exploration-exploitation tradeoff dilemma

Reinforcement Learning

- The multi-armed bandit is a problem in which a fixed limited set of resources must be allocated between competing choices in a way that maximizes their expected gain
- Classic reinforcement learning problem that exemplifies the exploration-exploitation tradeoff dilemma



AutoML for Dynamic Data

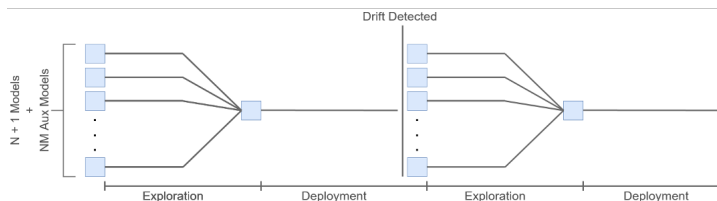
- Optimization Process

- Contains two phases: the exploration and deployment
- Exploration phase: computes several solutions to find one optimal Model
- Deployment phase: deployment of the optimal Model

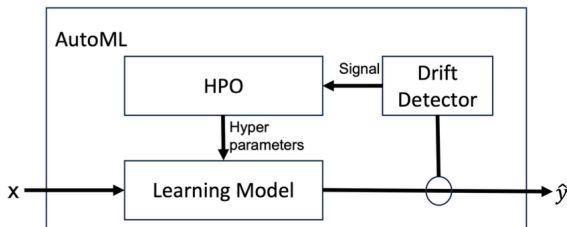
AutoML for Dynamic Data

● Optimization Process

- Contains two phases: the exploration and deployment
- Exploration phase: computes several solutions to find one optimal Model
- Deployment phase: deployment of the optimal Model

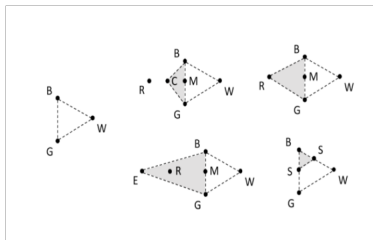


AutoML for Dynamic Data - Architecture



- Learning Model (Supervised Model)
- Drift Detector
- Hyper-Parameter Optimizer

AutoML for Dynamic Data

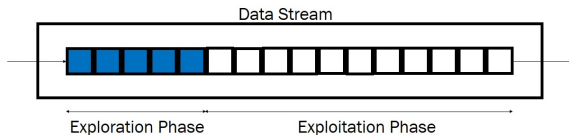


● Nelder-Mead Algorithm

- Uses a set of heuristics to optimize a loss function
- Starts with random hyperparameter values
- Auxiliary Models are computed in parallel
- Stopping criteria uses a distance metric

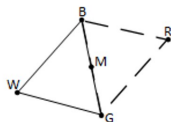
Approach

- Self Parameter Tuning Algorithm
 - Based on the Nelder-Mead optimisation algorithm
 - Adapted for data streams
- How to estimate the error of a Machine Learning algorithm?
 - Sample size estimation
- Explore different configurations
 - Parallel computing

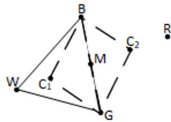


Nelder-Mead Algorithm

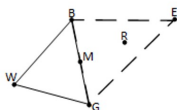
- Optimization algorithm to find a minimum of a function
- Use a simplex with k vertices, where $k = 1 + \text{number of parameters of the function to minimize}$
- Sort the different model configurations by the evaluation metric
- Apply Nelder-Mead Operators to obtain the updated parameters and substitute the Worst Model with the best configuration



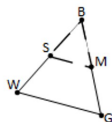
Reflection



Contraction



Expansion



Shrink

Nelder & Mead Algorithm

The vertices are ordered by the evaluation metric:

- best (B),
- good (G), which is the closest to the best vertex,
- worst (W).

For each Nelder-Mead operation, it is necessary to compute an additional set of vertexes:

- midpoint (M),
- reflection (R),
- expansion (E),
- contraction (C) and
- shrinkage (S)

and verify if the calculated vertices belong to the search space.

