OUKA
Osaka University Knowledge Archive

Title	Reinforcement Learning based Evolutionary Metric Filtering in Clustering
Author(s)	Ali, Ashour Mohamed Elsedfy Bassel
Citation	大阪大学, 2021, 博士論文
Version Type	VoR
URL	https://doi.org/10.18910/82282
rights	
Note	

Osaka University Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

Osaka University

Reinforcement Learning based Evolutionary Metric Filtering in Clustering

Submitted to

Graduate School of Information Science and Technology
Osaka University

January 2021

Bassel Ali Ashour Mohamed Elsedfy

Related Publications

1. **Journal paper**

   (a) Bassel Ali, Koichi Moriyama, Wasin Kalintha, Masayuki Numao, and Ken-ichi Fukui. Reinforcement learning based metric filtering for evolutionary distance metric learning. *Intelligent Data Analysis*, 24(6), pages 1345-1364, 2020.

2. **Peer-reviewed Conference paper**

   (a) Bassel Ali, Ken-ichi Fukui, Wasin Kalintha, Koichi Moriyama, and Masayuki Numao. Reinforcement learning based distance metric filtering approach in clustering. In *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI 2017)*, pages 1328-1335, 2017.

   (b) Bassel Ali, Wasin Kalintha, Koichi Moriyama, Masayuki Numao, and Ken-ichi Fukui. Reinforcement learning for evolutionary distance metric learning systems improvement. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO 2018)*, pages 155-156, 2018.

   (c) Bassel Ali, Koichi Moriyama, Masayuki Numao, and Ken-ichi Fukui. Reinforcement learning based evolutionary metric filtering for high dimensional problems. In*Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA 2020)*, pages 226-233, 2020.

3. **Other presentation**

   (a) Bassel Ali, Ken-ichi Fukui, Wasin Kalintha, Koichi Moriyama, and Masayuki Numao. Reinforcement learning based distance metric filtering approach in clustering. In *The 21st SANKEN International & The 16th SANKEN Nanotechnology Symposium, Osaka University*, 2017.

   (b) Bassel Ali, Wasin Kalintha, Koichi Moriyama, Masayuki Numao, and Ken-ichi Fukui. Reinforcement learning for evolutionary distance metric learning systems improvement. In 情報数理学シンポジウム *2019 (IPS2019)*, 2019.

   (c) Bassel Ali, Koichi Moriyama, Masayuki Numao, and Ken-ichi Fukui. Reinforcement learning based evolutionary metric filtering for high dimensional problems. In *IT*連携フォーラム *OACIS*第*39*回シンポジウム, 2020.

Abstract

Due to the growing amount of data, a technique that can direct attention to the important features can be useful. Conventional feature selection solves this problem by reducing the features number while Distance Metric Learning (DML) uses a distance metric over objects to offer insights on important features. However, in the previous work, feature selection and DML are performed separately and do not take advantage of each other's feedback. This research proposes a novel system that explores mutual feature selection and DML feedback and offers feature selection via metric filtering in the DML domain by combining Evolutionary Distance Metric Learning (EDML) and Reinforcement Learning (RL). EDML is a type of DML that relies on an evolutionary approach in its learning process to optimize its distance metric. RL is a learning technique used to derive a policy through a sequence of trials and errors and can explicitly select features for DML problems. In the proposed method, features represented by the elements of EDML distance transformation matrices are prioritized by a Differential Evolution algorithm. Then a selection control strategy using RL is learned by inserting and evaluating the prioritized elements. The proposed framework has the novelties of performing RL-based metric filtering in DML as well as adopting a two-way information exchange approach between RL and DML. In the first way, RL will learn and send feedback that will affect the EDML metric creation. In the second way, the evolutionary feature prioritizing of EDML is utilized by RL in its learning process. This directs attention to important portions of the input space in case the number of features is large and reaches good solutions with the aim to reduce the number of features while maintaining the clustering performance. A higher dimensional adaptation model is also explored to handle high dimensional DML problems. In this adaptation, a function approximation RL method creates the feature selection strategy to filter the metric while using a batch system to save DML evaluation time in high-dimensional input spaces. Moreover, different hybrid approaches are examined to explore different ways of information exchange. Additionally, Diagonal and Full distance metrics are both explored as well as different ways of formulating the RL policy. Results show a significant decrease in the number of features while maintaining accuracy.

Acknowledgement

I would first like to express my gratitude and appreciation to my main supervisor Associate Professor Ken-ichi Fukui who mainly guided and offered his valuable comments, expert advice, and ideas throughout my research and while writing this thesis paper. This research and thesis would not have been possible without his help.

I would also like to thank Professor Masayuki Numao for his support and help during my research. He provided me with valuable information and insights that led me in the right direction and always offered guidance whenever I needed it.

In addition to that, I would like to thank Associate Professor Koichi Moriyama for his valuable insight, support, and ideas that helped in forming the research structure from the beginning and his contributions during the research period. His patience and guidance made this research better.

I would also like to acknowledge the help of Wasin Kalintha who always offered valuable insights and opinions throughout the research and was always helpful whenever I needed his help. Also, I would like to offer my thanks to all the students in Numao Laboratory for their help and support of my work. I am grateful for their contributions and indebted to them.

I am also thankful for the support of the Network Joint Research Center for Materials and Devices in my research.

Finally, I express my profound gratitude and love to my parents and friends for their continuous encouragement and their support over the years from the beginning till now. This accomplishment would not have been possible without their help. I am grateful to them.

Contents

List of Figures **iii**

List of Tables **v**

1 Introduction **1**
1.1 Problem Overview 1
1.2 Research Contribution 3

2 Base Methods **6**
2.1 Evolutionary Distance Metric Learning (EDML) 6
2.1.1 Types of Distance Metric Learning 6
2.1.2 EDML Overview 7
2.1.3 EDML Process 8
2.2 Reinforcement Learning (RL) 9
2.2.1 RL Overview 9
2.2.2 RL as an MDP 11
2.2.3 Q-learning 12
2.2.4 Least-Squares Policy Iteration (LSPI) 13

3 Literature Review **15**
3.1 Conventional Feature Selection 15
3.2 Reinforcement Learning based Feature Selection 16
3.3 Feature Selection in High Dimensional Problems 18

4 Reinforcement Learning based Dimensional Metric Filtering (R-EDML) 20
4.1 Overview 20
4.2 Proposed Method 21
4.2.1 R-EDML Life Cycle 22
4.2.2 R-EDML Model as an MDP 25
4.2.3 RL Merge with EDML 30
4.3 Experiments and Results 31
4.3.1 Main Goal 31

4.3.2 Randomness Handling . . . 32
4.3.3 R-EDML Approaches . . . 32
4.3.4 Database . . . 47
4.3.5 Experiment settings . . . 47
4.3.6 Experiments . . . 52
4.3.7 Results and Observations . . . 54
4.3.8 Effect of the Acceptance Margin of F-measure . . . 56
4.3.9 Summary . . . 61

5 Adaptation to High Dimensional Metric Filtering (HDR-EDML) 62
5.1 Overview . . . 62
5.2 Proposed Method . . . 63
5.2.1 HDR-EDML Overview . . . 63
5.2.2 HDR-EDML Approaches . . . 65
5.3 Experiments and Results . . . 70
5.3.1 Database . . . 70
5.3.2 Experiment settings . . . 71
5.3.3 Experiments . . . 73
5.3.4 Results and Observations . . . 73
5.3.5 Summary . . . 76

6 Conclusion 78
6.1 Summary . . . 78
6.2 Discussion . . . 78
6.3 Research Applications . . . 80
6.4 Future Work . . . 81

List of Figures

1.1 Rapid data growth 1

2.1 EDML life cycle 8
2.2 EDML process 9
2.3 Reinforcement Learning life cycle 11
2.4 LSPI life cycle 14

4.1 R-EDML components 21
4.2 R-EDML life cycle (Diagonal R-EDML) 24
4.3 R-EDML RL phase pseudo-code 25
4.4 R-EDML satisfactory condition 28
4.5 R-EDML state action space 30
4.6 RL merge with EDML process 31
4.7 For each generation approach 33
4.8 For every N generations approach 34
4.9 Diagonal Change EDML approaches 35
4.10 Diagonal No Change EDML approach 36
4.11 Diagonal Resettable learning approach 37
4.12 Diagonal Appendable learning approach 38
4.13 Learn from Policy 39
4.14 Learn from Highest Accuracy 40
4.15 Frequent actions approach 41
4.16 Merge technique 42
4.17 Merge technique approaches combinations 43
4.18 Non-Merge technique approaches combinations 43
4.19 Selected Merge technique approach 44
4.20 Approaches mechanism part 1 - RL process between generation 1 and 2 45
4.21 Approaches mechanism part 2 - RL process between generation 2 and 3 45
4.22 Approaches mechanism part 3 - RL process between generation 3 and 4 46
4.23 Approaches mechanism part 4 - RL process of final generation 4 46
4.24 R-EDML margin evaluation graph - Wine data set 60
4.25 R-EDML margin evaluation graph - Glass data set 61

5.1 HDR-EDML process overview 65
5.2 ESARS approach 67
5.3 ESARS pseudo-code 68
5.4 L-LSPI approach 69
5.5 LSPI-1 approach 70

List of Tables

4.1 R-EDML UCI data sets 47
4.2 Class and cluster confusion matrix of data pairs 49
4.3 Approaches combinations filtering in Glass data set 51
4.4 Approaches combinations filtering in Wine data set 52
4.5 Approaches combinations filtering in Vehicle data set 52
4.6 Feature weighting results in EDML and ITML 56
4.7 ITML comparison results 56
4.8 F- measure comparison results (R-E refers to R-EDML) 57
4.9 Number of features comparison results (R-E refers to R-EDML) 58
4.10 Number of generations comparison results (R-E refers to R-EDML) . . . 59

5.1 HDR-EDML UCI data sets 71
5.2 Results comparison: F-measure(AVE±STD) 75
5.3 Results comparison: # Features (AVE±STD for EDML+l_1, LSPI-1, L-LSPI, and ESARS) 76
5.4 Run time comparison in low dimensional data (Total time (F-measure / # Features)) 76

Chapter 1

Introduction

1.1 Problem Overview

In real-world applications, massive amounts of data are created and need to be processed daily; however, the continuous growth of these data presents a challenge in the IT world regarding the ways to determine the important portions and select from such a large volume in an efficient and timely manner. Figure 1.1 shows the rapid growth of data in the last few years [1].

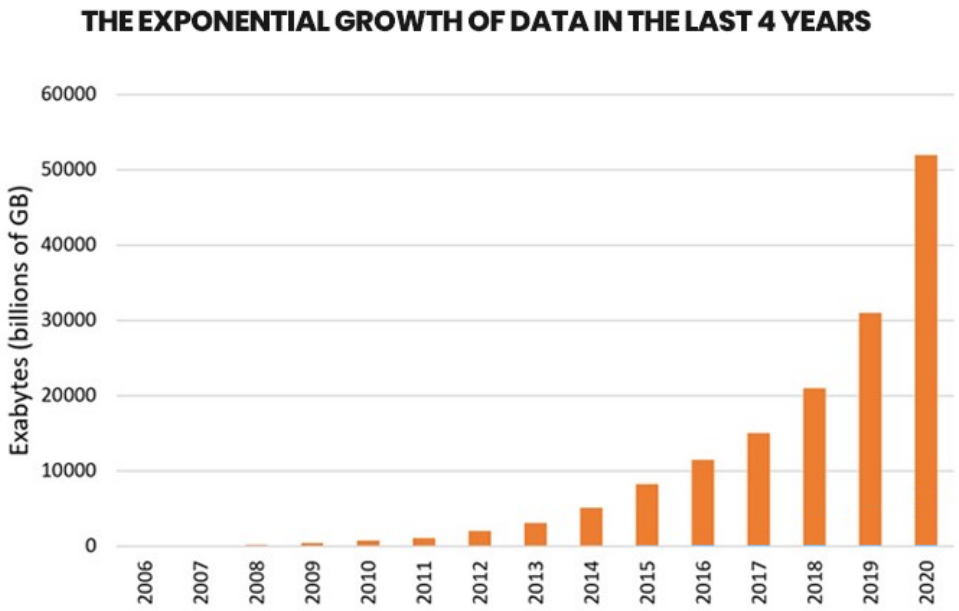

Figure 1.1: Rapid data growth

From the economic point of view, many organizations are unable to cope with the amount of data present in their warehouses and external data not in their possession.

[1] https://towardsdatascience.com

Because of the overload problem associated with them, a way to provide insights on the important features of the data is vital for business agility.

That is why today's data processing approaches and advances in computer science aim to extract important information only. To overcome the problems associated with data processing, machine learning, and data mining algorithms aim to process data and extract valuable information. However, these tools may be inefficient for the ever-growing amount of data over the last few decades. That is why data mining algorithms try to offer insights and select efficiently the important data features from large volumes of data.

Data processing algorithms in machine learning like clustering algorithms [1, 2] try to find structure in unlabeled data by grouping similar objects together and form a group. Forming these groups depends on what features the clustering algorithm uses to define what is similar and what is not. Another example is classification algorithms, classification is a predictive modeling problem where the model tries to predict the label for a given example of input data using the previously gathered labeled data.

In clustering algorithms, selecting the right features to group the data is very important as unimportant features increase the data processing time and can lead to worse results. Therefore, a way to provide insights into the important features of the input space plays an important role in clustering as it can lead to reducing the cost and time complexity of the system.

For that reason, feature selection [3, 4] can help in this problem. Feature selection can remove useless features from the original feature set and can alleviate high dimensionality problems as it reduces the search space and complexity. Also, regularization [5] is an effective approach in feature selection; for example, L_1 regularization is based on the L_1 norm as it can drive many parameters to zero and can avoid overfitting by reducing the model complexity.

In addition to feature selection, the distance between data points is very important and should be considered as it can give insights into the important features and can offer potential in feature reduction. Distance Metric Learning (DML) [6] is a machine learning technique that measures the distance between data points to capture important information. In clustering and classification, the definition of distance between data points greatly affects these tasks. Therefore, DML can help by learning a distance metric over objects to capture the important relationships among features. The learned

distance metric can be used to increase the accuracy of the classification and clustering tasks. However, DML uses elements in the matrix simultaneously, for which it requires access to all the features, including the unimportant ones in the learning process.

For that purpose input space filtering to reduce dimensionality is needed to DML. Methods like feature selection before DML can help reduce the DML dimensionality by reducing features. However, conventional feature selection processes neither put into account the DML evaluation with respect to the changed features nor the relationship between the selected features and their values in the metric. In addition to that, methods like L_1 regularization needs careful tuning of the hyper-parameter on each data set to balance between feature selection and performance of the task.

1.2 Research Contribution

This research offers a new way for input space filtering to reduce dimensionality in DML called distance metric filtering. The most difficult problem of distance metric filtering is that simultaneous optimization of the elements selection and their values are needed as well as combining the right number of features with the right distance metric.

To target these challenges in distance metric filtering, sequential feature selection with Reinforcement Learning (RL) [7] is introduced. RL is a machine learning technique that learns a policy according to feedback (reward) resulted from interaction with its environment. This work proposes RL-based feature selection in DML as a distance metric filtering technique and can include explicit feature selection to the DML process by learning the important features for any data set just by using the DML evaluation as feedback. This approach not only reduces the features but also selects the correct number of features along with their specific values in the metric. This is why this research has the potential to solve the challenges arising from distance metric filtering.

To provide a rich learning environment for RL in this research, Evolutionary Algorithms (EA) are introduced to the DML domain. The advantage of using EA is having multiple generations that can offer many ways of learning exchange for RL as well as using the EA information as input in the RL learning process. Furthermore, this evolutionary property is very unique in the DML domain as EAs are highly parallelizable and provide solutions to problems that mathematical programming finds hard to formulate. Other DML methods like Information-Theoretic Metric Learning (ITML) [8] can not provide such a rich environment as they possess neither the necessary generations nor

the evolutionary process that enhances the learning exploration possibilities for RL.

In this work, we propose a novel hybrid optimization framework in the DML field that combines a type of Evolutionary semi-supervised DML called Evolutionary Distance Metric Learning (EDML) [9] and RL-based feature selection called Reinforced EDML (R-EDML) [10, 11, 12]. EDML is a DML algorithm that adds EA in the DML process to optimize its transformation matrix.

Since EDML prioritizes features simultaneously instead of filtering or selecting them, RL can help in explicitly selecting and learning in a sequential manner not a simultaneous one like EDML, and can learn how to select features based on the accuracy resulted from EDML. This hybrid system can take advantage of the evolutionary feature weighting process and the optimization of the distance transformation matrix in EDML along with the RL decision making. This combination can direct attention to important portions of the DML input space in case the number of features is large and an optimal solution is hard to find, this allows for good sub-optimal solutions to be reached.

Another novelty is a mutual interaction and information exchange between RL and Evolutionary Algorithm (EA) in metric learning. This enables EA to focus the RL environment on the important features and enables RL to edit the input of EA before creating the next generation according to the learning feedback. This approach not only reduces the features but also selects the correct number of features along with specific transformations determined by the EDML EA. This enables clustering while scaling down dimensionality.

In real-world applications, the data is usually of high dimensional nature which presents a challenge to DML as it struggles to solve high-dimensional problems. In high-dimensional DML problems, high computational resources are not always available as a solution. Therefore, to target DML problems in high dimensional settings, an extension to R-EDML called High Dimensional Reinforced EDML (HDR-EDML) [13] is proposed. This extension is an adaption to handle high dimensional distance metric filtering problems where insights into the important features of high-dimensional input spaces are necessary as it can reduce the data processing time. HDR-EDML uses a function approximation RL learning process to create a feature selection control strategy in high-dimensional input space to filter the metric as well as a batch system that can reuse the evaluation obtained once.

Following this chapter, Chapter 2 discusses the literature regarding conventional

and RL-based feature selection. Chapter 3 describes the hybrid system base methods: EDML and RL. Chapter 4 describes the original hybrid method R-EDML along with its experimental results. Chapter 5 describes the extended hybrid method HDR-EDML along with its experimental results. Finally, Chapter 6 concludes this research.

Chapter 2

Base Methods

In this chapter, the proposed system's base methods will be described. Evolutionary Distance Metric Learning will be explained first, followed by Reinforcement Learning and the techniques used in it.

2.1 Evolutionary Distance Metric Learning (EDML)

2.1.1 Types of Distance Metric Learning

A variety of DML methods have been proposed [6, 14, 15]. Some of the methods include nearest neighbor classification [16, 17], image ranking [18, 19] and clustering algorithms [1, 2, 20, 21] with the aim to improve the clustering and classification accuracy by learning the distance metric from a data set. DML is divided into two learning techniques [6]:

1. Unsupervised DML: This can achieve dimensionality reduction as it identifies geometric relationships in the Euclidean data space. Additionally, it can convert the input space into a low dimensional space. Simultaneously, it can avoid losing data point relationships.

2. Semi-supervised DML: This uses auxiliary information like class labels and pairwise constraints of must-links and cannot-links in its learning. It aims to optimize a common metric transformation function by preserving similar classes and separating different classes.

2.1.2 EDML Overview

Evolutionary Distance Metric Learning (EDML) [9] used in this research uses a distance transformation matrix $\boldsymbol{M}$ to find the appropriate transformation of the input space where the diagonal elements of $\boldsymbol{M}$ represent scaling factors applied to corresponding features. EDML is a semi-supervised DML technique that uses an EA to optimize its $\boldsymbol{M}$. EDML relies on a clustering index with neighbor relation to evaluate inter- and intra-clusters and to optimize $\boldsymbol{M}$ based on the Mahalanobis distance defined as:

$$d_{i,j}^2 = (\boldsymbol{x_i} - \boldsymbol{x_j})^t \, \boldsymbol{M} (\boldsymbol{x_i} - \boldsymbol{x_j}) \tag{2.1.1}$$

where $\boldsymbol{x_i} = (x_{i,1}, \ldots, x_{i,v})^t$ is the i^{th} data point that has v-dimensional feature vector, and $\boldsymbol{M}$ is a $v \times v$ symmetric matrix. In EDML, matrix $\boldsymbol{M}$ is the variable to be optimized. This $\boldsymbol{M}$ comprises diagonal elements that represent the features weights and non-diagonal elements that represent the correlation between different features. The $\boldsymbol{M}$ optimization is conducted using self-adaptive differential evolution (jDE) algorithm [22]. Figure 2.1 describes the EDML life cycle which is summarized as follows:

1. jDE creates a generation of candidates which are the distance transformation matrices $\boldsymbol{M}$s, for each candidate $\boldsymbol{M}$ the following happens:

   (a) Using $\boldsymbol{M}$, the input space is transformed employing Mahalanobis distance in Equation 2.1.1.

   (b) The cluster structure is created by any clustering technique.

   (c) The clusters are evaluated with class labels and pairwise constraints using the clustering index like pairwise F-measure [23].

   (d) The evaluation result is sent back to jDE as the fitness for this candidate.

2. According to the fitness, jDE selects the individuals for the next generation using a probability-based mutation and cross over and creates the next generation.

3. Steps 1) and 2) are repeated until a threshold is satisfied, this threshold can be a certain number of generations or the desired accuracy margin. The result shows the matrix with the best performance among all its peers in all generations.

