Navigating Complex Machine Learning Challenges in Streaming Data

ECML Tutorial 2024

Heitor Murilo Gomes[1*], Marco Heyden[2]
Maroua Bahri[3,4]
https://heymarco.github.io/ecml24-streamingchallenges/

* Corresponding author: heitor.gomes@vuw.ac.nz

https://capymoa.org/

[1] Victoria University of Wellington, New Zealand, [2] KIT, Germany, [3] INRIA Paris, France,
[4] Sorbonne Université, France

Preprocessing

Reduction techniques

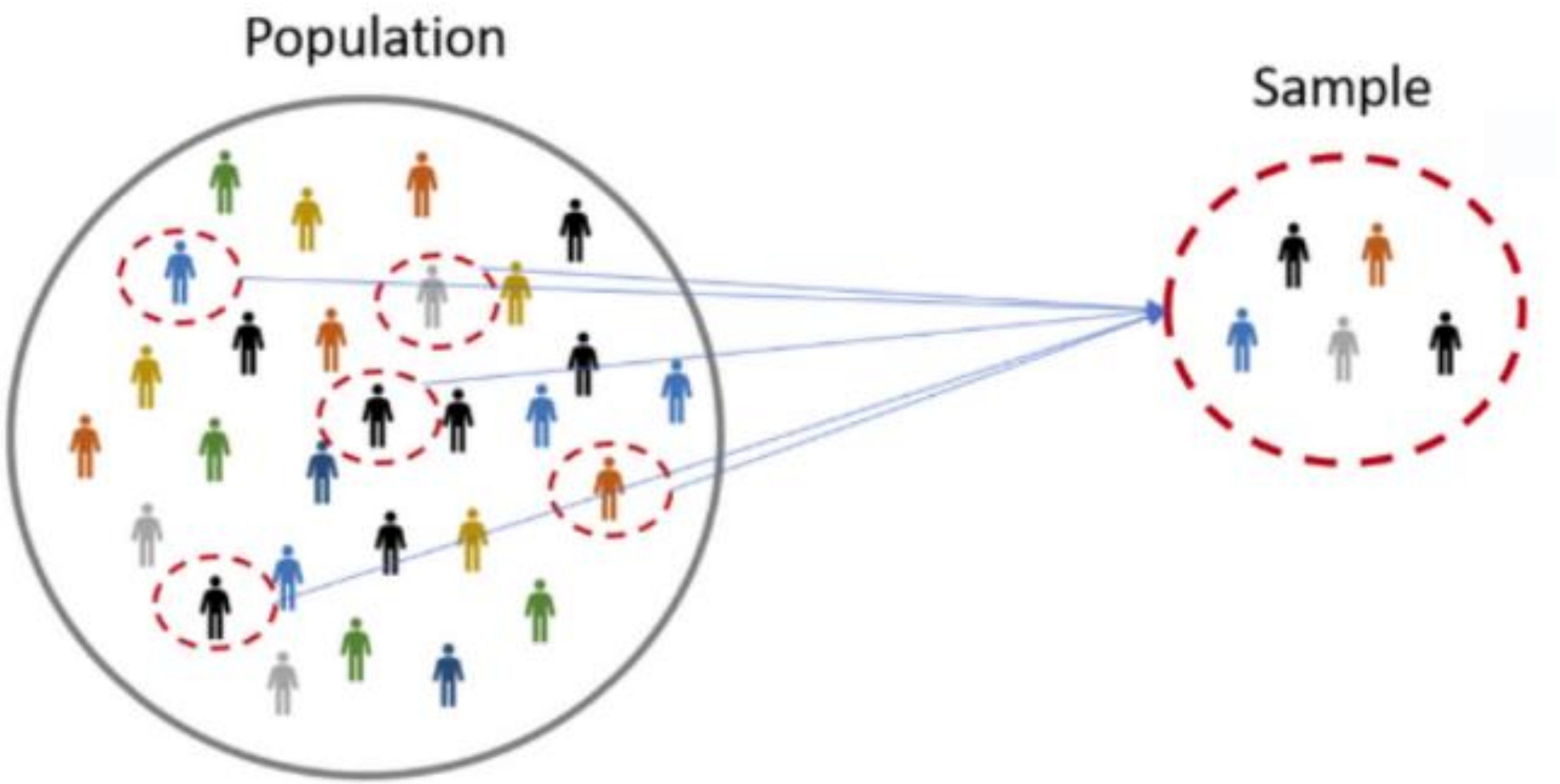
Summarization Techniques

- To address resource constraints (memory and time), we use summarization techniques
  - Dimension reduction
  - Sketches
  - Sampling
  - …

Bahri, M., Bifet, A., Maniu, S. and Gomes, H.M.,"Survey on feature transformation techniques for data streams." *IJCAI*, 2021.

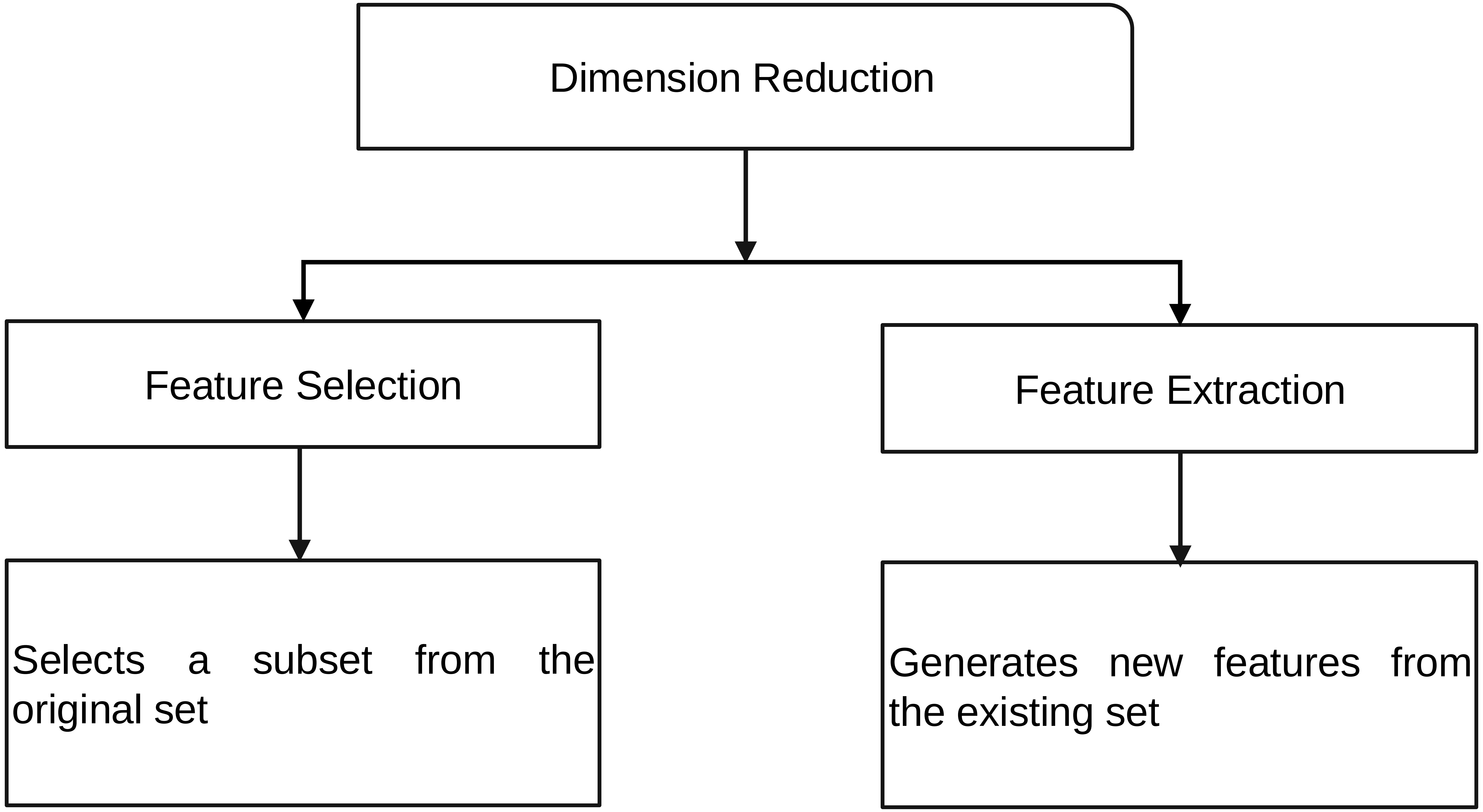
Sampling

- Maintain some “representative” instances and store synopsis from the stream in memory