Nelder Mead Algorithm

- The algorithm contains: ^a
 - Training function
 - Generate auxiliary Model
 - Update Simplex
 - Main function (exploration and deployment phases)

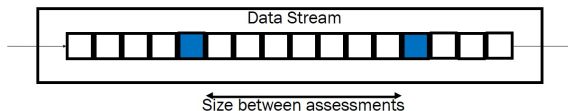
^aHyperparameter self-tuning for data streams B Veloso, J Gama, B Malheiro, J Vinagre Information Fusion 76, 2021

Algorithm 1: Nelder-Mead Algorithm

```
Input: A stream  $R^{(1)}, R^{(2)}, \dots, R^{(t)}, \dots$ , where each  $R^{(t)}$  is a partial matrix of ratings
1 Initialize the process by randomly creating configurations  $B, W, G$ 
2 for  $t \geq 1$  do
    /* Working on chunk  $V^{(t)}$  */
    3  $S^{(t)} = \text{Sample}(\bigcup_{n \leq t-1} R^{(n)})$ 
    4 if drift is detected then
        /* Exploration is enabled */
        5 Reconfigure  $G, W$  by random perturbations from  $B$ 
        6  $\text{simplex} = \{B, G, W\}$ 
        7  $\text{Train}(\text{simplex})$ 
        8  $\text{simplex} = \text{UpdateSimplex}(\text{simplex})$ 
        9  $\text{aux} = \text{UpdateAux}(\text{simplex})$ 
        10  $\text{Train}(\text{aux})$ 
        11 end
    12 else
        13 if Convergence is detected on simplex then
            /* Exploitation is enabled */
            14  $\text{simplex} = \{B\}$ 
            15  $\text{aux} = \emptyset$ 
            16 end
        17 else
            18  $\text{simplex} = \text{UpdateSimplex}(\text{simplex} \cup \text{aux})$ 
            19  $\text{aux} = \text{UpdateAux}(\text{simplex})$ 
            20 end
        21  $\text{Train}(\text{simplex} \cup \text{aux})$ 
        22 end
    23 end
24 Function  $\text{Train}(\text{models})$ :
    25 for each model  $m \in \text{models}$  do
        /* Update the current model using Gradient Descent */
        26  $\mathbf{p}_m^{(t)}, \mathbf{Q}_m^{(t)}, \hat{r}_m^{(t)} = \text{Optimize}(R^{(t)} \cup S^{(t)}, \mathbf{p}_m^{(t-1)}, \mathbf{Q}_m^{(t-1)}, \eta_m)$ 
    27 end
28 Should Function  $\text{UpdateSimplex}(\text{models})$ :
    /* Select best, good and worst model */
    29 Re-identify  $B, G, W$  from  $\text{models}$  based on the current associated losses
    30 return  $\{B, G, W\}$ 
31 Function  $\text{UpdateAux}(\text{models})$ :
    /* Contraction, shrinking, expansion, reflection */
    32 Generate auxiliary configurations  $M, E, R, S, C$  from  $\text{models}$ 
    33 return  $\{M, E, R, S, C\}$ 
```

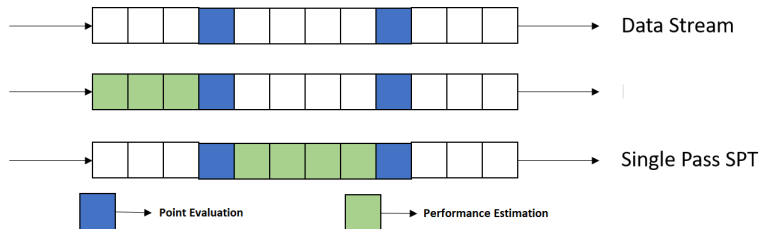
Sample Size Estimation

- How many predictions are needed for a fair performance estimation?
- How to select the appropriate moment to apply the Nelder-Mead operators?



- For each model, we need to estimate performance, for example, estimate the error of a configuration to calculate the sample size:
- $S_{size} = \frac{16\sigma^2}{M^2}$, where S_{size} is the sample size, σ is the standard deviation of the metric and M is the confidence level

Single Pass



Single Pass SPT:

- It uses the next bucket of instances (highlighted with green) to apply the Nelder-mead operators. Requires a single passage over the stream to apply the operators.

Beyond SPT: The Need for Exploration

- **The Limitation of Nelder-Mead:** Excellent at local optimization (exploitation) but vulnerable to getting stuck in local minima in complex search spaces. It was designed for continuous search space.
- **The Challenge of Concept Drift:** SPT relied on random perturbations when drift was detected.
- **The Solution:** An Evolutionary approach (MESSPT) to ensure continuous diversity within the model population.