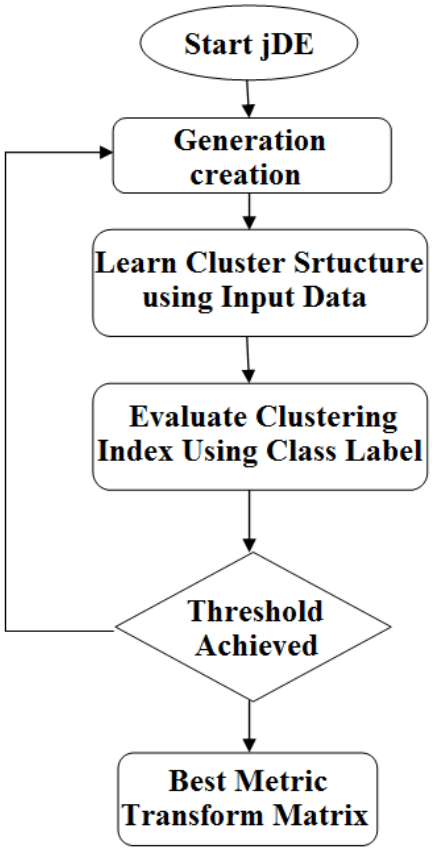

Figure 2.1: EDML life cycle

2.1.3 EDML Process

Figure 2.2 describes the process of the original EDML; the sequence of the events is top to bottom and left to right.

1. First EDML creates a new generation using the evolutionary algorithm, for example generation 1, this generation contains a population of distance transformation matrices: ***M1***, ***M2*** and ***M3***. This population of matrices is small just as an example.

2. Each matrix contains 5 diagonal elements which represent scaling values to the input space features; each element is represented by "x".

3. These elements are then used to transform the input space and a clustering operation is done and evaluated.

4. According to this evaluation, the evolutionary algorithm will either mutate the

current generation to create generation 2 or stop generating if a certain threshold is satisfied.

5. If the threshold is not satisfied after the evaluation, cross over and mutation will happen to generation 1 and generation 2 will be created, the red and orange elements are a simple representation of the changed values in the new generation after the evolutionary algorithm process is done.

6. This process will continue until a fixed number of generations are created or the threshold is satisfied. In this research, creating a generation, then doing clustering, evaluation, and then mutation to get the next generation is referred to as the EDML phase.

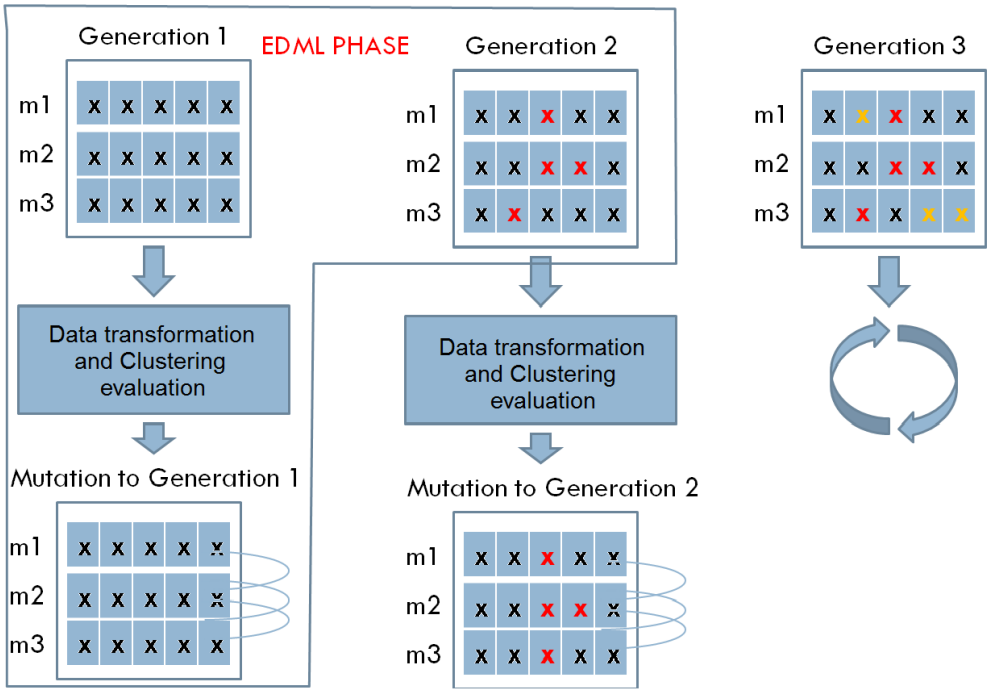

Figure 2.2: EDML process

2.2 Reinforcement Learning (RL)

2.2.1 RL Overview

Reinforcement Learning (RL) [7] is a learning technique that focuses on the interaction between an agent and the surrounding world. RL enhances the agent's behavior over time by learning from trials and errors. The agent produces an output that represents an action of the state of the world the agent is in. The state here represents the

virtual representation of the world at a specific time. Moreover, the world interacts with the agent's actions by giving a scalar value called a reward, informing the agent in this state how well its action is. The goal of the agent is to maximize the expected discounted cumulative reward. Reinforcement Learning is commonly used for control tasks in robotics, scheduling problems, or gameplay but not usually with data mining which this research is exploring.

One of the strongest points of RL is that a model of the environment is unnecessary. Through the agent's life span, the agent can learn a policy to follow through interaction and a reward system. The policy defines the agent's behavior at any given state. It is a function that maps any state of the environment with an action to be performed by the RL agent.

An episode is a sequence of interactions between the agent and the environment from the start state to the terminal state. The agent chooses an action using the policy derived from the value function, performs this action, and observes the reward of the environment. Afterward, the agent updates its estimate of the value function associated with the policy. Then, the optimal policy is inferred by choosing the highest state-action value. As for the learning process, RL can rely on tabular methods in finite input spaces or function approximation methods in huge or infinite spaces.

Figure 2.3 describes the RL life cycle: at each time step t, The agent executes an action A_t, reaches a state, and then receives an observation O_t and a reward R_t, according to these two the agent will execute an action, goes to another state and the process continues until the agent learns the optimal policy for the environment it is in.

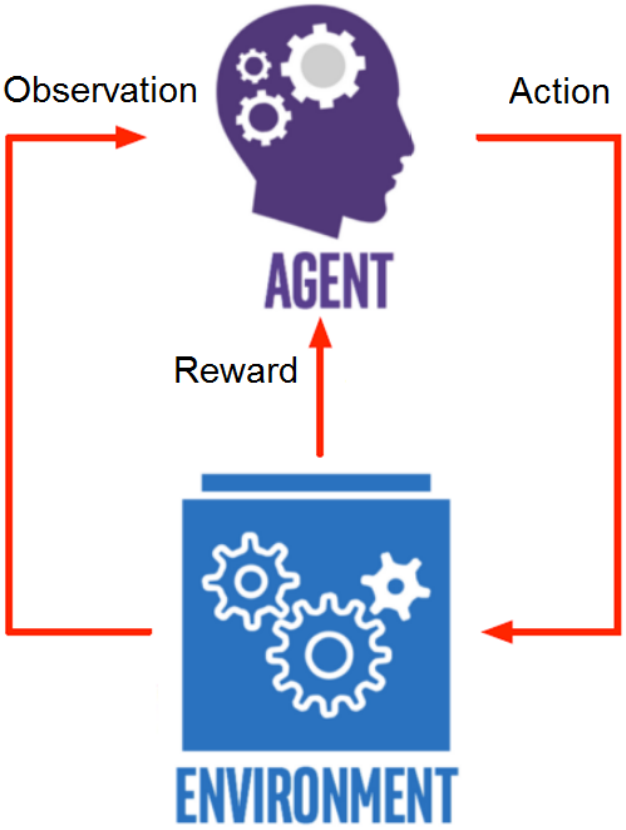

Figure 2.3: Reinforcement Learning life cycle

RL algorithms are divided into two types using the agent's information:

1. Model-based: where the agent creates a model of the environment and finds the optimal policy by performing a planning algorithm on the model.

2. Model-free: where the agent does not know the model of the environment. Regardless, the agent learns a policy by trial and error through a series of interactions with the environment.

2.2.2 RL as an MDP

Reinforcement Learning is based on Markov Decision Processes (MDP). Markov Decision Processes describe a fully observable environment, and every state in this environment has the Markov property, which means that any state can completely describe the history of all actions leading to this state. In other words, the state can be described as a sufficient statistic for the future.

Reinforcement Learning model can be described as an MDP model which has these main 4 elements:

1. $\boldsymbol{S}$: a set of states.
2. A: a set of actions.
3. $T(s, a, s')$: a transition function; the probability of transition from state s to state s' by action a.
4. $R(s, a)$: an expected reward of performing action a in state s.

2.2.3 Q-learning

Q-learning is a model-free RL technique that aims to maximize a value function Q. In Q-learning, the optimal policy is derived from the highest Q-value in the current state. This is carried out by iteratively updating the Q-value function.

Q-learning Process

In Q-learning the agent indulges in a sequence of state/action pairs, at the t^{th} step, the agent:

1. Observes the current state s.
2. Performs an action a.
3. Observes subsequent state transition $T(s, a, s')$.
4. Receives a reward $R(s, a)$.

This process is done for a number of episodes to optimize the Q-value function and to reach the optimal policy, described by the following equation:

$$Q\left(s,a\right) \leftarrow Q\left(s,a\right) + \alpha\left[r + \gamma max_{a'} Q\left(s^{'}, a^{'}\right) - Q\left(s,a\right)\right] \tag{2.2.1}$$

where s represents the state; a represents the action in state s; r represents the reward after taking an action in a state; γ $(0 \leq \gamma < 1)$ represents the discount factor, which determines how the sum of discounted rewards is calculated. Since future rewards are worth less than immediate rewards, the discount factor controls how future rewards are less than the immediate rewards. Finally, α $(0 < \alpha \leq 1)$ represents the learning rate, which controls the learning speed by controlling how the Q-values are updated. The low value refers to a slow update, which refers to slow learning, whereas the high value shows that learning can occur quickly.

2.2.4 Least-Squares Policy Iteration (LSPI)

LSPI Overview

Least-Squares Policy Iteration (LSPI) [24] uses a process called policy iteration to optimize its policy. Policy Iteration is an iterative process that uses a tabular representation to discover the optimal policy. It does that by generating a sequence of improving policies by following two steps: policy evaluation and policy improvement. The iteration converges to the optimal policy when there is no change in the policy evaluation process.

LSPI is an off-policy RL algorithm that combines value-function linear approximation and approximate policy iteration which uses a parametric representation to approximate the value function. LSPI relies on samples of experiences (tuples) to evaluate the policy. The data samples are in the form of state, action, reward, next state (s, a, r, s') known as SARS tuples. These data samples can be collected in any manner and reused efficiently by LSPI to evaluate the generated policies in all iterations and learn decision policies from them. Compared to other gradient descent methods [25, 26, 27], LSPI is sample efficient as it can reuse the same batch of experience samples in its policy evaluation and has no parameters to tune. The state-action value function $\widehat{Q}$ is approximated using a linear architecture:

$$\widehat{Q}(s, a; w) = \sum_{i=1}^{k} \phi_i(s, a) w_i \tag{2.2.2}$$

where ϕ is the feature vector, containing k features, describing the state-action pair. The weight vector w describes the weight of each element in the feature vector. The value function is the dot product of the feature vector and weight vector.

A greedy deterministic policy Π can be computed at any given state by the maximization of the approximate values overall actions in that state.

$$\Pi(s) = \text{argmax}_a \widehat{Q}(s, a) \tag{2.2.3}$$

LSPI Process

LSPI process in Figure 2.4 can be described as follows:

1. Data samples (s, a, r, s') are collected in any manner to cover the state-action space.

2. An initial random policy $\Pi(s)$ is created.

3. The collected data samples are used to approximate the value function of that policy (policy evaluation) using Equation 2.2.2 and update w by solving the linear system.

4. The policy is updated by choosing actions with the highest value functions (policy improvement) using Equation 2.2.3.

5. This process (steps 3 and 4) repeats for a number of iterations until the weights w of the approximate value function in the previous iteration does not change much from the weights $w^{'}$ in the current iteration according to a termination factor e, ($|w - w^{'}| < e$).

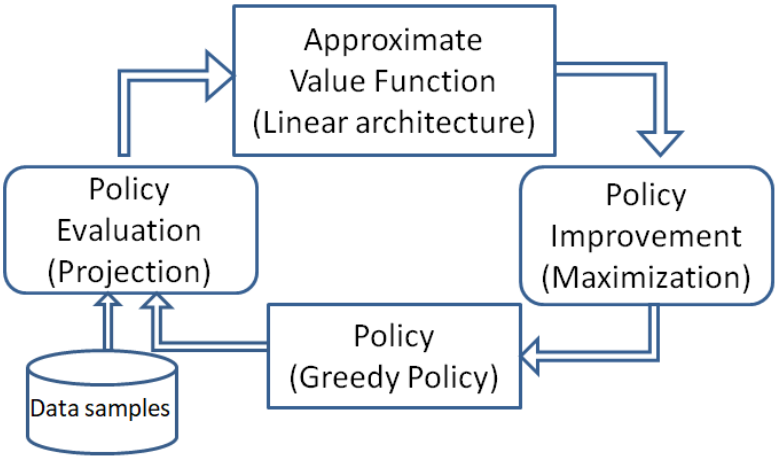

Figure 2.4: LSPI life cycle

Chapter 3

Literature Review

3.1 Conventional Feature Selection

Feature selection methods are categorized into 3 main classes Filter, Wrapper and Embedded [4]:

1. Filter class: Filter methods use a variable ranking approach for variable selection. They depend on general features like variable correlation for prediction or classification. This is done by suppressing the least interesting variables and selecting the rest for the prediction or classification task. The advantage of this method is time-efficient computations. The disadvantage of this method is that Filter methods ignore the variables relationships and this usually leads to selecting the same variables. Examples of Filter methods are:

   (a) Fisher method [28] which calculates the feature score as the ratio of interclass separation and variance, evaluates the features independently and collects the top-ranked ones as the final selected features.

   (b) RELIEF algorithm [29] which is a supervised approach that ranks the features using a relevance criterion and according to a threshold, it selects a subset of features.

2. Wrapper class: Unlike the previous class, Wrapper methods keep the relations between variables, they do so by choosing a subset of variables instead of individuals and this facilitates the detection of possible relations between these variables. However, they are time inefficient and suffer from the risk of overfitting if the number of observations is low. When a greedy search is used, they fail to obtain

optimal solutions. Examples of Wrapper methods are:

(a) Genetic Algorithm (GA) [30] which can be used to find the features subset, wherein the selected features can be represented by the chromosome bits.

(b) Particle Swarm Optimization (PSO) [31] which is an evolutionary technique that is based on swarm intelligence where each particle in the swarm can represent a solution and PSO can search for the optimal one by updating the velocity and the position of each particle.

3. Embedded class: The last class is a combination of the previous classes, so it tries to take advantage of the filter method variable selection process and does the feature selection and classification tasks simultaneously. An example of Embedded methods is SVM-RFE [32] which is a Suppor Vector Machine (SVM) [33] based feature selection. SVM-RFE performs a sequential feature selection in a backward elimination manner. Using a linear SVM, the features with high scores are the ones that separate the samples.

Regularization [5] is another effective approach in feature selection; for example, L_1 regularization is based on the L_1 norm as it can drive many parameters to zero and can avoid overfitting by reducing the model complexity. Given a model, we define the loss function L as the error of the model. This error is the difference between the true value y and the predicted value $\hat{y}$:

$$L = Error(y, \hat{y}) \tag{3.1.1}$$

By adding the L_1 regularization term to the calculation of the loss function, the model can avoid overfitting by adding constraints in L. Given n as the number of model variables, w as their weights, and λ as the regularization parameter that needs to be tuned, the loss function with L_1 regularization is defined as:

$$L = Error(y, \hat{y}) + \lambda \sum_{i=1}^{n} |w_i| \tag{3.1.2}$$

3.2 Reinforcement Learning based Feature Selection

RL has been successful when applied to feature selection. RL is chosen because of its ability to learn a strategy attentive towards the important parts of the data set which

can be useful for large data sets. RL-based feature selection transforms the feature selection process from a pre-processing step to an online sequential feature selection process. RL first selects the feature and depending on the current state of the machine learning model it interacts with, RL will select another feature or stop the selection process. This not only leads to feature reduction but also can speed up the learning process. The RL main applications in feature selection are discussed below:

Dulac-Arnold et al.[34] proposed an approach that used policy iteration approximation in classification, where the policy was redefined at each step until it converged. These authors converted classification into a sequential process where RL selected the features and classified the input into one of the available classes. In that way, classification and feature selection were done by a single component.

Norouzi et al.[35] suggested an RL-based attention control strategy for image recognition, in which a sequential block-based approach was used to increase the correct classification rate of partially occluded faces. In this approach, the faces were partitioned into blocks and their importance in the classification task was learned by an RL agent who learned the exact number and order of blocks needed for correct classification. This approach could reduce the number of features needed for image recognition especially if the image was incomplete or impartial.

T. Rückstieß [36] suggested RL-based sequential online feature selection in supervised learning domains to convert classification into a sequential decision process. The approach sequentially fed features to the classifier until a correct classification was achieved. This helped reduce the number of features in the classification process.

Hachiya et al. [37] proposed a new framework that used filter-type feature selection for RL. They used the conditional mutual information as feature selection to evaluate the independence between return and state-feature sequences. The conditional mutual information was approximated by a least-squares method.

Janisch et al. [38] tackled the problem where feature collection was costly with the goal to optimize a trade-off between the expected classification error and the feature cost. They defined the problem as a sequential decision-making problem and used Deep Q-learning [39] where individual actions are either requesting the feature values or terminating the episode by providing a classification decision. They used neural networks for value function approximations and showed that their approach outperformed the most recent methods specifically designed for the costly features classification.

To the best of our knowledge, none of the past RL-based feature selection researchers have tried to simultaneously optimize the DML elements selection and their values nor have used RL as a metric filtering technique in DML.

This research in comparison explores this problem in clustering and it does not just select the features; it selects the correct number of features along with specific scaling factors for each feature determined by the EDML EA in the transformation metric, thus filtering the distance metric.

3.3 Feature Selection in High Dimensional Problems

In real-world applications, data is usually of high dimensional nature in which selecting relevant information is an important task. In high-dimensional DML problems, the input space requires data processing resources with high computational power. Previous methods have been applied in the high-dimensional metric learning domain, where they handled high-dimensional data by using parallel computing [40], or reducing the distance metric into a low-rank matrix [41, 42], or adopting sparse learning by regularization [43]. Other methods have applied feature selection along with DML as discussed below:

Nguyen et al. [44] introduced an online learning algorithm using sparse coding for feature selection in high-dimensional spaces and applied it to simulated and real robotics domains. They created an MDP formulation that incorporates a principled way to factorize the state space compactly while capturing the comprehensive transition and reward dynamics information. In their work, they separated the state-attributes that defined the state from the informative state-features and applied feature selection on a large number of state features to capture the transition dynamics, while maintaining a compact state space.

Nezhad et al. [45] proposed a feature selection method based on deep learning architecture for high-dimensional clinical data and applied it to a specific medical problem to decrease the risk of heart disease. They used individual clinical data with many features and stacked auto-encoders for feature representation in higher-level abstraction. This approach applied deep learning to identify personalized features to control and predict the amount of left ventricular mass indexed (LVMI). This helped to identify significant risk factors affecting LVMI to body surface area.

C. Lian et al. [46] constructed a low-dimensional transformation matrix for dimensionality reduction of high-dimensional settings. They used $L_{2,1}$-norm sparsity regular-

ization of this low-dimensional transformation matrix to act as feature selection. To control the feature selection process, they relied on a hyper-parameter that controls this regularization and limits the influence of unreliable features.

Similarly, G. Kunapuli et al. [47] used a trace-norm, like the L_1 norm to introduce sparsity into the eigenvalues of the distance metric, thus performing feature selection along with metric learning. They relied on a hyper-parameter which controls the sparsity of the learned metric for input-space feature selection.

Compared to black-box methods like neural networks, this research uses LSPI as the RL method for feature selection in high-dimensional problems. LSPI is easy to implement and use, and its analysis and debugging are fairly transparent. Furthermore, LSPI can reuse samples previously obtained thus speeding up the learning process.

As for L_1 regularization methods, they need careful tuning of the hyper-parameter on each data set to balance between feature selection and performance of the task. If the hyper-parameter is too small it may fail to control the influence of unreliable features and if too big, it can remove important features. In comparison, our approach can learn the important features for any data set just by using the DML evaluation as feedback.

Chapter 4

Reinforcement Learning based Dimensional Metric Filtering (R-EDML)

4.1 Overview

The main goal of this research is to develop a technique that can direct attention to specific portions of data by creating a feature selection control strategy that aims to optimize the input space by reducing the number of used elements (features) in $\boldsymbol{M}$ of Equation 2.1.1 without affecting the accuracy.

To achieve this goal, a novel approach is tested by creating a new hybrid system called Reinforced EDML (**R-EDML**) [10, 11, 12] with two main components as shown in Figure 4.1:

1. **R** comes from Reinforcement Learning which is used as a control strategy model.
2. **EDML** comes from Evolutionary Distance Metric Learning which is a Distance Metric Learning technique and in this research, EDML is used in clustering.

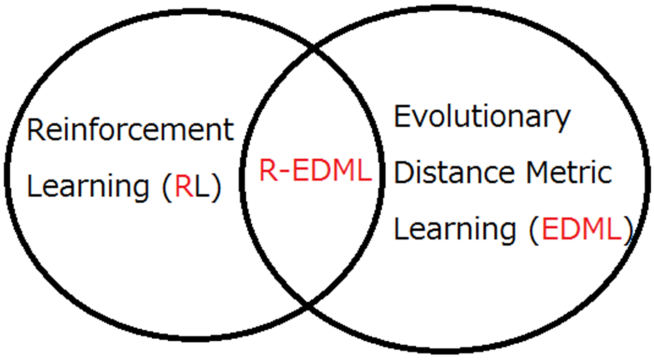

Figure 4.1: R-EDML components

In this hybrid system, it is important to use RL and EDML, EDML optimization mechanism combined with the RL decision-making process can create new ways of learning.

