Carlos Alexandre Siqueira da Silva

Novel semi-supervised algorithms based on extreme learning machine for unbalanced data streams with concept drift

Vitória, ES - Brazil

August 2020

Carlos Alexandre Siqueira da Silva

Novel semi-supervised algorithms based on extreme learning machine for unbalanced data streams with concept drift

Doctoral thesis submitted to the Graduate Program in Informatics of the Federal University of Espírito Santo as a partial requirement to obtain the degree of Doctor of Computer Science.

Federal University of Espírito Santo – UFES

Technological Center – CT

Graduate Program in Informatics – PPGI

Supervisor: Dr.-Ing. Renato Antônio Krohling

Vitória, ES - Brazil

August 2020

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

Silva, Carlos Alexandre Siqueira da, 1980-
S586n Novel semi-supervised algorithms based on extreme learning machine for unbalanced data streams with concept drift / Carlos Alexandre Siqueira da Silva. - 2020.
113 f. : il.

Orientador: Renato Antônio Krohling.
Tese (Doutorado em Informática) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Machine learning. 2. Semi-supervised learning. 3. Extreme learning machine (ELM). 4. Data streams. 5. Concept drift. 6. Unbalanced datasets. I. Krohling, Renato Antônio. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 004

"Do. Or do not. There is no try."
(Master Yoda)

Carlos Alexandre Siqueira da Silva

Novel semi-supervised algorithms based on extreme learning machine for unbalanced data streams with concept drift

Esta Tese foi julgada aprovada para a obtenção do Título de Doutor em Ciência da Computação, e aprovada em sua forma final pelo Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo.

Vitória, ES - Brasil, 06 de agosto de 2020

Prof. Dr.-Ing. Renato Antônio Krohling
Orientador

Prof. Dr. Antônio de Pádua Braga
Membro externo

Prof. Dr. Daniel Cruz Cavalieri
Membro externo

Prof. Dr. Celso Alberto Saibel Santos
Membro interno

Prof. Dr. Vinícius Fernandes Soares Mota
Membro interno

Acknowledgements

First of all, I would like to thank my family, for all the support they have always given me. Even from a distance, they always accompanied and encouraged me.

I deeply thank my wife Geciane, for all the support, encouragement, affection and especially for the patience to deal with me in times of greatest stress. Thank you for always being with me, and for listening to me talking about neural networks and ELMs.

My sincere gratitude to my supervisor, Prof. Renato Krohling, for all his dedication and patience. Thank you for trusting me and guiding me for this long period, for pointing me in the path of academic research.

I am grateful to all PPGI professors and staff for making it possible to develop my research. I also thank the members of the examination committee for this doctoral thesis, for kindly making available their time and knowledge to improve this work.

I would like to thank my friends at Labcin, for all the conversations, suggestions and insights; and more broadly, to all colleagues at UFES, for sharing the experiences and anxieties of academic life, making the journey a little less arduous. I also thank my friends at Ifes, for their companionship and encouragement. Each of you, in your own way, contributed to the completion of this work.

I take this opportunity to thank the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), for financial support of this work, and the Instituto Federal do Espírito Santo (Ifes), for granting me a period of exclusive dedication to the doctorade.

Statement of Originality

The research work contained in this thesis has not been previously submitted for any degree or diploma at any other university. This thesis is an original work and does not contain any material previously published by another person, except where appropriate references are made. It is the product of my own work and all sources and received assistance in its development have been cited and recognized.

Carlos Alexandre Siqueira da Silva

August, 2020

Vitória-ES, Brazil

Resumo

Nos dias atuais, *streams* de dados são importantes fontes de informação e, com a popularização de dispositivos móveis e sistemas de sensores que coletam todos os tipos de dados, grandes quantidades de informações são geradas a uma velocidade cada vez maior. Esse crescimento no fornecimento de dados apresenta alguns problemas para os algoritmos tradicionais de aprendizado de máquina. Tarefas como classificação, regressão ou clusterização de dados têm algumas limitações em relação a conjuntos de dados muito grandes, variações ou fluxos contínuos de dados. Em geral, algoritmos que funcionam em uma dessas situações podem não funcionar em outras. Além disso, os fluxos de dados apresentam novos desafios aos algoritmos de aprendizado de máquina. O alto custo de se rotular manualmente instâncias para o treinamento de algoritmos de classificação dificulta o uso de métodos totalmente supervisionados. Conjuntos de dados desbalanceados tendem a fazer com que os algoritmos ignorem uma ou mais classes. Além disso, *concept drifts* nos fluxos de dados exigem que os modelos sejam atualizados periodicamente. Para minimizar os problemas mencionados, nesta tese foram propostos algoritmos semi-supervisionados e on-line baseados em Extreme Learning Machine (ELM). O primeiro algoritmo proposto denominado Semi-Supervised Online Elastic ELM (SSOE-ELM), superou outros da literatura em acurácia e tempo de treinamento, mostrando bons resultados em casos de bases desbalanceadas. O SSOE-ELM usa amostras rotuladas e não rotuladas para treinamento e recebe dados sequencialmente em blocos de uma ou mais instâncias, atualizando continuamente o modelo. Em geral, como um algoritmo baseado em Extreme Learning Machine, seu treinamento é muito rápido em comparação com algoritmos baseados em gradiente descendente. O segundo algoritmo proposto, denominado Semi-Supervised Online Elastic ELM with Forgetting Parameter (SSOE-FP-ELM), é uma extensão do SSOE-ELM para lidar com fluxos de dados com *concept drift*. O SSOE-FP-ELM usa um parâmetro de esquecimento híbrido que considera instâncias rotuladas e não rotuladas para detectar casos de *concept drift* gradual e abrupto. Resultados experimentais mostram que os dois algoritmos propostos superam outros na literatura em acurácia e poder de generalização, indicando serem alternativas viáveis para a classificação de fluxos de dados.

Palavras-chave: Aprendizado de máquina; Aprendizado semi-supervisionado; Extreme Learning Machine (ELM); *Streams* de dados; *Concept drift*; Bases desbalanceadas.

Abstract

Data streams are important sources of information nowadays, and with the popularization of mobile devices and sensor systems that collect all kinds of data, more and more information is generated at an ever increasing speed. This growth in data supply poses some problems for traditional machine learning algorithms. Tasks such as data classification, regression, or data clustering presents some limitations regarding very large datasets, data streams, or variations in data. In general, algorithms that works in one of these situations may not work in others. In addition, data streams pose further challenges to machine learning algorithms. The high cost of labeling instances for training classification algorithms makes it difficult to use fully supervised algorithms. Unbalanced datasets tend to cause algorithms to ignore one or more classes. Moreover, concept drifts in data streams require algorithms to be retrained from time to time. To minimize the problems mentioned, in this thesis semi-supervised and online algorithms based on Extreme Learning Machine (ELM) were proposed. The first proposed algorithm named Semi-Supervised Online Elastic ELM, for short, SSOE-ELM, overperform others in the literature in accuracy and training time, showing good results in cases of unbalanced datasets. SSOE-ELM uses labeled and unlabeled samples for training, and receives data sequentially in chunks of one or more instances, continuously updating the network. In general, as an Extreme Learning Machine based algorithm, its training is very fast compared to gradient descent based algorithms. The second proposed algorithm named Semi-Supervised Online Elastic ELM with Forgetting Parameter, for short, SSOE-FP-ELM, is an extension of SSOE-ELM to deal with data streams with concept drift. SSOE-FP-ELM uses a hybrid forgetting parameter that considers labeled and unlabeled instances to detect gradual and abrupt concept drift cases. Experimental results show that the two proposed algorithms outperform others in the literature in accuracy and generalization ability, showing suitable alternatives for data streams classification.

Keywords: Machine learning; Semi-supervised learning; Extreme learning machine (ELM); Data streams; Concept drift; Unbalanced datasets.

List of Figures

List of Tables

List of Algorithms

Contents

1 Introduction

Data streams are important sources of information nowadays. With the increase of mobile devices such as smartphones, sensors and health monitoring devices, technological gadgets in vehicles or shops, more and more information is generated at an ever increasing speed.

This growth in data supply poses some challenges for traditional machine learning algorithms. Tasks such as data classification, regression, or data clustering have some limitations regarding very large datasets, data streams, or variations in data. In general, algorithms that works in one of these situations may not work in the others.