Micro Evolutionary Self Hyper-Parameter Tuning (MESSPT)

Micro Evolutionary Algorithm

- The algorithm contains: ^a
 - Training function
 - Generate Auxiliary Model
 - Update Simplex
 - Main function (exploration and deployment phases)

^aImproving hyper-parameter self-tuning for data streams by adapting an evolutionary approach, A. Moya, B. Veloso, J. Gama, S. Ventura; DAMI. 38, 2024

Algorithm 1 Learning process of MESSPT

Input c_pop , $offsprings$, F , CR , $best$

```
1: procedure LEARN ONE( $x, y$ )           ▷  $x$  is the input and  $y$  the label
2:   if no convergence (exploration phase) then
3:     for  $ind \in c\_pop$  do
4:        $ind \leftarrow update\_metric(ind, x, y)$            ▷ testing
5:        $ind \leftarrow update\_model(ind, x, y)$            ▷ training
6:     end for
7:     for  $ind \in offsprings$  do
8:        $ind \leftarrow update\_metric(ind, x, y)$ 
9:        $ind \leftarrow update\_model(ind, x, y)$ 
10:    end for
11:    Update model and metric from offsprings
12:     $best \leftarrow update\_best(c\_pop, offsprings)$ 
13:    if grace period then
14:       $F \leftarrow update\_F()$            ▷ Reducing F
15:       $CR \leftarrow update\_CR()$        ▷ Increasing CR
16:       $c\_pop \leftarrow generate\_next\_pop(best)$ 
17:      check convergence
18:      if no convergence then
19:         $offsprings \leftarrow generate\_offspring\_pop(c\_pop)$ 
20:      end if
21:    end if
22:  else           ▷ (deployment phase)
23:    check concept drift
24:    if concept drift is detected then
25:      Restart
26:    else
27:       $best \leftarrow update\_metric(best, x, y)$ 
28:       $best \leftarrow update\_model(best, x, y)$ 
29:    end if
30:  end if
31: end procedure
```

Micro Evolutionary Algorithm

Micro-Evolutionary approach

- **4 individuals per population.**
- **Elitist** approach (only best saved per generation).
- **Adapting** operators to achieve **fast convergence.**

Codification of each individual

Each individual wraps a ML model (trainable), the metric updated and **the configuration of hyperparameters.**

Selection criterion

- Current best kept.
- 3 new individuals consists of: ML model of the current best, current metric and a random configuration (explore).

Micro Evolutionary Algorithm - Operators

- The algorithm contains:
 - Generate Next Population
 - Generate Offspring Population

Algorithm 2 Generate next population

Input c_pop , $best$

```
1: procedure GENERATE_NEXT_POP( $best$ )      ▷ Generate next population
2:    $c\_pop \leftarrow \{\}$ 
3:    $model\_best, conf\_best, metric\_best \leftarrow extractFromInd(best)$ 
4:    $c\_pop.add(best)$                       ▷ Add the best
5:   for  $i \in 1, \dots, 3$  do                ▷ Add random
6:      $conf\_r \leftarrow randConf()$ 
7:      $new\_ind \leftarrow create\_ind(model\_best, conf\_r, metric\_best)$ 
8:      $c\_pop.add(new\_ind)$ 
9:   end for
10: end procedure
```

Algorithm 3 Generate offspring population

Input c_pop , $offspring$, $best$

```
1: procedure GENERATE_OFFSPRING_POP( $c\_pop$ ) ▷ Generate new offspring
   pop
2:    $model\_best, conf\_best, metric\_best \leftarrow extractFromInd(best)$ 
3:   for  $ind \in c\_pop$  do
4:      $model\_ind, conf\_ind, _ \leftarrow extractFromInd(ind)$ 
5:      $conf\_r1, conf\_r2 \leftarrow selectConfFromRand(c\_pop)$ 
6:      $conf\_child \leftarrow mutOperator(conf\_best, conf\_r1, conf\_r2)$ 
7:      $conf\_child \leftarrow cross(conf\_ind, conf\_child)$ 
8:      $new\_child \leftarrow create\_ind(model\_best, conf\_child, metric\_best)$ 
9:      $offspring.add(new\_child)$ 
10:  end for
11: end procedure
```