1. What EDML can achieve with RL:

   (a) EDML has its own feature prioritizing process which can be taken advantage of and used as a prefiltering process before RL-based metric filtering.

   (b) EDML has multiple generations which allow RL to explore in multiple environments and exchange feedback.

   (c) EDML has its EA which can be edited using RL to explore different variations in the evolutionary process.

2. What RL can achieve with EDML:

   (a) RL uses the EDML generation and matrices as an environment, it learns according to the DML evaluation which is based on the metric values so the feature selection process is related to specific metric values to these features.

   (b) The RL space can be changed according to the EA, so the feature selection process can be focused on the output of the EA by customizing the RL space.

Without using RL or EDML, RL and EDML unique feedback exchange will not happen and none of these advantages can be achieved.

4.2 Proposed Method

Since the Evolutionary Algorithm (EA) in EDML constantly mutates and changes the elements inside the distance matrices for every generation it creates. The approach

is to merge the RL sequential decision process in the EDML evolutionary generation process, taking place by inserting the elements in a sequential manner in the matrices and by learning the correct number of elements for any given EDML generation. This information is either used to change the generation itself or just record the data for later comparisons. RL compared to other feature selection algorithms is suited best for EDML as RL can learn a feature selection control strategy, unrequired for a model of the environment, tailored to each EDML generation.

To create the feature selection control strategy to filter the metric, R-EDML uses a tabular RL method called Q-learning (described in Section 2.2.3). Q-learning is chosen for its ability to learn an optimal solution given enough exploration which is not a problem in non-high-dimensional settings.

In this study, we focus on two types of EDML matrices: One is diagonal EDML where R-EDML will select from only the diagonal elements as they represent the features. The second is the Full Matrix EDML where R-EDML selects from both diagonal and non-diagonal elements to use the full power of EDML matrices transformations.

Following this, the R-EDML life cycle of this model is described, then R-EDML modeling the problem as an MDP is introduced as RL is based on Markov Decision Processes (MDP). Finally, how RL merges with the original EDML process described in Figure 2.2 is shown.

4.2.1 R-EDML Life Cycle

R-EDML uses the EDML generation as the RL environment. The overall R-EDML life cycle is divided into two phases:

1. The EDML phase
2. The RL phase

Both phases run after each other in a loop for a specific number of generations. The EDML phase prepares the generation for the RL phase, whereas the RL phase runs between steps 1) and 2) in the EDML cycle (described in Section 2.1.2) where it learns which subset of features to be used, then performs steps a) through d) in the EDML cycle. RL then gives feedback to the EDML phase to create the new generation. For every iteration, before the RL phase starts, the highest EDML F-measure so far (described in Section 4.3.5) will be recorded to be used in the satisfactory condition

that RL uses in its learning process. The detailed steps are as follows:

1. **EDML phase:**

   EDML creates a generation G of candidates then evaluates them using the K-means clustering algorithm and selects the elite results for the new generation using an EA. This generation's population is a set of distance matrices $\boldsymbol{M}$*'s* responsible for the data set transformation (The diagonal elements of $\boldsymbol{M}$*'s* correspond to the features in the data set, whereas non-diagonal elements correspond to the correlation between different features).

2. **RL phase:**

   (a) Elements of each $\boldsymbol{M}$ are stored for future reference. In the case of Diagonal R-EDML, the diagonal elements for each $\boldsymbol{M}$ are stored. In the case of Full Matrix R-EDML, both the diagonal and non-diagonal elements are stored.

   (b) All the elements in all $\boldsymbol{M}$s for this generation are reset to zero.

   (c) Several RL episodes will start, in each episode:

      i. RL learns which elements to select by sequentially inserting the elements in $\boldsymbol{M}$s and using the clustering evaluation with each insertion as learning feedback to know the element's importance.

      ii. According to this evaluation, RL either stops inserting new elements or continues. The termination of each episode depends on either achieving the satisfactory condition after evaluation or inserting all the elements.

   (d) From these episodes, RL learns which subset of elements $\boldsymbol{M}_e$ give the best accuracy $\boldsymbol{M}_a$ for the best $\boldsymbol{M}$ in G.

   (e) The selected elements will either be saved for later comparisons or will be fed back to EDML and used in creating the next generation.

   (f) The feedback (important features learned) from the RL phase to the EDML phase is constructed in two ways: Change EDML and No Change EDML.

      i. **Change EDML:** the current generation is changed before being passed to the EA, this change is:

         A. Resetting the non selected elements to zero or decreasing their value by a certain fixed ratio.

B. Setting the selected elements to their original values.

ii. **No Change EDML:** the result is not fed to EDML, and all the elements are returned to their original values. EA will continue independently from RL.

EDML phase will use EA to create a new generation and RL will start learning a subset of elements for the new generation. The cycle repeats for n generations. Given the number of selected elements of the best $\boldsymbol{M}$ in all generations so far ($\boldsymbol{M}^*$) in terms of fewest features and close EDML accuracy $EDML_a$ up to a margin β (small positive value) as $\boldsymbol{M}_e^*$. The selection of the new $\boldsymbol{M}^*$ satisfies both these conditions [$(EDML_a - \boldsymbol{M}_a) \leq \beta$] and [$\boldsymbol{M}_e < \boldsymbol{M}_e^*$]. After n generations, $\boldsymbol{M}^*$ is the R-EDML process output.

Figure 4.2 shows an example of Diagonal R-EDML life cycle with each generation having a population of 3 matrices ***M1***, ***M2*** and ***M3***, each with different color (green, red and blue) to uniquely distinguish their 3 diagonal elements (features) . Grey cells represent non diagonal elements while white cells represent empty value cells., whereas Figure 4.3 shows the R-EDML RL phase pseudo-code.

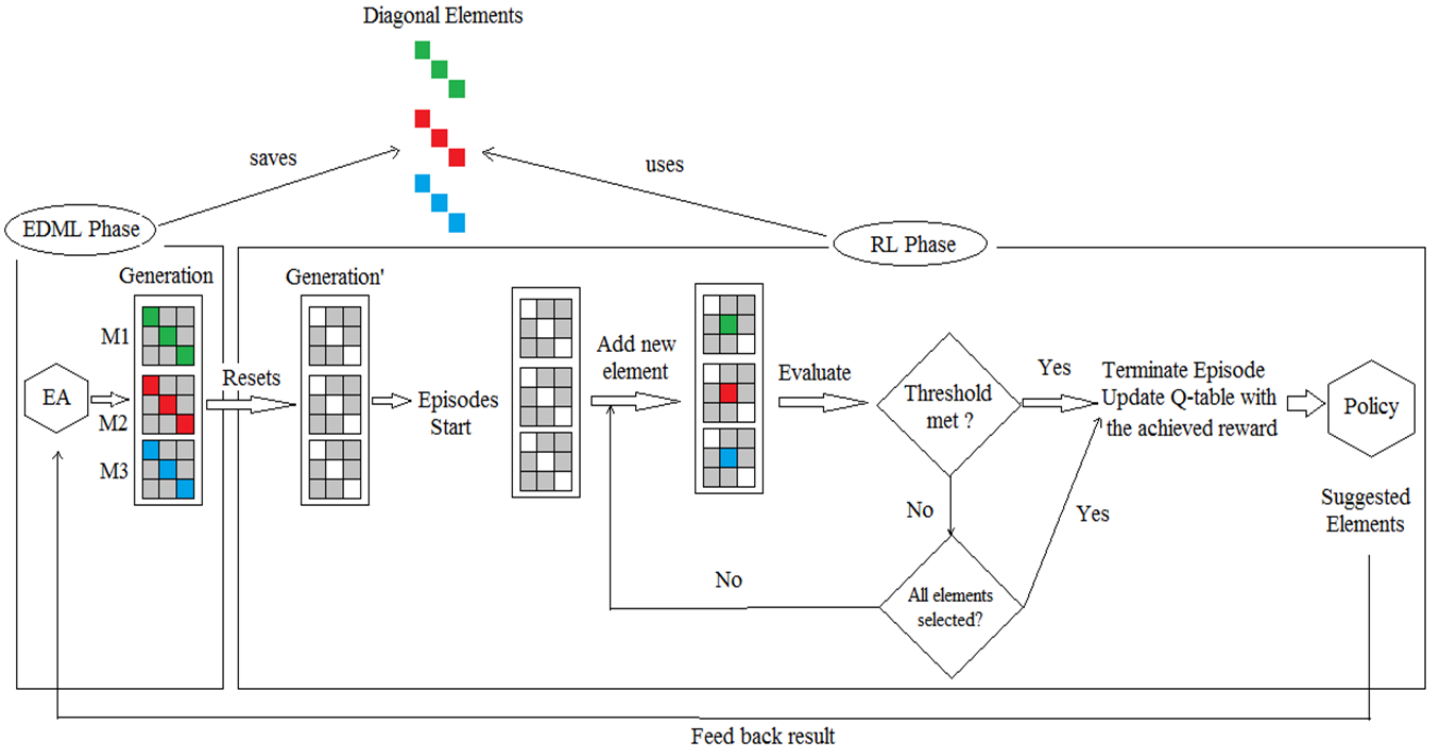

Figure 4.2: R-EDML life cycle (Diagonal R-EDML)

```
Foreach Generation / N Generations Do
  For 1 to Number_of_Episodes Do
      Reset Generation elements
      Set r to 0
      For all M elements in the generaton Do
        Select an unselected element using epsilon-greedy method
        Set r to r - AP
        Evaluate using K-means
        If satisfactory condition was met
          Break
        End If
      End For
      If satisfactory condition was met
        Set r to r +SR
      End If
      Update Q(s,a) with r
  End For
  Return Generation to initial state or change it
End For
```

Figure 4.3: R-EDML RL phase pseudo-code

4.2.2 R-EDML Model as an MDP

RL is based on MDP processes characterized by the Markov property. In MDP, every Markov state captures all the relevant information from history and independently describes the sequence of states leading to this state in the environment. Therefore, R-EDML models the problem as an MDP, MDP modeling has the following components:

1. States
2. Actions
3. State Transition function
4. Reward function
5. Terminal function

As for R-EDML, the following concepts are described:

1. Satisfactory condition
2. Policy

The states definition (s) is the current selected elements in the EDML transformation matrices $\boldsymbol{M}$s. The state is described by saving the indices of these elements in $\boldsymbol{M}$s.

The actions definition (a) is selecting the index of an element in $\boldsymbol{M}$s. Therefore, in diagonal EDML, selecting a feature is represented by inserting the values of the selected diagonal element index to $\boldsymbol{M}$s. In this process, an already selected element cannot be selected again.

The state transition function definition $[T(s, a, s')]$ is described as follows: from a state s with certain elements in the matrices, the agent will take an action a by selecting an unselected element; then, this element is inserted in the generation's matrices and a new state s' is created and evaluated according to the newly inserted element. The environment in this model is deterministic so this is a deterministic transition, i.e., $T(s, a, s') = 1$ for the new state s' under the state-action pair (s, a).

The final goal of the agent is the maximization of the expected reward, in other words, maintaining the EDML accuracy with the least number of actions (elements) possible, the reward function and terminal functions are constructed in the following manner to achieve that. The environment in this model is deterministic which makes the expected reward deterministic as well.

The reward function definition $[R(s, a)]$ is described as follows: For every action, a small negative reward called action punishment (AP) is given to the agent, this influences the agent's behavior to use as few elements as possible while maintaining accuracy. A positive reward called satisfactory reward (SR) is given to the agent if the satisfactory condition is met and this terminates the episode. The total reward after an episode finishes is described as:

$$R_{total} = E_{selected} \times AP + SR \tag{4.2.1}$$

where $E_{selected}$ is the number of the selected elements after the episode is finished. While the episode is running, given the satisfactory condition as θ, a set of selected elements is denoted as B and the currently selected element after the t^{th} generations of EDML is denoted as m^t; the value assignments of AP and SR are described as:

$$SR = \begin{cases} 10, & \text{if } \theta \text{ is satisfied} \\ 0, & \text{otherwise} \end{cases}$$

$$AP = \begin{cases} 0, & \text{if } \mathrm{m}^t \in B \\ -1, & \text{otherwise} \end{cases}$$

The terminal function definition is described as follows: The episode in R-EDML is terminated if the satisfactory condition (θ) is met or if all the elements in $\boldsymbol{M}$ are selected.

The satisfactory condition (θ) is the criteria by which the reward and terminal functions judge how good or bad a certain state is. This condition focuses on reducing the number of features while obtaining an accuracy close to the EDML accuracy up to a certain range. Given the F-measure (described in Section 4.3.5) after performing an action as F_{1a}, the condition is satisfied if these conditions are met:

1. The F-measure (F_{1a}) is close to a fixed margin ϕ as compared with the best EDML accuracy [$(F_{1a} - \textit{Best Accuracy}) \geq$-$\phi$]. This ensures that F_{1a} can have any value higher than the best EDML accuracy but limits the best EDML accuracy when it is higher than F_{1a}. This means the Best accuracy will not exceed ϕ if it is higher than F_{1a}, whereas F_{1a} will exceed ϕ if is higher than the Best accuracy.

2. The number of elements of F_{1a} is equal to or less than the fewest elements recorded to achieve the best accuracy.

Figure 4.4 shows a detailed description of the current satisfactory condition

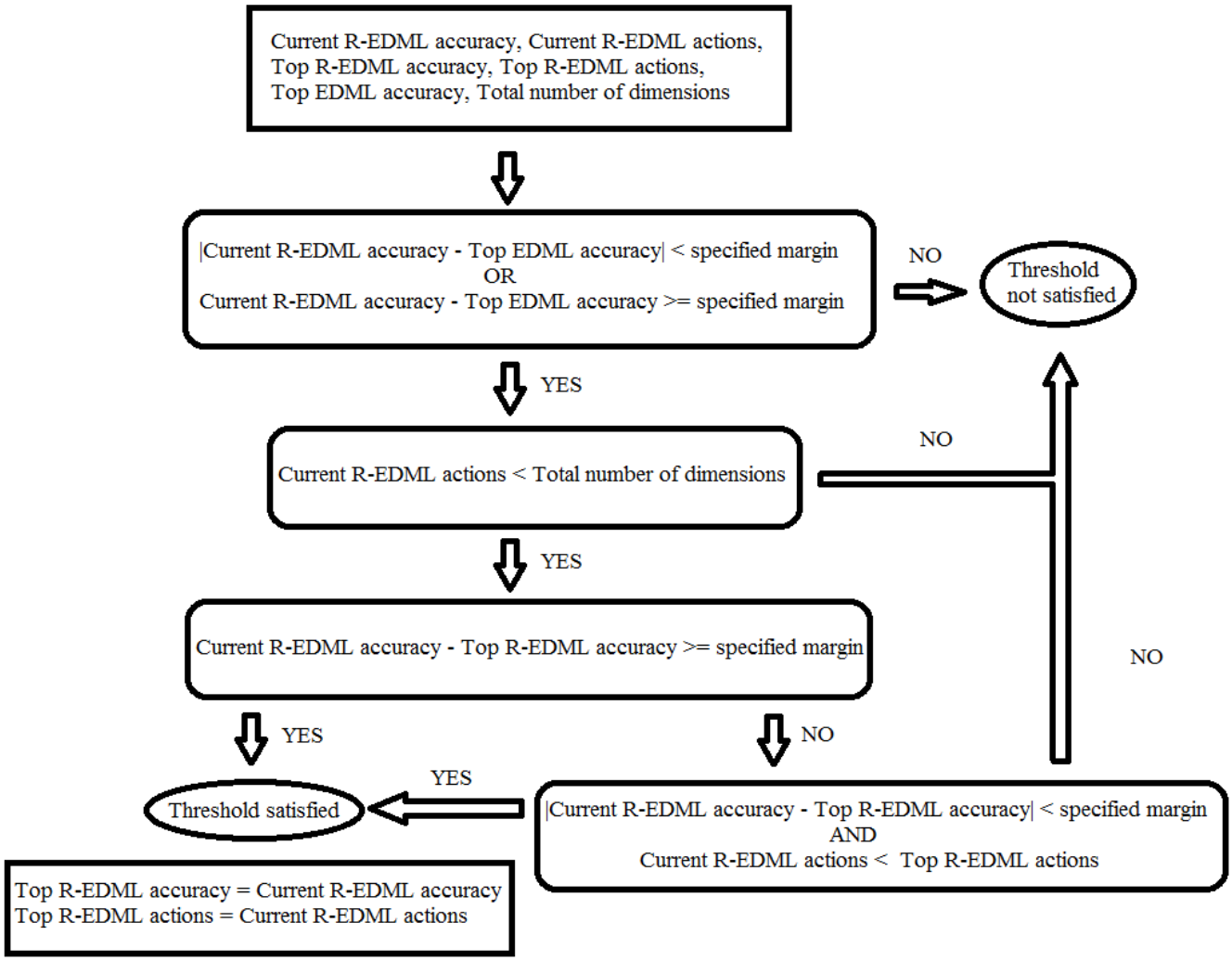

Figure 4.4: R-EDML satisfactory condition

After both conditions are met, the values of the current best EDML accuracy and the fewest elements recorded are updated according to F_{1a} to improve the agent performance by imposing a harder goal in the next phases. The comparison in the satisfactory condition shows our acceptance of the slight degradation as well as the improvement of evaluation score within the accepted margin as long as the number of features is reduced. **The policy** (Π), is applied in two ways:

1. The agent learns a new policy for each generation (**policy separation**) based on the assumption that EDML EA mutates and changes the elements' values after each generation, so a new policy tailored for each generation is tested.

2. A unified policy across all generations is used that is continuously updated for all

generations (**policy unification**).

The policy used in this research is an Epsilon greedy policy, which allows the agent to be greedy for rewards with a probability of $1 - \epsilon$ and also lets the agent explore as well with a probability of ϵ. This greedy exploration method is adopted because it is believed that an optimal solution is not guaranteed if the number of features is large. Given state s, action a, $A(s)$ as the set of available actions in s where $a \in A(s)$ and A^* as the action with the highest value function where $A^* \leftarrow argmax_a Q(s, a)$. Epsilon greedy policy Π can be described as follows:

$$\Pi(a|s) = \begin{cases} 1 - \epsilon + \epsilon/|A(s)|, & \text{if } a = A^* \\ \epsilon/|A(s)|, & \text{if } a \neq A^* \end{cases}$$

An example of R-EDML's state action space in a generation with 3 matrices is described in Figure 4.5. In this example, there are 4 states $s1$, $s2$, $s3$ and $s4$, every state represents a generation with a population of 3 matrices ***m1***, ***m2*** and ***m3*** and each matrix has the indices of 3 elements e1, e2, and e3. From each state s, different numbers of possible states can be created according to which element is selected to be inserted. With each action, a small negative action punishment (AP-) is given to the agent and a big satisfactory reward is given (SR+) if the satisfactory condition is met. The episode is terminated if the satisfactory condition is met or all elements are selected.

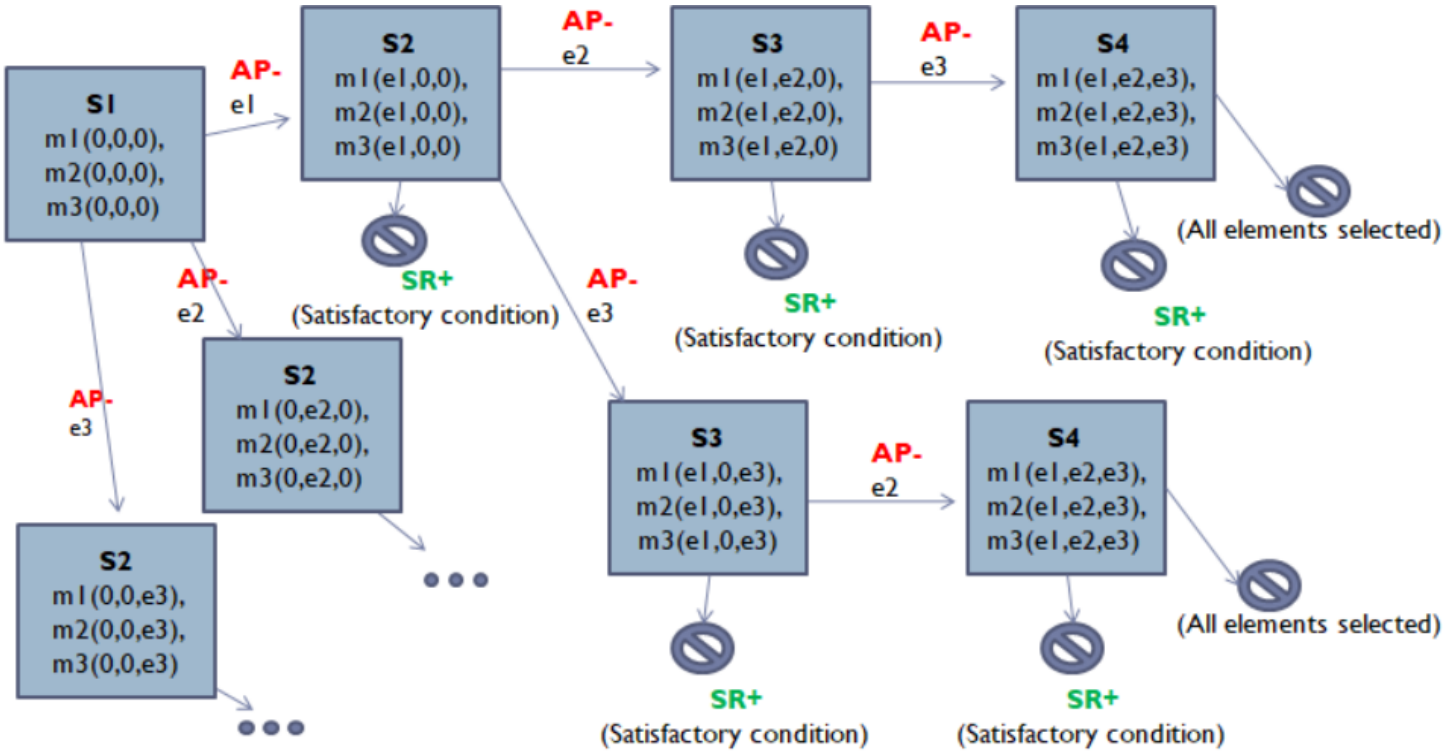

Figure 4.5: R-EDML state action space

4.2.3 RL Merge with EDML

Figure 4.6 shows how RL merges with the original process of EDML (described in Section 2.1.3), the sequence of the events is left to right, top to bottom.