In the early 2000s, the term *big data* gained popularity, bringing a large set of (sometimes ambiguous) definitions. Laney (2001) described big data in three dimensions, Volume, Variety, and Velocity, giving rise to the "Three V's of big data". The exact definitions of these dimensions vary with respect to what is considered big data and what is not. Gandomi & Haider (2015) presents the following definitions about the Three V's.

- **Volume** refers to the order of magnitude of data, including characteristics such as number of features, number of samples, and storage space.
- **Variety** refers to differences in data structure. Data can be structured as tabular data or unstructured as images, videos, audios, and social network posts.
- **Velocity** refers to the rate at which data is generated or received, or the rate at which data must be processed.

Later, other dimensions were being created. The "Six V's of big data" adds Veracity, Variability and Value to the original three dimensions.

In Qiu et al. (2016) some problems concerning machine learning techniques for big data processing are presented. Figure 1 shows these critical issues of machine learning for big data.

In the Figure 1, the first three boxes of the bottom row correspond to the original 3 V's of big data, while the last two boxes refer to the Veracity and Value dimensions. In order to develop machine learning algorithms adapted to big data characteristics, some of these problems must be solved.

The third dimension (Velocity) can be addressed by fast training algorithms, capable of receiving and processing training instances or even entire data partitions quickly and continuously. In big data problems, the Velocity dimension is associated with the continuous

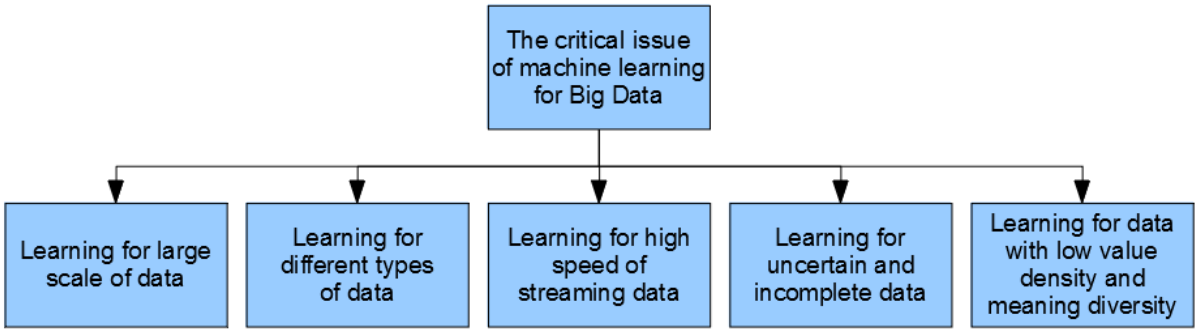

Figure 1 – The critical issues of machine learning for big data (QIU et al., 2016)

arrival of data, at high speed. In order to be able to treat this information without accumulating or discarding data, an algorithm capable of receiving, storing and processing the data in a fast way is necessary. The neural network training algorithm Extreme Learning Machine (ELM) was proposed by Huang, Zhu & Siew (2004) and Huang, Zhu & Siew (2006) as a competitive alternative to the widely used algorithm backpropagation (RUMELHART et al., 1988). Different from gradient descent-based learning methods, ELM does not perform an iterative adjustment of weights. The input weights of the network are randomly generated, and output weights are calculated analytically. Thus, the training time of the ELM algorithm is a few hundred (or thousands) times less than the backpropagation algorithm (HUANG; ZHU; SIEW, 2006). Furthermore, ELM is not susceptible to getting stuck in local minima or slow convergence, common backpropagation problems. These features make ELM a suitable choice for machine learning algorithms to solve big data issues.

The first dimension (Volume) can be addressed by sample selection and online sequential learning techniques. In order to allow online learning, algorithms like Online Sequential ELM (OS-ELM) (LIANG et al., 2006), Regularized Online Sequential ELM (ReOS-ELM) (HUYNH; WON, 2011), Dynamic ELM (DELM) (XU; WANG, 2017), Elastic ELM (E^2LM) (XIN et al., 2015) and Forgetting Parameter ELM (FP-ELM) (LIU; WU; JIANG, 2016) were proposed, which enable sequential training with partitions or even with a single sample of the training set when the full dataset is not available. To allow sample selection, algorithms like Extreme Active Learning Machine (EALM) (HORTA; CASTRO; BRAGA, 2015) and Sample Selected ELM (SS-ELM) (AN et al., 2018) were proposed, allowing the model to be trained with fewer samples, faster and without loss of accuracy.

The fourth dimension (Veracity) concerns the reliability of the data. Uncertainties, missing data, unlabeled data, noise, unbalanced classes and concept drifts are the key challenges in obtaining reliable data. Each of these factors is handled by different techniques.

For unlabeled data, semi-supervised algorithms are indicated. In order to solve the

problem of the absence or the reduced amount of labeled training samples, algorithms like Semi-Supervised ELM (SS-ELM) (HUANG et al., 2014), Unsupervised ELM (US-ELM) (HUANG et al., 2014), Self-Organized Maps ELM (SOM-ELM) (MICHE et al., 2015) and Hessian Semi-Supervised ELM (HSS-ELM) (KRISHNASAMY; PARAMESRAN, 2016) were proposed, which consider that the training dataset has few or no labeled samples.

For concept drift cases, two approaches are commonly used: supervised algorithms that take advantage of the labels to detect a drift; and unsupervised algorithms that uses only input data to detect changes. Supervised approaches often present good results, but have the disadvantage of needing many labeled samples to work properly. With the increasing size of datasets, it becomes difficult to obtain labeled samples to properly train the machine learning algorithms. Zhao, Wang & Park (2012) and Liu, Wu & Jiang (2016) proposed supervised approaches to concept drift detection using forgetting parameters. Unsupervised approaches do not require labels, working directly on input data. However, they are more susceptible to false alarms, since a change in input data does not necessarily indicate a change in the labels. Sethi & Kantardzic (2017) works in an unsupervised approach based on class boundaries. Costa, Rios & Mello (2016) presents an unsupervised algorithm that uses phase spaces to detect concept drifts.

For unbalanced datasets, two common approaches are oversampling and undersampling methods. There are also hybrid methods, which alternate between the previous two according to predefined criteria. Both are based on the principle of dataset balancing: in undersampling, the majority class is reduced by taking out samples; In oversampling, the minority class is expanded by the addition of synthetic samples. Lin et al. (2017) proposes using clustering techniques as a preprocessing step before undersampling, and Kang et al. (2017) uses a noise filter in the minority class while undersampling the majority class. Some oversampling algorithms are Synthetic Minority Over-Sampling Technique (SMOTE) (CHAWLA et al., 2002) and Sigma Nearest Oversampling based on Convex Combination (SNOCC) (ZHENG; CAI; LI, 2016). Both undersampling and oversampling bring problems to the dataset. In undersampling, reducing the majority class may cause important data to be discarded. According to Pozzolo, Caelen & Bontempi (2015), undersampling increases the variance of the classifier, perturb the *a priori* probability of the training set and induces a warping in the posterior distribution. In oversampling, creating synthetic samples can lead to undesirable behaviors, like overfitting. According to Sáez, Krawczyk & Woźniak (2016), oversampling may cause a class distribution shift, since new artificial samples are being created on the basis of previously introduced ones, and may increase noise levels. In addition, both methods make it difficult or even impossible to detect concept drift by interfering with data distribution and changing the number of labels.

1.1 Problem definition

It is desirable for a machine learning algorithm to solve big data problems that it reaches as many dimensions as possible. In addition, it is important that this algorithm minimizes the problems of the previously presented algorithms.

In view of the challenges presented by the big data dimensions, and the importance of initially treating the two initial V's (Volume and Velocity) before taking into account the other dimensions, it is necessary to first find algorithms that address these two problems. The next step is to check if it is possible to adapt the selected algorithm to face some of the characteristic challenges of the Veracity dimension, such as: partially labeled datasets, unbalanced datasets and concept drift affected datasets

Due to its characteristics (fast training, online updating, incremental learning), the ELM neural network training algorithm and its extensions appear to be a viable option for dealing with large datasets and data streams, which would make it a good alternative for the initial two V's of big data problems. Some of the problems of the Veracity dimension, such as partially labeled datasets, unbalanced datasets and the occurrence of concept drifts, can be addressed through semi-supervised learning methods, dataset balancing techniques, concept drift detectors and forgetting parameters to adjust the model. All of these features have already been addressed separately by ELM-based algorithms (KROHLING,).