Our DE proposal

Mutation and cross operators

- **Mutation:** $u_i^G = x_{best}^G + F(x_{r_1} - x_{r_2})$

- **Cross:**

$$x_{ij}^{G+1} = \begin{cases} u_{ij}^G, & \text{if } r \leq CR \text{ or } l = j \\ x_{ij}^G & \text{otherwise} \end{cases} \quad (1)$$

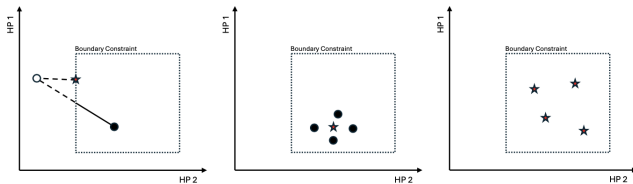
Creating a new offspring population (grace period)

- Mutation and cross operators acts over the configuration of hyperparameters (Grace period).
- Each individual consists of: ML model of the current best individual, the metric updated and the mutated and crossed hyperparameter configuration.

Boundary Constraint

Boundary Constraint

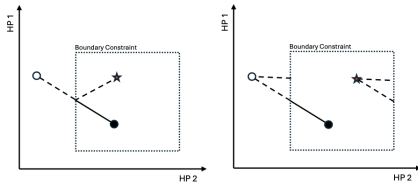
Adaptive boundary constraint-handling scheme for constrained optimization, E Juarez-Castillo, HG Acosta-Mesa, E Mezura-Montes Soft Computing, 2019



(a) Boundary

(b) Centroid

(c) Random



(d) Reflection

(e) Wrapper

The Bottleneck of External Drift Detectors

- **The Previous Paradigm:** Both MESSPT and SPT react only after a drift alarm is triggered (e.g., by ADWIN, DDM).
- **The Problem:** False positives trigger unnecessary restarts; False negatives keep sub-optimal models running.
- **The New Vision:** What if the optimization algorithm could naturally contract and expand, sensing the environment without relying on an external detector...

Fish Search School

The Proposal: FSS-SPT

Fish School Search - Stream Parameter Tuning (FSS-SPT)

A novel adaptation of the Fish School Search (FSS) algorithm designed explicitly for the dynamic context of data streams.

Key Advantages:

- **Single Pass:** Processes data sequentially without storing large batches.
- **No External Detectors:** Does not rely on drift detectors to trigger re-optimization.
- **Dynamic Adaptation:** Automatically switches between exploration and exploitation modes.
- **Swarm Intelligence:** Handles multi-local minimum problems effectively.

FSS Core Concepts

Inspired by the behavior of fish schools searching for food.

- **Agents (Fish):** Represent candidate solutions (hyper-parameter sets).
- **Weight:** Represents the success of a fish (fitness/accuracy).
- **Operators:**
 - 1 **Individual Movement:** Local search around current position.
 - 2 **Feeding:** Updates weight based on fitness improvement.
 - 3 **Collective Instinctive Movement:** Whole school moves towards favorable directions.
 - 4 **Collective Volitive Movement:** Controls expansion (exploration) or contraction (exploitation) of the school.

Adapting FSS for Data Streams

Evaluation Strategy:

- Uses non-overlapping evaluation windows (e.g., every 1000 samples).
- Optimization minimizes the error of the classifier/regressor.

The Two-Phase Window Process:

Phase 1 (First 500 samples)

- Assess fitness.
- Execute *Individual Movement*.

Phase 2 (Next 500 samples)

- Assess fitness again.
- Execute *Feeding, Instinctive, and Volitive* movements.

Exploration vs. Exploitation

The critical innovation is the **Collective Volitive Movement**:

- **School Weight Increases:**

- Implies the school is finding food (good solutions).
- Action: The school **contracts** (Exploitation).

- **School Weight Decreases:**

- Implies the current region is poor (e.g., after a concept drift).
- Action: The school **expands** (Exploration).

Result: The algorithm adapts to concept drift automatically without needing an alarm.

1. Individual Movement Operator (Local Search)

Objective: Perform a local search around the fish's current position to find an immediate better solution.

- Each fish explores a new candidate position (n) by adding a random displacement to its current position (x).
- The magnitude of the displacement is controlled by the current **individual step size** ($Step_{ind}$).
- **Movement Logic:** A fish only moves to the new position if the new position's fitness ($f(n)$) is better than its current fitness ($f(x)$).