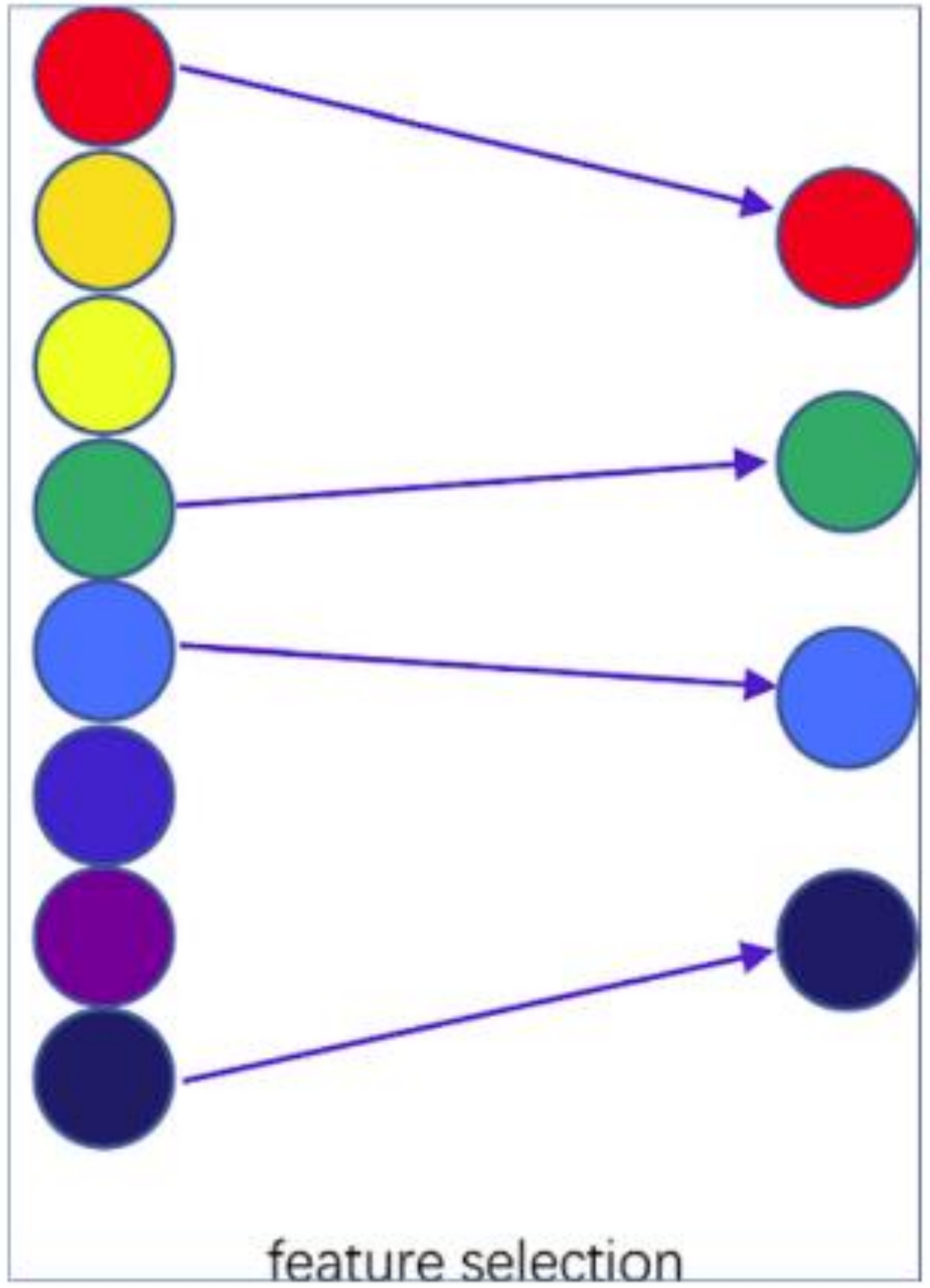


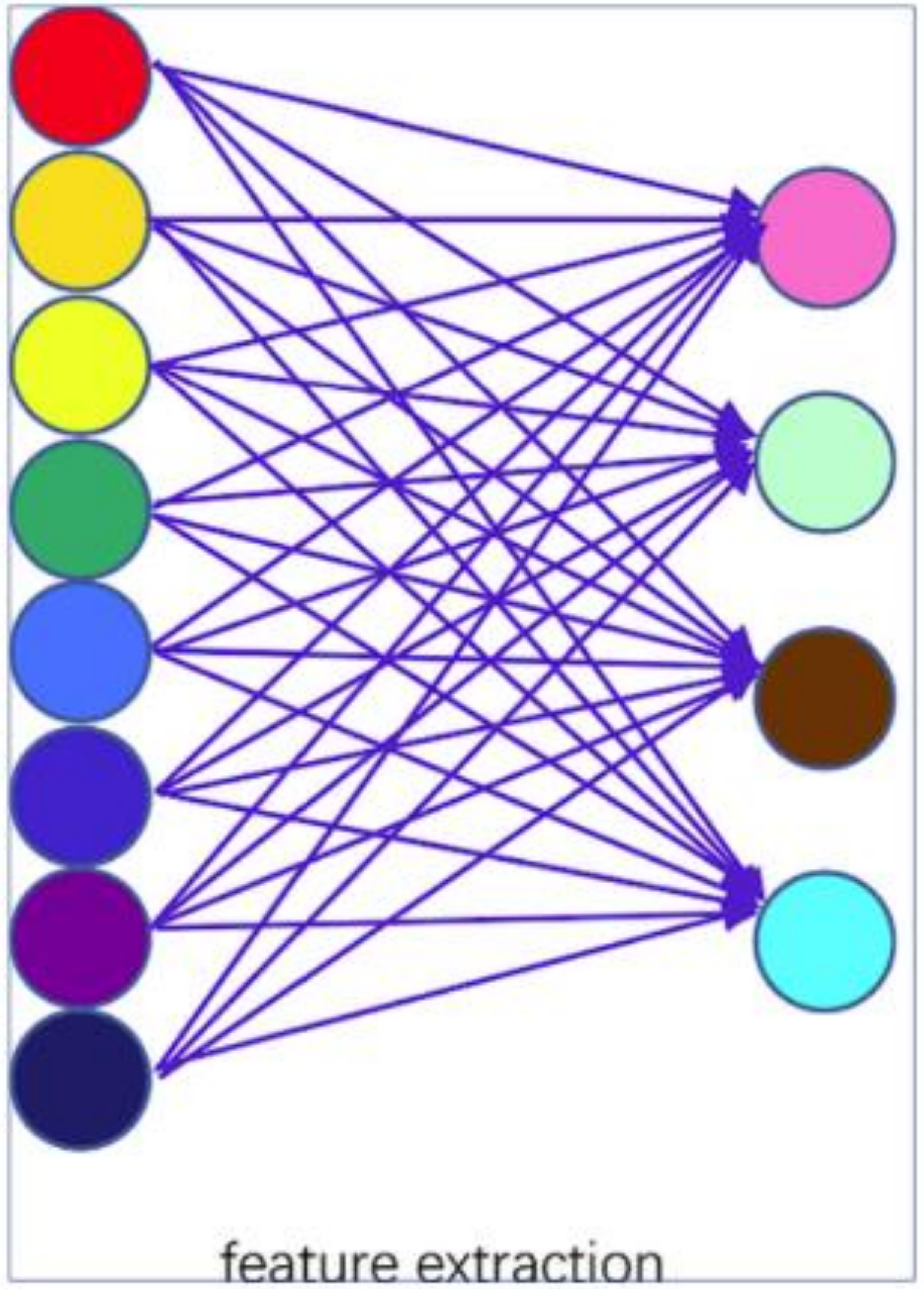
Dimensionality Reduction

- Reduce the number of attributes



Dimensionality Reduction





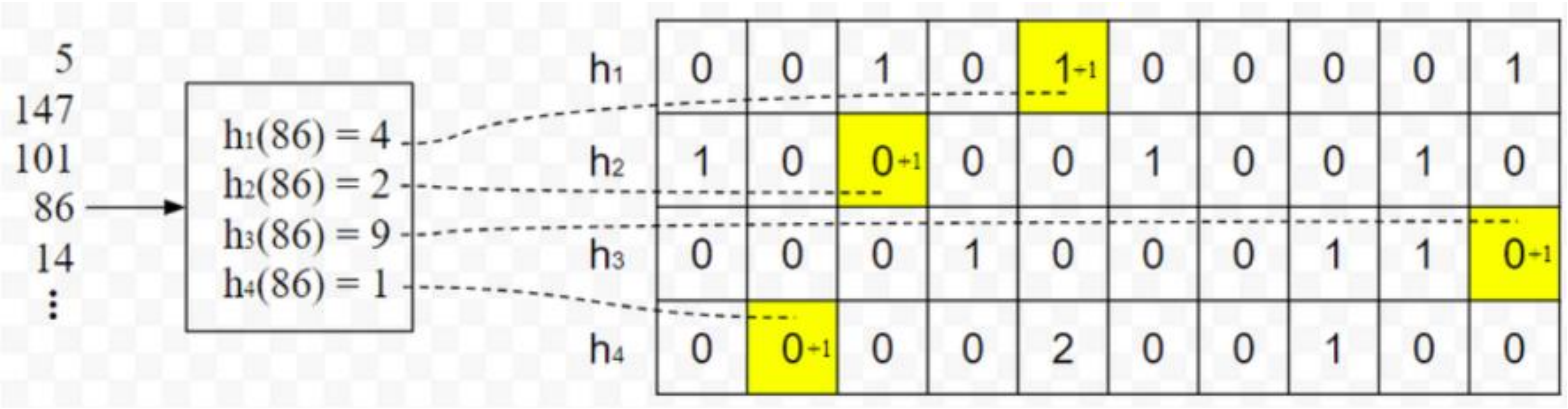
Examples:

Random projection, PCA, feature hashing, UMAP, …

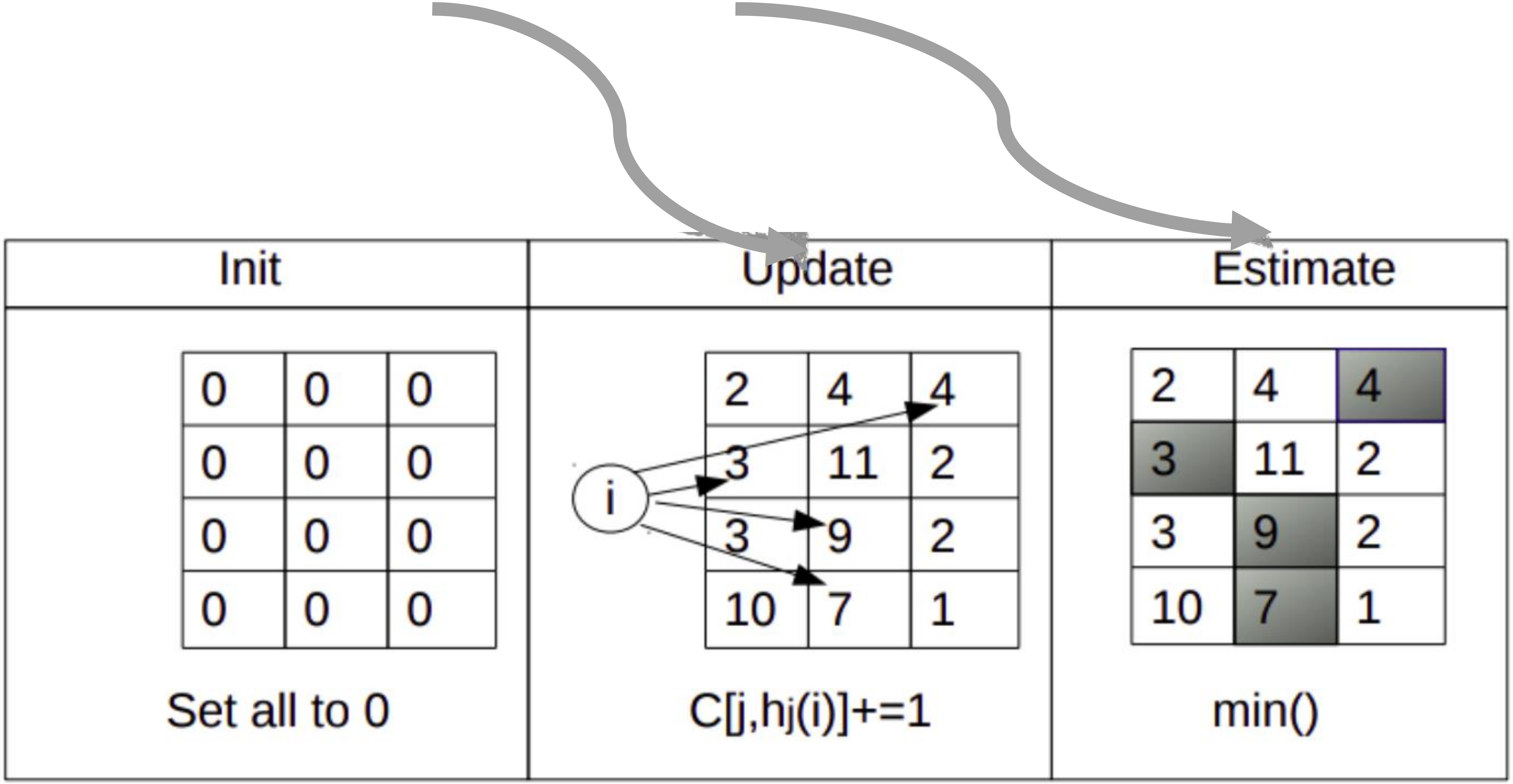
Sketching

- A data structure of a fixed-size
  - **Examples:** Bloom filter, Count-min sketch

Count-min sketch

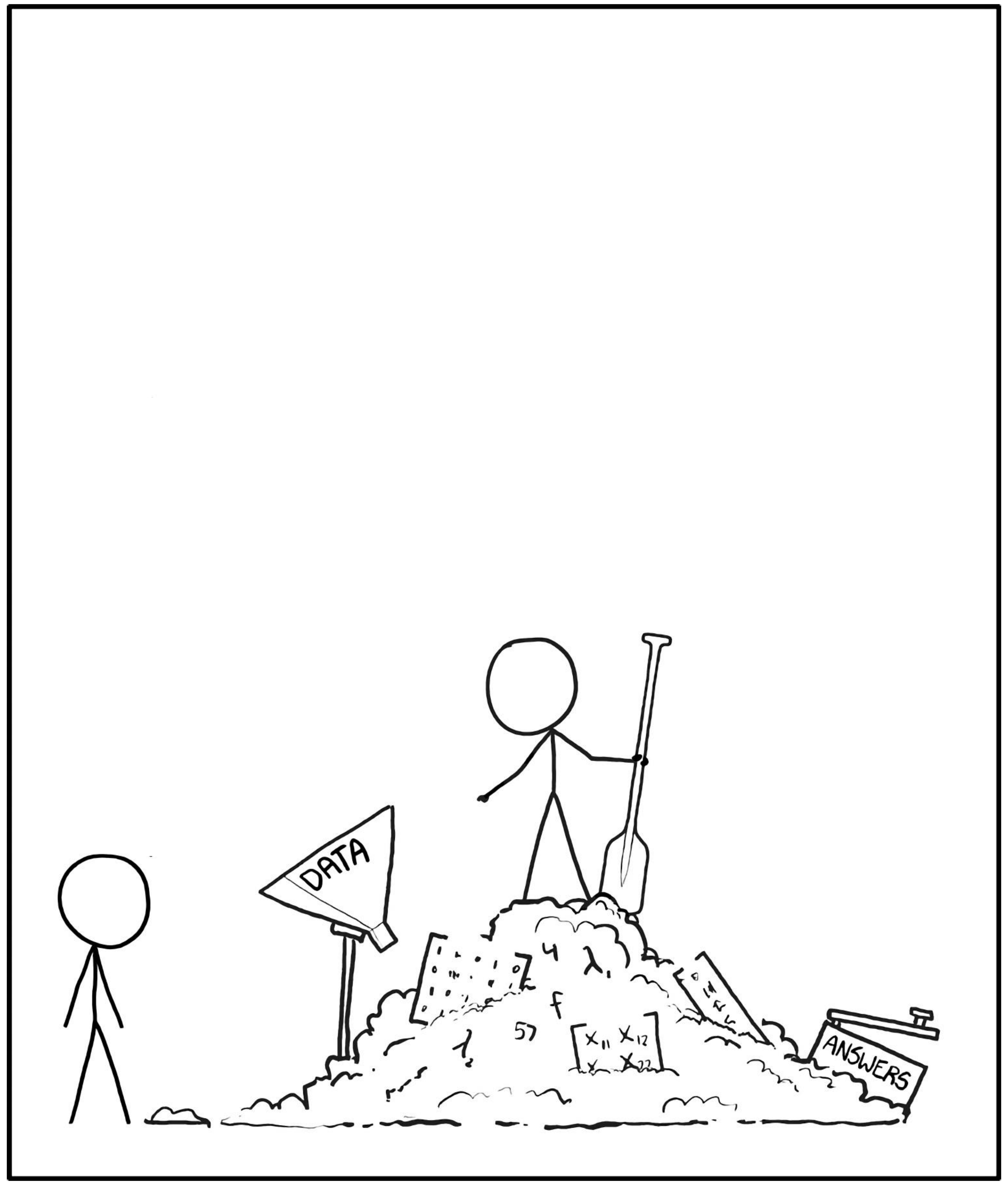


Count-Min Sketch



Automated Machine Learning (AutoML)