1- RL will be inserted between EDML generations, so after each EDML phase (which is creating a generation, then doing clustering, evaluation, and then mutation to get the next generation), the RL will start.

2- Each generation in this figure has 3 distance transform matrices: ***M1***, ***M2*** and ***M3*** and the diagonal elements are denoted by "x" and the red elements are an indication of the changed elements due to the evolutionary process of EDML.

3- First thing before RL start is that the diagonal elements will be reset, which is denoted by empty cells.

4- RL will start the learning episodes, for example in this figure only 2 episodes will be run. In each episode, RL will sequentially insert the original diagonal elements and evaluate, and according to the evaluation, either a new element will be added or not.

5- After all the episodes finish, Q-learning will learn a policy (from Equation 2.2.1) which will suggest a number of elements for this generation, in this example the policy

suggested elements whose indices are 2 and 4. In this research, this phase is the RL phase which is taking an EDML generation, performing learning episodes, and learning.

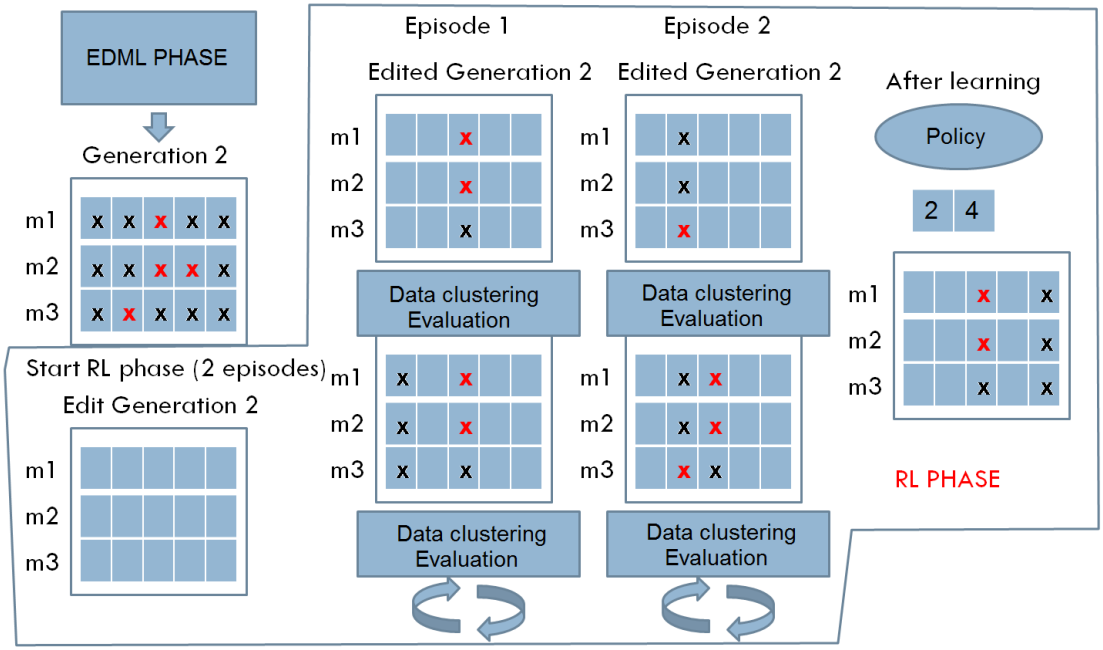

Figure 4.6: RL merge with EDML process

4.3 Experiments and Results

4.3.1 Main Goal

The goal of this section is to test the effect of the learned selection control strategy on the EDML transformation matrices. F-measure (F_1 score) described in Section 4.3.5 is used for clustering evaluation in the following experiments. To examine the different ways of combining EDML and RL in this hybrid system, a series of variations, approaches, and experiments are tested on different real data sets. K-means is the clustering algorithm used in this study using a K-nearest neighbor graph of cluster centroids.

The results of R-EDML are compared with the embedded feature weighting conducted by EDML, which prioritizes the features, and also with the EDML accuracy [10] as well as conventional feature selection [11] to compare R-EDML with the original EDML as well as feature selection applied on EDML in terms of features number and accuracy. Moreover, policy separation is applied to all tests apart from the policy unification test, whereas Diagonal R-EDML is applied to all tests apart from the Full Matrix R-EDML which is compared to Information-Theoretic Metric Learning (ITML) [8].

4.3.2 Randomness Handling

Both RL and EDML have a random factor in their processes. In RL, the Epsilon greedy policy has a random factor for the exploration process, as for EDML, the evolutionary algorithm has a random factor in the mutation and cross over processes that change and optimize the values by creating new generations. In addition to that, EDML starts with an initial random population. To handle this randomness and to limit its effect and ensure a unified start for all tests, two approaches are adopted:

1. Instead of using the initial randomly created generation that EDML creates, a unified initial population is created with fixed random values to ensure the same starting point for all tests. This fixed initial generation does not affect the evolutionary process in any way.
2. Each approach for each test will be run multiple times and the average result is taken.

4.3.3 R-EDML Approaches

Since RL has not been used with EDML before, a multitude of approaches are tested and compared to examine different ways of information exchange between RL and EDML, the approaches are as follows and will be described each in details:

1. For each generation / N generations
2. Change / No Change EDML
3. Resettable / Appendable learning
4. learn from Policy (P) / Highest Accuracy (HA)
5. Frequent actions feature
6. Merge technique

Approaches Description

1. **For each generation / N generations**:

   Since the RL phase runs on matrices, it can run any time independent from the EDML phase. Thus, we view the effect of how often the RL phase runs.

(a) For each generation: Run RL for each EDML generation. Each EDML phase creates 1 generation and after the EDML phase finishes, RL episodes will run (RL phase).

(b) For every N generations: Run RL each time EDML finishes N generations. Each EDML phase creates N generations and after the EDML phase finishes, RL episodes will run (RL phase).

Figure 4.7 describes For each generation approach with each generation having a population of 3 matrices and each has 5 diagonal elements, after each EDML phase (1 generation), the RL phase will be run. The purpose of this approach is to check how RL can learn with each generation EDML creates. Figure 4.8 describes For every N generations approach with each generation having a population of 3 matrices and each has 5 diagonal elements, after each EDML phase (N generations), the RL phase will be run. The purpose of this approach is to check how RL can learn not with each generation EDML creates but after a batch of generations.

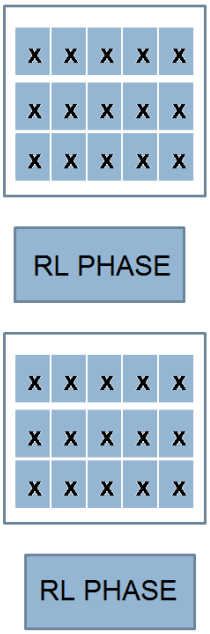

Figure 4.7: For each generation approach

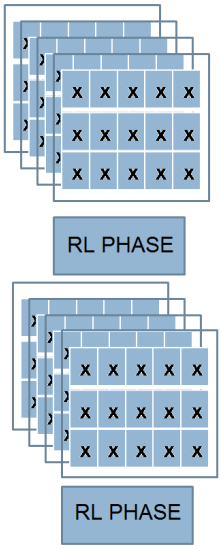

Figure 4.8: For every N generations approach

2. **Change / No Change EDML**:

   In this hybrid system, we test if the RL phase can offer better results by affecting the EDML phase or by running as an independent post process. It is believed in the early stages of EDML generations not to Change EDML as the generations are immature. The EDML should be changed after the generations are matured and are close to convergence.

   (a) Change EDML:

   This approach is done after the RL phase finishes (after all episodes end and the learning process is finished) and before passing the generation to the EDML evolution algorithm. In this approach, the generation that RL has learned for changes according to this learning process, meaning all elements will be reset to zero except the learned elements suggested by the learning process, they will return to their original values. The purpose of this approach is to let RL affect the EDML generation according to what it learns so that the learned elements will have more priority and effect in the evolutionary algorithm than the non-learned elements in the next EDML phase. In this research, two ways of changing EDML are tested:

   i. Change EDML

   After the RL phase ends, the learned elements will return to their original values m_{ii}^{t}, while the non-learned elements will be reset then the

generation is passed to the EDML phase.

ii. Change EDML 2

After RL phase ends, the learned elements will return to their original values m_{ii}^t, while the non learned elements will have an 80 % value reduction instead of being reset ($m_{ii}^t \leftarrow m_{ii}^t \times 0.2$) then the generation is passed to the EDML phase. The reason behind this is that lesser elements weights will have a smaller effect in the next generation

Figure 4.9 shows the process of both Change EDML and Change EDML 2 in the Diagonal case with the same example of generation (population of 3 matrices and 5 elements each).

In this example after generation 2 is created, it undergoes a reset operation inside the RL phase for each episode and after the RL phase ends, in this example it learns that elements whose indices are 2 and 4 are sufficient for this generation. Change EDML approach will return only the learned elements back to their original values (denoted by x) and reset the rest (denoted by empty cells). Change EDML 2 approach will return only the learned elements back to their original values (denoted by x) and reduce the rest by a certain value (denoted by small red x).

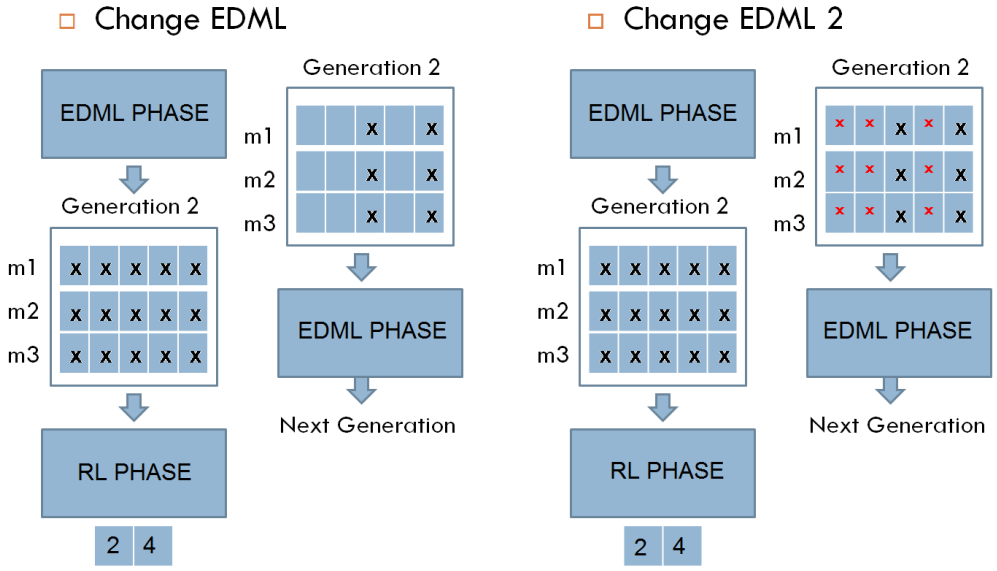

Figure 4.9: Diagonal Change EDML approaches

(b) No Change EDML:

This approach is done after the RL phase finishes (after all episodes end

and the learning process is finished) and before passing the generation to the EDML evolution algorithm. In this approach, the generation that RL has learned for returns to its initial state, meaning all elements will return to their original values (instead of being reset like at the start of the RL phase). The purpose of this approach is to run RL in parallel and independently from EDML and not let it affect EDML generations, instead just act as an observer that records the results and picks the best after learning.

Figure 4.10 shows the process of No change EDML in Diagonal case with the same example of generation (population of 3 matrices and 5 elements each). In this example after generation 2 is created, it undergoes a reset operation inside the RL phase for each episode and after the RL phase ends, in this example it learns that elements whose indices are 2 and 4 are sufficient for this generation. Regardless of this suggestion, the No change EDML approach will return generation 2 back to its initial state and pass it to the EDML phase to create the next generation while the policy result will be saved for later comparisons.

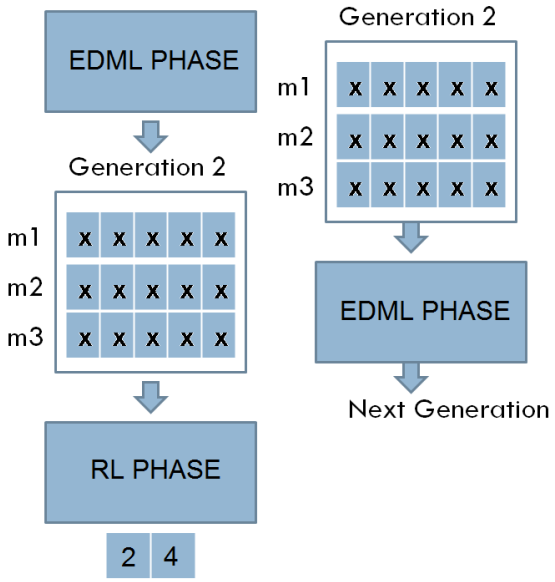

Figure 4.10: Diagonal No Change EDML approach

3. **Resettable / Appendable learning**:

These approaches test whether the next RL phase will perform better if affected by the previous RL phases. Resettable learning explores RL phases disconnected

from one another, whereas Appendable learning connects RL phases.

(a) Resettable learning:

This approach is done before the RL phase start (before each episode start). Since as described before in the process methodology of R-EDML, before each episode, the generation matrices are reset from their elements to zero (Section 4.2.1, RL phase step b) to allow RL to sequentially insert them and learn. In this approach, this resettable state of the generation before the start of each episode will not be changed or affected by what was learned before in the previous RL phases. The purpose of this approach is to investigate every generation independently regardless of what the previous RL phase in the previous generation has learned. The goal is to see if this approach can give any insights or advantages if each learning process was independent of the other.

Figure 4.11 describes this approach in the diagonal case, wherein each learning process, different elements are learned (2 and 4 in first RL phase, 0 and 3 in second RL phase, and so on) and since each RL phase does not affect the next RL phase, the generation will always be reset of its elements before each episode of each RL phase.

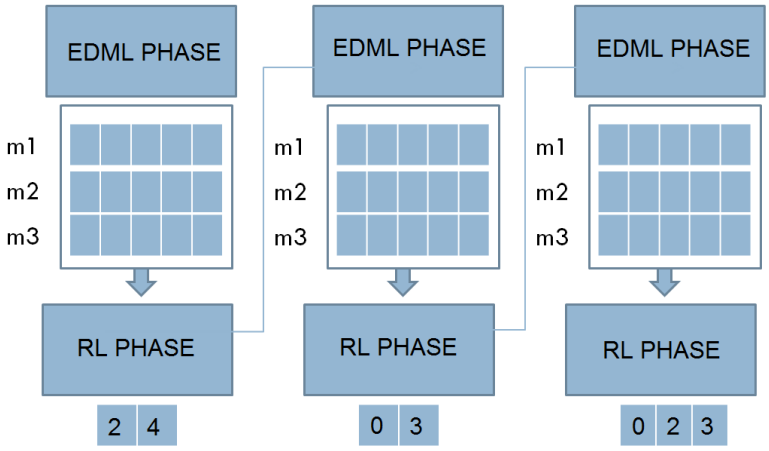

Figure 4.11: Diagonal Resettable learning approach

(b) Appendable learning:

This approach is done before the RL phase start (before each episode start). In this approach, the initial resettable state of the generation before the

start of each episode will be changed according to what was learned in the previous RL phases. So before every episode starts, instead of resetting all elements, only the elements that were not learned in the previous RL phases will be reset, and the learned elements will stay unchanged. This means that the initial generation state before each episode will not be resettable. In this approach, every RL phase appends its learned elements to the initial state of the next RL phase. This appending cycle will continue until the appended elements are equal to the total number of elements. In that case, the appended elements list will be emptied and the appending cycle starts again. The purpose of this approach is to investigate how the RL phases can benefit one another. The goal is to see if this approach can give any insights or advantages if each learning process was depending on the other.

Figure 4.12 describes this approach in the diagonal case, wherein each RL phase, the learned elements are appended to the initial state of the next RL phase and this appending process continues until all the elements are added in the initial state, then it is reset and the appending cycle starts again.

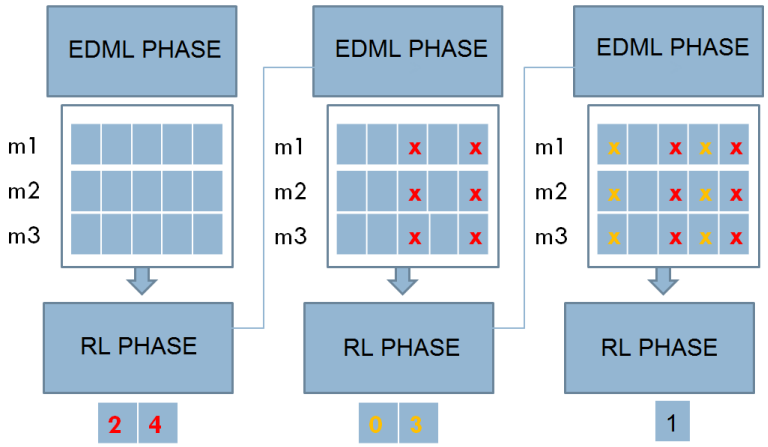

Figure 4.12: Diagonal Appendable learning approach

4. **Learn from Policy (P) / Highest Accuracy (HA)**:

Two different RL phase outputs are explored; one uses the policy learned, and the second uses the highest result in the phase. The idea behind this is to check two types of feedback to the EDML phase, one from the policy and the other from the episode with the best result.

(a) Learn from Policy (P):

For the current generation, after RL finishes its episodes, the learned elements are chosen by the RL policy described in Section 4.2.2. For the current generation, after RL finishes its episodes, the learned elements that can be used in the next EDML phase will be the ones learned from the RL policy. The purpose of this approach is to use the original policy of RL in R-EDML. Figure 4.13 shows this approach, where the policy learned that elements 2 and 4 are enough for this generation, and these elements will be fed to the next EDML phase.

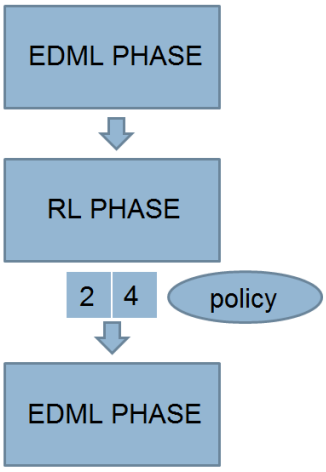

Figure 4.13: Learn from Policy

(b) Learn from Highest Accuracy (HA):

The other type of learning that is explored in the RL phase is learning from Highest Accuracy, which means for the current generation after the RL finishes its episodes, the learned elements that can be used in the next EDML phase will be the elements of the best accuracy episode. Given $\boldsymbol{M^b}$ as the matrix of the best episode, k as the number of episodes, F_e as a set of all the F-measures from all the current RL phase episodes and a_k as the F-measure for $\boldsymbol{M_k}(k = 1, ..., n)$, $\boldsymbol{M^b}$ is defined as:

$$\boldsymbol{M^b} = argmax_{M_k}\{a_k | a_k \in \mathrm{F}_e\}$$

The purpose of this approach is to explore another way of choosing the elements, not according to policy, but according to the best episode result in this RL phase.

Figure 4.14 shows this approach, where the policy is not used and instead

the best episode result was elements 0 and 3, and these elements will be fed to the next EDML phase.

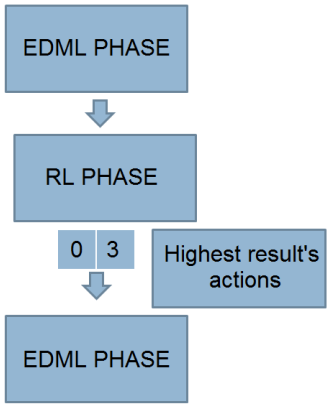

Figure 4.14: Learn from Highest Accuracy

5. **Frequent Items approach**:

   This approach is tested to take advantage of the elements frequently chosen in each RL phase. This feature is implemented to test whether better results will be achieved if these elements are fixed in the next phases. This feature was only tested with the Resettable option since the Appendable option forces the actions selected from the last RL phase to be added to the next generation. The idea is: depending on a frequency threshold, if an action's a_{ii}^{t} frequency reached this threshold ($Frequency_{a_{ii}^{t}} \geq$ (Frequency Threshold)), it will be fixed in the next RL phases.