Movement Formula

$$n_{ik} = x_{ik} + \text{rand}(-1, 1) \cdot Step_{ind}$$

- n_{ik} : Candidate position of fish i in dimension k .
- x_{ik} : Current position of fish i in dimension k .
- $\text{rand}(-1, 1)$: A random number between -1 and 1 .

2. Feeding Operator (Weight Update)

Objective: Update the fish's weight based on its individual performance improvement. This drives the collective movements.

- After the individual movement, the fish calculates its fitness variance, $\Delta f = f(n) - f(x)$.
- The weight w_i of fish i is updated if its fitness improved (i.e., $\Delta f > 0$, where fitness minimization is the goal, or Δf is scaled to represent improvement).
- The weight update is normalized by the maximum fitness variance, $\max(\Delta f)$, observed across the population.

Weight Update Formula

$$w_i(t + 1) = w_i(t) + \frac{\Delta f_i}{\max(\Delta f)}$$

- $w_i(t + 1)$: New weight of fish i .
- Δf_i : Fitness variance (improvement) for fish i .

3. Collective Instinctive Movement (School Direction)

Objective: Move the entire school towards the most favorable direction based on individual successes.

- A **resultant direction vector** ($I(t)$) is computed by summing the weighted displacement ($\Delta x_i \Delta f_i$) of all fish.
- This movement ensures that fish that found better solutions pull the entire school in their direction, favoring regions of the search space where recent improvements were found.
- All fish in the school move in this single collective direction.

Resultant Direction Vector $I(t)$

$$I(t) = \frac{\sum \Delta x_i \Delta f_i}{\sum \Delta f_i}$$

Position Update

$$x_i(t+1) = x_i(t) + I(t)$$

4. Collective Volitive Movement (Exploration/Exploitation)

Objective: Dynamically adjust the school's concentration (inwards/outwards) to balance exploitation and exploration, adapting to concept drift without external detectors.

- 1 **Calculate Barycenter ($B(t)$):** The weighted average position of the entire school.

$$B(t) = \frac{\sum x_i w_i(t)}{\sum w_i(t)}$$

- 2 **Check Total Weight:** Sum of all fish weights, $W(t) = \sum w_i(t)$.

4. Collective Volitive Movement (Exploration/Exploitation)

Objective: Dynamically adjust the school's concentration (inwards/outwards) to balance exploitation and exploration, adapting to concept drift without external detectors.

- 3 **Exploitation (Inward Move):** If the total weight has **increased** (the school is improving), the fish move **inwards** towards the barycenter to exploit the promising region.

$$x(t+1) = x(t) - \text{step}_{vol} \cdot \text{rand}(0, 1) \cdot \frac{x(t) - B(t)}{\text{dist}(x(t), B(t))}$$

- 4 **Exploration (Outward Move):** If the total weight has **decreased** (stuck or concept drift), the fish move **outwards** away from the barycenter to explore a new region.

$$x(t+1) = x(t) + \text{step}_{vol} \cdot \text{rand}(0, 1) \cdot \frac{x(t) - B(t)}{\text{dist}(x(t), B(t))}$$

Fish Search School Algorithm

- The algorithm contains:
 - Training function
 - Generate auxiliary Model
 - Update Fish
 - Main function (call operators functions)

Algorithm 1 Pseudocode of the FSS-SPT algorithm.

```
1: initialize all fish randomly
2: for eval window = 500, 1000, 1500, 2000, ..., N do
3:   if eval window % 1000 != 0 then
4:     for each fish do
5:       Evaluate fish fitness function
6:       Execute individual movement
7:     end for
8:   end if
9:   if eval window % 1000 == 0 then
10:    for each fish do
11:      Evaluate fish fitness function
12:      Execute feeding operator
13:      Execute collective, instinctive movement
14:    end for
15:    Calculate barycenter
16:    for each fish do
17:      Execute the collective volitive movement
18:    end for
19:    Update step size
20:   end if
21: end for
```

Fish Search School

Fish Search approach

- **50 individuals (fish) in the search school.**
- **Adapting** operators to achieve **fast convergence.**
- **Continuous** exploration and exploitation mode

Codification of each individual

Each individual wraps a ML model (trainable), the metric updated and **the configuration of hyperparameters.**



Our Fish School proposal

Operators

- **Step Size:** $step_{ind}(t+1) = step_{ind}(t) - \frac{step_{indInitial} - step_{indFinal}}{iterations}$
- **Individual Movement:** $n_{ik} = x_{ik} + rand(-1, 1)currentStep_{ind}$