AutoML

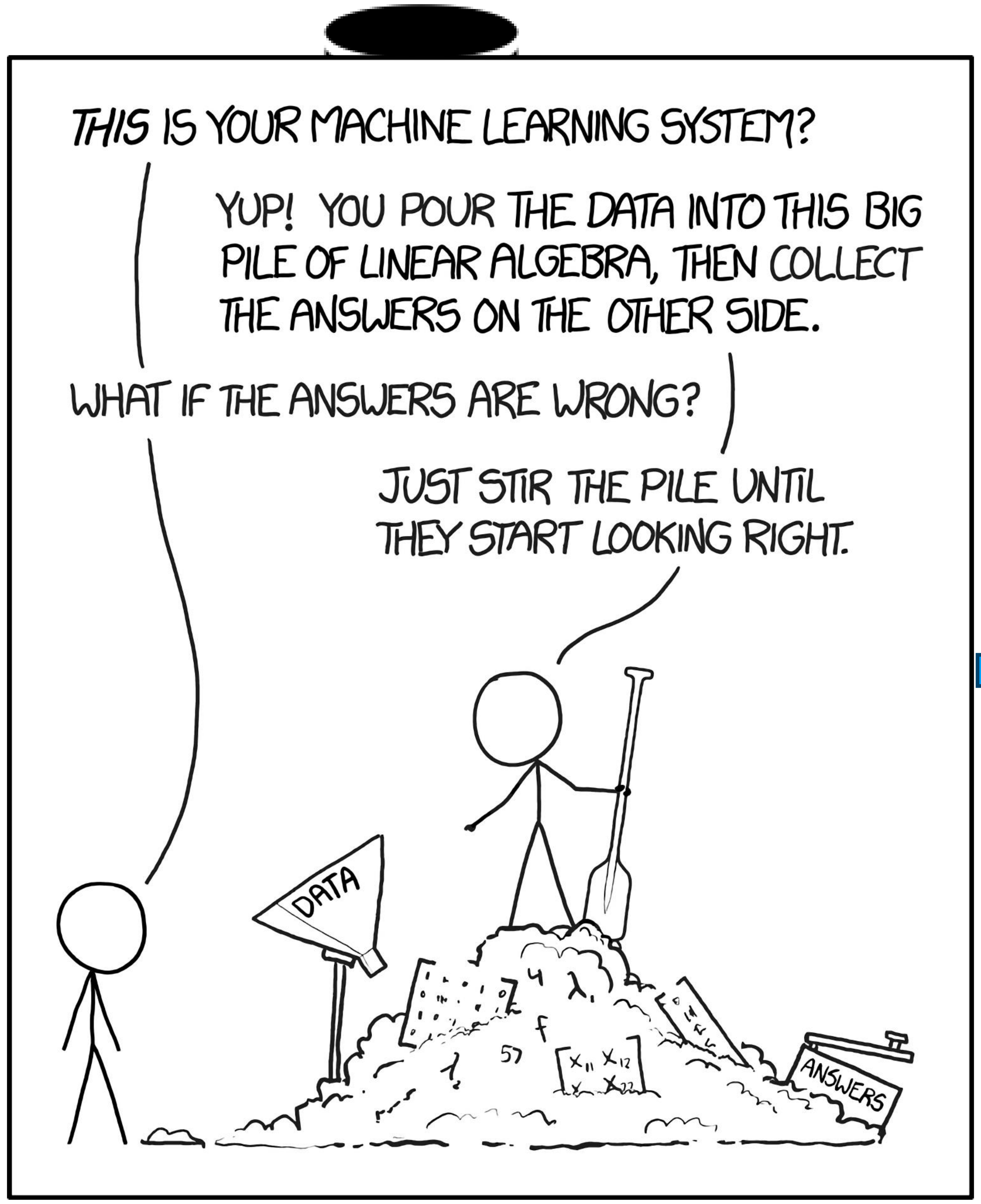


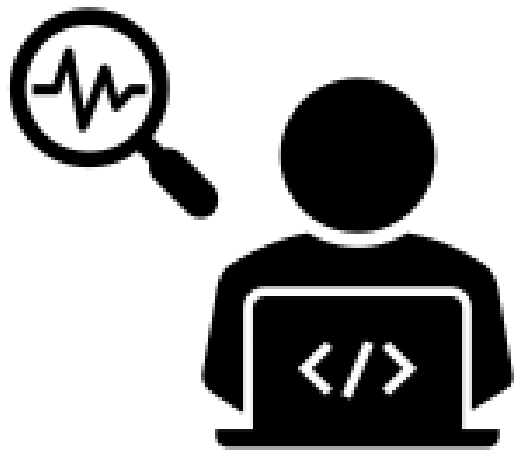
[1]

[1] Munroe, Randall. “Xkcd Cartoon.” *Xkcd*. Accessed 28 Aug. 2022, xkcd.com/1838

Traditional ML and AutoML

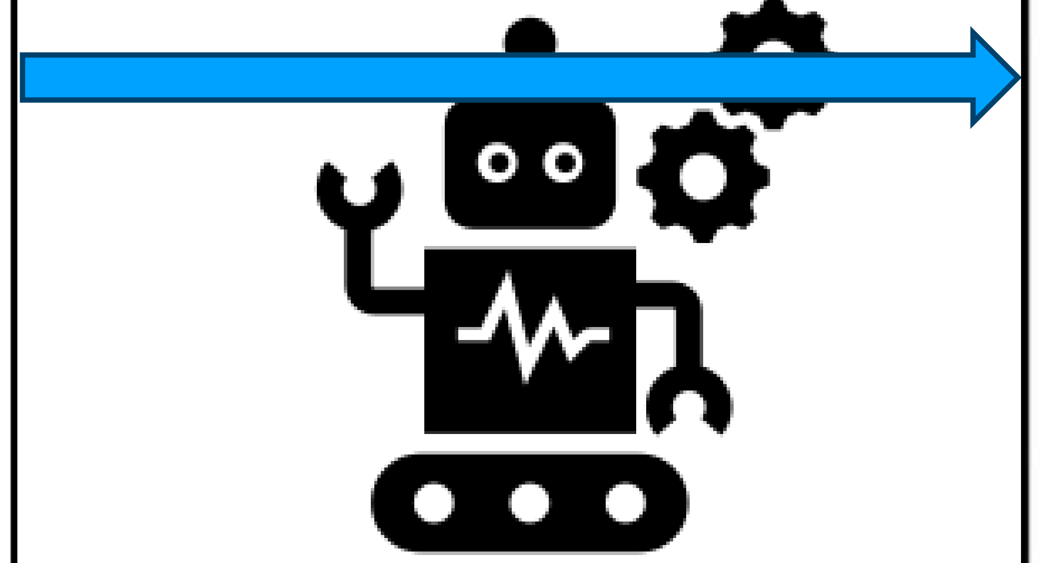
Traditional ML practice





Human expert

AutoML



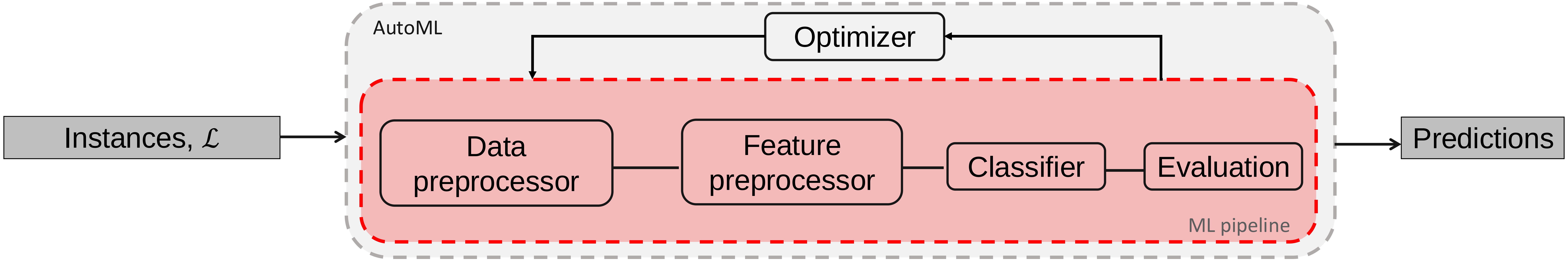
Learning
Hyper-parameter optimizatio
Model evaluation

What is Behind the Magic?

- Data collection
- Data cleaning
- Other data preprocessing
- Hyperparameters optimization
- Model selection

→ Using optimization techniques to automatically detect the best ML algorithm with the best hyperparameter configuration is defined as a *Combined Algorithm Selection and Hyperparameter (CASH)* problem

The AutoML Problem



CASH-Problem [2]

$$\underbrace{A^*,\Lambda^* \in \operatorname*{arg\,min}_{P^{(j)}\in \mathrm{P},\lambda\in\Lambda^{(j)}}}_{\text{Minimize the Loss of a Pipeline configuration}} \underbrace{\frac{1}{K}\sum_{i=1}^{K}}_{\text{Sum of Losses on k-folds}} \overbrace{\mathcal{L}\left(\underbrace{\mathrm{P}_{A,\Lambda}\left(D_{train}^{(i)}\right)}_{\text{Pipeline fitted on } D_{train}^{(i)}}, D_{valid}^{(i)}\right)}^{\text{Loss on } D_{valid}^{(i)}}$$

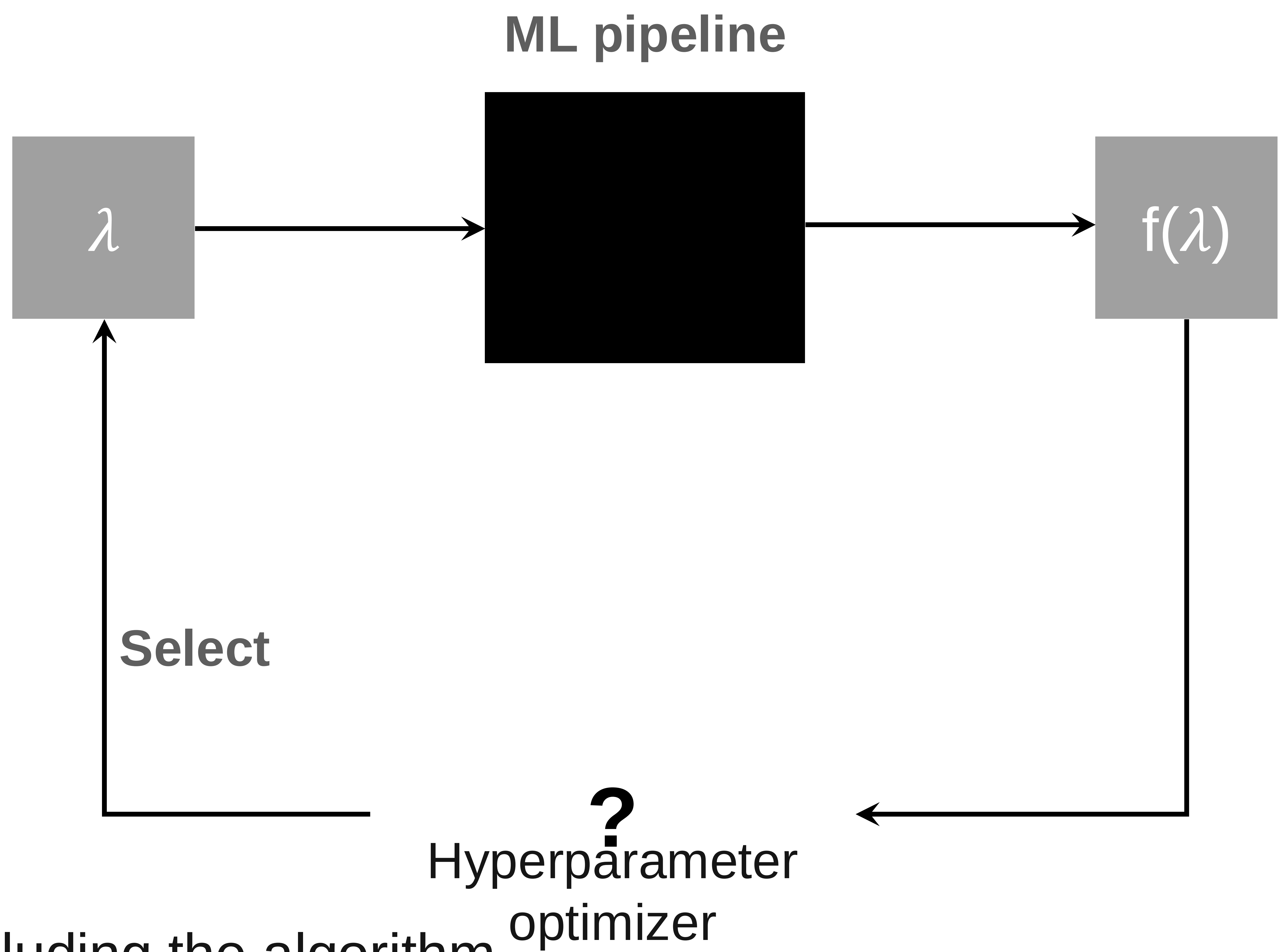
A	Algorithms
Λ	Hyperparameter Configuration
$P_{A,\Lambda}$	Configured ML Pipeline
$\mathcal{L}$	Metric / Loss
$D^{(i)}$	Dataset

*Combined Algorithm Selection and Hyper-parameter optimization problem

[2] Kotthoff, et al. “Auto-WEKA 2.0: Automatic Model Selection and Hyperparameter Optimization in WEKA”. *Journal of Machine Learning Research*, vol. 18, 2017, p. 25:1-25:5

Hyperparameter optimization (HPO)

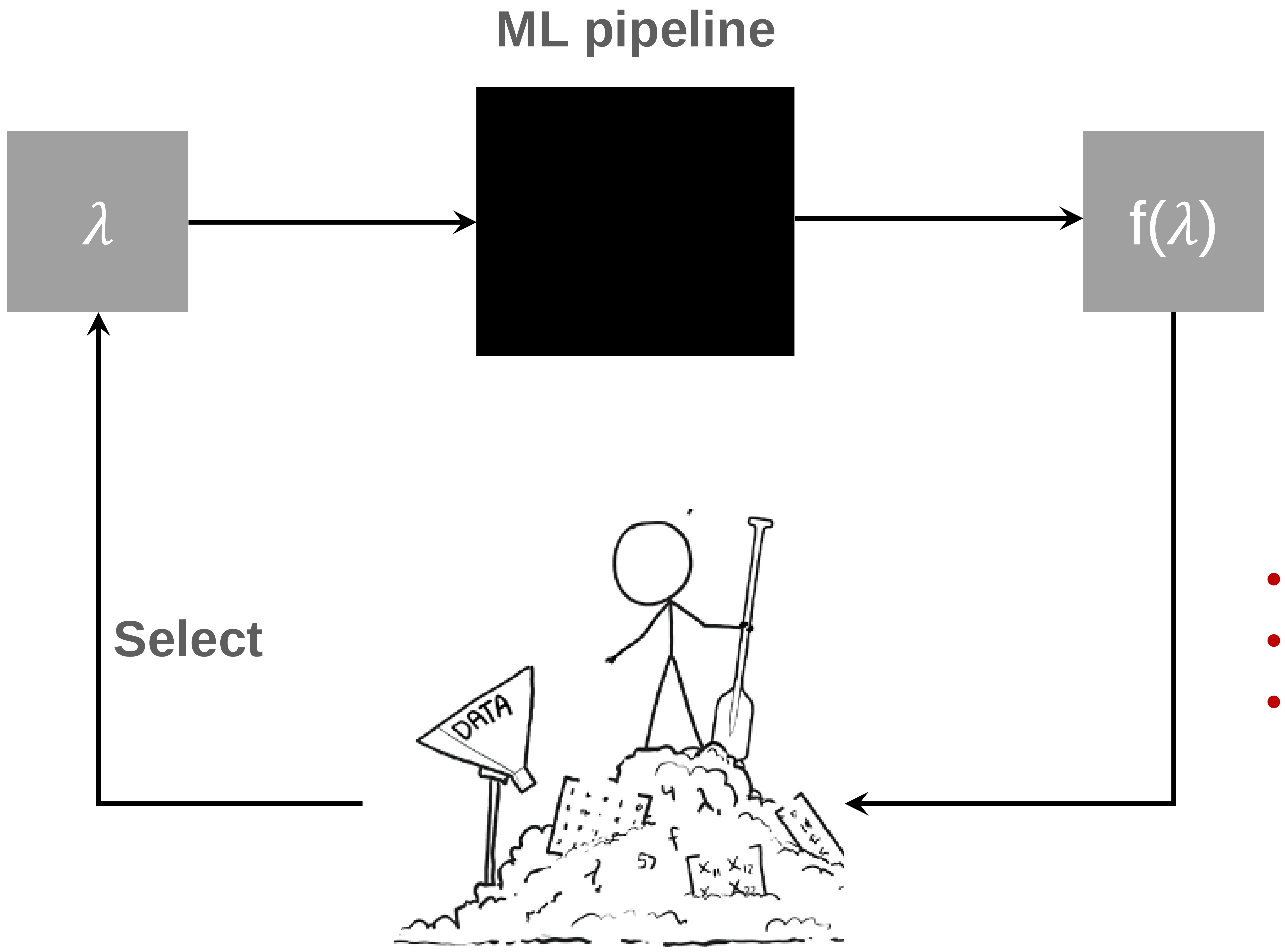
- Black box optimization



HPO = CASH when including the algorithm

Hyperparameter optimization (HPO)