This thesis is intended to answer the following question: Is an ELM-based algorithm suitable for solving big data problems in the Volume, Velocity and Veracity dimensions at the same time and without a high computational cost?

1.2 Objectives

This thesis aims to propose algorithms for data classification that allow to handle three dimensions of big data: Volume, Velocity and Veracity. For this, the following steps were necessary:

- Identify the desirable characteristics for an algorithm to deal with the big data dimensions of Volume, Velocity and Veracity.
- Identify algorithms that address one or more dimensions of big data problems, their advantages and performance limitations.
- Propose computational solutions in the form of algorithms for the three dimensions mentioned, comparing them with others in the literature.

1.3 Scientific contributions

- A novel semi-supervised and online ELM-based algorithm for unbalanced datasets named SSOE-ELM, with small training time and competitive accuracy for data classification.
- An extension of the previous algorithm named SSOE-FP-ELM, with a forgetting mechanism for data streams with two types of concept drift (features and labels).
- A hybrid forgetting parameter computation for semi-supervised algorithms that takes into account labeled and unlabeled samples.

To allow the reproducibility of the experiments in this thesis, all algorithms are fully described in pseudo-code and all the parameters used are reported. Python source codes are available at https://bitbucket.org/carlosalexandress/ssoe-elm-ssoe-fp-elm. The datasets used in experiments are in the public domain, and the datasets used in case studies are available for research purposes.

1.4 Publications

- **Classificação de grandes bases de dados utilizando algoritmo de Máquina de Aprendizado Extremo** published in the proceedings of the SBPO 2016 - Brazilian Symposium on Operational Research (SILVA; KROHLING, 2016).
- **Semi-Supervised Online Elastic Extreme Learning Machine for Data Classification** published in the proceedings of the IEEE IJCNN 2018 - International Joint Conference on Neural Networks (SILVA; KROHLING, 2018).
- **Restricted boltzmann machine to determine the input weights for extreme learning machines** published in the Expert Systems With Applications journal (PACHECO; KROHLING; SILVA, 2018).
- **Semi-Supervised Online Elastic Extreme Learning Machine with Forgetting Parameter to deal with concept drift in data streams** published in the proceedings of the IEEE IJCNN 2019 - International Joint Conference on Neural Networks (SILVA; KROHLING, 2019).
- **Semi-supervised classification of non-stationary data streams with Extreme Learning Machine-based algorithms** submitted to the Expert Systems With Applications journal (Under review) (SILVA; KROHLING, 2020).

1.5 Organization of this thesis

This thesis is organized as follows: Chapter 2 presents a short background on learning algorithms. Chapter 3 presents two proposed solutions to address the target problems: a semi-supervised algorithm for unbalanced datasets and a new version of previous algorithm to deal with concept drift cases. Chapter 4 presents experimental results and discussions. In Chapter 5 the proposed algorithms are applied to a case study, and Chapter 6 presents some conclusions, limitations and directions for future works.

2 Background on Learning Algorithms

This chapter addresses the fundamentals of learning algorithms that underlie all the approaches taken during the thesis.

2.1 Unsupervised, supervised and semi-supervised learning

According to Chapelle, Schölkopf & Zien (2006), in machine learning, there are originally three types of tasks: supervised, unsupervised and semi-supervised learning. As presented by Roh, Heo & Whang (2019) and Jing & Tian (2020), a fourth type, which has gained attention recently, is self-supervised learning.

In **unsupervised learning**, there is a set of unlabeled samples $(X) = \{(x_i)|x_i \in R^d, i = 1, \ldots, N\}$, where N is the number of samples and d is the number of features. These samples are drawn from a common distribution on X. The main objective of unsupervised learning is to find (hidden) patterns in the data X.

Nowadays, it is easy to get a large amount of unlabeled data from web sites, security cameras, weather stations or even smartphones and tablets. Applications where data arrives continuously, such as video streammings or social network posts are inexhaustible sources of unlabeled data. This data allow for tasks such as clustering, dimensionality reduction, density estimation or autoencoding.

- Clustering - to group similar samples, according to any similarity measure;
- Dimensionality Reduction - to transform each high-dimensional sample in a lower dimensional feature vector without a considerable loss of information;
- Density Estimation - to estimate the parameters of the underlying distribution that generated X.
- Autoencoding - to learn a mapping function $f(x) = x'$, where $x \approx x'$, creating an intermediate representation of x that can be sparse or compressed compared to the original inputs.

The main problem of unsupervised learning is that this unlabeled data does not allow straightforward mappings from inputs to outputs, as in classification tasks.

On the other hand, in **supervised learning** there is a set of pairs (X, Y), consisting of examples and labels, so that $(X, Y) = \{(x_i, y_i)|x_i \in R^d, y_i \in R^m, i = 1, \ldots, N\}$, where N is the number of pairs, d is the number of features and m is the number of classes. The

main objective of supervised learning is to learn a mapping function $f : X \rightarrow Y$ where $f(x_i) = y_i$. Once the model is trained, it is possible to predict the label y' of an unknown sample x' by $f(x') = y'$.

Based on the nature of Y, supervised learning can be categorized into *classification*, when the output Y is discrete; or *regression*, when the output Y is continuous.

The main problem of supervised learning is to get enough labeled data to accurately train the classification model, as the process of manually labeling data is time-consuming and costly.

In **self-supervised learning**, there is a set of pairs (X, Z), consisting of examples and *pseudo labels*, so that $(X, Z) = \{(x_i, z_i)|x_i \in R^d, i = 1, \ldots, N\}$, where N is the number of pairs and d is the number of features. According to Jing & Tian (2020), unlike supervised learning, in self-supervised learning the pseudo label is automatically generated for a pre-defined pretext task without involving any human annotation. The main objective of self-supervised learning is to use an unlabeled dataset to learn intermediate representations of the data that will contribute to subsequent classification tasks.

Finally, **semi-supervised learning (SSL)** is a mixture of the supervised and unsupervised paradigms. SSL is a hybrid technique that uses both labeled and unlabeled data to perform supervised tasks (as classification or regression) or unsupervised tasks (as unsupervised learning guided by constraints). SSL has attracted the attention of many researchers over the past years ((CHAPELLE; SCHÖLKOPF; ZIEN, 2006), (PRAKASH; NITHYA, 2014), (ZHU, 2017), (SAWANT; PRABUKUMAR, 2018)).

In semi-supervised classification, like in the supervised case, the main goal of SSL is to find a function $f : X \rightarrow Y$ to predict the label y of an unknown sample x. To train a model, SSL use a small labeled set $(X, Y) = \{(x_i, y_i)|x_i \in R^d, t_i \in R^m, i = 1, \ldots, N_L\}$ and a large unlabeled set $(X') = \{(x'_i)|x'_i \in R^d, i = 1, \ldots, N_U\}$, where d is the number of features, m is the number of classes, N_L is the number of labeled samples and N_U is the number of unlabeled samples.

Most of the problems encountered in real world are essentially semi-supervised. Monitoring systems (city traffic, personal health, natural disasters and many others), intelligent personal assistants and social network analyse systems work with a large amount of data, but only a small portion of them are labeled. Therefore, SSL is the focus of this work. SSL can be divided into several models, as described in the following.

2.1.1 Generative models

Generative models describes general learning representations. A model can estimate the joint distribution $P(X, Y)$, and compute the conditional probability $P(X|Y = y)$ directly from training data. Generative models are especially useful for creating labeled

synthetic instances for training classifiers.

Kingma et al. (2014) proposed a stacked architecture with two layers: a generative feature extractor and a generative semi-supervised model. Maaløe et al. (2015) improved this architecture adding to the formulation a set of stochastic variables. Adiwardana, Matsukawa & Whang (2016) proposed a semi-supervised Deep Convolutional Generative Adversarial Networks (DCGAN) where the discriminator loss consists of three components: supervised loss, based on the cross-entropy loss from the predicted distribution; unsupervised loss, based on the loss from classifying unlabeled data points as real; and GAN sample loss, based on the loss from classifying generated images as fake. Narayanaswamy et al. (2017) proposed to use a generalized variational autoencoders architecture for semi-supervised learning.