- **Feeding:**

$$\Delta f = f(\vec{n}) - f(\vec{x})$$

$$\Delta \vec{x} = \vec{n} - \vec{x}$$

$$w_i(t+1) = w_i(t) + \frac{\Delta f_i}{\max(\Delta f)}$$

- **Collective Instinctive Movement:**

$$\vec{l}(t) = \frac{\sum \Delta \vec{x}_i \Delta f_i}{\sum \Delta f_i}$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{l}(t)$$

- **Collective Volitive Movement:**

$$\vec{B}(t) = \frac{\sum \vec{x}_i w_i(t)}{\sum w_i(t)}$$

$$\vec{x}(t+1) = \vec{x}(t) (-or+) step_{vol} rand(0, 1) \frac{\vec{x}(t) - \vec{B}(t)}{dist(\vec{x}(t), \vec{B}(t))}$$

Research Question

How to perform hyper-parameter tuning on a stream-based algorithm?

- Does the exploitation phase of the HPO converge to high-quality solutions?
- When drift is present in the data, how well does the HPO adapt to changes?
- Does the boundary constraint have an impact on the performance?

Evaluation

- Baselines:
 - SPT
 - MESSPT
 - FSS
- Data:
 - Synthetic and real-world datasets
 - Temporal order
- Protocol:
 - Predictive sequential (prequential) evaluation protocol for data streams (J.Gama 2013)
- Metric:
 - Root-mean-square error (RMSE)
 - Accuracy

Datasets

TASK: Classification

Dataset	Type	Drift	Nr. Instances	Nr. Features
ENRON ¹	Real	-	1702	1000
NOMAO ¹	Real	-	34465	120
RandomRBF ²	Synthetic	-	20000	4
Agrawal ²	Synthetic	-	20000	9
Hyperplane ²	Synthetic	Yes	20000	2
SEA Drift ²	Synthetic	Yes	20000	3

TASK: Regression

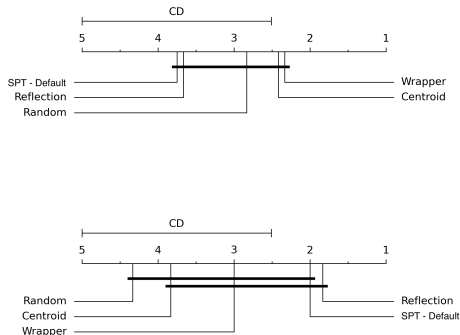
Dataset	Type	Drift	Nr. Instances	Nr. Features
Tetuan ³	Real	-	52417	6
Metro ⁴	Real	-	48204	9
2DPlanes ²	Synthetic	-	20000	10
MV ²	Synthetic	-	20000	10
Friedman Drift ²	Synthetic	Yes	20000	10
Friedman ²	Synthetic	-	20000	10

SPT HPO

TASK: Classification – Evaluation Metric: Accuracy					
Dataset	Boundary	Reflection	Wrapper	Random	Centroid
ENRON	0.919	0.919	0.927	0.919	0.919
NOMAO	0.894	0.905	0.914	0.913	0.935
RandomRBF	0.522	0.519	0.518	0.517	0.525
Agrawal	0.987	0.986	0.987	0.987	0.984
Hyperplane	0.952	0.953	0.954	0.953	0.953
SEA_Drift	0.912	0.935	0.957	0.962	<u>0.961</u>

TASK: Regression – Evaluation Metric: RMSE					
Dataset	Boundary	Reflection	Wrapper	Random	Centroid
Tetuan	4615.590	4567.231	4570.758	4682.291	4931.213
Metro	2424.115	3208.708	3262.022	3303.232	<u>2967.850</u>
Friedman	2.190	2.187	2.193	2.227	2.203
MV	6.199	6.057	6.094	6.103	7.036
Friedman Drift	2.652	2.655	2.655	2.660	2.724
Planes2D	1.094	1.094	<u>1.095</u>	1.096	1.094

Table 2: Average Performance (Best scores highlighted with **bold and second best score is underlined)**