- Black box optimization: Human optimization

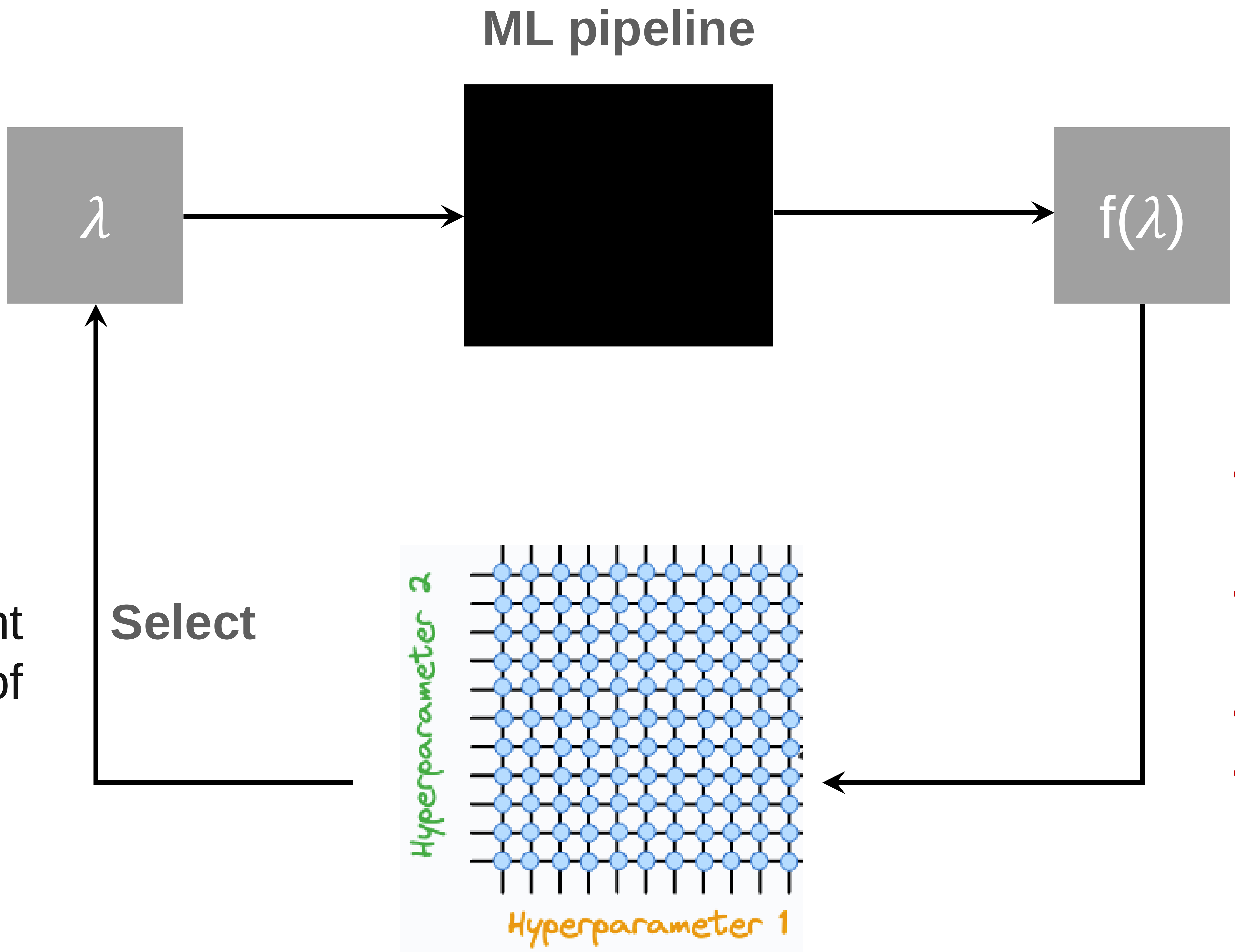


- Better understanding of the results

- Tedious task
- Expert knowledge
- Inefficient

Hyperparameter optimization (HPO)

- Black box optimization: Grid search

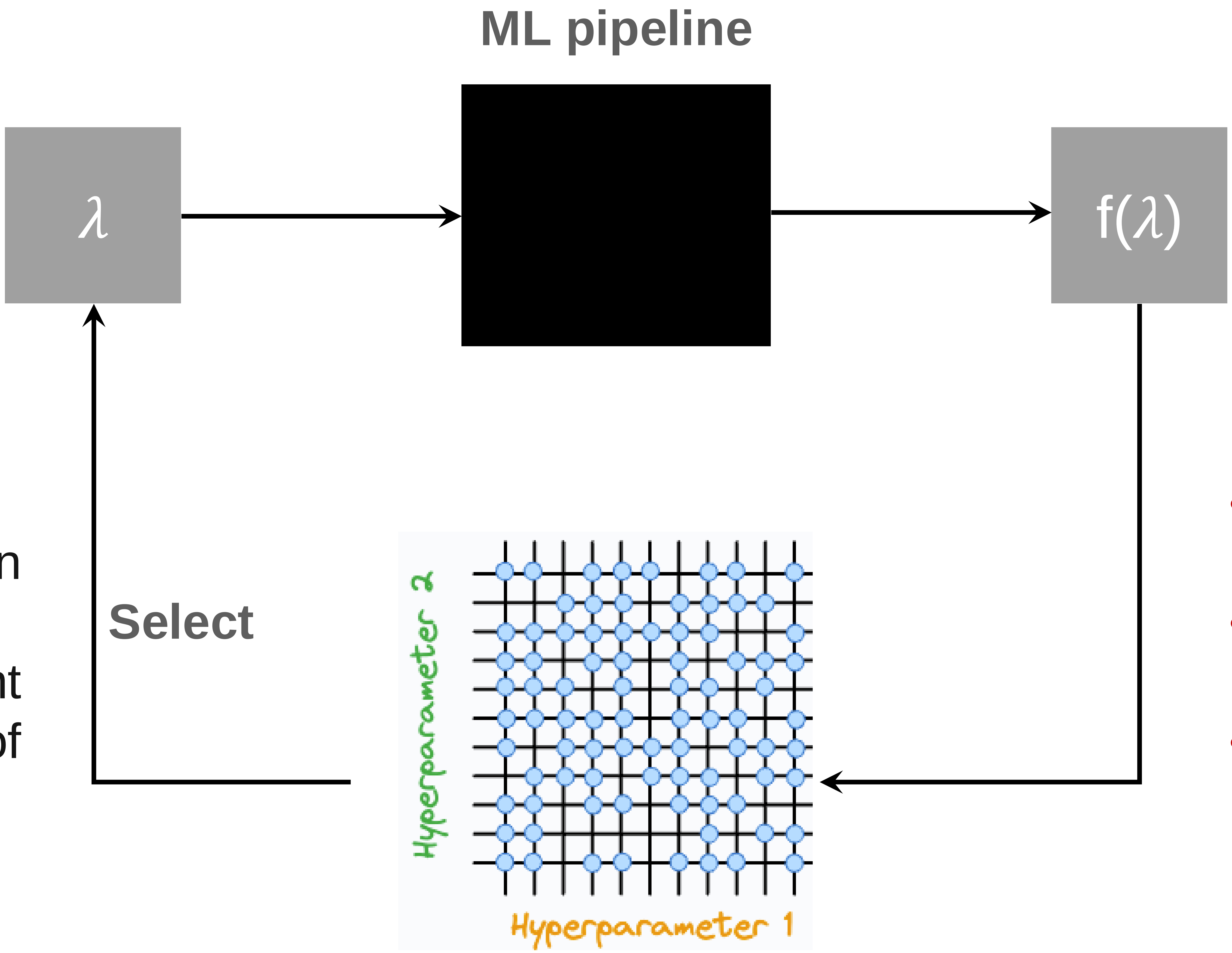


- Evaluates all combination
- Easy to implement
- Handle all types of hyperparameters

- Scale badly with dimensions
- Grid needs to be defined
- Inefficient
- Expensive

Hyperparameter optimization (HPO)

- Black box optimization: Random search

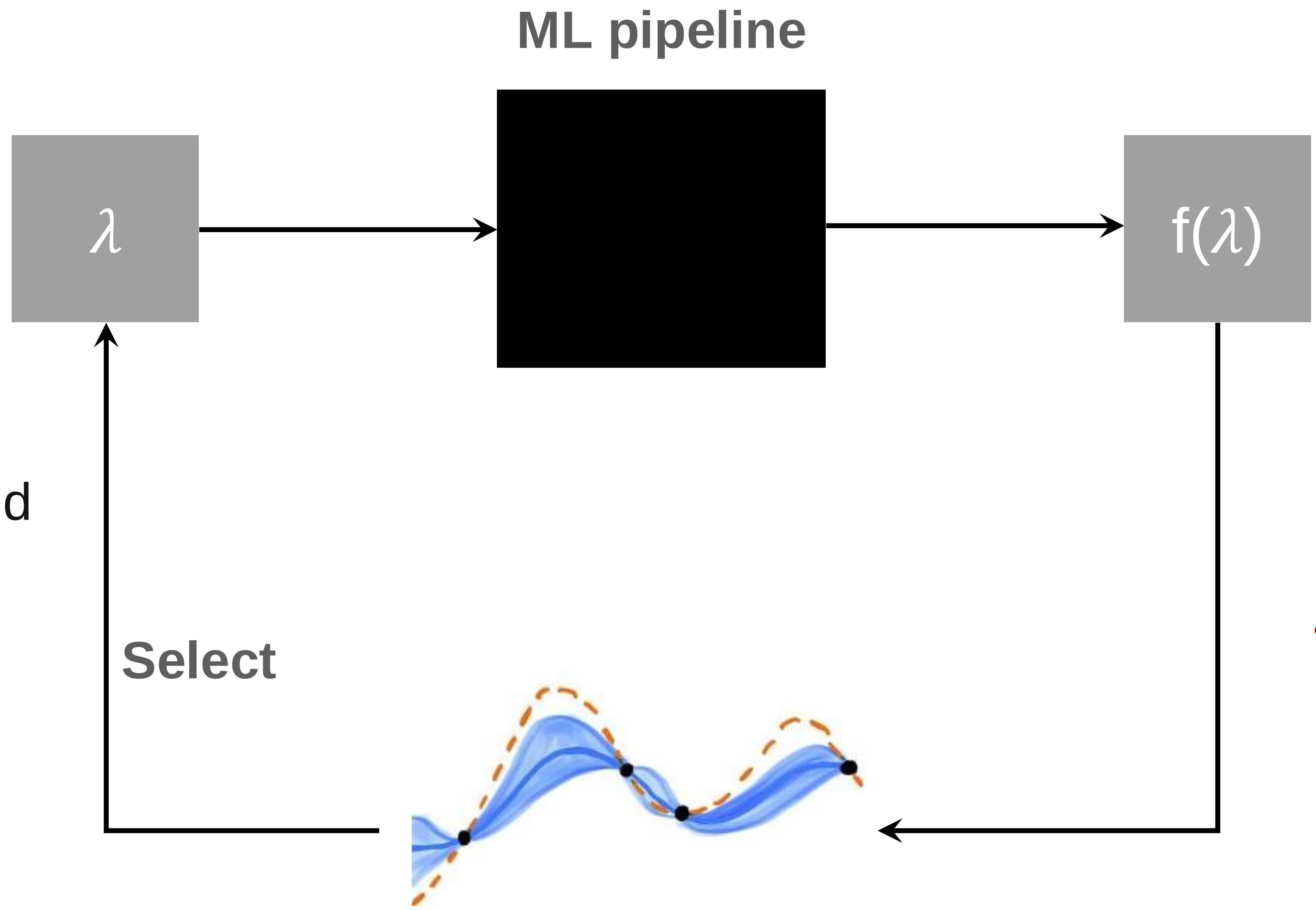


- Variation of grid search
- More efficient than grid search
- Easy to implement
- Handle all types of hyperparameters

- Scale badly with dimensions
- Unnecessary evaluations
- Inefficient

Hyperparameter optimization (HPO)