   Figure 4.15 describes in detail the process of this approach. In this example, given a frequency threshold of 2, which means the action will be considered frequent if it was selected 2 times. First, every action frequency counter will be set to zero, so for the 5 actions (from action 0 to action 4) each has a frequency value of zero before the first RL phase starts. Then after the first RL phase finishes, the learned elements/actions frequency counter will increase (actions 0 and 2 will have a frequency value of 1) and after the RL phase finishes, each action frequency value will be compared to the frequency threshold. Since none of them have a frequency of 2, then none of them will be considered frequent. Then the next RL phase starts and after it finishes, action 0 and 1 will have their frequency value increased, which will make action 0 considered a frequent action. This will make action 0 fixed before the next RL phase starts (meaning it will be added to the

initial resettable state of the generation in the next RL phase).

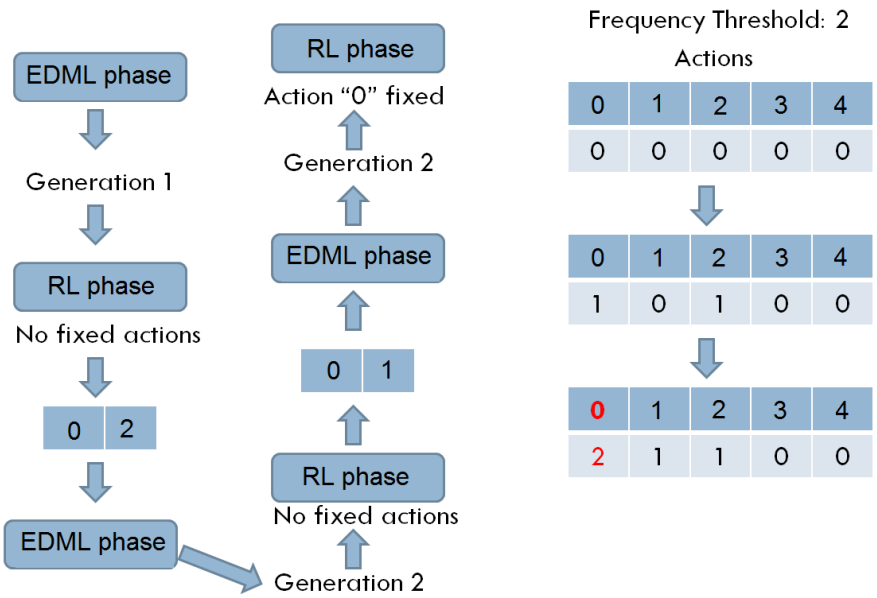

Figure 4.15: Frequent actions approach

6. **Merge technique approach**:

   All the previous approaches have been tested in different combinations but not all at the same time; this approach combines all of them. The idea is to examine the performance if EDML is unchanged by RL from the beginning and is allowed to mature and converge through the evolutionary process, and then RL is allowed to change it. This approach works as follows: given the total number of generations N, the RL phases acting on the first half of the generations will be No Change EDML with Resettable method to give the EDML a chance to converge before changing it, whereas the second half will be Change EDML with the Appendable method to start changing the matured EDML generations. Figure 4.16 describes this approach, given 10 generations, the first half and second half will be treated differently after each RL phase.

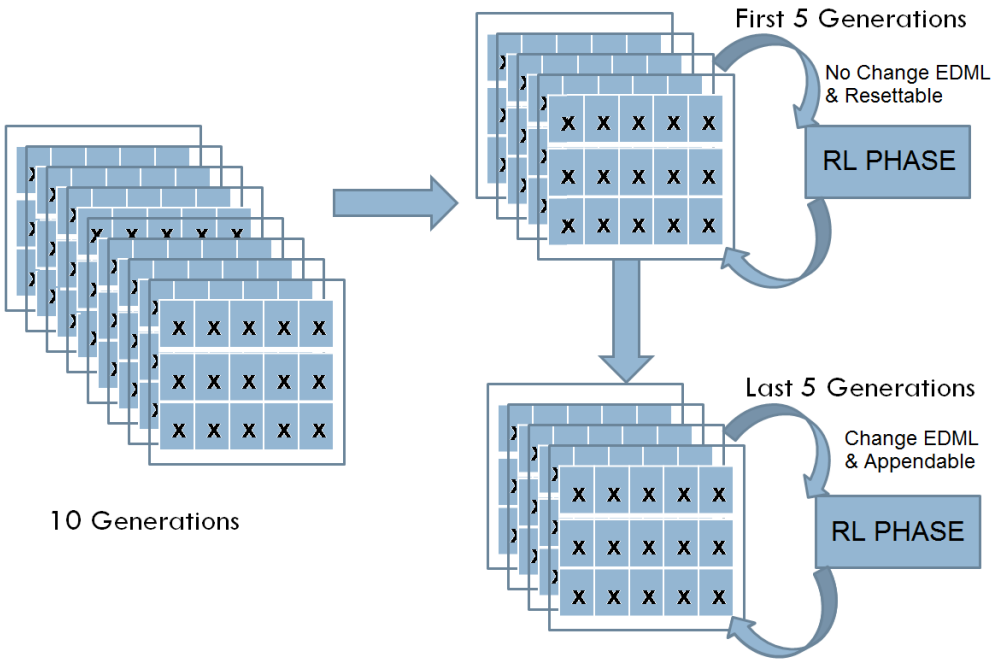

Figure 4.16: Merge technique

Approaches Combinations

This section shows how all the previous approaches from the last section are combined and used together. A multitude of combinations are explored, the purpose of this is to examine the different ways each combination can change the R-EDML result. Some combinations wait for EDML to converge first then change it, others change EDML before it converges, others explore not changing EDML and run RL independently in parallel with it and others merge all these ways together. The approaches are divided into 2 main types, Merge technique approaches, and Non-Merge technique approaches. All these combinations are tested and filtered according to the accuracy and number of selected features. The experiments results that will be shown in this research are the results of the filtered approaches only. Figures 4.17 and 4.18 show the Merge technique approaches combinations and the Non-Merge technique approaches combinations.

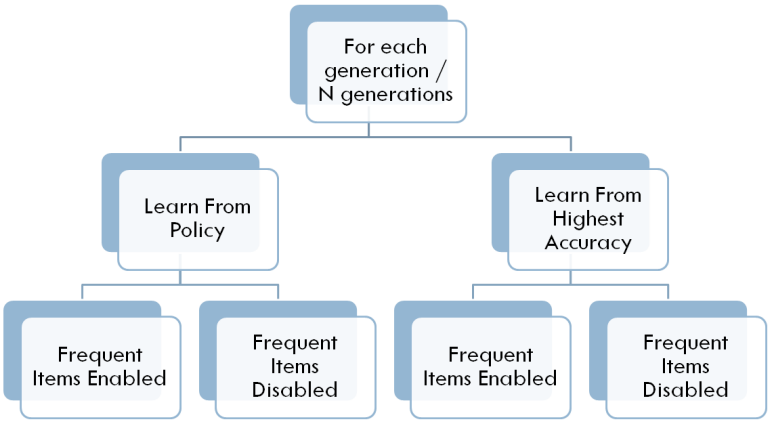

Figure 4.17: Merge technique approaches combinations

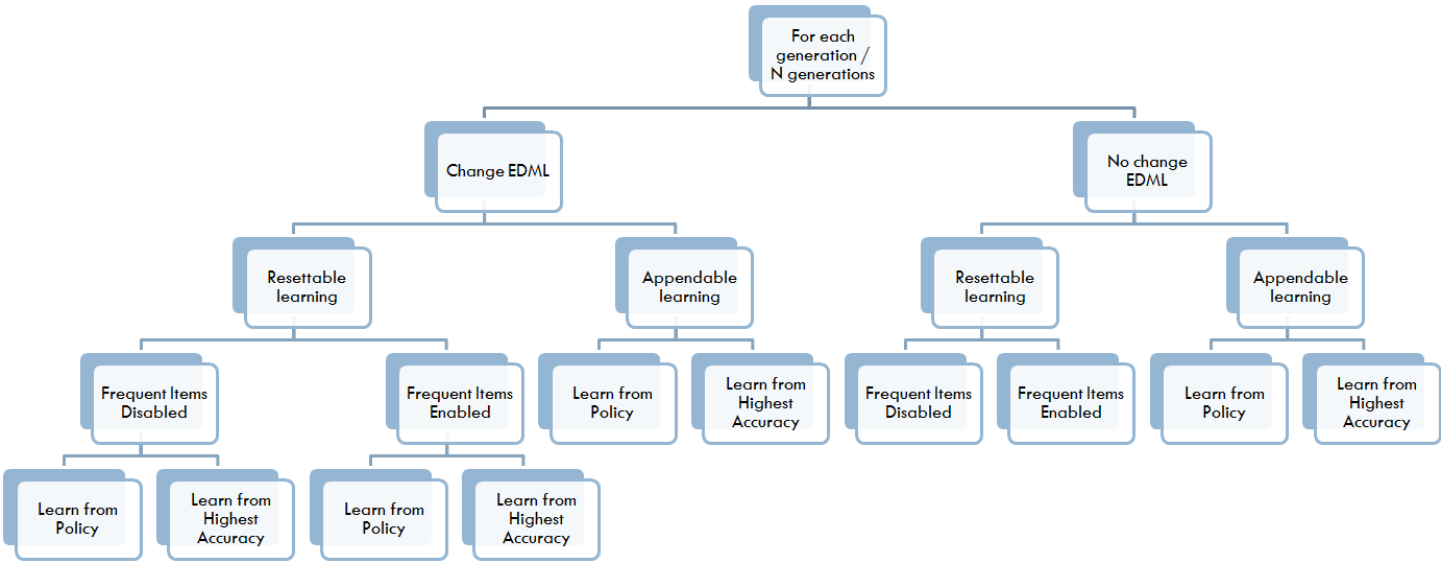

Figure 4.18: Non-Merge technique approaches combinations

Approaches Combinations Mechanism

This section describes the mechanism model of one of the approaches combination to offer a detailed description of the R-EDML approach process. The selected approach will be a Merge technique approach since it combines all the approaches in one; Figure 4.19 shows the selected Merge technique approach (For each generation + Learn From policy + Frequent Items Enabled).

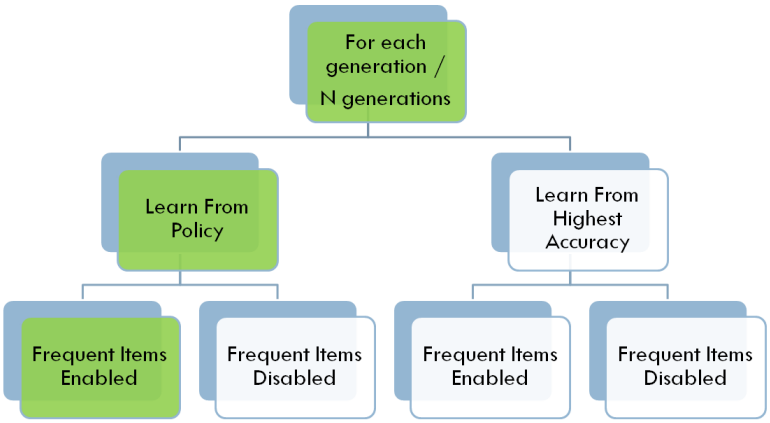

Figure 4.19: Selected Merge technique approach

Example Settings

2 episodes for each RL phase, 3 matrices per generation, 5 features, and a frequency threshold of 1.

Given 4 generations, this approach will use the following approaches:

Generation 1∼2: No Change EDML + Resettable Learning + Learn from Policy + Frequent Items

Generation 3∼4: Change EDML + Appendable Learning + Learn from Policy

Figures 4.20, 4.21, 4.22 and 4.23 show the mechanism of every generation in this selected approach.

1. Generation 1: before learning, all elements are considered not frequent (nf). After learning, elements 2 and 4 will be considered frequent (f) but generation 1 won't be changed before passing it to the EDML phase. Figure 4.20 describes this process.

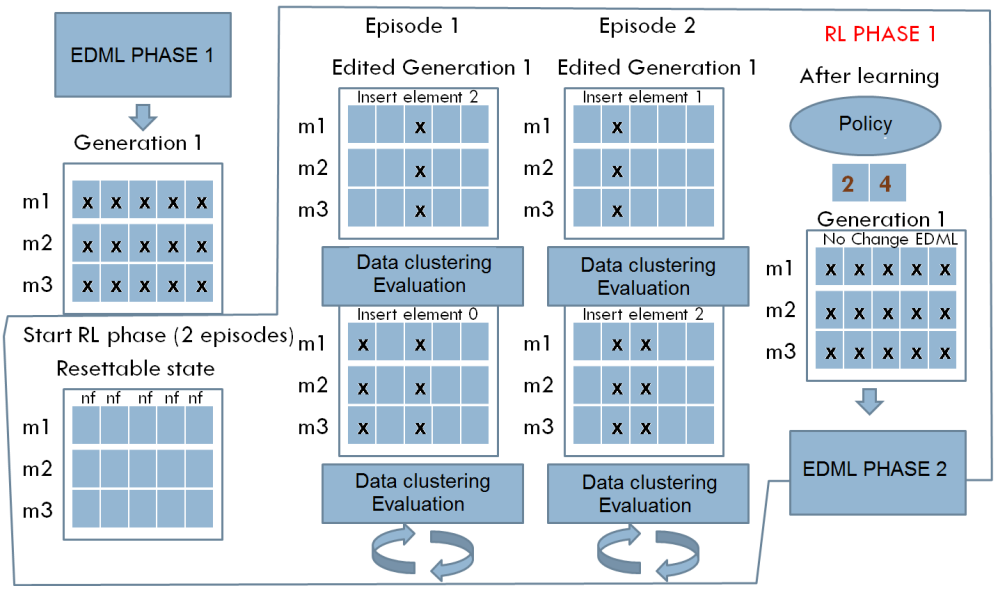

Figure 4.20: Approaches mechanism part 1 - RL process between generation 1 and 2

2. Generation 2: before learning, all elements are considered not frequent (nf) except for elements 2 and 4 which will be fixed in the resettable state of each episode, after learning, elements 2, 3, and 4 will be considered frequent (f) but generation 2 will not be changed before passing it to EDML phase. Figure 4.21 describes this process.

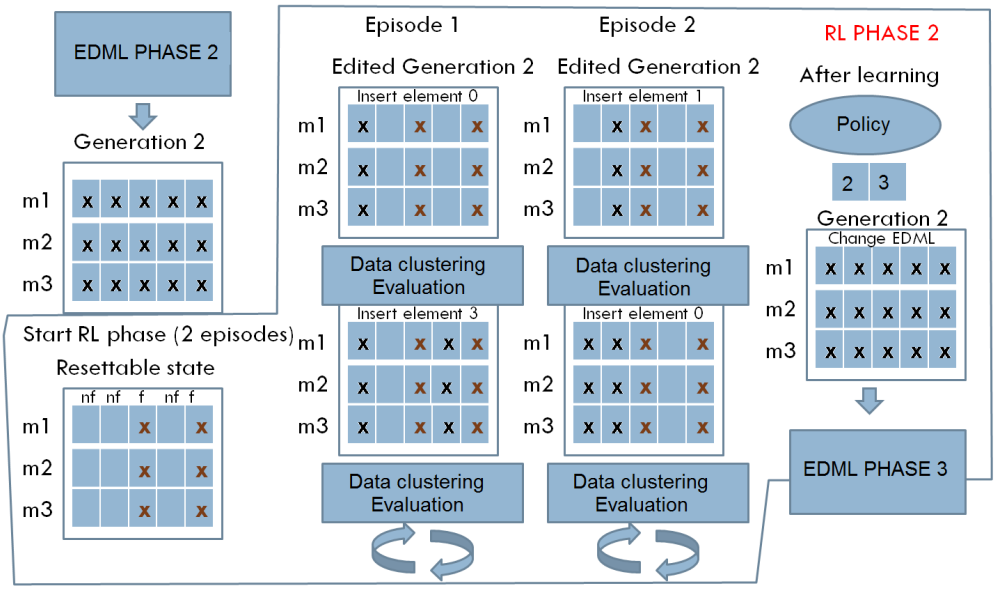

Figure 4.21: Approaches mechanism part 2 - RL process between generation 2 and 3

3. Generation 3: before learning, no elements will be fixed as the Frequent Items

approach is not enabled with the Appendable learning approach, after learning, element 0 is learned and the generation will be changed before passing it to the EDML phase. Figure 4.22 describes this process.

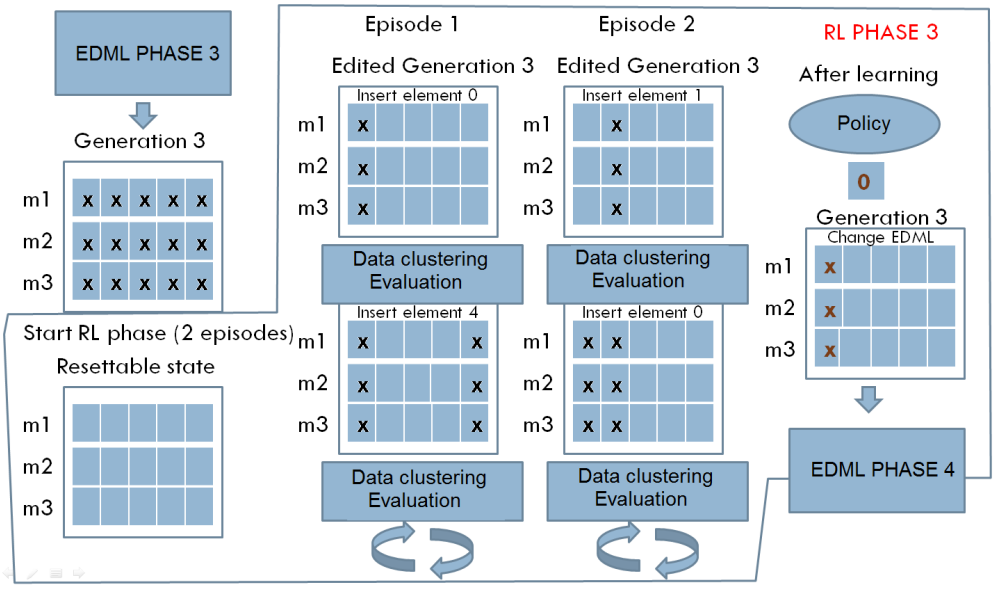

Figure 4.22: Approaches mechanism part 3 - RL process between generation 3 and 4

4. Generation 4: before learning, element 0 which is learned from the last RL phase will be fixed in the Resettable state of each episode, after learning, elements 1 and 4 are learned and the generation will be changed. Figure 4.23 describes this process.

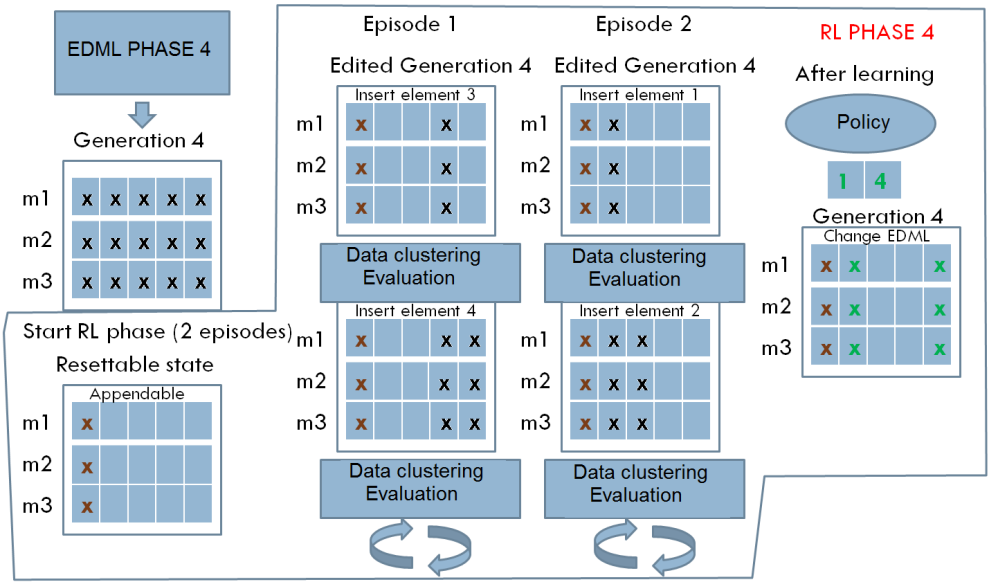

Figure 4.23: Approaches mechanism part 4 - RL process of final generation 4

4.3.4 Database

The tests are performed on different types of real data sets, to cover testing R-EDML against a variety of data sets. The real data set tests are carried out on the UCI machine learning database [48]. The data sets are Iris, Glass, Wine, Vehicle, and Segment; each with a different number of data points, features, and classes; Table 4.1 shows the data sets details. The UCI data sets are used as is, i.e., we did not add any noise nor remove any values in the data sets. We chose those UCI data sets as they are reliable and they do not have missing values and are correctly labeled as well as containing little noise to consider in this research.

Table 4.1: R-EDML UCI data sets

Data set	**Data points**	**Features**	**Classes**
Iris	150	4	3
Glass	214	9	6
Wine	178	13	3
Vehicle	846	18	4
Segment	2310	19	7

4.3.5 Experiment settings

R-EDML Parameters

R-EDML parameters are divided into EDML parameters, RL parameters, and Approaches parameters. Multiple parameters are used owing to a hybrid system combining multiple techniques like EDML and RL and each parameter section manages the customization of a separate part of the R-EDML model.

1. **EDML parameters:**

   EDML parameters like the number of population, clusters, and neighbors are used from previous papers [49, 50] which show after preliminary experiments that using these parameters will make EDML converge.