MESSPT HPO

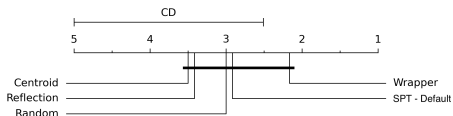
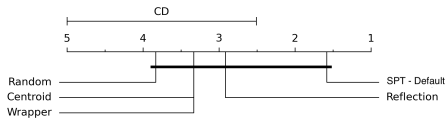
TASK: Classification – Evaluation Metric: Accuracy

Dataset	Boundary	Reflection	Wrapper	Random	Centroid
ENRON	0.918	0.918	0.918	0.918	0.918
NOMAO	0.945	0.945	0.937	0.919	0.930
RandomRBF	0.529	0.529	0.529	0.517	0.519
Agrawal	0.988	0.983	0.985	0.987	<u>0.986</u>
Hyperplane	0.952	0.952	0.952	0.952	0.952
SEA_Drift	0.961	0.954	0.954	0.954	<u>0.955</u>

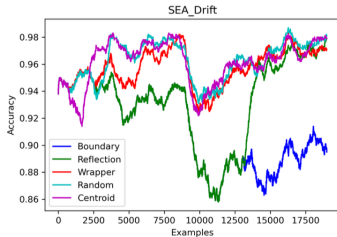
TASK: Regression – Evaluation Metric: RMSE

Dataset	Boundary	Reflection	Wrapper	Random	Centroid
Tetuan	4514.984	4514.984	4559.847	4514.019	4648.248
Metro	3995.966	4016.770	<u>3272.120</u>	3782.606	3084.842
Friedman	2.217	2.281	2.204	2.231	2.233
MV	<u>5.594</u>	5.591	5.594	5.653	5.820
Friedman Drift	2.716	2.735	<u>2.641</u>	2.704	2.629
Planes2D	1.095	1.095	1.095	1.095	<u>1.096</u>

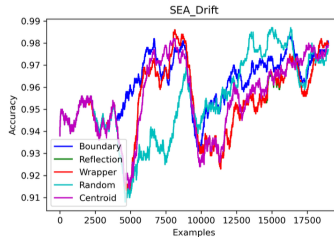
Table 5: Average Performance (Best scores highlighted with **bold and second best score is underlined)**



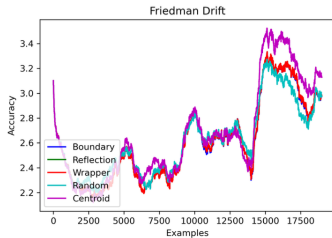
Drift



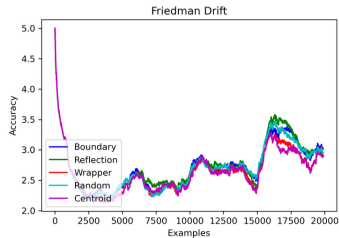
(a) SSPT - Classification Task - SEA



(b) MESSPT - Classification Task - SEA



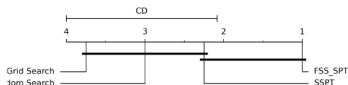
(c) SSPT - Regression Task - Friedman



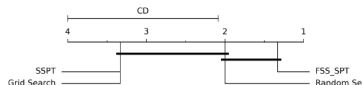
(d) MESSPT - Regression Task - Friedman

FSS SPT HPO

TASK: Classification – Evaluation Metric: Accuracy				
Dataset	Random Search	Grid Search	SSPT	FSS_SPT
ENRON	<u>0.920</u>	0.919	0.919	0.928
NOMAO	0.882	0.715	<u>0.905</u>	0.938
RandomRBF	0.424	0.416	<u>0.524</u>	0.537
Agrawal	0.775	0.682	<u>0.959</u>	0.984
Hyperplane	0.867	0.868	<u>0.952</u>	0.954
SEA_Drift	0.868	0.864	<u>0.949</u>	0.967
TASK: Regression – Evaluation Metric: RMSE				
Dataset	Random Search	Grid Search	SSPT	FSS_SPT
Tetuan	4387.729	4331.839	4717.855	<u>4347.961</u>
Metro	1878.074	1960.757	2426.537	<u>1891.569</u>
Friedman	<u>2.122</u>	2.798	2.208	2.070
MV	<u>3.976</u>	19.502	6.201	3.936
Friedman Drift	<u>2.554</u>	3.318	2.642	2.493
Planes2D	<u>1.091</u>	2.519	1.093	1.090