- Black box optimization: Bayesian optimization

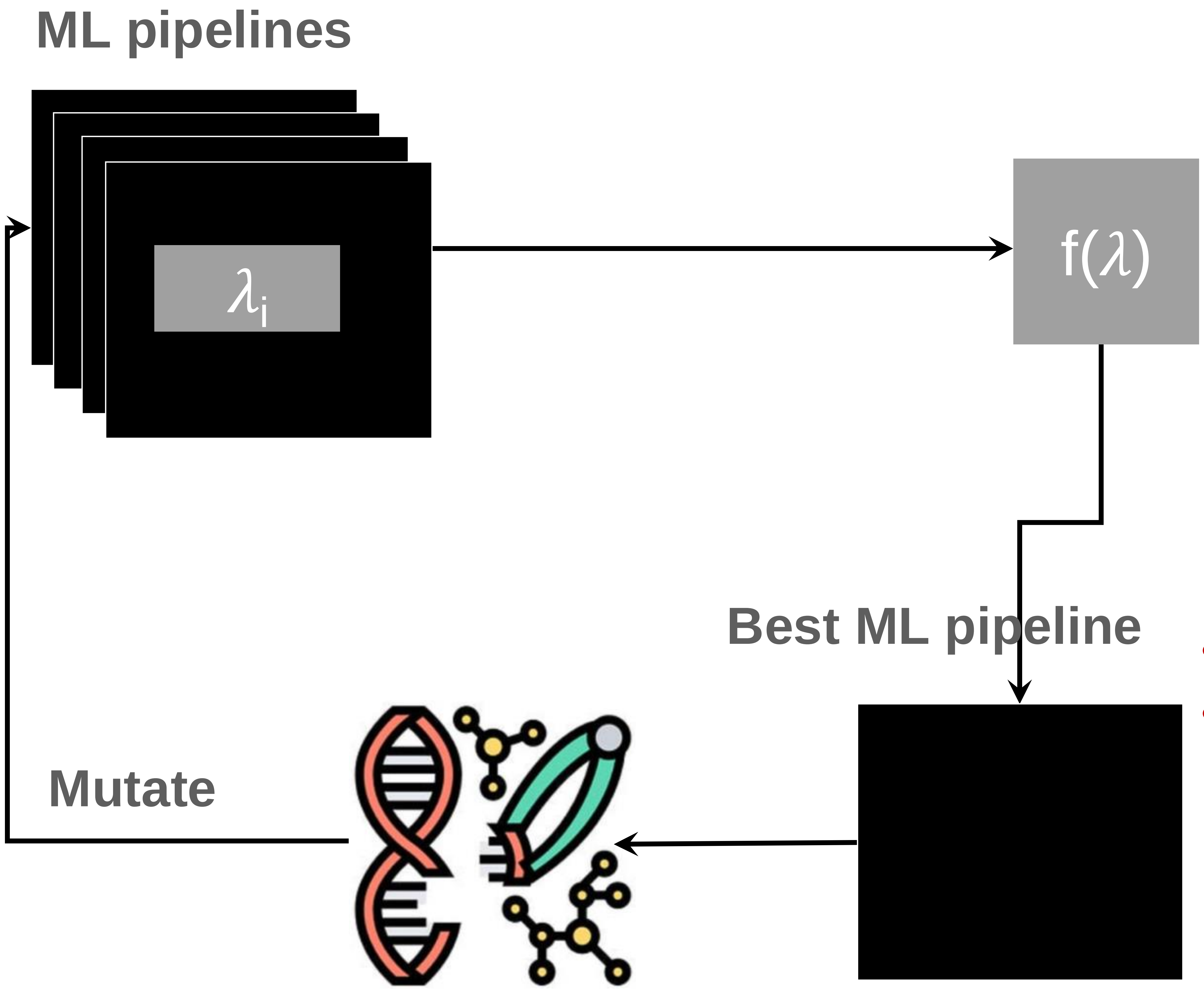


- Overcome GS and RS limitation
- Trade-off exploration & exploitation
- Data efficient

- Scale badly with parallel resources

Hyperparameter optimization (HPO)

- Black box optimization: Evolutionary algorithms



- Easy to implement
- Handle complex configurations
- Can identify global optimums

- Population size
- Mutation rates

AutoML Systems

- Only for supervised learning
- Not semi-supervised or unsupervised learning
- Data streams
- Expensive

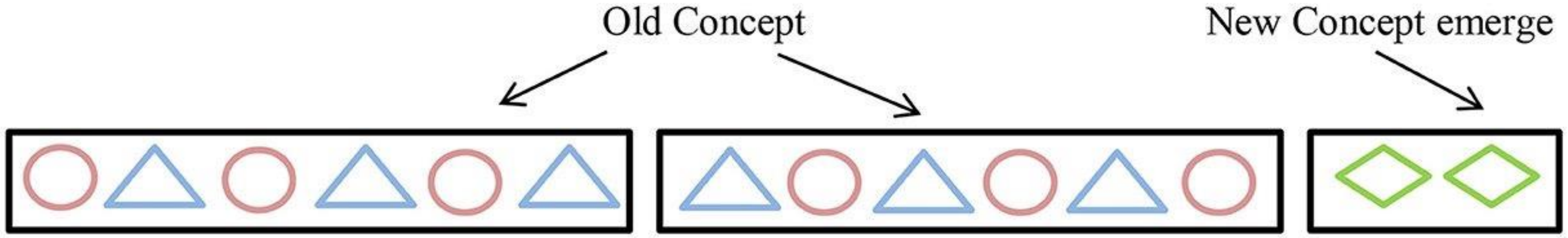
AutoML for Data Streams

Algorithms Have Hyperparameters!

- Machine learning algorithms have multiple hyperparameters:
  - Hoeffding tree: *grace period*, *max depth*, *split criterion*, *confidence, leaf prediction …*
  - Adaptive random forest: *ensemble size, features per tree, leaf prediction, lambda, change detector …*
  - Online bagging: *ensemble size, base learner, parameters of the learner*
  - kNN: *number of neighbors, window size, search technique*
  - Clustream: *window size, number of clusters, number of kernels, kernel factor*
  - …

Challenges

- Many algorithms
- Many hyperparameters
- Hyperparameters can be of different types
- High-dimensional parameter space
- Instances can face concept drift

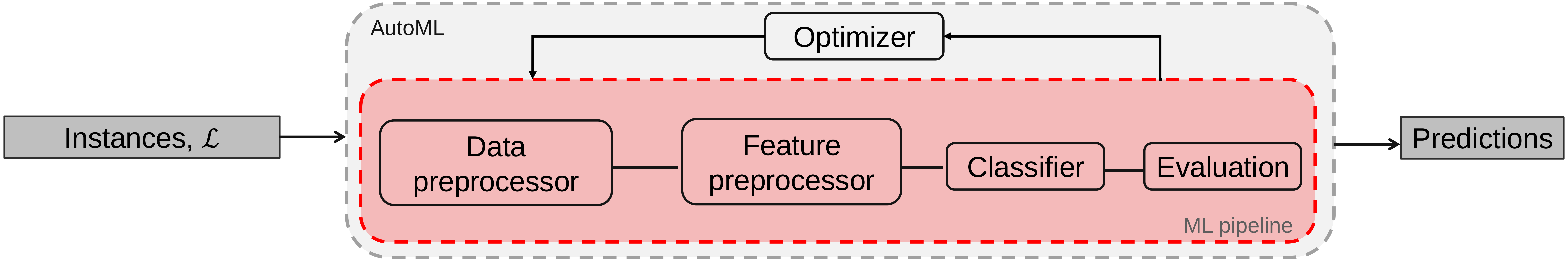


- Expertise in ML

→Combination of algorithm configuration and selection

Setting up your autoML

- Define the algorithms to consider
- Define the configuration space of each algorithm
- Choose the optimization strategy
- Define your evaluation metric



The Stream CASH Problem

Loss on S^V

Stream CASH-Problem [3]

$$\underbrace{A^*, \Lambda^* \in \operatorname*{arg\,min}_{P^{(j)} \in \mathrm{P}, \lambda \in \Lambda^{(j)}}}_{\text{Minimize the Loss of a Pipeline configuration}} \overbrace{\mathcal{L}\big(\underbrace{P_{A,\Lambda}(S^T)}_{\text{Pipeline fitted on } S^T}, S^V\big)}^{\text{Loss on } S^V}$$

Minimize the
Loss of a
Pipeline
configuration

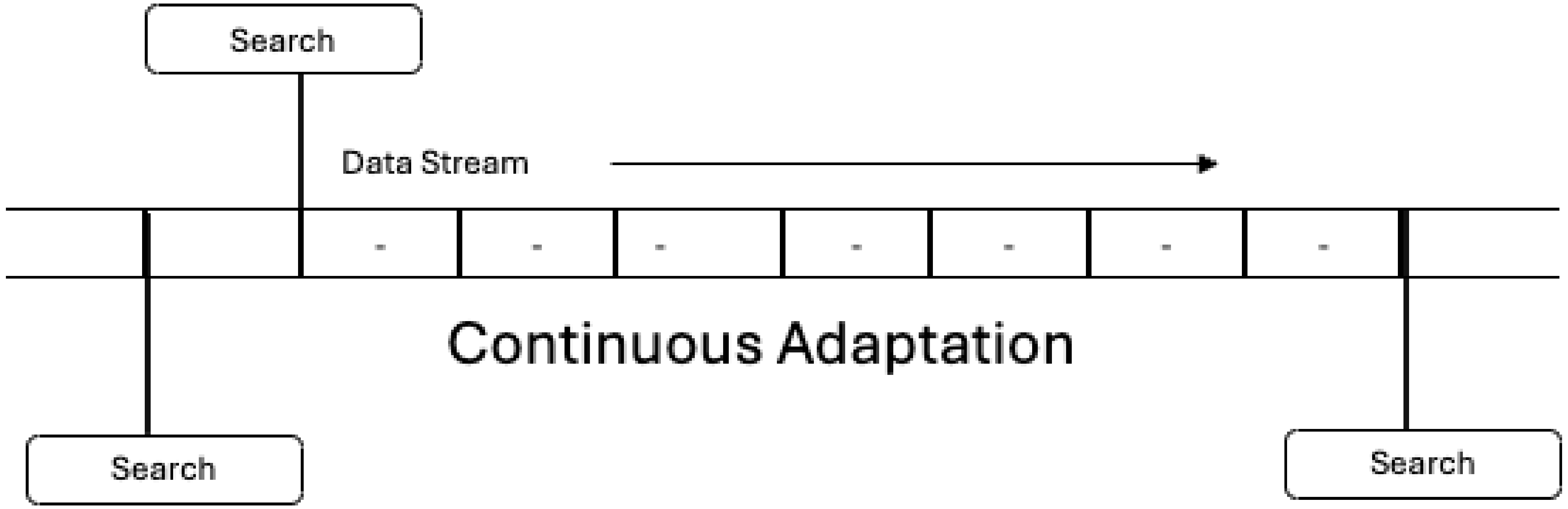
Pipeline fitted
on
S^T

A	Algorithms
Λ	Hyperparameter Configuration
$P_{A,\Lambda}$	Configured ML Pipeline
$\mathcal{L}$	Metric / Loss
S^T	Training Data Points
S^V	Validation Data Points

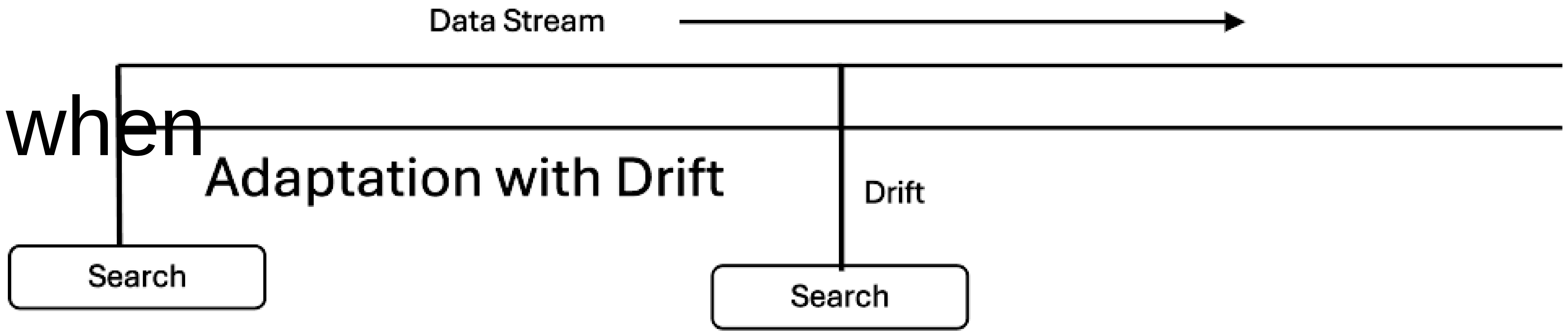
[3] C Kulbach, *et al*. "Evolution-based Online Automated Machine Learning". The Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2022.

Adaptation strategies

- Continuous adaptation
  - Divided windows
  - Periodically run multiple algorithms



- Adaptation with drift
  - Uses a sliding window
  - Run the search process when a drift is detected



EvoAutoML

Strategy: **Continuous adaptation**

Evolutionary mutation for new pipeline search

Select the best-performing pipeline

Mutate and replace the worst-performing pipeline

C Kulbach, *et al*. "Evolution-based Online Automated Machine Learning". The Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2022.

AutoClass

Strategy: **Continuous adaptation**

Probability distribution for new pipeline search

Select the best-performing pipeline

Sample a new pipeline from it

Replace worst-performing pipeline

M Bahri, N Geogantas. "AutoClass: AutoML for Data Stream Classification”. 8th Workshop on Real-time Stream Analytics, Stream Mining, CER/CEP & Stream Data Management in Big Data in conjunction with the IEEE International Conference on Big Data (Big Data), 2023.