2.1.2 Self-training

Self-Training or Self-Learning is an SSL method whose training is divided into two phases: labeled training and unlabeled set classification. In the first phase, the labeled samples are used to train the model, as in supervised case. In the second phase, the unlabeled samples are classified by the trained model, and the most confident samples and the predicted labels are appended to the labeled training set. These two steps are repeated until all unlabeled training samples are labeled, or until the model meets a stopping condition.

One problem of the self-training method is that if the initial training set is insufficient to train the model, many classification errors can occur in the unlabeled samples in the second phase. By incorporating these incorrectly labeled samples into the training set, these errors will be propagated to the next phases, making the classification model ineffective.

Triguero, García & Herrera (2015) presents a survey on self-training methods. Tanha, Someren & Afsarmanesh (2017) applied self-training in ensembles of decision trees. To increase training dataset volume using self-training methods, Dupre et al. (2019) proposed a technique based on iterative learning cycle with thresholding, while Wu et al. (2018) proposed a method using density peaks.

2.1.3 Co-training

Co-Training is an SSL method that uses two different views of the same data to train two classification models. Each sample is described by two feature sets with different information, and each feature set is used to train one classifier. Similar to self- training, co-training consists of two phases: labeled training of classifiers and crossed unlabeled set classification. In the first phase, the two classifiers are trained using labeled training samples, as in supervised case. In the second phase, the most confident samples of each

classifier and the predicted labels are appended to the labeled training set. These two steps are repeated until all unlabeled training samples are labeled, or until the model meets a stopping condition.

One problem with co-training is that the two classifiers need to be independent and complementary. If one classifier can correctly classify samples that the other can not, the final result will be improved. On the other hand, if the two classifiers attribute the same labels to the samples, then there is no advantage in co-training, and using only one of the classifiers would be more computationally efficient.

Qiao et al. (2018) presents a deep approach of co-training framework. Zhan & Zhang (2017) presents an inductive approach for multi-label classification. Romaszewski, Głomb & Cholewa (2016) proposed an algorithm using co-training strategies to hyperspectral data classification.

2.1.4 Active learning

Active learning is a special case of SSL where an algorithm interactively asks an oracle (usually a human expert) to label some unlabeled samples to train a model. Since manually labeling samples is time-consuming, the active learning method selects the unlabeled samples that will bring greater advantage to the model training. One problem with active learning is the need for human intervention (oracle) to label unlabeled samples. As the unlabeled training set grows, more human effort is needed to train the model. Chen & Wang (2017) and Su et al. (2016) proposes applications of active and semi-supervised learning. Samiappan & Moorhead (2015) presents a framework for semi-supervised image classification exploiting both active learning and co-training.

2.1.5 Graph-based models

Graph-based models are SSL methods that assume that data points (both labeled and unlabeled) are embedded in a low-dimensional manifold. This low-dimensional manifold can be expressed by a weighted graph where the nodes are data points, and edges are a similarity measure between these points (SUBRAMANYA; TALUKDAR, 2014). Yang, Cohen & Salakhutdinov (2016) and Ma et al. (2016) present graph-based frameworks for semi-supervised learning. Sawant & Prabukumar (2018) presents a review of graph-based semi-supervised learning methods.

2.2 Reinforcement learning

Along with supervised and unsupervised learning, reinforcement learning is one of the basic paradigms of machine learning. According to Sutton & Barto (2018), reinforcement

learning is a computational approach to understanding and automating goal-directed learning and decision-making. It emphasizes learning by an agent from direct interaction with its environment, without relying on exemplary supervision or complete models of the environment. Reinforcement learning is a growing research field, mainly with the advent of deep architectures (LI, 2018).

Figure 2 shows the relationship between supervised, unsupervised and reinforcement learning, also placing semi-supervised learning within the broad area of machine learning. The study of reinforcement learning algorithms is outside the scope of this thesis, which focuses on semi-supervised algorithms and their advantages over unsupervised and supervised algorithms.

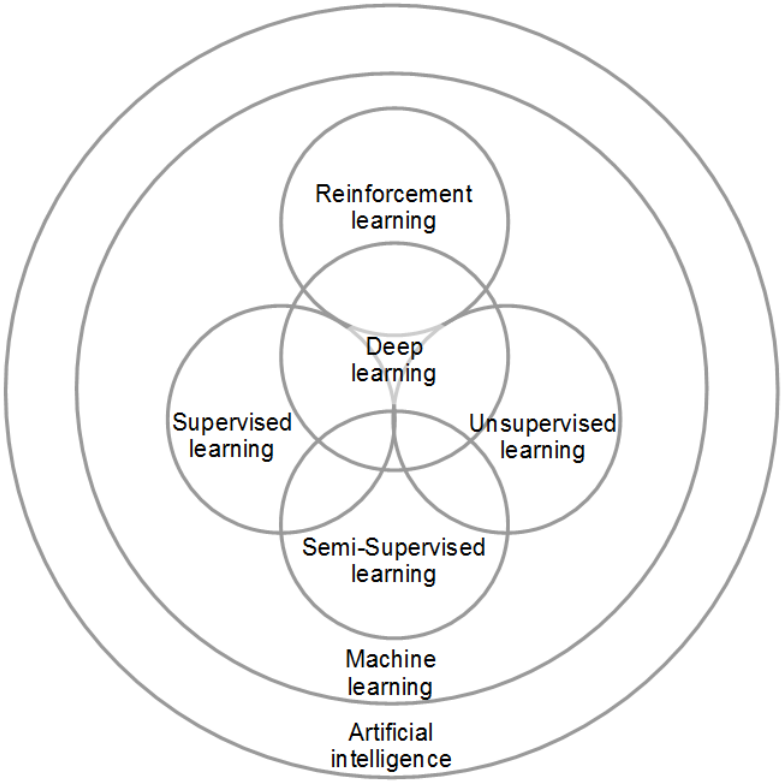

Figure 2 – Relationship among learning paradigms, adapted from Li (2018).

2.3 Batch and online learning

According to how data is received and processed, learning algorithms can be divided into two main cases: batch learning and online learning.

In *batch learning*, all data is previously available to the algorithm, and this data is processed in a one-step way. Batch algorithms do not take into account the arrival time information of data, and need to retrain their models every time a new sample arrive.

On the other hand, *online learning* allows the updating of the model with each new

sample. It is not necessary to have the complete dataset at the beginning of the training phase, and even though the full dataset is available, it is not necessary to process all data in one single step. In addition, online algorithms can use the arrival time of samples to update the model through aging mechanisms that prioritize more recent data or forgetting mechanisms that discard outdated data.

In online learning, a complete set of samples X divided in k partitions, X_k, is given. X_k can have a single sample or a set of N_k samples, and in datasets coming from continuous streaming, the number k of partitions is unknown.

Online learning can be applied in unsupervised or supervised tasks. In unsupervised tasks such as online or incremental clustering ((PHAM; DIMOV; NGUYEN, 2004), (LIN et al., 2004), (BARBAKH; FYFE, 2008), (HYDE; ANGELOV; MACKENZIE, 2017)), online PCA ((ARTAC; JOGAN; LEONARDIS, 2002), (DAGHER, 2010)) or online dimensionality reduction ((LAW; JAIN, 2006), (YAN et al., 2006)), each partition k is represented by a set of unlabeled samples $(X'_k) = \{(x'_i)|x'_i \in R^d, i = 1, \ldots, N_U\}$, where d is the number of features and N_U is the number of unlabeled samples. These samples are drawn from a common distribution on X'.

In supervised tasks such as classification or regression ((YING; ZHOU, 2006), (LIANG et al., 2006), (YING; PONTIL, 2008), (YAO, 2010), (WANG; ZHAO; HOI, 2014), (LIU; WU; JIANG, 2016)), each partition k is represented by a pair $(X_k, Y_k) = \{(x_i, y_i)|x_i \in R^d, y_i \in R^m, i = 1, \ldots, N_L\}$, where d is the number of features, m is the number of classes and N_L is the number of labeled samples.

This work will focus on the online learning paradigm, since most of the problems addressed can not be easily handled by batch algorithms. Online learning is an area that has been attracted much research interest (BLUM, 1998), (SMALE; YAO, 2006), (SHALEV-SHWARTZ, 2012), especially in the last few years ((GUO et al., 2018), (DING et al., 2018), (CAO et al., 2018)) due to the increasing volume of information generated and made available daily by mobile devices such as smartphones, sensors and health monitoring devices, technological devices in vehicles or shops and many others. As the volume of data increases, it becomes necessary to develop algorithms that can work with small partitions of data at a time and are able to update over time.