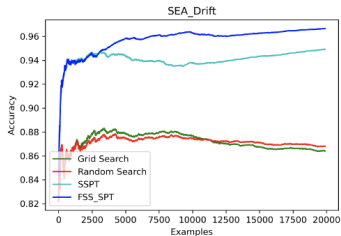


(a) TASK: Classification

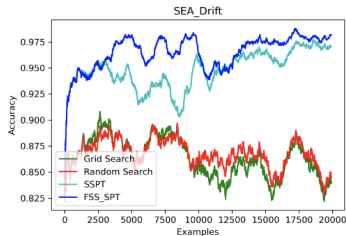


(b) TASK: Regression

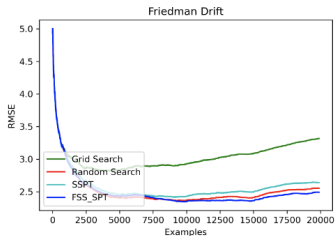
Drift



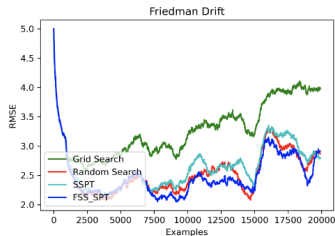
(a) SEA Drift - Average



(b) SEA Drift - Window



(c) Friedman Drift - Average



(d) Friedman Drift - Window

Conclusions

- **Dynamic Adaptation:** Automatic selection of hyperparameter settings has a substantial impact on the outcomes of stream-based algorithms. Our approaches successfully maintain optimal parameters on the fly as working conditions change.
- **Superior Performance:** The proposed dynamic approaches achieve the lowest prediction error. Methods like the Micro Evolutionary (MESSPT) and Fish Search School (FSS) algorithms consistently outperform static baselines across both classification and regression tasks.

Conclusions

- **Resilience to Concept Drift:** The dual-phase architecture (exploration and deployment) effectively recovers model performance when concept drift is detected, as demonstrated in datasets like SEA Drift and Friedman Drift.
- **Effective Boundary Handling:** Implementing structured boundary constraints (such as Wrapper and Reflection) ensures valid hyper parameter configurations are maintained without degrading the learning process.

Future Research Lines

- **Complex Search Spaces:** Apply the proposed method to more complex hyper parameter spaces, such as hierarchical configurations or deep learning network architectures.
- **Multi-Objective Tuning:** Expand the optimization process to simultaneously balance predictive performance (Accuracy and RMSE) with strict stream constraints like memory consumption, processing time or carbon consumption.
- **Advanced Drift Integration:** Explore tighter coupling between the hyper parameter optimizers and the concept drift detectors to enable proactive tuning before performance drops significantly.

Future Research Lines

- **Distributed Environments:** Adapt the population-based algorithms to run across distributed or decentralized nodes for ultra-high-velocity data streams.
- **Hyper parameter Importance:** Implement techniques to dynamically evaluate the relative importance of individual hyper parameters, allowing the algorithm to prune the search space and accelerate convergence.

Future Research Lines

- **Green AI:** By optimizing responsively the optimization methods, we can reduce the carbon footprint compared to the deep retraining cycles of static models.
- **Fairness & Bias Mitigation:** Data stream demographics change over time. Continuous re-parameterization prevents the model from consolidating obsolete historical biases (temporal bias).

Publications

- Veloso, B., Gama, J., & Malheiro, B. (2018). Self hyper-parameter tuning for data streams. In International conference on discovery science.
- Bahri, M., Veloso, B., Bifet, A., & Gama, J. (2020). AutoML for stream k-nearest neighbors classification. In 2020 IEEE International Conference on Big Data.
- Veloso, B., Caroprese, L., König, M., Teixeira, S., Manco, G., Hoos, H. H., & Gama, J. (2021). Hyper-parameter optimization for latent spaces. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases.

Publications

- Veloso, B., Gama, J., Malheiro, B., & Vinagre, J. (2021). Hyperparameter self-tuning for data streams. Information Fusion.
- Moya, A. R., Veloso, B., Gama, J., & Ventura, S. (2024). Improving hyper-parameter self-tuning for data streams by adapting an evolutionary approach. Data Mining and Knowledge Discovery.
- Veloso, B., Neto, H. A., Buarque, F., & Gama, J. (2026). Fish swarm parameter self-tuning for data streams. Data Mining and Knowledge Discovery.

Thank You!



- Bruno Veloso
- bveloso@fep.up.pt