   (a) Total number of generations: 2000

   (b) Diagonal EDML Generation population (number of transformation matrices in each generation):

i. 20 matrices (Iris data set).

ii. 46 matrices (Glass and Wine data sets).

iii. 56 matrices (Vehicle and Segment data sets).

(c) Full Matrix EDML Generation population:

i. 30 matrices (Iris data set).

ii. 135 matrices (Glass data set).

iii. 273 matrices (Wine data set).

iv. 513 matrices (Vehicle and Segment data sets).

(d) K-means clusters:

i. 20 clusters (Iris, Glass, and Wine data sets) .

ii. 50 clusters (Vehicle and Segment data sets).

(e) K-nearest neighbors: 5

(f) Generations number per EDML phase:

i. 1 generation (in case of for each generation).

ii. 20 generations (in case of for each N generations).

2. **RL parameters**:

(a) Number of episodes:

i. per generation: 4 (in case of for each generation approaches where N = 1).

ii. per 20 generations: 40 (in case of for each N generations approaches where N = 20).

(b) Epsilon (Epsilon greedy policy) = 0.1

(c) AP (action punishment) = -1

(d) SR (satisfactory reward) = $+10$

The idea behind choosing these values is the following: The number of episodes is chosen according to preliminary experiments. As for the Epsilon value, it is set to 0.1 because the state-action space is not big to require a lot of exploration. For the rewards assignments, these values are selected after some preliminary experiments because it is believed that they are suitable for the range of features in the target data sets (4 to 19 features).

3. **Approaches parameters:**

   They include the reduction fixed ratio of Change EDML 2 which is set to 80% and the frequency threshold which is set to 10 (described in Section 4.3.3) with the idea that in case of for each generation, 100 RL phases will run and if an element is selected at least tenth of that number it will be considered frequent and important.

Evaluation Measure

As for the evaluation process, the evaluation measure used is the F-measure (F_1 score) which is the harmonic average of recall and precision where precision is the measure of the same class among each cluster, whereas recall is the measure of the same cluster among each class. The F-measure used here is the pairwise F-measure [23] which is similar to normal F-measure but is evaluated in a pairwise manner on the clustering result. Pairwise manner evaluation borrows the idea of precision and recall to evaluate the clustering result and is more suitable to evaluate this problem.

Given $C(\boldsymbol{x_i})$ as the cluster index of $\boldsymbol{x_i}$; $C(\boldsymbol{x_j})$ as the cluster index of $\boldsymbol{x_j}$; $T(\boldsymbol{x_i})$ as the class index of $\boldsymbol{x_i}$ and $T(\boldsymbol{x_j})$ as the class index of $\boldsymbol{x_j}$, the class and cluster confusion matrix of data pairs is defined in Table 4.2:

Table 4.2: Class and cluster confusion matrix of data pairs

	$T(\boldsymbol{x_i}) = T(\boldsymbol{x_j})$	$T(\boldsymbol{x_i}) \neq T(\boldsymbol{x_j})$
$C(\boldsymbol{x_i}) = C(\boldsymbol{x_j})$	a	b
$C(\boldsymbol{x_i}) \neq C(\boldsymbol{x_j})$	c	d

The precision P, recall R, and F-measure F_1 are defined as follows:

$$P = \frac{a}{a+b} \tag{4.3.1}$$

$$R = \frac{a}{a+c} \tag{4.3.2}$$

$$F_1 = 2 \cdot \frac{P \cdot R}{P+R} \tag{4.3.3}$$

which ensures an incremental behavior in the performance with every generation having

better accuracy or at least equal to the generation before. This gives RL an incentive to keep updating its goal and to keep up with the incremental accuracy.

Approaches Combinations Filtering

Different approaches that investigate different ways of information exchange between RL and EDML have been examined. However, some combinations (described in Section 4.3.3) are thought to be the best due to the following assumptions:

1. It is best to let EDML EA matures first before changing it, this will allow the EA to utilize its full power.

2. If RL changed EDML before EA matures, it won't take full advantage of the EA process to optimize the matrices.

3. Until EA matures, RL should act as an independent post-process to observe and test different selection variations till EA becomes more mature then RL should experiment in changing it.

4. Appendable learning approach links the RL phases together to use the learning experience from previous RL phases in the current RL phase. Appendable learning is best suited with Change EDML approach as after EA matures, RL can start changing EDML using all the combined RL phases experiences.

5. Resettable learning approach does not take advantage of the past RL phase learning experience so it is best suited with No Change EDML approach as it gives RL the chance to explore with all elements freely in every RL phase without changing EDML.

6. Frequent item approach notices the consistent selection of elements by multiple RL phases and takes advantage of it in the learning process, so it is best to use it.

7. For every n generations approach is better than For each generation approach as it lets the EA work independently for several generations instead of RL changing each generation, this way the EA can perform better and RL can inject changes every once in a while without hindering the original evolutionary process itself.

8. For each generation approach is best suited with No change EDML, as it lets the RL explore with each generation without hindering the EA process.

9. Learn from policy approach is better than Learn from highest accuracy approach as it uses the RL power to learn the elements with the metric values instead of searching for the best result regardless of the metric value relationship with the elements selection.

Initial tests are carried out to filter out the best approach combinations according to the best pair of accuracy and number of selected features. The tests are also conducted to choose the best margin ϕ in the RL threshold (described in Section 4.2.2). Tables 4.3, 4.4, and 4.5 show the best 3 combinations out of all the approach combinations in Glass, Wine, and Vehicle data sets. The three top combinations are:

1. Combination 1: No Change EDML and Resettable [For each generation].
2. Combination 2: Merge technique, Change EDML 1, and Frequent Items [For each 20 generations].
3. Combination 3: Change EDML 2, Resettable, and Learn from Policy [For each 20 generations].

It is important to note that combination 1 fulfills assumptions 5 and 8, while combination 2 fulfills assumptions 1, 2, 3, 4, 5, 6, 7, and 9 and finally, combination 3 fulfills assumptions 7 and 9. The best-selected combination to achieve the best pair of F-measure and number of features across all data sets is combination 2. Combination 2 achieved the best results as it does not change EDML in the beginning and changes EDML later on, also it uses Resettable learning with No Change EDML and uses Appendable learning with Change EDML, and also takes advantage of the Frequent item approach which injecting the RL changes into EDML every n generations, not with every generation.

Table 4.3: Approaches combinations filtering in Glass data set

Combination	**F-measure**	**# Features**
Combination 1	0.5344	3
Combination 2	**0.5391**	**1**
Combination 3	0.5324	1.9

Table 4.4: Approaches combinations filtering in Wine data set

Combination	**F-measure**	**# Features**
Combination 1	0.879	4.2
Combination 2	**0.955**	**3.3**
Combination 3	0.9286	2.9

Table 4.5: Approaches combinations filtering in Vehicle data set

Combination	**F-measure**	**# Features**
Combination 1	0.4453	2.2
Combination 2	**0.4643**	**2.1**
Combination 3	0.4538	1.6

4.3.6 Experiments

The following experiments use diagonal EDML, which means in the symmetrical transformation matrices, only the diagonal elements which represent scaling factors to the actual dimensions will be used in R-EDML, the other non-diagonal elements represent the correlation between different dimensions. Therefore, removing these elements is the same as removing features from the input space. The only exception is the Full Matrix R-EDML experiment that uses the entire EDML matrix. All experiments are performed 25 times and the average result is recorded for each data set. The approaches shown in this section are the best-filtered approaches from all the combinations of approaches.

R-EDML vs. EDML

The idea behind this experiment is to compare R-EDML to normal EDML to check if this hybrid system will improve EDML in terms of features and accuracy. Although EDML does not explicitly select features, it has its feature prioritizing process. We observed the EDML optimal matrices results and the ratios between all the elements' weights and we picked a threshold of 0.05. The prioritizing method of EDML important features number is as follows: after all generations are created, the diagonal elements' weights of the optimal matrix are analyzed. Weights that are less than the threshold (0.05) are immediately discarded, and the rest is filtered according to the following formula:

Given **M*** as the optimal EDML matrix with elements m^*_{ii}, given m_{max} as the maximum value in **M*** and m_{min} as the minimum value in **M*** whose value is bigger than 0.1, L^* is the list of prioritized important elements from **M***. The following equation describes the EDML important feature prioritizing process:

$$L^* = \{m^*_{ii} | m^*_{ii} \geqslant 0.05 \text{ and } [(m_{max} - m^*_{ii}) - (m_{max} - m_{min})] \leqslant 0.1\} \tag{4.3.4}$$

R-EDML vs. Conventional Feature Selection

Conventional feature selection is applied to EDML to test if it is superior to R-EDML in terms of feature reduction. Two types of feature selection are tested, both concentrate on different aspects of the features:

1. **Feature scoring**: Gives score to each feature based on gain ratio [51] (Higher scores are better) and focuses on the quality of each feature.

2. **Feature subset selection**: Uses a greedy forward selection algorithm to select the best features and is based on Pearson's correlation [4] that indicates the quality of the features, focusing on the relation between features instead of individual features.

Full Matrix R-EDML (non-diagonal)

Full Matrix R-EDML uses a combination of diagonal and non-diagonal elements. The purpose of this test is to take advantage of the Full Matrix capability in transforming the input space and see if the result can be improved while trying to use diagonal and non-diagonal elements. Full Matrix R-EDML is compared to another semi-supervised DML technique called ITML, which is the most famous DML method that learns a Mahalanobis distance metric by exploiting a notion of a margin between pairs of samples. A weight prioritizing process similar to that of EDML is carried out on ITML, later compared with R-EDML. A GitHub implementation [1] is used for ITML.

R-EDML Policy Unification

In all the previous tests, in each R-EDML generation, a new policy tailored for the generation is learned because the assumption is that the R-EDML evolution algorithm

[1] `https://metric-learn.github.io/metric-learn/_modules/metric_learn/itml.html#ITML_Supervised`

changes and mutates the elements of each generation. In this experiment, the policy is unified across all generations to determine if a unified updated policy has potential validity.

4.3.7 Results and Observations

Table 4.6 shows the feature weighting results of EDML and ITML, whereas Table 4.7 shows the comparison result between Full Matrix R-EDML and ITML. Tables 4.8, 4.9, and 4.10 show the comparison between all the previous experiments among all data sets.

In Table 4.10, # Generations refers to the number of generations needed to reach the best result (highest F-measure and lowest number of features), which is a measurement to verify the approach that converges faster.

In Table 4.9, # Features in Diagonal EDML refers to the number of the selected diagonal elements. In Full Matrix EDML, # Features refers to diagonal elements (whether these diagonal elements are explicitly selected by R-EDML or they are not explicitly selected but non-diagonal elements associated with these diagonal elements are selected; in this case, they are considered selected).

R-EDML vs. EDML Feature Weighting

In Table 4.6, EDML EA important feature weighting has decreased the required features for every data set. Even though feature weighting is not a considered feature selection, EDML still uses all the features. Typically, R-EDML explicitly selected fewer features than EDML important features in Table 4.9.

Diagonal vs. Full Matrix R-EDML

Surprisingly, Full Matrix R-EDML selected fewer features than Diagonal R-EDML, even though the former used a Full Matrix instead of the diagonal one. In Table 4.9, Full Matrix R-EDML offered the best feature reduction in 3 out of 5 data sets as compared with the diagonal approach, where each policy method offered the least features in 2 out of 5 data sets. In Table 4.10, Full Matrix R-EDML converged faster in Vehicle data set, whereas in Iris, Glass, and Wine data sets Diagonal R-EDML converged faster. As for the F-measure, Table 4.8 shows that Full Matrix R-EDML offered better accuracy compared with Diagonal one, with higher accuracy in 3 out of 5 data sets.

Policy Unification vs. Separation

In comparison to the policy separation approach, Table 4.9 shows that the unified policy selected fewer features in 2 out of 5 data sets, the same number of features in 2 out of 5 data sets, and more features in only 1 data set. In terms of features, this shows better results. Regarding the F-measure, Table 4.8 shows that this approach offered the same accuracy in 4 out of 5 data sets. As for the number of generations to converge, Table 4.10 shows that the unified approach converged faster in 4 out of 5 data sets. This shows potential in the policy unification method.

R-EDML vs. EDML with Conventional Feature Selection

R-EDML (Diagonal policy separation, Diagonal policy unification, and Full Matrix) showed better results in terms of F-measure, number of features, and convergence compared with conventional feature selection. In Table 4.8, R-EDML achieved better accuracy in all data sets. In Table 4.9, R-EDML selected fewer features in all data sets, and in Table 4.10, R-EDML needed a fewer number of generations to converge as it reached the best result in fewer generations in 4 out of 5 data sets. Thus, the R-EDML feature selection strategy has led to a high average feature reduction % while keeping a high F-measure: 65% in Iris, 88% in Glass, 74% in Wine, 88% in Vehicle, and 94% in Segment. This shows that this method offers a great advantage.

Full Matrix R-EDML vs. Information-Theoretic Metric Learning (ITML)

Since ITML uses a Full Matrix, only Full Matrix R-EDML is comparable to it. In Table 4.7, the comparative result is displayed for each data set showing the F-measure and the number of features. Even though ITML does not explicitly select features, but prioritize them instead, R-EDML selected fewer features than ITML. Since no advantage in F-measure between the two techniques is noted, we conclude that R-EDML is better than ITML.

Table 4.6: Feature weighting results in EDML and ITML

Data set	# Feat.	EDML Imp. Feat.	ITML Imp. Feat.
Iris	4	2	2.1
Glass	9	3.3	3.7
Wine	13	6.3	6
Vehicle	18	3.6	7.4
Segment	19	4.8	6.8

Table 4.7: ITML comparison results

Data set	ITML(F-measure	# Feat.)	Matrix R-EDML(F-measure	# Feat.)
Iris	0.91	4	**0.92**	**1.3**
Glass	0.45	9	**0.56**	**1.8**
Wine	**0.98**	13	0.97	**2.2**
Vehicle	**0.52**	18	0.43	**1.5**
Segment	0.54	19	**0.75**	**1**

4.3.8 Effect of the Acceptance Margin of F-measure

This section investigates the effect of the acceptance margin of F-measure (ϕ in Section 4.2.2). We performed experiments on Wine and Glass data sets in the Diagonal R-EDML while changing the control parameter. This parameter is the margin ϕ used in the threshold comparison to examine how the F-measure and the number of features change in the top three approaches combinations in Tables 4.3, 4.4, and 4.5. Figures 4.25 and 4.24 shows the results of the two data sets using three different margins $\phi = 0.01$, 0.04, and 0.08. Results show that as the value of the margin decreases, R-EDML gets an F-measure close to the best EDML accuracy. The results also show that as the F-measure increases, the number of features increases as well. When the value of the margin increases, R-EDML is not restricted with an accuracy close to the best accuracy; thus, the F-measure and the number of features decrease. Combination 2 (in red) which has the merge approach showed better results compared with the other combinations in both F-measure and number of features. Moreover, Combination 2 showed no increase in the number of features no matter how small the margin is in the Glass data set,

Table 4.8: F- measure comparison results (R-E refers to R-EDML)

Data set	**EDML**	**Feat. subset**	**Feat. scoring**	**R-E policy sep**	**R-E policy uni**	**Matrix R-E**
Iris	**0.92**±0.007	**0.92**±0.006	0.91±0.006	**0.92**±0.004	**0.92**±0.010	**0.92**±0.003
Glass	0.54±0.002	0.54±0.001	0.53±0.001	0.54±0.002	0.54±0.002	**0.56**±0.003
Wine	0.95±0.003	0.87±0.007	0.93±0.006	0.96±0.009	0.93±0.020	**0.97**±0.007
Vehicle	0.43±0.003	0.42±0.002	0.44±0.008	**0.46**±0.010	**0.46**±0.010	0.43±0.003
Segment	0.72±0.007	0.65±0.010	0.73±0.005	0.73±0.009	0.73±0.009	**0.75**±0.008

Table 4.9: Number of features comparison results (R-E refers to R-EDML)

Data set	**EDML**	**Feat. subset**	**Feat. scoring**	**R-E policy sep**	**R-E policy uni**	**Matrix R-E**
Iris	4	2.5	2.0	1.4	1.4	**1.3**
Glass	9	4.5	3.5	**1.0**	1.2	1.8
Wine	13	5.5	7.0	3.3	**1.4**	2.2
Vehicle	18	3.5	6.5	2.1	1.6	**1.5**
Segment	19	4.5	12.0	**1.0**	**1.0**	**1.0**

Table 4.10: Number of generations comparison results (R-E refers to R-EDML)

Data set	**EDML**	**Feat. subset**	**Feat. scoring**	**R-E policy sep**	**R-E policy uni**	**Matrix R-E**
Iris	1244	786	956	778	**486**	870
Glass	1010	1146	989	726	**482**	786
Wine	1013	1174	1169	**910**	1098	948
Vehicle	473	101	322	464	390	**98**
Segment	441	**267**	354	376	359	392

whereas in the Wine data set, the number of features displayed a small increase when the margin is reduced compared with the other approaches. This shows potential in RL in the EDML process (Combination 2) compared with running RL independently (Combination 1 and Combination 3).

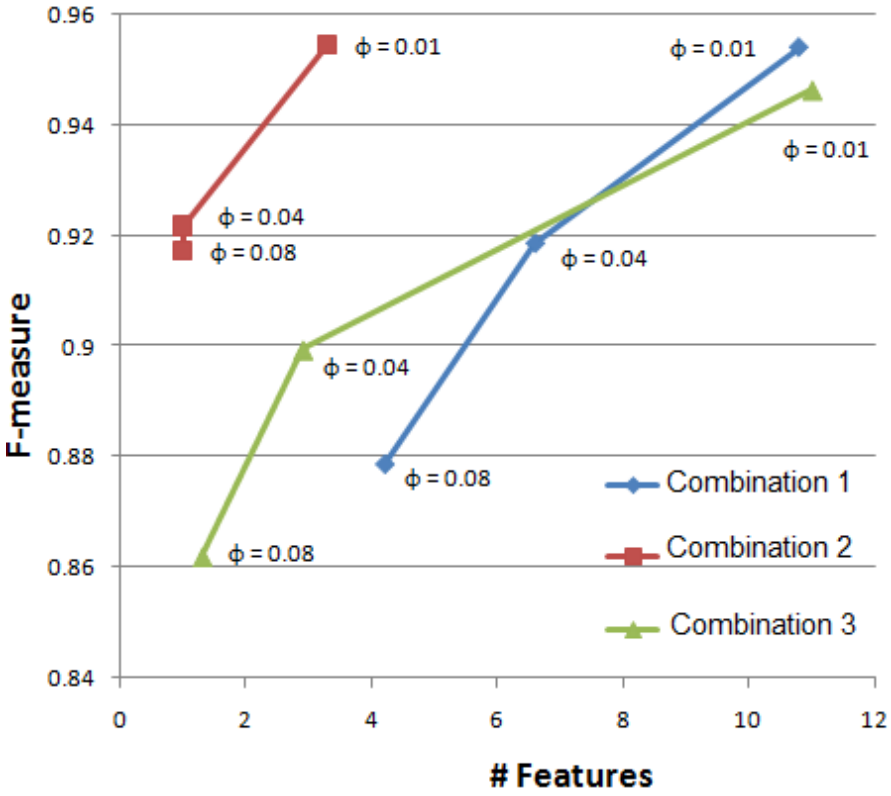

Figure 4.24: R-EDML margin evaluation graph - Wine data set

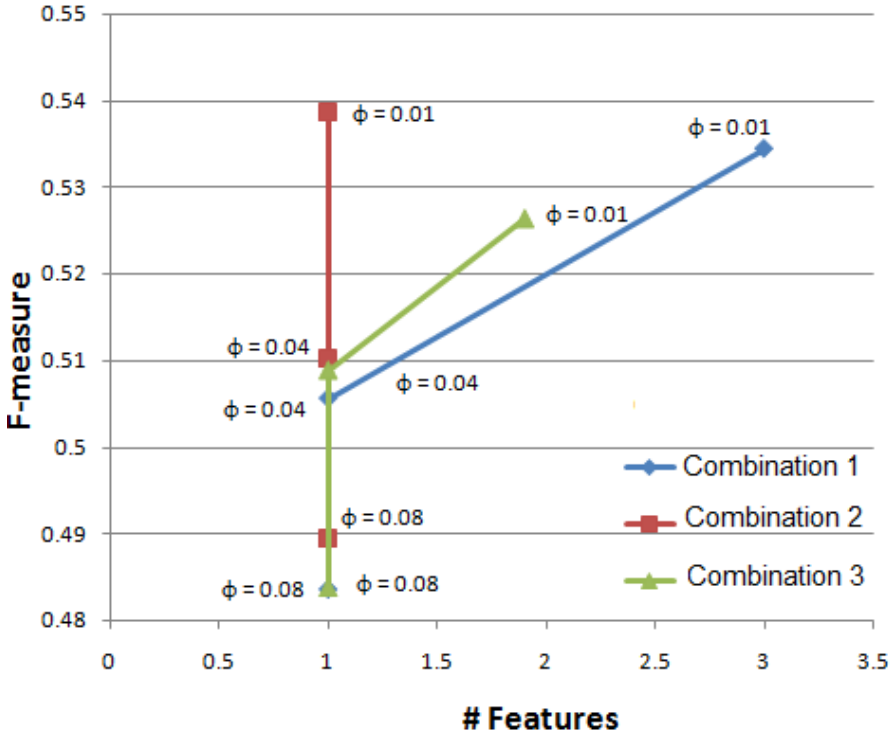

Figure 4.25: R-EDML margin evaluation graph - Glass data set

4.3.9 Summary

R-EDML is composed of two machine learning techniques, RL and EDML to create a feature selection strategy using RL by performing metric filtering in EDML. In the R-EDML life cycle, RL and EDML both work after each other and exchange feedback to perform this metric filtering. Since R-EDML is based on RL, it models the problem as an MDP. Various experiment settings, parameters, and information exchange approaches have been tested to determine the best way to merge RL and EDML.