2.4 Concept drift

Concept Drift is the name given to modifications in the behavior of data streams along time. This topic has attracted the attention of many researchers over time ((COSTA; RIOS; MELLO, 2016), (SETHI; KANTARDZIC, 2017), (LIU; WU; JIANG, 2016), (WANG; ABRAHAM, 2015), (BUDIMAN; FANANY; BASARUDDIN, 2016)). Formally, it is a change in the joint probability distribution $P(X, Y)$ of the input data samples X and their

corresponding class labels Y:

$$P(X, Y) = P(Y|X).P(X) \tag{2.1}$$

According to Tsymbal (2004), concept drift can be abrupt, characterized by a large and easily noticeable change in all data, or gradual, when data change is more subtly during processing. Concept drift can also affect input data, class labels, or both. A study on concept drifts is provided by Gama et al. (2014), defining two types of drifts: real and virtual. Real concept drift refers to changes in outputs (labels). Such changes can happen either with or without change in input data. Virtual drift happens if the distribution of the incoming data changes without affecting the outputs. In this thesis, real and virtual drifts are referred to as label and feature drifts, respectively.

For label changes (Label Concept Drift, or LCD), supervised approaches are used, and for feature changes (Feature Concept Drift, or FCD), unsupervised methods are recommended. Sethi & Kantardzic (2017) investigated supervised and unsupervised methods for concept drift tracking.

Supervised methods continuously evaluate some model metrics (class boundaries, accuracy) to detect the concept drift. Although these methods present good results, they require a large amount of labeled samples to evaluate the model at each training step. Barros & Santos (2018) presents an extensive empirical study of concept drift detectors, verifying their performance in several datasets with abrupt and gradual concept drifts for fully supervised datasets. Wang & Abraham (2015) presented a framework for drift detection in unbalanced datasets and data streams. Budiman, Fanany & Basaruddin (2016) and Xu & Wang (2017) proposed online algorithms based on ELM to address concept drifts in supervised data streams. Another approach that deals with concept drift only in fully supervised cases is proposed by Anderson et al. (2019). This framework takes copies of the classifiers and reuses them if the concept drift is recurrent (a concept that is no longer valid becomes valid again). However, this approach has the additional cost of training and keeping multiple classifiers in memory.

Unsupervised methods evaluate the distribution of input data to detect concept drift. These methods are good alternatives to datasets with few labeled samples, but many classification errors may occur because a change in feature distribution does not necessarily imply changing the labels. An unsupervised approach using dynamic systems tools is proposed by Costa, Rios & Mello (2016) for concept drift detection.

For real-world streaming problems, where few labeled samples are available for training, a semi-supervised method for concept drift tracking is necessary.

2.5 Extreme Learning Machine based algorithms

In this section, the standard Extreme Learning Machine (ELM) algorithm and some ELM-based variations are presented.

2.5.1 Standard ELM

The neural network training algorithm Extreme Learning Machine, proposed by Huang, Zhu & Siew (2004) and Huang, Zhu & Siew (2006), is a competitive option to the widely used algorithm backpropagation (RUMELHART et al., 1988) in Single-Layer Feedforward Neural Networks (SLFN).

Different from gradient descent-based learning methods, ELM does not perform an iterative adjustment of weights. The input weights of the network are randomly generated and maintained during the training phase. The network output weights are calculated analytically using the least squares method (HUANG; ZHU; SIEW, 2006). Thus, the training time of the ELM algorithm is a few hundred (or thousands) times less than the backpropagation algorithm, according to Huang, Zhu & Siew (2006). Furthermore, ELM is not susceptible to getting stuck in local minima or slow convergence, common in backpropagation.

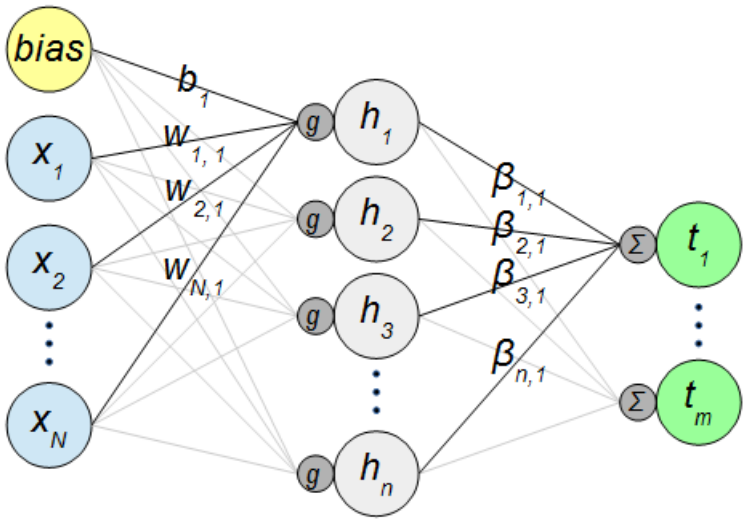

Figure 3 – Illustration of SLFN trained with ELM

Figure 3 shows a SLFN trained with ELM algorithm. Consider a training set $(x_i, t_i), x_i \in R^d, t_i \in R^m, i = 1, \ldots, N$, where N is the number of training samples, d is the number of features and m is the number of classes; an activation function $g(x)$, a sigmoidal function for example; and a number of neurons n in the hidden layer. The relationship between the input data x_i and the network output values t_i is given by:

$$t_i = \sum_{j=1}^{n} \beta_i g(w_j x_i + b_j), \ \ i = 1, ..., N \tag{2.2}$$

where β_i is the output weights, w_j and b_j are the input weights and bias. In ELM, w_j and b_j are set randomly before the training phase, and are not adjusted.

Equation (2.2) can be written in the matrix form $H\beta = T$, where H is the hidden layer matrix (feature matrix) given by:

$$H = \begin{bmatrix} g(w_1 x_1 + b_1) & \dots & g(w_n x_1 + b_n) \\ \vdots & \ddots & \vdots \\ g(w_1 x_N + b_1) & \dots & g(w_n x_N + b_n) \end{bmatrix}_{N \times n} \tag{2.3}$$

The linear system $H\beta = T$ is solved by employing the Moore-Penrose generalized inverse (pseudo-inverse) of the H matrix, represented by $H^\dagger$. There are different ways to calculate the pseudo-inverse of a matrix, including the orthogonal projection, iterative methods, and decomposition into singular values (SVD) (BEN-ISRAEL; GREVILLE, 2003). The solution of ELM and $H^\dagger$ matrix computation are given by:

$$H^\dagger = (H^T H)^{-1} H^T \tag{2.4}$$

$$\beta = H^\dagger T \longrightarrow \beta = (H^T H)^{-1} H^T T \tag{2.5}$$

The addition of a small value on the diagonal of a matrix guarantees this matrix is non-singular, enabling the orthogonal projection method to be used in the computation of the pseudo-inverse. As demonstrated by Huang et al. (2012), this regularization factor makes the solution obtained more stable and with greater generalization ability, and its value can be obtained through a parameter optimization technique or calculated as suggested by Araújo et al. (2019). Thus, for a given regularization factor α, $H^\dagger$ matrix and β matrix can be obtained as follows:

$$H^\dagger = (H^T H + \alpha I)^{-1} H^T \tag{2.6}$$

$$\beta = (H^T H + \alpha I)^{-1} H^T T \tag{2.7}$$

The ELM training pseudocode, as proposed by Huang, Zhu & Siew (2006), is described in Algorithm 1.

2.5.2 Semi-Supervised ELM (SS-ELM)

The SS-ELM algorithm is a graph-based semi-supervised version of ELM proposed by Huang et al. (2014). The SS-ELM takes advantage of the structural relationship between the input data when there are no labels, considering that the changes in data occur smoothly. This relationship is given by the Smoothness Assumption, formalized in the smoothness function described as:

$$S(f) = \frac{1}{2} \sum_{i \sim j} A_{ij} (f_i - f_j)^2 = f^T L f \tag{2.8}$$

Algorithm 1 ELM training

Input: Training set $(X, T) = \{(x_i, t_i)|x_i \in R^d, t_i \in R^m, i = 1, \ldots, N\}$;
Activation function $g(x)$;
Number of hidden neurons n;
Regularization parameter α;
Output: input weights W, bias b and output weights β
1: Initialize input weights W and bias b randomly drawn from an uniform distribution in [-1, 1];
2: Compute feature matrix H according to (2.3);
3: Compute output weights β according to (2.7);
4: **Return**

where L is the graph Laplacian of labeled and unlabeled input data, computed as follows:

$$L = D - A \tag{2.9}$$

$$D_{ii} = \sum_{j=1}^{N} A_{ij} \tag{2.10}$$

$$A_{ij} = e^{-||x_i - x_j||^2/2\sigma^2} \tag{2.11}$$

where N is the number of labeled and unlabeled samples, D is a diagonal matrix, and A is the adjacency matrix of input data.