OnlineAutoML

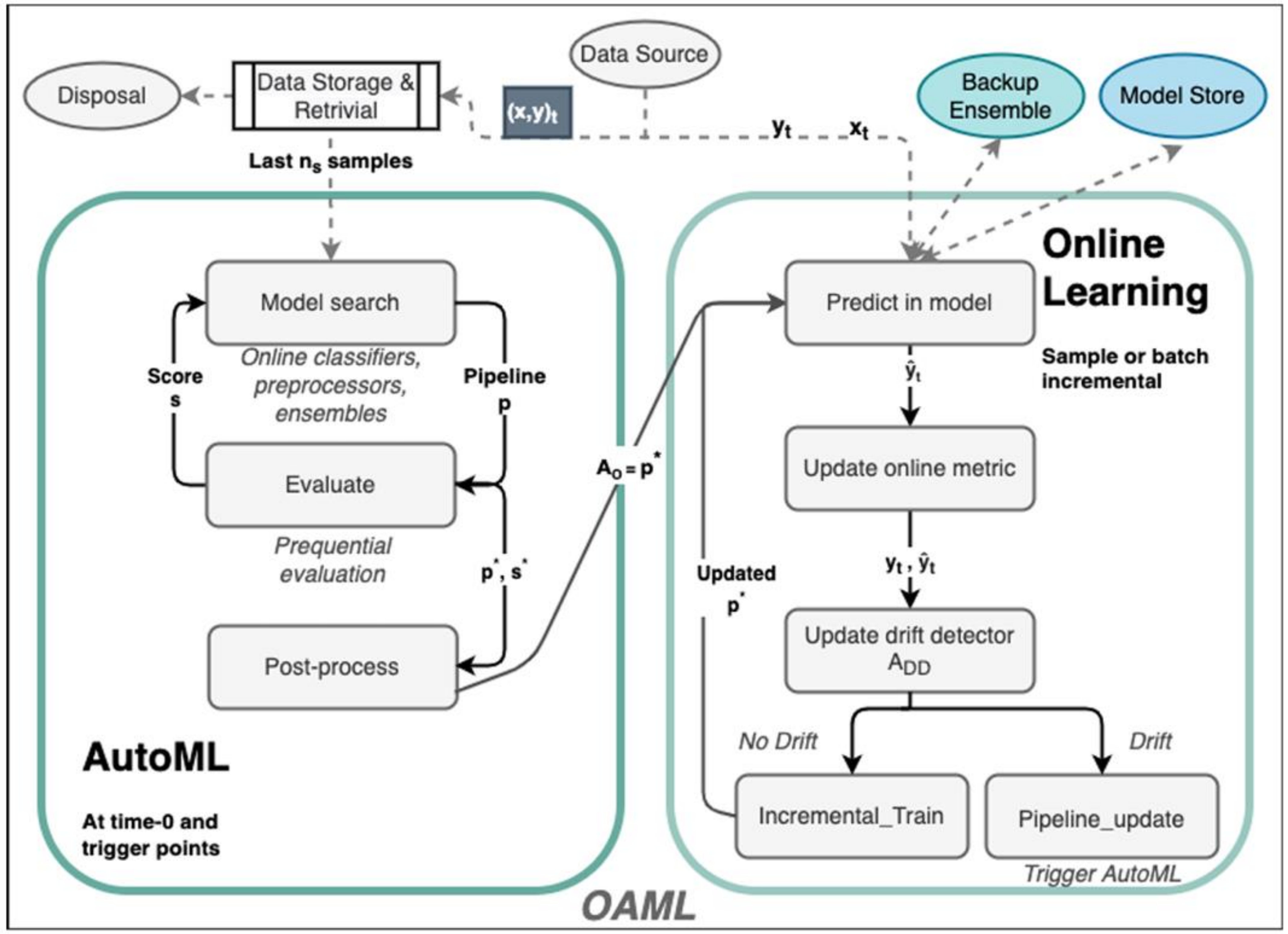
Strategy: **Adaptation with drift**

Genetic algorithm for pipeline search

Detect the drift

Offline search within the last sliding window

Use of the best pipeline for online learning



B Celik *et al*. "Online automl: An adaptive automl framework for online learning”. Machine Learning, 112(6), 1897-1921, 2023.

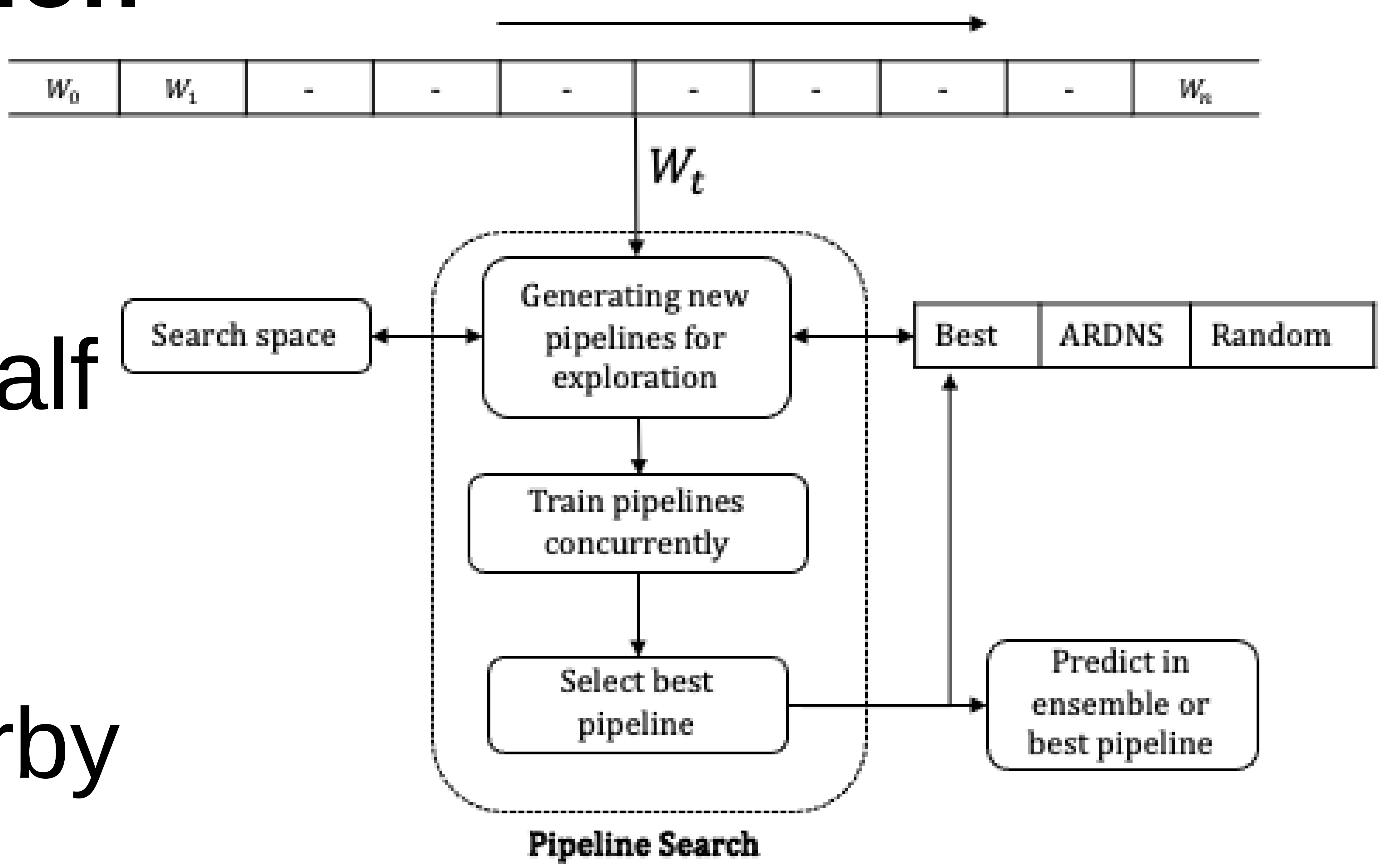
ASML

Strategy: **Continuous adaptation**

Select the best pipeline

Random search to generate half pipelines

Adaptive random directed nearby search to half pipelines



N Verma *et al*. "ASML: A Scalable and Efficient AutoML Solution for Data Streams". International Conference on Automated Machine Learning, 2024

AutoClass

In- &
Output

Algorithm 1 autoClass training

Input:
Data stream S, Ensemble size s, sampling rate w, loss function $\mathcal{L}$, configuration space $\mathcal{A}, \Lambda$
Output:
Set of suited algorithms configurations:
$\mathcal{M} = \{M^{(1)}, \ldots, M^{(s)}\}$

$\mathcal{M} \leftarrow \emptyset$ ▷ Initialization
while $|\mathcal{M}| < s$ AND $\mathcal{M}$ is $\emptyset$ **do**
  $M \leftarrow \text{Add}(\mathcal{A}, \Lambda)$ ▷ Add the algorithms in A with the default parameters
  $\mathcal{M} \leftarrow \mathcal{M} \cup M$
end while
$t \leftarrow 0$
while $HasNext(S)$ **do** ▷ Start the data stream
  $(x, y) \leftarrow Next(S)$
  if $t \mod w == 0$ **then** ▷ Each w instances
    $M^{best} \leftarrow \min_{M \in \mathcal{M}} \mathcal{L}(M(S^T), S^V)$
    $M^{worst} \leftarrow \max_{M \in \mathcal{M}} \mathcal{L}(M(S^T), S^V)$
    $M^{mut} \leftarrow \text{Mutate}(M^{best})$
    $\mathcal{M} \leftarrow \mathcal{M} \cup M^{mut}$ ▷ Add the new generated configuration
    $\mathcal{M} \leftarrow \mathcal{M} \setminus M^{worst}$ ▷ Remove the weakest configuration
  end if
  for $M \in \mathcal{M}$ **do** ▷ Update the ensemble
    $M.\text{fit}(x, y)$
  end for
  $t \leftarrow t + 1$
end while

AutoClass Approach

Input:
Ensemble size: s
Sampling rate: w
loss function: $\mathcal{L}$
configuration space: A, Λ

Output:
Ensemble of best configurations

AutoClass

AutoClass Approach

Input:
Ensemble size: s
Sampling rate: w
loss function: $\mathcal{L}$
configuration space: A, Λ

Output:
Ensemble of best configurations

Algorithm 1 autoClass training

Input:
Data stream S, Ensemble size s, sampling rate w, loss function $\mathcal{L}$, configuration space $\mathcal{A}, \Lambda$
Output:
Set of suited algorithms configurations:
$\mathcal{M} = \{M^{(1)}, \dots, M^{(s)}\}$

$\mathcal{M} \leftarrow \emptyset$ ▷ Initialization
while $|\mathcal{M}| < s$ AND $\mathcal{M}$ is $\emptyset$ **do**
  $M \leftarrow \mathrm{Add}(\mathcal{A}, \Lambda)$ ▷ Add the algorithms in A with the default parameters
  $\mathcal{M} \leftarrow \mathcal{M} \cup M$
end while
$t \leftarrow 0$
while $HasNext(S)$ **do** ▷ Start the data stream
  $(x, y) \leftarrow Next(S)$
  if $t \mod w == 0$ **then** ▷ Each w instances
    $M^{best} \leftarrow \min_{M \in \mathcal{M}} \mathcal{L}(M(S^T), S^V)$
    $M^{worst} \leftarrow \max_{M \in \mathcal{M}} \mathcal{L}(M(S^T), S^V)$
    $M^{mut} \leftarrow \mathrm{Mutate}(M^{best})$
    $\mathcal{M} \leftarrow \mathcal{M} \cup M^{mut}$ ▷ Add the new generated configuration
    $\mathcal{M} \leftarrow \mathcal{M} \setminus M^{worst}$ ▷ Remove the weakest configuration
  end if
  for $M \in \mathcal{M}$ **do** ▷ Update the ensemble
    $M.\mathrm{fit}(x, y)$
  end for
  $t \leftarrow t + 1$
end while

In- &
Output

AutoClass

Initialization

$$M^{(i)}_{\vec{A},\vec{\lambda}}$$

$$\vdots$$

$$M^{(s)}_{\vec{A},\vec{\lambda}}$$

Algorithm 1 autoClass training

Input:
Data stream S, Ensemble size s, sampling rate w, loss function $\mathcal{L}$, configuration space $\mathcal{A}$, Λ
Output:
Set of suited algorithms configurations:
$\mathcal{M} = \{M^{(1)}, \dots, M^{(s)}\}$