The database used is UCI for its reliability and a different number of data sets were tested with a variety of data points, classes, and features. The evaluation measure used in R-EDML is the F-measure and the clustering algorithm used is K-means. For experiments, R-EDML feature selection was tested against EDML feature weighting, EDML with conventional feature selection, and ITML. Diagonal and Full matrix R-EDML were tested as well as different ways of updating the RL policy. Results show a feature reduction superiority in R-EDML while maintaining the clustering accuracy. Finally, margin variation testing was performed to select the best accuracy margin to be used in R-EDML.

Chapter 5

Adaptation to High Dimensional Metric Filtering (HDR-EDML)

5.1 Overview

R-EDML has performed well in small dimensional data sets compared to other conventional feature selection methods but is unsuitable in high dimensional data as its sequential evaluation is time-consuming and its table-based learning process can handle only a limited input space.

High Dimensional Reinforced EDML (HDR-EDML) [13] extends R-EDML to scale up the dimensionality by using a function approximation learning process that can handle bigger input spaces as well as a batch system that can reuse the evaluation obtained once. HDR-EDML aims to minimize the used elements (features) in $\boldsymbol{M}$ while maintaining the clustering accuracy.

One novelty of HDR-EDML compared to R-EDML is that it combines EDML and Least-Squares Policy Iteration (LSPI) [24] which is an off-policy RL method suitable for learning in high dimensional settings that uses value-function linear approximation and approximate policy iteration. Another novelty is a mutual interaction and a two-way information exchange between RL and Evolutionary Algorithm (EA) in metric learning. This enables EA to focus the RL environment on the important features and enables RL to edit the input of EA before creating the next generation according to the learning feedback. This approach not only reduces the features but also selects the correct number of features along with specific transformations determined by the EDML EA. This enables clustering while scaling down dimensionality.

5.2 Proposed Method

In HDR-EDML, we focus on diagonal EDML where HDR-EDML will select from only the diagonal elements as they represent the features. RL is based on Markov Decision Processes (MDP). Therefore, R-EDML modeled the problem as an MDP. HDR-EDML models the problem the same way.

To create the feature selection control strategy to filter the metric, HDR-EDML uses a function approximation RL method called Least-Squares Policy Iteration (LSPI) (described in Section 2.2.4). LSPI can derive good solutions from large or infinite search spaces present in high dimensional settings.

LSPI is chosen in this research for the following reasons:

- LSPI is an off-policy method that can reuse samples previously obtained by other policies.
- LSPI has no parameters to tune and will either converge or reach a near-optimal policy in the policy space.
- LSPI offers the strength of policy iteration, where policy iteration generally results in a small number of very large steps directly in policy space, in linear function approximation we can jump directly to the least-squares solution (LSPI process, step 3) where the sum squared error of all w updates to be zero.

5.2.1 HDR-EDML Overview

HDR-EDML is based on the R-EDML process with the extension of using an RL approximation method and adopting a two-way information exchange approach (EDML $\leftarrow$ LSPI and EDML $\rightarrow$ LSPI). HDR-EDML process is as follows:

1. EA creates Generation G with a population of $\boldsymbol{M}$s.
2. HDR-EDML creates a batch of random samples of SARS experiences (s,a,r,s') from G.
3. HDR-EDML controls the available actions (elements) that will create these samples by limiting the elements to the ones prioritized by EA. It uses a threshold on elements in $\boldsymbol{M}$ in this limitation process, thus controlling the batch action space which will later be used by LSPI (EDML $\rightarrow$ LSPI). Then, the samples are created:

(a) Several random policies P_r will create the batch by starting from random initial states s.

(b) To create initial s, randomly selected initial indices edit $\boldsymbol{M}$s (to be $\boldsymbol{M}'$s).

(c) P_r performs a on $\boldsymbol{M}'$s and evaluates by clustering the data with Equation 2.1.1.

(d) P_r sets r according to a's evaluation.

(e) P_r creates s' that will be the $\boldsymbol{M}'$s with the new a.

(f) The policies continue until either a random number of actions is performed or θ is satisfied.

(g) After the desired number of samples is acquired, the whole process stops.

4. LSPI will use this batch to learn a policy P that will filter all $\boldsymbol{M}$s in G by selecting specific elements (features).

5. P edits G to $G^{'}$ by removing the non-selected elements from $\boldsymbol{M}$s and keeping the selected ones.

6. $G^{'}$ will be used by EA to create the next generation (EDML $\leftarrow$ LSPI).

The output of HDR-EDML is one specific $\boldsymbol{M}$ selected from all Gs (using the same output selection conditions in the R-EDML process). This $\boldsymbol{M}$ has a subset of features whose accuracy is close to EDML which uses all features.

Figure 5.1 shows an overview of HDR-EDML process: generation $G1$ has a population of 2 $\boldsymbol{M}$s, each with 6 diagonal elements. After creating a batch and running LSPI, Policy P selects and inserts only elements 0, 3, and 5 in $G1$ (making it into $G1^{'}$) before running EA. This way an RL policy tailored to the EDML generation is believed to give better results as it helps to learn the best features that suit the matrices of the current generation.

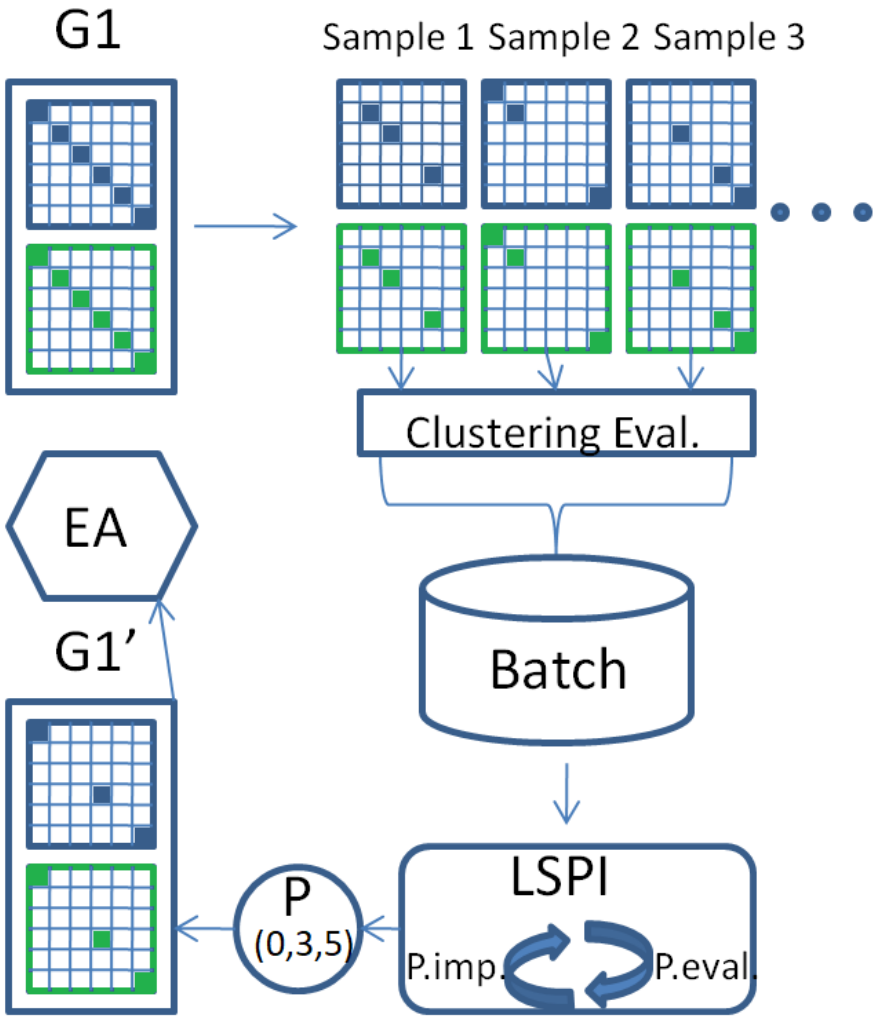

Figure 5.1: HDR-EDML process overview

5.2.2 HDR-EDML Approaches

Like R-EDML, the overall HDR-EDML life cycle is divided into two phases: The EDML phase and the RL phase. The EDML phase prepares the generation for the RL phase, whereas the RL phase learns a policy on this generation. The output of HDR-EDML is one specific $\boldsymbol{M}$ with selected fewer features that have an accuracy close to EDML which uses all the features, from all the EDML generations (using the same output selection conditions in the R-EDML process described in Section 4.2.1).

Three approaches are examined to test the effect of the information exchange between RL and EDML phases on the result. These approaches are: **ESARS**, **L-LSPI**, and **LSPI-1**.

1. **EDML SARS (ESARS)** has a two-way information exchange between EDML

and RL:

(a) EDML $\xrightarrow{feedforward}$ RL: EDML will control the creation of the batch's SARS samples to limit the input space to the elements that EDML deems important only.

(b) EDML $\xleftarrow{feedback}$ RL: RL will learn and send feedback that will affect the EDML creation of the next generation.

In this process, EDML and RL phases will run after each other in a loop for a fixed number of generations.

(a) EDML phase: EDML evaluates the candidates using any clustering algorithm and selects the elite results for the new generation using an EA. This generation's population is a set of distance matrices $\boldsymbol{M}$s responsible for the data set transformation. EDML will create n generations. Generation # n (G) has the best $\boldsymbol{M}_n^*$ which gives the best EDML accuracy so far until generation # n.

(b) RL phase:

i. The SARS batch (step 1 in the LSPI process, Section 2.2.4) for G (called ESARS) is created randomly from only the diagonal elements in $\boldsymbol{M}_n^*$ whose weights are larger than a threshold δ. This focuses the learning on this limited space instead of the whole input space.

ii. LSPI learns a policy Π (steps 2 through 5 in the LSPI process, Section 2.2.4).

iii. The matrices in G will be changed according to Π's selected elements. This will make the EA focus more on Π's selected elements in the next generation creation.

(c) Steps a) and b) repeats for several generations.

The output is the $\boldsymbol{M}$ with the fewest selected features whose accuracy is close to EDML (using the same output selection conditions in the R-EDML process in Section 4.2.1).

Figure 5.2 shows the ESARS approach: EDML EA creates n generations, then ESARS batch created from EDML feedforward is created. A Policy P learns from

this batch and applies to generation #n (blue). RL feedback from P will change the EA (P-EA) next-generation creation.

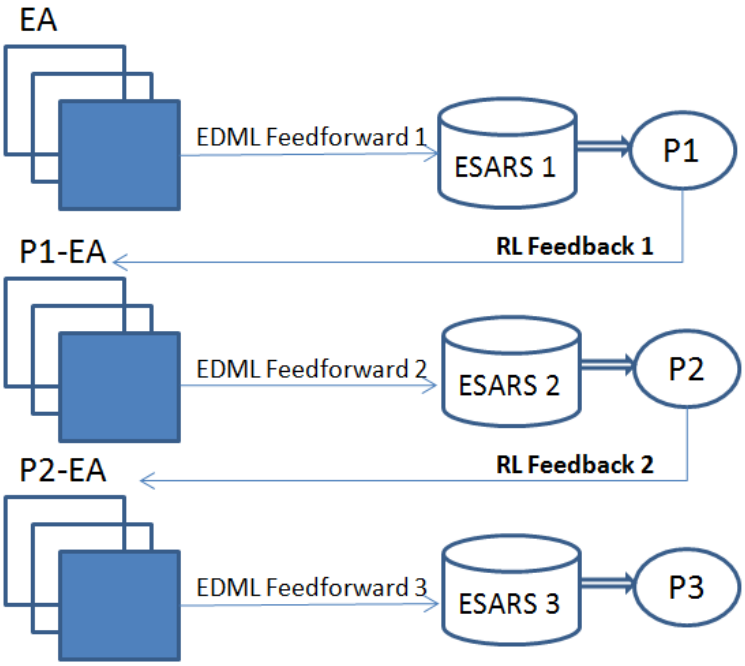

Figure 5.2: ESARS approach

Figure 5.3 shows the pseudo-code of ESARS.

Algorithm 1 ESARS pseudo code

function ESARS(n,m,e)
  //n: total # generations
  //e: LSPI termination factor
  //m: # generations per RL phase
  //D: Batch of samples (s, a, r, s')
  //Π_0: Initial policy, given as w_0
  //g_i: generation # i
  //$\boldsymbol{M}^*$: Best $\boldsymbol{M}$ with close EDML F-measure and least features
  for $i \leftarrow 1, n$ **do**
    if $(i\%m = 0)$ **then**
      create D from important elements of g_i
      $\Pi' \leftarrow \Pi_0$ $\{w' \leftarrow w_0\}$
      repeat
        //LSPI
        $\Pi \leftarrow$ policyImprov.(Π') $\{w \leftarrow w'\}$
        $\Pi' \leftarrow$ policyEval.(D, Π) $\{w' \leftarrow$ policyEval.$(D, w)\}$
      until $(\Pi \approx \Pi')$ $\{\|w - w'\| < e)\}$
      $g_i' \leftarrow$ Edit(g_i) {edit g_i according to Π'}
      $\boldsymbol{M}^* \leftarrow$ SelectBest() {select $\boldsymbol{M}^*$ from all generations}
      $g_{i+1} \leftarrow$EA(g_i') {EA creates g_{i+1} from g_i'}
    else
      $g_{i+1} \leftarrow$EA(g_i) {EA creates g_{i+1} from g_i}
    end if
  end for
  return $\boldsymbol{M}^*$
end function

Figure 5.3: ESARS pseudo-code

2. **Linked LSPI (L-LSPI)** is a special case of ESARS. While having the same process as ESARS, L-LSPI however has only a one-way information exchange between EDML and RL (EDML $\xleftarrow{feedback}$ RL).

   (a) EDML phase: The same as ESARS's EDML phase.

   (b) RL phase: The same as the ESARS's RL phase except for the SARS batch creation step (step i) in ESARS's RL phase). A batch of SARS samples is created randomly from all the diagonal elements of G to cover the entire state-action (input) space.

Figure 5.4 shows the L-LSPI approach: EDML EA creates n generations, then

the SARS batch is created. A Policy P learns from this batch and applies to generation $\#n$ (blue). RL feedback from P will change the EA (P-EA) next-generation creation.

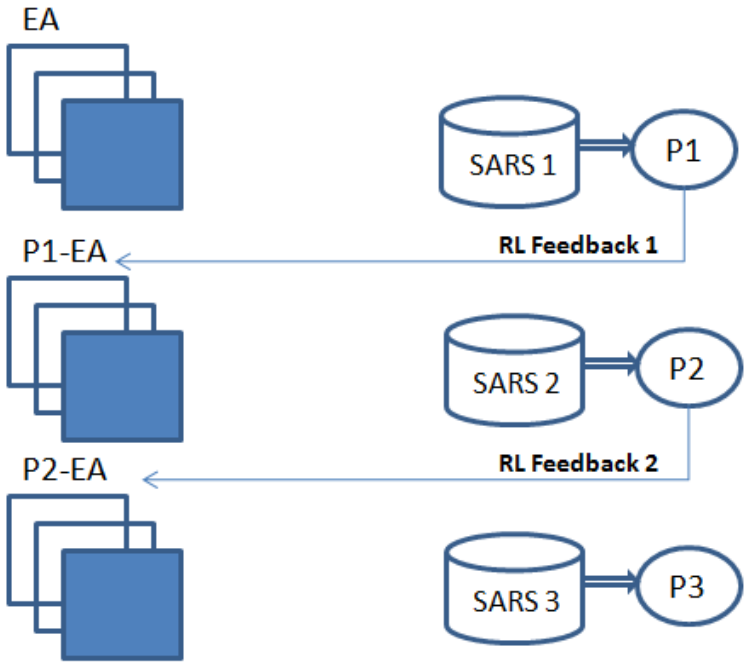

Figure 5.4: L-LSPI approach

3. **LSPI-1** has no information exchange between EDML and RL where RL will learn one time directly from the last EDML generation without affecting EDML itself. LSPI-1 is a special case of L-LSPI, wherein LSPI-1, n is equal to the total number of generations, and G will not be changed after learning the policy. LSPI-1 process is the same as L-LSPI except it does not have step iii) in the RL phase nor step c) afterward.

   Figure 5.5 shows the LSPI-1 approach: EDML EA creates all generations, then the SARS batch is created from all elements, and Policy P learns from it and applies to the last EDML generation (blue).

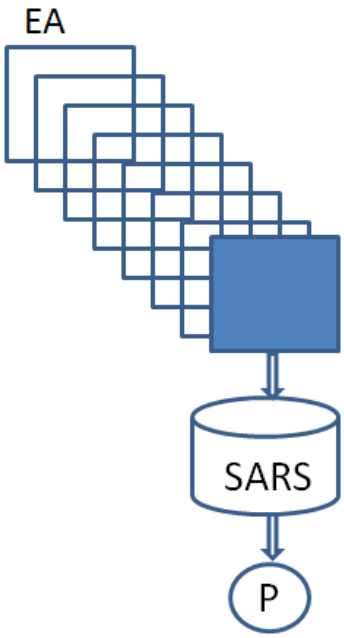

Figure 5.5: LSPI-1 approach

5.3 Experiments and Results

The goal of this section is to test the effect of the learned selection control strategy on high-dimensional EDML transformation matrices.

5.3.1 Database

The tests are carried out on the UCI machine learning database [48]. The data sets used are Human Activity Recognition using Smartphones, Isolet, Parkinson's Disease Classification, Musk, and Libras Movement. The data sets classes, instances, and features number are described in Table 5.1. The UCI data sets are used as is, i.e., we did not add any noise nor remove any values in the data sets. We chose those UCI data sets as they are reliable due to correct labels, no missing values, and little noise to consider in this research.

Table 5.1: HDR-EDML UCI data sets

Data set	# Features	# Classes	# Instances
Activity	561	6	7352
Isolet	617	26	6238
Parkinson	752	2	756
Musk	166	2	476
Libras	90	15	360

5.3.2 Experiment settings

In this study, K-means using a K-nearest neighbor graph of cluster centroids is the clustering algorithm used. The average result of 20 trials is taken and rounded up. The same preparations for R-EDML randomness handling in Section 4.3.2 are done for HDR-EDML as well as the same evaluation method in Section 4.3.5.

HDR-EDML Parameters

Multiple parameters are used owing to a hybrid system and combining multiple techniques like EDML and LSPI. Furthermore, HDR-EDML uses some hyperparameters as well. These parameters values are selected according to preliminary experiments and are divided as follows:

1. **EDML parameters:**

   Preliminary experiments selected the number of generations (from a range of 500–2500 generations) that led to a convergence in EDML, the population size (from a range of 50–300 matrices) of each generation that gave the best results, and the appropriate number of clusters (from a range of 20–60 clusters) for evaluation.

   (a) # Generations:

      i. 2000 (Musk and Libras).

      ii. 1000 (Activity, Isolet, and Parkinson).

   (b) Population size:

      i. 100 (Musk and Libras).

      ii. 250 (Activity, Isolet, and Parkinson).

   (c) # K-means clusters: 50.

2. **RL parameters**:

   Preliminary experiments selected the total number of batch samples that covers appropriately the state-action space (from a range of 1000–3000 samples), a termination factor that gives a good approximation (from a range of 10^{-3}–10^{-6}), and reward values that are proportional to the dimensionality.

   (a) # Batch samples:

      i. 3000 (LSPI-1).

      ii. 1000 (L-LSPI and ESARS), since L-LSPI and ESARS have multiple RL phases.

   (b) Termination factor e in policy iteration: 10^{-6}.

   (c) Action reward (AP):

      i. $-1/d$ (LSPI-1 and L-LSPI), d = number of all diagonal elements.

      ii. $-1/d^{'}$ (ESARS), $d^{'}$ = number of important elements.

   (d) Threshold reward (TR): +1.

3. **HDR-EDML parameters:**

   Preliminary experiments selected the RL phase frequency (every 1/6–1/2 of total generations) that allows enough information exchange between RL and EDML. Furthermore, an accuracy margin is selected that allows an accuracy degradation compared to EDML up to 2% only while reducing features. In addition to that, EDML EA weight assignments differ for each data set, so a suitable weight threshold is selected by sorting the weights and selecting the median as the minimum.

   (a) # Generations per RL phase:

      i. 600 (Musk and Libras).

      ii. 300 (Activity, Isolet, and Parkinson).

   (b) Accuracy margin β: 0.02.

   (c) Weight threshold δ:

      i. 0.002 (Activity, Isolet, and Parkinson).

      ii. 0.01 (Libras).

      iii. 0.004 (Musk).

5.3.3 Experiments

To examine the different ways of combining EDML and RL in this hybrid system, the three approaches described in Section 5.2.2 will be compared against K-means (KMN), ITML [8], EDML, and EDML with l_1 norm regularization. ITML is the most famous DML method that learns a Mahalanobis distance metric by exploiting a notion of a margin between pairs of samples. While l_1 [5] regularization based on the l_1 norm is an effective approach in feature selection as it can drive many parameters to zero and can avoid overfitting by reducing the model complexity. For the EDML with l_1 regularizer, the l_1 hyperparameter was tuned by a grid search for each data set. R-EDML is not included in high-dimensional tests as its tabular representation and sequential decision making make it not applicable to high-dimensional data sets which is why HDR-EDML is proposed.