The ELM cost function can be updated to take into account the classification error, the system complexity and the smoothness of data transition. The cost function, with a tradeoff parameter λ, is described by:

$$\min_f \ \frac{1}{2}\{||\beta||^2 + ||f - T||^2 + \lambda f^T L f\} \tag{2.12}$$

To increase the generalization of the model for unbalanced datasets, a diagonal matrix of penalty coefficients J is defined. Given a penalty coefficient C, a training sample x_i belonging to class t_i and the number N_{t_i} of labeled samples of class t_i, the matrix J is defined as:

$$J = diag(C_1, \ldots, C_N), \quad C_i = C/N_{t_i}, \ i = 1, ..., N \tag{2.13}$$

For unlabeled data samples, the C_i value in J matrix is zero. Adding J to (2.12), and replacing $f = H\beta$, we have the cost function and the solution for SS-ELM as follows:

$$\min_\beta \ \frac{1}{2}\{||\beta||^2 + ||JH\beta - T||^2 + \lambda(H\beta)^T L H\beta\} \tag{2.14}$$

$$\beta = (I + H^T J H + \lambda H^T L H)^{-1} H^T J T \tag{2.15}$$

The SS-ELM training pseudocode, as proposed by Huang et al. (2014), is described in Algorithm 2.

Algorithm 2 SS-ELM training

Input: Labeled training set $(X, T) = \{(x_i, t_i)|x_i \in R^d, t_i \in R^m, i = 1, \ldots, N_L\}$;
Unlabeled training set $(X') = \{(x'_i)|x'_i \in R^d, i = 1, \ldots, N_U\}$;
Activation function $g(x)$;
Number of hidden neurons n;
Tradeoff parameter λ;
Penalty coefficient C;

Output: input weights W, bias b and output weights β

1: Initialize input weights W and bias b randomly drawn from an uniform distribution in [-1, 1];
2: With labeled training set (X, T) and unlabeled training set (X'), compute feature matrix H similarly to ELM;
3: Compute Laplacian matrix L according to (2.9)-(2.11);
4: Compute penalty coefficients matrix J according to (2.13);
5: Compute β according to (2.15);
6: **Return**

2.5.3 Online Sequential ELM (OS-ELM)

The OS-ELM algorithm is an online sequential version of ELM proposed by Liang et al. (2006). Its formulation starts with the β calculation formula presented in (2.5), and it is divided into two phases: initialization and sequential. Consider a training set arriving sequentially in partitions. Each partition k is represented by (x_i, t_i), $x_i \in R^d$, $t_i \in R^m, i = 1, \ldots, N_k$, where N_k is the number of samples in partition k, d is the number of features and m is the number of labels. In the initialization phase, we have $k = 0$. Replacing $K = H^T H$ in (2.7), the solution of the initialization phase of OS-ELM is given by:

$$K_0 = H_0^T H_0 \tag{2.16}$$

$$\beta_0 = K_0^{-1} H_0^T T_0 \tag{2.17}$$

In sequential phase, we have $k \geq 1$. The recursive solution of OS-ELM in the sequential phase is given by:

$$K_k = K_{k-1} + H_k^T H_k \tag{2.18}$$

$$\beta_k = \beta_{k-1} + K_k^{-1} H_k^T (T_k - H_k \beta_{k-1}) \tag{2.19}$$

Replacing $P_k = K_k^{-1}$ and applying the Woodbury Formula (GOLUB; LOAN, 1989)(PRESS et al., 2007) in (2.19), we obtain the recursive solution of OS-ELM as follows:

$$P_k = P_{k-1} - P_{k-1} H_k^T (I + H_k P_{k-1} H_k^T)^{-1} H_k P_{k-1} \tag{2.20}$$

$$\beta_k = \beta_{k-1} + P_k H_k^T (T_k - H_k \beta_{k-1}) \tag{2.21}$$

The OS-ELM training pseudocode, as proposed by Liang et al. (2006), is described in Algorithm 3.

Algorithm 3 OS-ELM training

Input: Initial training set $(X_0, T_0) = \{(x_i, t_i) | x_i \in R^d, t_i \in R^m, i = 1, \ldots, N_0\}$;
Activation function $g(x)$;
Number of hidden neurons n;
Output: input weights W, bias b, intermediate matrix P_k and output weights β
Initialization phase:
1: Initialize input weights W and bias b randomly drawn from an uniform distribution in [-1, 1];
2: With initial training set (X_0, T_0), compute the feature matrix H_0 similarly to ELM;
3: Compute K_0 according to (2.16);
4: Compute β_0 according to (2.17);
Sequential phase:
5: **for** each new training set $(X_k, T_k) = \{(x_i, t_i) | x_i \in R^d, t_i \in R^m, i = 1, \ldots, N_k, k \geq 1\}$ **do**
6: Compute feature matrix H_k similarly to ELM;
7: Compute P_k according to (2.20);
8: Compute β_k according to (2.21);
9: **end for**
10: **Return**

2.5.4 Semi-supervised Online Sequential ELM (SOS-ELM)

The SOS-ELM algorithm is a graph-based semi-supervised and online sequential version of the ELM. It was proposed by Jia et al. (2016) joining the SS-ELM (HUANG et al., 2014) and OS-ELM (LIANG et al., 2006).

Like SS-ELM, SOS-ELM takes advantage of the structural relationship between the input data when there are no labels, considering that changes in data occur smoothly. Similar to OS-ELM, the SOS-ELM is divided into initialization and sequential phases, recursively updating β according to each new training partition. The formulation of SOS-ELM algorithm starts with the SS-ELM solution, described in (2.15).

Consider a training set arriving sequentially in partitions. Each partition k is represented by labeled samples $(x_i,\ t_i)$, $x_i \in R^d$, $t_i \in R^m, i = 1, \ldots, N_{k_L}$ and unlabeled samples (x'_i), $x'_i \in R^d, i = 1, \ldots, N_{k_U}$, where N_{k_L} and N_{k_U} are the number of labeled and unlabeled samples in partition k, respectively, d is the number of features and m is the number of labels. In the initialization phase, we have $k = 0$, and SOS-ELM solution is given by:

$$K_0 = I + H_0^T (J_0 + \lambda L_0) H_0 \tag{2.22}$$

$$\beta_0 = K_0^{-1} H_0^T J_0 T_0 \tag{2.23}$$

In sequential phase we have $k \geq 1$, and SOS-ELM recursively updates K and β according to following equations:

$$K_k = K_{k-1} + H_k^T(J_k + \lambda L_k)H_k \tag{2.24}$$

$$\beta_k = \beta_{k-1} + K_k^{-1}H_k^T[J_k T_k - (J_k + \lambda L_k)H_k\beta_{k-1}] \tag{2.25}$$

Replacing $P_k = K_k^{-1}$ and applying the Woodbury Formula (GOLUB; LOAN, 1989)(PRESS et al., 2007) in (2.25), we obtain the recursive solution of SOS-ELM given by:

$$P_k = P_{k-1} - P_{k-1}H_k^T(I + (J_k + \lambda L_k)H_k P_{k-1}H_k^T)^{-1}(J_k + \lambda L_k)H_k P_{k-1} \tag{2.26}$$

$$\beta_k = \beta_{k-1} + P_k H_k^T[J_k T_k - (J_k + \lambda L_k)H_k\beta_{k-1}] \tag{2.27}$$

The SOS-ELM training pseudocode, as proposed by Jia et al. (2016), is described in Algorithm 4.