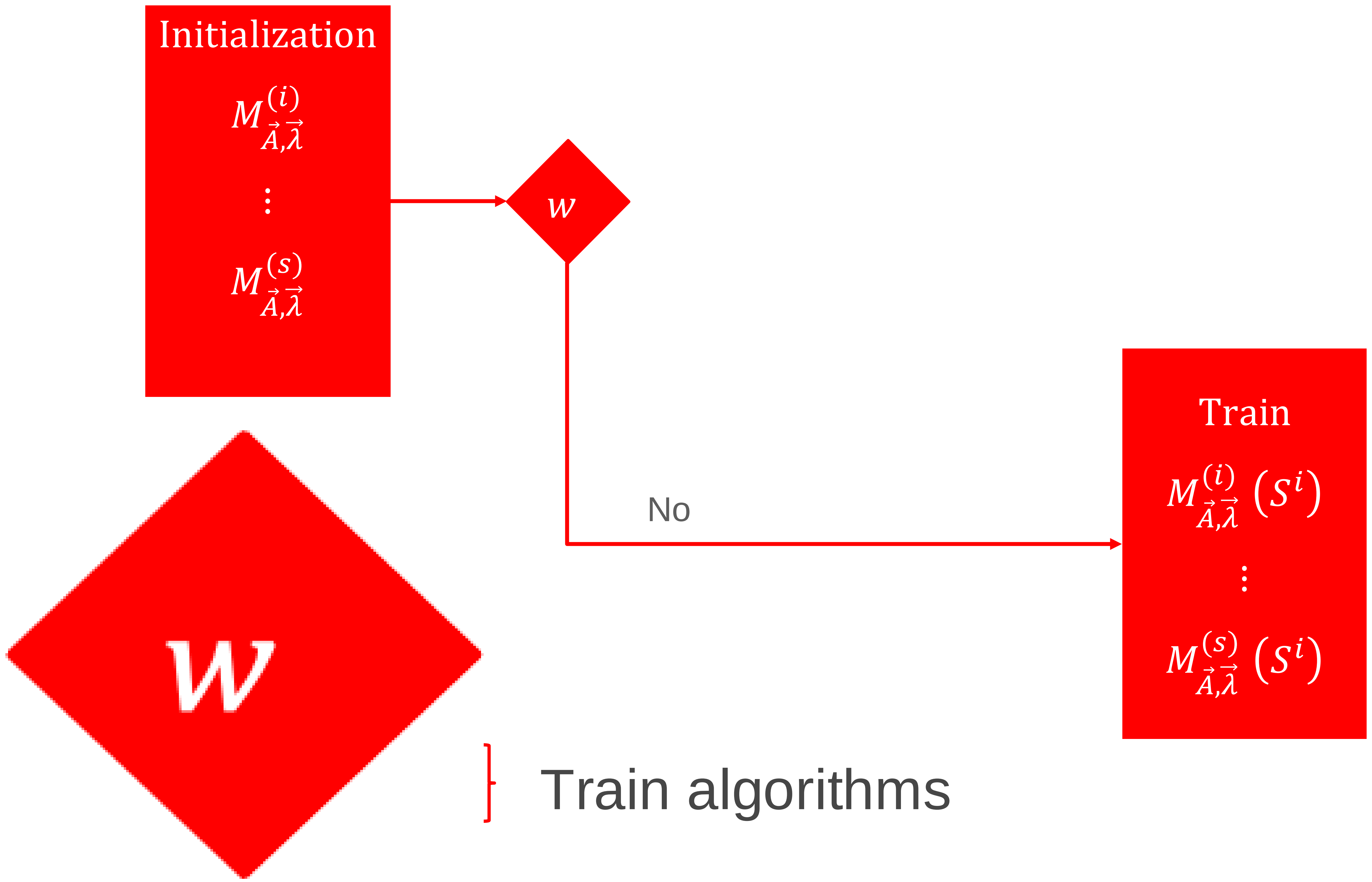
$\mathcal{M} \leftarrow \emptyset$ ▷ Initialization
while $|\mathcal{M}| < s$ AND $\mathcal{M}$ is $\emptyset$ **do**
    $M \leftarrow \mathrm{Add}(\mathcal{A}, \Lambda)$ ▷ Add the algorithms in A with the default parameters
    $\mathcal{M} \leftarrow \mathcal{M} \cup M$
end while
$t \leftarrow 0$
while $HasNext(S)$ **do** ▷ Start the data stream
    $(x, y) \leftarrow Next(S)$
    if $t \mod w == 0$ **then** ▷ Each w instances
        $M^{best} \leftarrow \min_{M \in \mathcal{M}} \mathcal{L}(M(S^T), S^V)$
        $M^{worst} \leftarrow \max_{M \in \mathcal{M}} \mathcal{L}(M(S^T), S^V)$
        $M^{mut} \leftarrow \mathrm{Mutate}(M^{best})$
        $\mathcal{M} \leftarrow \mathcal{M} \cup M^{mut}$ ▷ Add the new generated configuration
        $\mathcal{M} \leftarrow \mathcal{M} \backslash M^{worst}$ ▷ Remove the weakest configuration
    end if
    for $M \in \mathcal{M}$ **do** ▷ Update the ensemble
        $M.\mathrm{fit}(x, y)$
    end for
    $t \leftarrow t + 1$
end while

Initialization

AutoClass Approach

Initialization:
Generating methods from the configuration space with default parameters

AutoClass

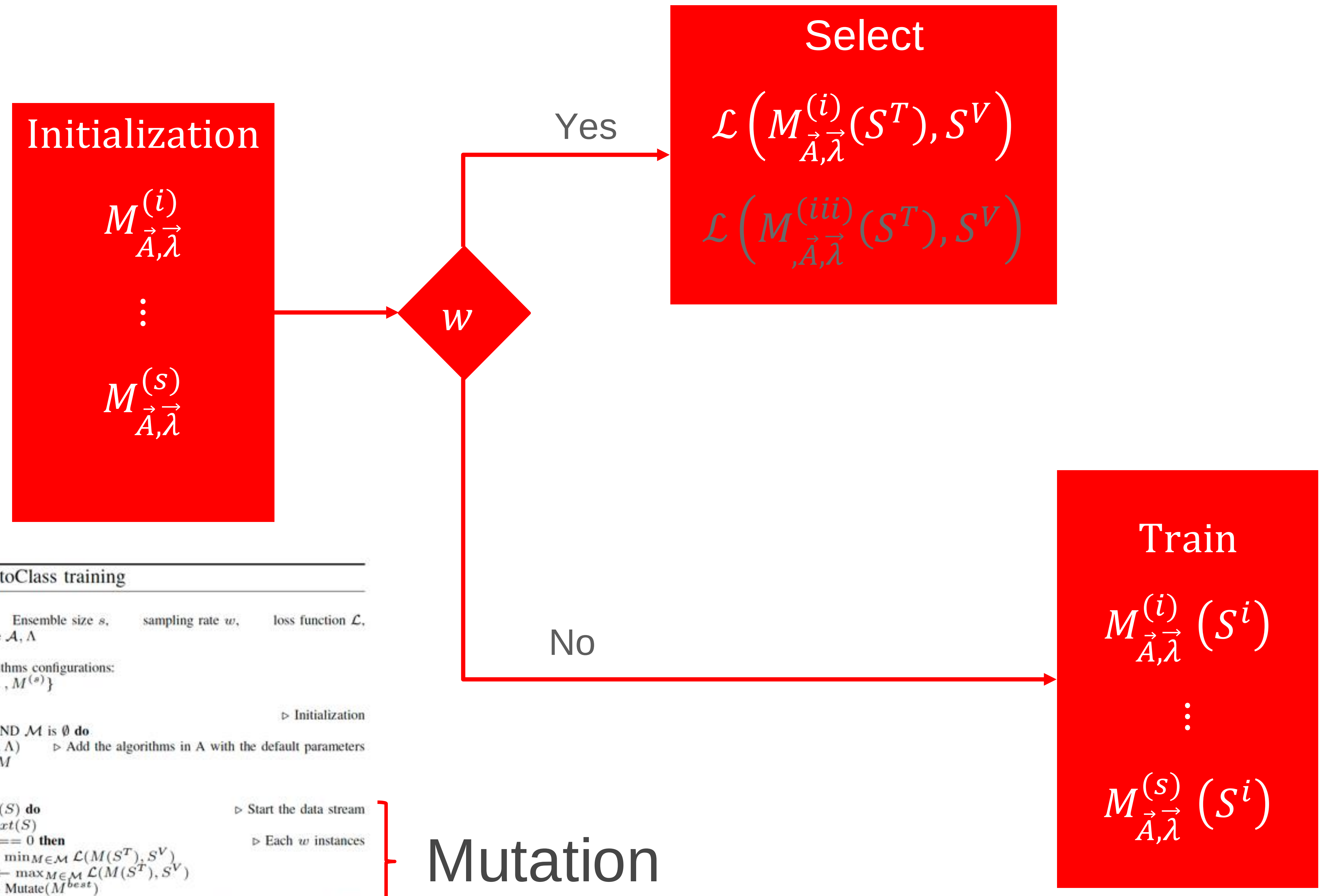


AutoClass Approach

Mutation:

1. Select a good & weakest pipelines

AutoClass



Algorithm 1 autoClass training

```
Input:
Data stream S,   Ensemble size s,   sampling rate w,   loss function L,
configuration space A, Λ
Output:
Set of suited algorithms configurations:
M = {M^(1), ..., M^(s)}

M ← ∅                                              ▷ Initialization
while |M| < s AND M is ∅ do
    M ← Add(A, Λ)        ▷ Add the algorithms in A with the default parameters
    M ← M ∪ M
end while
t ← 0
while HasNext(S) do                                ▷ Start the data stream
    (x, y) ← Next(S)
    if t  mod w == 0 then                          ▷ Each w instances
        M^best ← min_{M∈M} L(M(S^T), S^V)
        M^worst ← max_{M∈M} L(M(S^T), S^V)
        M^mut ← Mutate(M^best)
        M ← M ∪ M^mut                 ▷ Add the new generated configuration
        M ← M\M^worst                 ▷ Remove the weakest configuration
    end if
    for M ∈ M do                                   ▷ Update the ensemble
        M.fit(x, y)
    end for
    t ← t + 1
end while
```

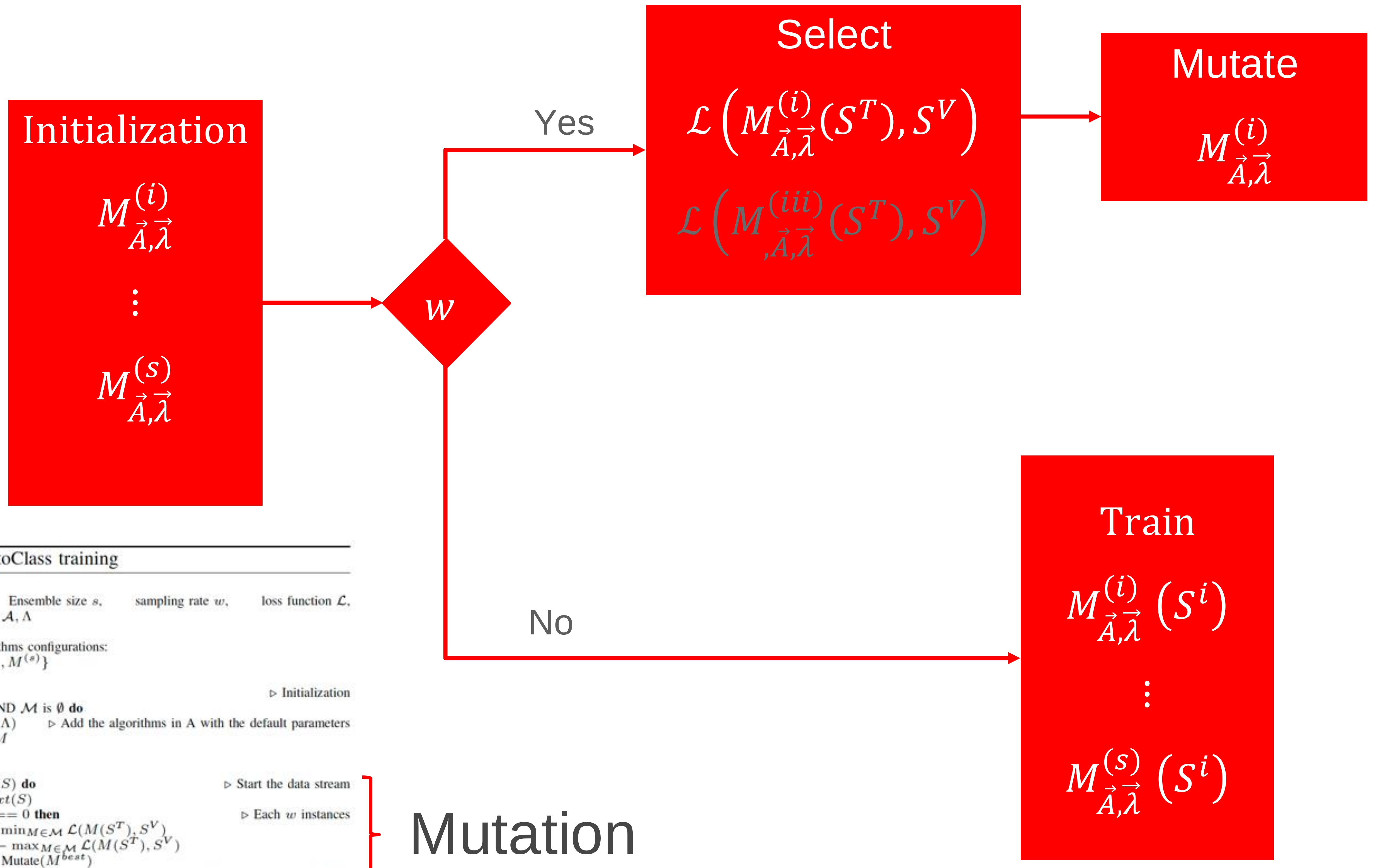
Mutation

AutoClass Approach

Mutation:

1. Select a good & weakest pipelines

AutoClass



Algorithm 1 autoClass training

Input:
Data stream S, Ensemble size s, sampling rate w, loss function $\mathcal{L}$, configuration space $\mathcal{A}$, Λ
Output:
Set of suited algorithms configurations:
$\mathcal{M} = \{M^{(1)}, \ldots, M^{(s)}\}$

$\mathcal{M} \leftarrow \emptyset$ ▷ Initialization
while $|\mathcal{M}| < s$ AND $\mathcal{M}$ is $\emptyset$ **do**
  $M \leftarrow \text{Add}(\mathcal{A}, \Lambda)$ ▷ Add the algorithms in A with the default parameters
  $\mathcal{M} \leftarrow \mathcal{M} \cup M$
end while
$t \leftarrow 0$
while $HasNext(S)$ **do** ▷ Start the data stream
  $(x, y) \leftarrow Next(S)$
  if $t \mod w == 0$ **then** ▷ Each w instances
    $M^{best} \leftarrow \min_{M \in \mathcal{M}} \mathcal{L}(M(S^T), S^V)$
    $M^{worst} \leftarrow \max_{M \in \mathcal{M}} \mathcal{L}(M(S^T), S^V)$
    $M^{mut} \leftarrow \text{Mutate}(M^{best})$
    $\mathcal{M} \leftarrow \mathcal{M} \cup M^{mut}$ ▷ Add the new generated configuration
    $\mathcal{M} \leftarrow \mathcal{M} \setminus M^{worst}$ ▷ Remove the weakest configuration
  end if
  for $M \in \mathcal{M}$ **do** ▷ Update the ensemble
    $M.\text{fit}(x, y)$
  end for
  $t \leftarrow t + 1$
end while