Furthermore, HDR-EDML ESARS is compared with R-EDML in terms of total computational time and memory to check if the added novelties in ESARS can improve R-EDML. Since R-EDML is not applicable in high-dimensional data sets, 3 small UCI [48] data sets used previously in R-EDML [11] are chosen. The data sets are Glass, Wine, and Vehicle; their information is described in Section 4.3.4. The same experiment settings used for R-EDML [10, 11, 12] (described in Section 4.3.5) are used in this comparison compared to the current HDR-EDML settings with 2000 generations created for each data set. CPU Xeon Gold 6234 3.3 GHz and 256 GB RAM is used for the low dimensional tests.

5.3.4 Results and Observations

Compared to the other HDR-EDML approaches, it is thought that the ESARS approach will yield the best results, the reasoning behind this assumption is as follows:

1. ESARS special batch creation performs a prefiltering process before metric filtering is done RL.
2. This approach uses the EA feature weighing power to focus learning on the elements deemed important by EDML.
3. It saves evaluation time by limiting the RL state-action space.
4. It offers the power of mutual information exchange cycle where RL feedback edits the EA which later on edits the RL learning space.

F-measure Results

Table 5.2 shows the comparison result between the proposed 3 approaches (LSPI-1, L-LSPI, and ESARS) with KMN, ITML, EDML, and EDML with l_1 norm regularization in all UCI data sets in terms of F-measure.

The proposed methods' F-measure is close to EDML and EDML+l_1 in all data sets because β is set to 0.02 with the exception of HDR-EDML outperforming EDML+l_1 in Isolet. HDR-EDML also outperformed ITML in 2 data sets (Activity and Musk). This shows F-measure maintainability.

Comparing the 3 proposed approaches together, ESARS achieved the best accuracy in 2 data sets (Activity and Isolet), L-LSPI achieved in 2 (Parkinson and Musk), and LSPI-1 achieved it in Parkinson only. However, by comparing L-LSPI vs. ESARS, ESARS performed better on Activity and Isolet data sets. This shows the benefit of 2-way (ESARS) information exchange between RL and EDML compared to 1-way (L-LSPI) or 0-way (LSPI-1).

Number of Features Results

Table 5.3 shows the comparison result between the proposed 3 approaches (LSPI-1, L-LSPI, and ESARS) with KMN, ITML, EDML, and EDML with l_1 norm regularization in all UCI data sets in terms of the number of features.

KMN, EDML, and ITML used all the features as they do not explicitly select features. Compared to EDML+l_1, the proposed methods selected fewer features in 4 out of 5 data sets. The feature reduction percentage of L-LSPI is about 91% in Activity, while of ESARS 65% in Isolet, 93% in Parkinson, 94% in Musk, and 73% in Libras. L-LSPI and ESARS could significantly reduce the features with a maximum drop in F-measure of 2%.

ESARS selected the least features in 4 data sets (Isolet, Parkinson, Musk, and Libras), while L-LSPI selected the least features only in Activity. LSPI-1 did not select fewer features in any data sets compared to the other approaches. This shows the benefit of a 2-way (ESARS) information exchange between RL and EDML compared to 1-way (L-LSPI) or 0-way (LSPI-1) information exchange. The ESARS approach can focus its learning on only the EDML prioritized elements which reduced the search space, time, and the number of features.

Table 5.2: Results comparison: F-measure(AVE±STD)

Data set	**KMN**	**ITML**	**EDML**	**EDML+l_1**	**LSPI-1**	**L-LSPI**	**ESARS**
Activity	0.58±0.008	0.63±0.012	0.67±0.009	0.67±0.005	0.65±0.015	0.66±0.008	**0.68**±0.025
Isolet	0.43±0.004	**0.50**±0.014	0.49±0.013	0.47±0.009	0.48±0.007	0.47±0.028	**0.50**±0.011
Parkinson	0.74±0.005	0.75±0.004	**0.76**±0.000	**0.76**±0.000	**0.76**±0.005	**0.76**±0.004	0.75±0.000
Musk	0.50±0.020	0.61±0.005	**0.67**±0.000	**0.67**±0.004	0.66±0.000	**0.67**±0.000	0.66±0.005
Libras	0.34±0.007	0.40±0.004	**0.41**±0.000	**0.41**±0.004	0.39±0.004	0.40±0.008	0.39±0.005

Table 5.3: Results comparison: # Features (AVE±STD for EDML+l_1, LSPI-1, L-LSPI, and ESARS)

Data set	KMN	ITML	EDML	EDML+l_1	LSPI-1	L-LSPI	ESARS
Activity	561	561	561	147.9±3.67	161.7±2.93	**49.2**±4.31	64.4±2.62
Isolet	617	617	617	**158.6**±4.40	411.3±4.32	292.6±2.78	216.5±3.98
Parkinson	752	752	752	170.4±4.23	109.8±3.21	98.2±2.48	**50.1**±3.32
Musk	166	166	166	62.2±2.60	36.4±1.80	24.3±3.62	**9.4**±2.77
Libras	90	90	90	39.7±1.90	54.1±3.03	51.2±2.87	**23.6**±2.40

Computational Time and Memory Results

Table 5.4 shows HDR-EDML and R-EDML computational time comparison. ESARS computational time is much less than R-EDML in all data sets while showing competitive feature reduction and F-measure. This shows the effectiveness of HDR-EDML compared to R-EDML in higher-dimensional data. Also, even though ESARS selects fewer features than EDML+l_1, due to the added RL evaluations it requires roughly double the computational time. However, ESARS focuses its learning only on the important elements, so the added evaluations are kept to a minimum compared to R-EDML.

Regarding memory, in HDR-EDML the state-action values are not stored. Instead, only the feature functions ϕ and weights w are stored for state-action value function computations. However, in R-EDML, the action value for every state is stored which makes it impractical in high-dimensional spaces as the number of states and actions will increase substantially.

Table 5.4: Run time comparison in low dimensional data (Total time (F-measure / # Features))

Method	Glass	Wine	Vehicle
R-EDML	5.8 min (0.54 / 1.0)	3.7 min (0.90 / 2.9)	29.3 min (0.45 / 2.2)
HDR-EDML (ESARS)	**1.4** min (0.53 / 2.1)	**1.2** min (0.94 / 4.3)	**7.8** min (0.46 / 2.5)

5.3.5 Summary

R-EDML has performed well in small dimensional data sets. However, in real-world applications, high-dimensional data is commonly used and R-EDML is unsuitable and time-consuming due to its table-based learning process. Therefore, HDR-EDML is in-

troduced which is an extension of R-EDML to handle high-dimensional data. In HDR-EDML, the learning process is function-approximation based and a batch system allows HDR-EDML to reuse the learning samples which saves time. A different number of approaches were tested that tested different ways of information exchange between RL and EDML.

The high-dimensional database used is UCI for its reliability and a different number of data sets were tested with a variety of data points, classes, and features. The evaluation measure used in R-EDML is the F-measure and the clustering algorithm used is K-means. Various experiment settings and parameters have been tested to determine the best way to merge RL and EDML. For experiments, HDR-EDML was tested against K-means, EDML, EDML with L_1 regularization, and ITML in terms of the number of selected features and F-measure. Results show a feature reduction superiority in HDR-EDML while maintaining the clustering accuracy. Finally, computational time and memory are shown between R-EDML and HDR-EDML and showed better memory handling and less learning time needed for HDR-EDML.

Chapter 6

Conclusion

6.1 Summary

The constant growth in data has presented a challenge in the data processing world on how to manage and filter this data. In this research, a new hybrid system R-EDML is introduced that takes advantage of the sequential decision making in Reinforcement Learning (RL) to perform metric filtering in Evolutionary Distance Metric Learning (EDML) and produce an optimal distance metric with similar performance while extremely reducing the feature space. R-EDML learns a control strategy that directs attention to the important features in the input space and an extension for high-dimensional problems called HDR-EDML is introduced. The experiments performed on UCI data sets show consistent superiority of R-EDML and HDR-EDML as they reduce the required features while maintaining the clustering performance which shows promising potential in using these new methods and a chance for future improvements.

6.2 Discussion

Regarding the experiments performed in this research along with the observed results, this research has combined two techniques and showed a lot of ways of merging and taking advantage of each technique's strong point. However since this is a novel combination, a lot of new different ways of merging must exist and need to be discovered and taken advantage of and there is a huge exploration potential with different methods as well. It was also important to notice that other feature selection methods do not offer the same reduction capability with DML compared to our approach. This is due to the fact that their feature selection process does not consider the metric values in the DML

process and they do not also care about the evaluation feedback with respect to the selected features. This novelty in feature selection provided by our approach showed this advantage in DML methods.

Another thing to notice is that most works in this research were done on diagonal EDML and few works were done on full matrix EDML. However, full matrix EDML showed promising results which are encouraging to engage in more experiments and show potential for further research as more ideas and more experiments can be custom made for full matrix capabilities. In this research, two RL methods were tested with one DML method. This can be expanded as many RL algorithms can offer different advantages with different DML methods and the RL policy can be learned in many different ways and can be adapted according to the DML method used. Regarding the data sets used, only real data sets were used to show the metric filtering capability in real-world applications. However, synthetic data sets can also be explored to test different aspects of this new approach.

Regarding EDML, EDML feature weighting has offered an advantage as it acts as a pre-filtering process before RL can perform metric filtering which enhances the result and saves time as un-needed features can be skipped in the RL phase. It is also important to note that full matrix EDML might be beneficial to be used in small to medium data sets as it's transformation power will be utilized to the maximum without worrying about time. For that reason, R-EDML focuses on both diagonal and full matrix DML as either method can be beneficial according to the data set size. As for diagonal EDML, it can be more beneficial to be used in very large data sets as it focuses more on selection rather than selection and transformation optimization. That is why HDR-EDML focuses only on diagonal in its learning.

Regarding the accuracy margin used in the experiments to make the f-measure close to the EDML, it can be adjusted according to the desired accuracy and the feature selection will adapt accordingly. That being stated, some approaches showed a consistent level of feature selection regardless of how small the accuracy margin gets. We believe that with further investigation we can determine the factor that is responsible for this advantage. It is also important to note that all experiments have either maintained the F-measure according to the margin or increased the F-measure in some cases. This means there might be ways to explore increasing the accuracy rather than maintaining it. Regarding EDML convergence, the number of generations needed was less in our

approach. Further investigation can provide insights or recognize patterns that can be beneficial in determining ways to increase convergence and reduce future computational time.

As for RL methods selection, the approximation method showed faster learning time than the table-based one which is expected as the input space is not explored as much as the table-based. However, the table-based way showed better results which are also expected. Therefore, according to the data set size and time allowed for learning whichever way can be chosen to determine which tradeoff is needed.

In Regards to the EDML and RL information exchange, mutual information exchange offered an advantage in feature selection and computational time compared to one way or zero way. This shows the effect of mutual feature selection and DML feedback instead of only one-way feedback or no feedback. Finally, regarding memory, HDR-EDML memory handling is vital for large data sets as R-EDML saves every state action value and can not be used in case data sets are large in size. However, R-EDML memory handling will not be a problem in small to medium data sets as the memory needed will not be a problem in that case.

6.3 Research Applications

In this research, two main methods were introduced, an RL-based dimensional metric filtering technique called R-EDML and an adaptation technique to high-dimensional metric filtering called HDR-EDML. Both methods have their own strengths and weaknesses. R-EDML strength is represented in its accurate and optimal solution as well as completely covering the entire input space given enough exploration. However, R-EDML suffers from memory and computational time issues due to its sequential evaluation and table-based learning. As for HDR-EDML, its strength comes from its ability to handle high-dimensional data due to its batch system and approximation based learning as it can save time and memory. However, this advantage comes with a trade-off regarding the input space exploration as it is very hard to accurately cover the entire space and reach an optimal solution often.

When deciding which method to use, it is important to take into consideration the size of the data and its dimensionality. For high-dimensional data, it is better to use HDR-EDML as it is more appropriate to handle this kind of data. As for smaller data sets, R-EDML will be better to use as it will give a more accurate result and neither

evaluation time nor memory needed will be a problem.

This research showed very good results when applied to EDML. However, this research goal is the simultaneous optimization of element selection and their values which can have a wide variety of applications not just in EDML. Other possible applications can include any DML method as RL can optimize the selection of the distance metric elements with their values given proper environment description and enough input space exploration. Other applications include classification where RL can select features according to the classification result which can be useful in high-dimensional input spaces like image data and can save future data processing if parts of the input space can be discarded if seemed unimportant by RL.

6.4 Future Work

Even though these new hybrid systems had good results that show potential in them, some issues need to be handled and more experiments can be done. Future issues include handling noisy data which is a usual occurrence in real-world applications. Therefore, it would be worth investigating our approach performance on noisy data by adding synthetic noise to the UCI data. Another issue is that the bigger the data dimensionality becomes, the harder it is to efficiently cover the input space and learn from it the correct number of important features. So different ways of RL methods need to be explored to efficiently handle very big or infinite input spaces. Furthermore, more performance measurements like Purity and Entropy can be added along with the F-measure.

This research shows a big potential for more ideas as this is a novel area that past researchers did not explore. These ideas can also be extended to other data mining techniques. Some future work ideas include:

1. **Incorporating multiple RL agents to handle different parts of the state-action space and combine their learning**: In RL, the state-action space exploration plays a big role in the quality of the learning. In our research, only one RL agent is used to cover the entire input space which is acceptable if the input space is not big. However, it can be challenging to efficiently cover the space when the dimensionality increases. In our research, multiple random policies try to cover the space before learning. Another idea to explore is dividing the state-action space into sections and several RL agents will be deployed where each one

handles a section. A variation would be multiple agents all cover the entire space then share their experience to learn from one another.

2. **Optimization of the matrix M instead of using EA**: In our approach, the optimization of the distance metrics is done via the EDML Evolutionary Algorithm. Another idea is to use RL or another optimization method to optimize the elements in M instead of using EA.

3. **Making the policy of RL affected by the EDML weights**: The EDML feature weights in our research are usually changed and affected by the RL feedback. However, the RL policy itself is not affected by the EA weights changing. In this idea, the EA feature weights can have significance in the value function definition which can affect the policy and the learning process in general.

4. **Increase EDML weights according to selected features**: The EDML features weights in our research are reset to zero if they seemed unimportant by the RL phase. However, the important EDML feature weights are not affected by the RL feedback. An idea to consider is to increase the feature weights which are deemed important by RL before passing them to the EA to create the next generation.

Bibliography

[1] Bruno Magalhães Nogueira, Yuri Karan Benevides Tomas, and Ricardo Marcondes Marcacini. Integrating distance metric learning and cluster-level constraints in semi-supervised clustering. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4118–4125. IEEE, 2017.

[2] Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015.

[3] Manoranjan Dash and Huan Liu. Feature selection for classification. *Intelligent Data Analysis*, 1(1-4):131–156, 1997.

[4] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[5] Andrew Y Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM, 2004.

[6] Fei Wang and Jimeng Sun. Survey on distance metric learning and dimensionality reduction in data mining. *Data Mining and Knowledge Discovery*, 29(2):534–564, 2015.

[7] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[8] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216, 2007.

[9] Ken-ichi Fukui, Satoshi Ono, Taishi Megano, and Masayuki Numao. Evolutionary distance metric learning approach to semi-supervised clustering with neighbor rela-

tions. In *Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference*, pages 398–403. IEEE, 2013.

[10] Bassel Ali, Ken-ichi Fukui, Wasin Kalintha, Koichi Moriyama, and Masayuki Numao. Reinforcement learning based distance metric filtering approach in clustering. In *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI 2017)*, pages 1328–1335. IEEE, 2017.

[11] Bassel Ali, Wasin Kalintha, Koichi Moriyama, Masayuki Numao, and Ken-ichi Fukui. Reinforcement learning for evolutionary distance metric learning systems improvement. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 155–156, 2018.

[12] Bassel Ali, Koichi Moriyama, Wasin Kalintha, Masayuki Numao, and Ken-Ichi Fukui. Reinforcement learning based metric filtering for evolutionary distance metric learning. *Intelligent Data Analysis*, 24(6):1345–1364, 2020.

[13] Bassel Ali, Koichi Moriyama, Masayuki Numao, and Ken-ichi Fukui. Reinforcement learning based evolutionary metric filtering for high dimensional problems. In *Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA 2020)*, pages 226–233. IEEE, 2020.

[14] Liu Yang and Rong Jin. Distance metric learning: A comprehensive survey. *Michigan State Universiy*, 2(2):4, 2006.

[15] Panagiotis Moutafis, Mengjun Leng, and Ioannis A Kakadiaris. An overview and empirical comparison of distance metric learning methods. *IEEE transactions on cybernetics*, 47(3):612–625, 2016.

[16] Dor Kedem, Stephen Tyree, Fei Sha, Gert R Lanckriet, and Kilian Q Weinberger. Non-linear metric learning. In *Advances in neural information processing systems*, pages 2573–2581, 2012.

[17] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[18] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 11–14. Springer, 2009.

[19] Jun Li, Xun Lin, Xiaoguang Rui, Yong Rui, and Dacheng Tao. A distributed approach toward discriminative distance metric learning. *IEEE transactions on neural networks and learning systems*, 26(9):2111–2122, 2014.

[20] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems*, pages 521–528, 2003.

[21] Tomer Hertz, Aharon Bar-Hillel, and Daphna Weinshall. Boosting margin based distance functions for clustering. In *Proceedings of the twenty-first International Conference on Machine Learning*, page 50. ACM, 2004.

[22] Janez Brest, Sao Greiner, Borko Boskovic, Marjan Mernik, and Viljem Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657, 2006.

[23] Ken-ichi Fukui and Masayuki Numao. Neighborhood-based smoothing of external cluster validity measures. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 354–365. Springer, 2012.

[24] Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of machine learning research*, 4(Dec):1107–1149, 2003.

[25] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[26] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[27] Jonathan Baxter and Peter L Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

[28] Quanquan Gu, Zhenhui Li, and Jiawei Han. Generalized fisher score for feature selection. *arXiv preprint arXiv:1202.3725*, 2012.

[29] Huan Liu and Hiroshi Motoda. *Computational methods of feature selection*. CRC Press, 2007.

[30] David E Goldenberg. Genetic algorithms in search, optimization and machine learning, 1989.

[31] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

[32] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

[33] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[34] Gabriel Dulac-Arnold, Ludovic Denoyer, Philippe Preux, and Patrick Gallinari. Datum-wise classification: a sequential approach to sparsity. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 375–390. Springer, 2011.

[35] Ehsan Norouzi, Majid Nili Ahmadabadi, and Babak Nadjar Araabi. Attention control with reinforcement learning for face recognition under partial occlusion. *Machine Vision and Applications*, 22(2):337–348, 2011.

[36] Thomas Rückstieß. *Reinforcement Learning in Supervised Problem Domains*. PhD thesis, Universität München, 2016.

[37] Hirotaka Hachiya and Masashi Sugiyama. Feature selection for reinforcement learning: Evaluating implicit state-reward dependency via conditional mutual information. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 474–489. Springer, 2010.

[38] Jaromír Janisch, Tomáš Pevnỳ, and Viliam Lisỳ. Classification with costly features using deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3959–3966, 2019.

[39] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[40] Rong Jin, Shijun Wang, and Yang Zhou. Regularized distance metric learning: Theory and algorithm. In *Advances in neural information processing systems*, pages 862–870, 2009.

[41] Han Liu, Zhizhong Han, Yu-Shen Liu, and Ming Gu. Fast low-rank metric learning for large-scale and high-dimensional data. In *Advances in Neural Information Processing Systems*, pages 819–829, 2019.

[42] Jason V Davis and Inderjit S Dhillon. Structured metric learning for high dimensional problems. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 195–203, 2008.

[43] Guo-Jun Qi, Jinhui Tang, Zheng-Jun Zha, Tat-Seng Chua, and Hong-Jiang Zhang. An efficient sparse metric learning in high-dimensional space via l 1-penalized log-determinant regularization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 841–848, 2009.

[44] Trung Nguyen, Zhuoru Li, Tomi Silander, and Tze Yun Leong. Online feature selection for model-based reinforcement learning. In *International Conference on Machine Learning*, pages 498–506, 2013.

[45] Milad Zafar Nezhad, Dongxiao Zhu, Xiangrui Li, Kai Yang, and Phillip Levy. Safs: A deep feature selection approach for precision medicine. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 501–506. IEEE, 2016.

[46] Chunfeng Lian, Su Ruan, and Thierry Denoeux. Dissimilarity metric learning in the belief function framework. *IEEE Transactions on Fuzzy Systems*, 24(6):1555–1564, 2016.

[47] Gautam Kunapuli and Jude Shavlik. Mirror descent for metric learning: A unified approach. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 859–874. Springer, 2012.

[48] Catherine L Blake and Christopher J Merz. Uci repository of machine learning databases [http://www. ics. uci. edu/˜ mlearn/mlrepository. html]. irvine, ca: University of california. *Department of Information and Computer Science*, 55, 1998.

[49] Wasin Kalintha, Ken-ichi Fukui, Satoshi Ono, Taishi Megano, Koichi Moriyama, and Masayuki Numao. Semi-supervised evolutionary distance metric learning for clustering. *The Japanese Society for Artificial Intelligence*, 29:1–4, 2015.

[50] Wasin Kalintha, Satoshi Ono, Masayuki Numao, and Ken-ichi Fukui. Kernelized evolutionary distance metric learning for semi-supervised clustering. *Intelligent Data Analysis*, 23(6):1271–1297, 2019.

[51] Asha Gowda Karegowda, AS Manjunath, and MA Jayaram. Comparative study of attribute selection using gain ratio and correlation based feature selection. *International Journal of Information Technology and Knowledge Management*, 2(2):271–277, 2010.