Algorithm 4 SOS-ELM training

Input: Initial training partition with labeled training set $(X_0, T_0) = \{(x_i, t_i)|x_i \in R^d, t_i \in R^m, i = 1, \ldots, N_L\}$ and unlabeled training set $(X'_0) = \{(x'_i)|x'_i \in R^d, i = 1, \ldots, N_U\}$;
Activation function $g(x)$;
Number of hidden neurons n;
Tradeoff parameter λ;
Penalty coefficient C;

Output: input weights W, bias b, intermediate matrix P_k and output weights β

Initialization phase:

1: Initialize input weights W and bias b randomly drawn from an uniform distribution in [-1, 1];
2: With initial labeled training set (X_0, T_0) and initial unlabeled training set (X'_0), compute feature matrix H_0 similarly to ELM;
3: Compute Laplacian matrix L_0 and penalty matrix J_0 similarly to SS-ELM;
4: Compute K_0 according to (2.22);
5: Compute β_0 according to (2.23);

Sequential phase:

6: **for** each new training partition k, where $k \geq 1$, with labeled training set (X_k, T_k) and unlabeled training set (X'_k) **do**
7: Compute feature matrix H_k similarly to ELM;
8: Compute Laplacian matrix L_k and penalty matrix J_k similarly to SS-ELM;
9: Compute P_k according to (2.26);
10: Compute β_k according to (2.27);
11: **end for**
12: **Return**

2.5.5 Elastic ELM (E^{2}LM)

Elastic ELM (E^{2}LM) is a variation of ELM proposed by Xin et al. (2015) as an online sequential alternative to be used in the MapReduce framework. The three main

characteristics of this algorithm are: adaptation of ELM to the MapReduce framework; decomposition of β computation into two matrices U and V; and incremental, decremental, and correctional online sequential training. This work is not intended to focus on the MapReduce framework. However, the remaining two characteristics of E^{2}LM is detailed in this section.

According to Huang et al. (2012), the ELM output weights β can be computed by (2.7), where α is a regularization factor. Consider a training set arriving sequentially in partitions. Each partition k is represented by $(x_i, t_i), x_i \in R^d, t_i \in R^m, i = 1, \ldots, N_k$, where N_k is the number of samples in partition k, d is the number of features and m is the number of labels. Replacing $U = H^T H$ and $V = H^T T$ in (2.7), the initialization phase of E^{2}LM where $k = 0$ is given by:

$$U_0 = H_0^T H_0 \tag{2.28}$$

$$V_0 = H_0^T T_0 \tag{2.29}$$

$$\beta_0 = (\alpha I + U_0)^{-1} V_0 \tag{2.30}$$

In the sequential phase of E^{2}LM, for each training partition k, where $k \geq 1$, we calculate ΔU and ΔV of the current partition as follows:

$$\Delta U = H_k^T H_k \tag{2.31}$$

$$\Delta V = H_k^T T_k \tag{2.32}$$

obtaining U_k e V_k through recursive formulas. The incremental (I), decremental (D) and correctional (C) learning equations are given by:

$$U_k = \begin{cases} U_{k-1} + \Delta U & (I) \\ U_{k-1} - \Delta U & (D) \\ U_{k-1} + \Delta U_I - \Delta U_D & (C) \end{cases} \tag{2.33}$$

$$V_k = \begin{cases} V_{k-1} + \Delta V & (I) \\ V_{k-1} - \Delta V & (D) \\ V_{k-1} + \Delta V_I - \Delta V_D & (C) \end{cases} \tag{2.34}$$

The calculation of β_k in sequential phase is:

$$\beta_k = (\alpha I + U_k)^{-1} V_k \tag{2.35}$$

The E^{2}LM training pseudocode, as proposed by Xin et al. (2015), is described in Algorithm 5.

2.5.6 Forgetting Parameter ELM (FP-ELM)

The Forgetting Parameter ELM (FP-ELM) is a variation of OS-ELM proposed by Liu, Wu & Jiang (2016) to deal with concept drift in non-stationary environments.

Algorithm 5 E^{2}LM training

Input: Initial training set $(X_0, T_0) = \{(x_i, t_i)|x_i \in R^d, t_i \in R^m, i = 1, \ldots, N_0\}$;
Activation function $g(x)$;
Number of hidden neurons n;
Regularization parameter α;
Output: input weights W, bias b, intermediate matrices U_k and V_k and output weights β
Initialization phase:
1: Initialize input weights W and bias b randomly drawn from an uniform distribution in [-1, 1];
2: With initial training set (X_0, T_0), compute feature matrix H_0 similarly to ELM;
3: Compute matrix U_0 according to (2.28);
4: Compute matrix V_0 according to (2.29);
Sequential phase:
5: **for** each new training set $(X_k, T_k) = \{(x_i, t_i)|x_i \in R^d, t_i \in R^m, i = 1, \ldots, N_k, k \geq 1\}$ **do**
6: Compute feature matrix H_k similarly to ELM;
7: Compute matrix U_k according to (2.33);
8: Compute matrix V_k according to (2.34);
9: **end for**
10: Compute β_k according to (2.35);
11: **Return**

The main idea of FP-ELM is to apply a forgetting mechanism in the model, reducing the influence of previous training samples in the classification results.

Like OS-ELM, in FP-ELM the training samples arrive sequentially in partitions. Each partition k is represented by (x_i, t_i), $x_i \in R^d$, $t_i \in R^m, i = 1, \ldots, N_k$, where N_k is the number of samples in partition k, d is the number of features and m is the number of labels. For $k = 0$, we have the solution of the initialization phase of FP-ELM as follows:

$$K_0 = H_0^T H_0 \tag{2.36}$$

$$\beta_0 = (\alpha I + K_0)^{-1} H_0^T T_0 \tag{2.37}$$

where α is a regularization factor as proposed in Huang et al. (2012), to get a more stable solution and avoid a singular matrix inversion.

In sequential phase, where $k \geq 1$, FP-ELM needs to compute the forgetting parameter γ. For each new data chunk X_k, the value of γ_k is given by:

$$\gamma_k = reg(\theta - \eta * Error_k) \tag{2.38}$$

$$reg(x) = \begin{cases} 0, & x < 0 \\ x, & 0 \leq x \leq 1 \\ 1, & x > 1 \end{cases} \tag{2.39}$$

where $Error_k$ is the current learner's error, θ and η are control parameters and $reg(x)$ is a function to regulate γ_k in the range [0,1].

With the new value of γ_k, the recursive solution of FP-ELM in the sequential phase is given by:

$$K_k = \gamma_k^2 K_{k-1} + H_k^T H_k \tag{2.40}$$

$$\beta_k = \beta_{k-1} + (\alpha I + K_k)^{-1}(H_k^T(T_k - H_k\beta_{k-1}) - \alpha(1-\gamma_k^2)\beta_{k-1}) \tag{2.41}$$

The FP-ELM training pseudocode, as proposed by Liu, Wu & Jiang (2016), is described in Algorithm 6.

Algorithm 6 FP-ELM training

Input: Initial training set $(X_0, T_0) = \{(x_i, t_i)|x_i \in R^d, t_i \in R^m, i = 1, \ldots, N_0\}$;
Activation function $g(x)$;
Number of hidden neurons n;
Control parameters θ, η;
Regularization parameter α;
Output: input weights W, bias b, intermediate matrix P_k and output weights β
Initialization phase:
Initialize input weights W and bias b randomly drawn from an uniform distribution in [-1, 1];
With initial training set (X_0, T_0), compute feature matrix H_0 similarly to ELM;
Compute K_0 according to (2.36);
Compute β_0 according to (2.37);
Sequential phase:
for each new training set $(X_k, T_k) = \{(x_i, t_i)|x_i \in R^d, t_i \in R^m, i = 1, \ldots, N_k, k \geq 1\}$ **do**
    Compute feature matrix H_k similarly to ELM;
    Compute current learner's error $Error_k$;
    Compute the forgetting parameter γ_k according to (2.38);
    Compute P_k according to (2.40);
    Compute β_k according to (2.41);
end for
Return

2.5.7 Remarks

In this chapter, some variants of the Extreme Learning Machine were discussed. The algorithms proposed in the next chapter are based on some ideas presented by these ELM variants, adapting them to new problems and scenarios. The semi-supervised learning process and the ability to learn incrementally are the guiding concepts behind the development of the new algorithms.

3 Development of new semi-supervised algorithms based on Extreme Learning Machine

This chapter describes the steps of developing the algorithms proposed in this thesis.