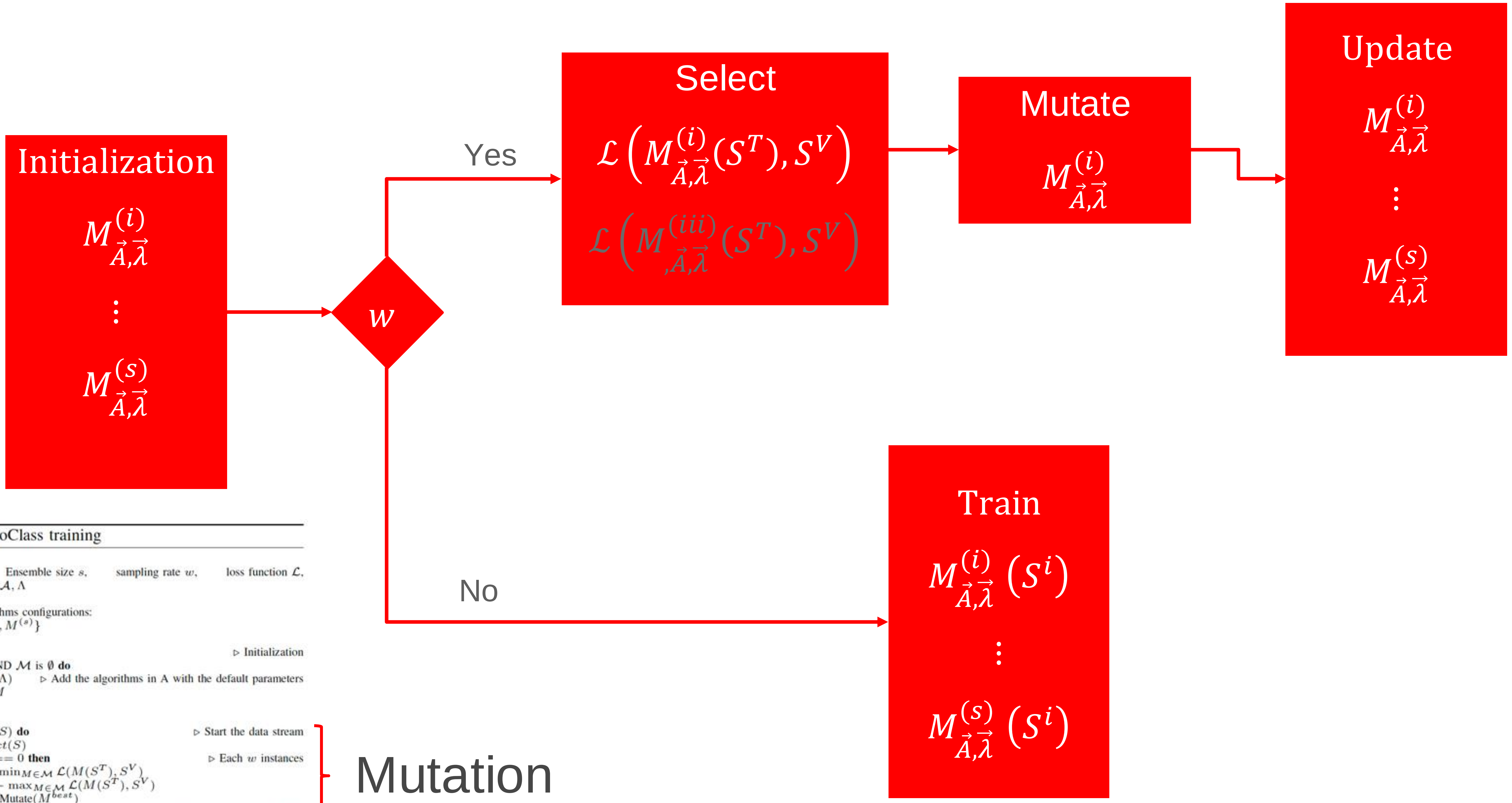
Mutation

AutoClass Approach

Mutation:

1. Select a good & weakest pipelines
2. Generate a new configuration by mutating the good one

AutoClass



Algorithm 1 autoClass training

```
Input:
Data stream S,   Ensemble size s,   sampling rate w,   loss function 𝓛,
configuration space 𝒜, Λ
Output:
Set of suited algorithms configurations:
𝓜 = {M^(1), ..., M^(s)}

𝓜 ← ∅                                                    ▷ Initialization
while |𝓜| < s AND 𝓜 is ∅ do
    M ← Add(𝒜, Λ)          ▷ Add the algorithms in A with the default parameters
    𝓜 ← 𝓜 ∪ M
end while
t ← 0
while HasNext(S) do                                  ▷ Start the data stream
    (x, y) ← Next(S)
    if t mod w == 0 then                                 ▷ Each w instances
        M^best ← min_{M∈𝓜} 𝓛(M(S^T), S^V)
        M^worst ← max_{M∈𝓜} 𝓛(M(S^T), S^V)
        M^mut ← Mutate(M^best)
        𝓜 ← 𝓜 ∪ M^mut                    ▷ Add the new generated configuration
        𝓜 ← 𝓜\M^worst                     ▷ Remove the weakest configuration
    end if
    for M ∈ 𝓜 do                                     ▷ Update the ensemble
        M.fit(x, y)
    end for
    t ← t + 1
end while
```

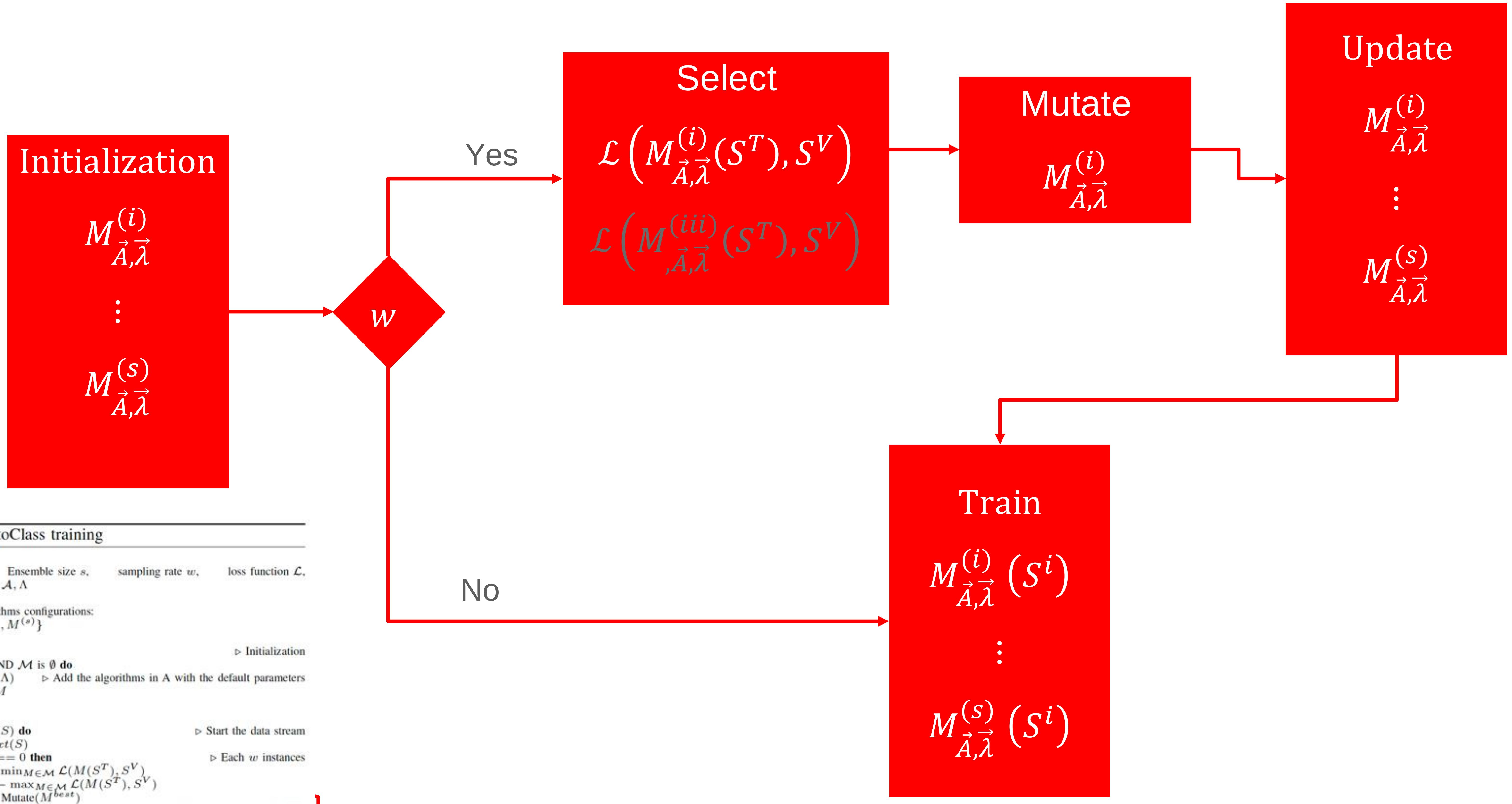
Mutation

AutoClass Approach

Mutation:

1. Select a good & weakest pipelines
2. Generate a new configuration by mutating the good one
3. Remove the weakest configuration

AutoClass



Algorithm 1 autoClass training

```
Input:
Data stream S,   Ensemble size s,   sampling rate w,   loss function ℒ,
configuration space 𝒜, Λ
Output:
Set of suited algorithms configurations:
ℳ = {M^(1), ..., M^(s)}

ℳ ← ∅                                          ▷ Initialization
while |ℳ| < s AND ℳ is ∅ do
    M ← Add(𝒜, Λ)        ▷ Add the algorithms in A with the default parameters
    ℳ ← ℳ ∪ M
end while
t ← 0
while HasNext(S) do                            ▷ Start the data stream
    (x, y) ← Next(S)
    if t mod w == 0 then                       ▷ Each w instances
        M^best ← min_{M∈ℳ} ℒ(M(S^T), S^V)
        M^worst ← max_{M∈ℳ} ℒ(M(S^T), S^V)
        M^mut ← Mutate(M^best)
        ℳ ← ℳ ∪ M^mut               ▷ Add the new generated configuration
        ℳ ← ℳ\M^worst               ▷ Remove the weakest configuration
    end if
    for M ∈ ℳ do                               ▷ Update the ensemble
        M.fit(x, y)
    end for
    t ← t + 1
end while
```

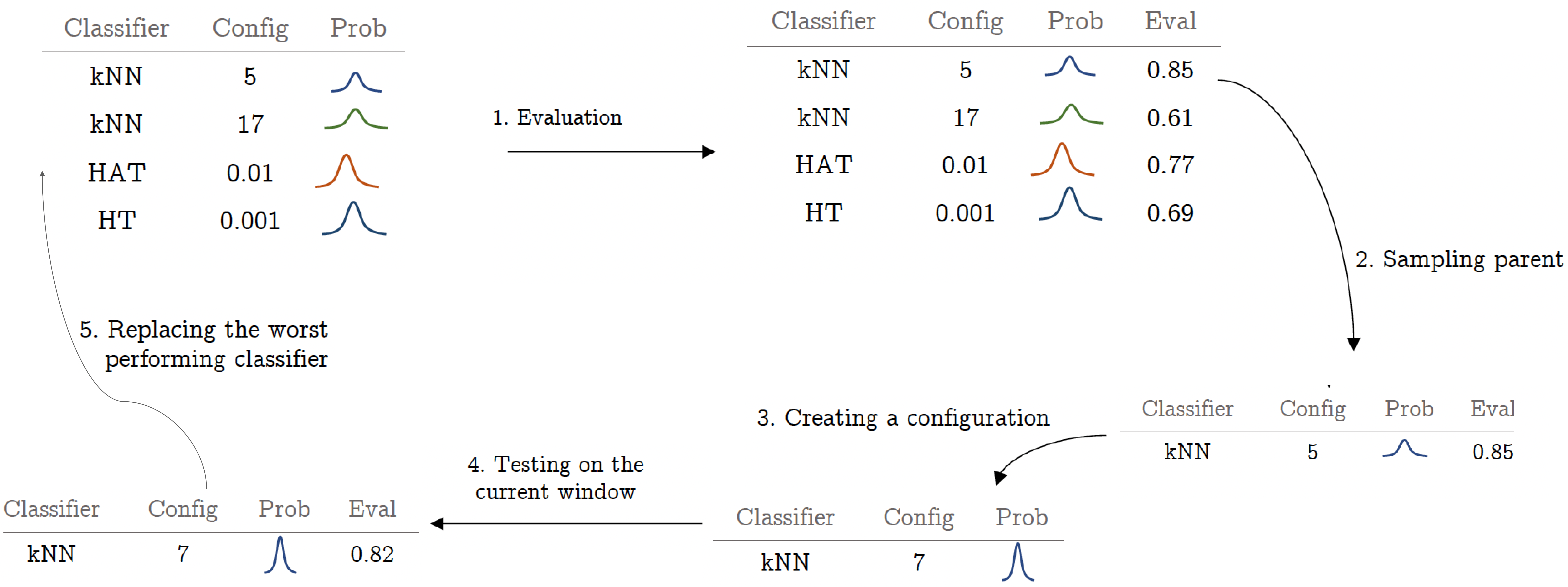
Training

AutoClass Approach

Mutation:

1. Select a good & weakest pipelines
2. Generate a new configuration by mutating the good one
3. Remove the weakest configuration

AutoClass


Classifier Config Prob
kNN 5
kNN 17
HAT 0.01
HT 0.001
1. Evaluation
Classifier Config Prob Eval
kNN 5 0.85
kNN 17 0.61
HAT 0.01 0.77
HT 0.001 0.69
2. Sampling parent
Classifier Config Prob Eval
kNN 5 0.85
3. Creating a configuration
Classifier Config Prob
kNN 7
4. Testing on the current window
Classifier Config Prob Eval
kNN 7 0.82
5. Replacing the worst performing classifier

Practical examples

05_ECML2024_automl.ipynb