3.1 The algorithm SSOE-ELM

The first proposed algorithm of this thesis is the Semi-Supervised Online Elastic Extreme Learning Machine (SSOE-ELM) (SILVA; KROHLING, 2018), a semi-supervised version of Elastic ELM (XIN et al., 2015).

SSOE-ELM is a semi-supervised algorithm trained in an online way, receiving labeled and unlabeled samples continuosly. Similarly to SS-ELM (HUANG et al., 2014) and SOS-ELM (JIA et al., 2016), SSOE-ELM is a graph-based algorithm that uses similarity information between labeled and unlabeled training samples, through the Laplacian matrix, to infer information about labels when there are not enough labeled samples to train the model.

SSOE-ELM is divided into two main process: semi-supervised learning and online update. The idea behind the SSOE-ELM operation is to use the SS-ELM semi-supervised mechanism in each training partition separately. The intermediate matrices generated by the SS-ELM (Laplacian matrix L and penalty matrix J) are inserted in the decomposition formulas of the β calculation proposed by the Elastic ELM, generating the U and V matrices. Thus, at each training partition arrived, the U and V matrices are online updated, without the need to calculate β matrix until the end of training process. Figure 4 shows the schematic diagram illustrating SSOE-ELM with one training partition k.

Consider a training set arriving sequentially in partitions. Each partition k is represented by labeled samples $(x_i,\, t_i)$, $x_i \in R^d$, $t_i \in R^m, i = 1, \ldots, N_{k_L}$ and unlabeled samples (x'_i), $x'_i \in R^d, i = 1, \ldots, N_{k_U}$, where N_{k_L} and N_{k_U} are the number of labeled and unlabeled samples in partition k respectively, d is the number of features and m is the number of labels. SSOE-ELM needs to adapt the SS-ELM formulation to work as an online algorithm. In SS-ELM, the training process is made in batch mode as follows:

$$H = g(XW + b) \tag{3.1}$$

$$\beta = (I + H^T JH + \lambda H^T LH)^{-1} H^T JT \tag{3.2}$$

where X is the input matrix, T is the output matrix (labels), W and b are the input weights and bias, $g()$ is an activation function, H is the feature matrix of input data, L is

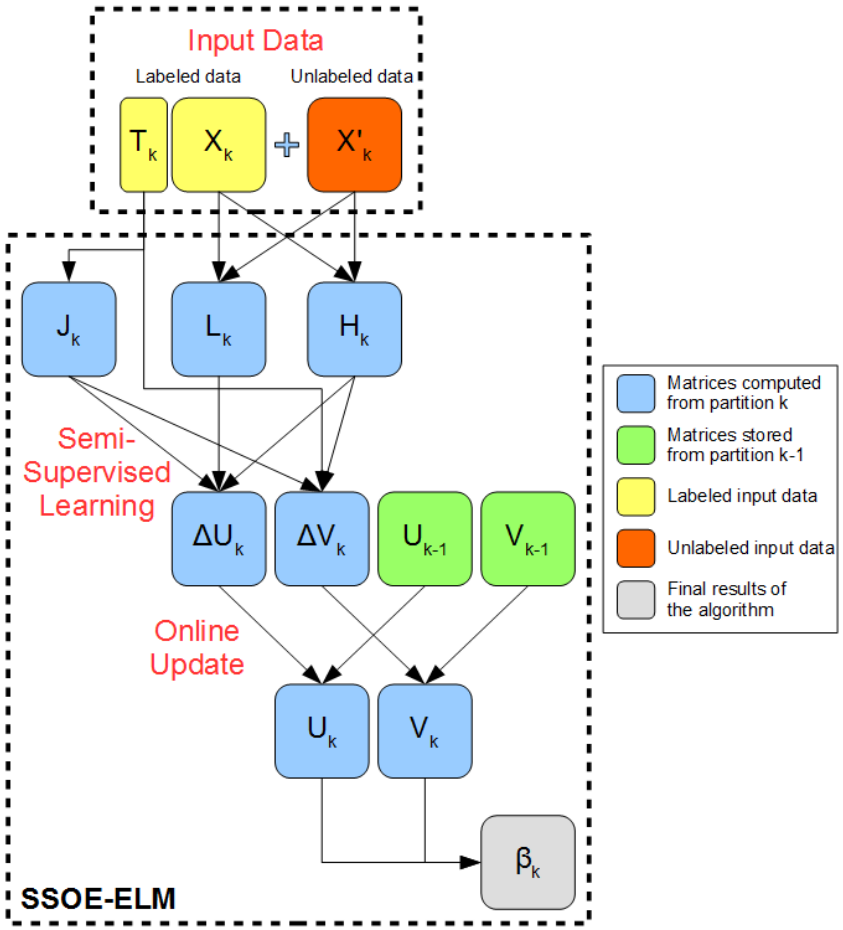

Figure 4 – Schematic diagram of SSOE-ELM with semi-supervised learning and online update steps for one training partition k

the graph Laplacian matrix of labeled and unlabeled training samples, J is the penalty coefficient matrix for unbalanced datasets, β is the network output weigths matrix and λ is a tradeoff parameter.

In the proposed algorithm SSOE-ELM, the graph Laplacian L of labeled and unlabeled input data is computed for each training partition k, as follows:

$$L_k = D_k - A_k \tag{3.3}$$

$$D_k[i,i] = \sum_{j=1}^{N_k} A_k[i,j] \tag{3.4}$$

$$A_k[i,j] = \begin{cases} adj(x_i, x_j), & \text{if } x_i \text{ and } x_j \text{ are neighbors} \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

where N_k is the number of labeled and unlabeled samples in partition k, D_k is a diagonal matrix with the degree of the vertices and A_k is the adjacency matrix of training partition X_k. The adjacency value $adj(x_i, x_j)$ can be computed by the Gaussian function $e^{-||x_i - x_j||^2/2\sigma^2}$, or simply fixed to 1. Likewise, penalty diagonal matrix J for unbalanced

datasets is also calculated for each training partition k, as follows:

$$J_k[i,i] = \begin{cases} C/N_{t_i}, & \text{if } x_i \text{ is a labeled sample} \\ 0, & \text{if } x_i \text{ is an unlabeled sample} \end{cases} \tag{3.6}$$

where $i = 1, ..., N_k$, C is a penalty coefficient, N_k is the number of labeled and unlabeled samples in partition k and N_{t_i} is the number of labeled samples from partition k that have the same label as sample x_i.

Based on SS-ELM formulation presented in (3.2), and considering only one training partition k, the β matrix of SSOE-ELM is calculated as follows:

$$\beta_k = (\alpha I + H_k^T J_k H_k + \lambda_k H_k^T L_k H_k)^{-1} H_k^T J_k T_k \tag{3.7}$$

where α is a regularization factor to avoid singular matrices and increase the stability of the solution and λ_k is the tradeoff factor between labeled and unlabeled data computed in training partition k.

As an online algorithm, SSOE-ELM is divided in two phases: initialization and sequential. Decomposing (3.7) into two intermediate matrices U and V as in Elastic ELM, we have the initialization phase of SSOE-ELM algorithm, where $k = 0$, as follows:

$$U_0 = H_0^T (J_0 + \lambda_0 L_0) H_0 \tag{3.8}$$

$$V_0 = H_0^T J_0 T_0 \tag{3.9}$$

$$\beta_0 = (\alpha I + U_0)^{-1} V_0 \tag{3.10}$$

Unlike SS-ELM, which use a fixed value for the parameter λ (usually obtained empirically), SSOE-ELM calculates the value of λ for each training partition, according to the number of samples received. The parameter assumes values between zero and one, where $\lambda = 0$ means to disregard the unlabeled data, and $\lambda = 1$ means giving same weight to the labeled and unlabeled data. Considering N_{k_L} and N_{k_U} respectively the number of labeled and unlabeled samples of a partition k, λ_k is calculated by:

$$\lambda_k = \frac{N_{k_U}}{N_{k_U} + N_{k_L}} \tag{3.11}$$

As in SOS-ELM, it is assumed that the graph Laplacian L_k of a given training partition k has no relation to the graph Laplacian of the previous or later partitions. This limitation was imposed on the model because otherwise it would be necessary to store all graph Laplacian already computed, which contradicts the online and incremental nature of the proposed algorithm.

In the sequential phase of the SSOE-ELM algorithm, where $k \geq 1$, matrices ΔU e ΔV are calculated based on the matrix H_k of the hidden layer of the new training partition

as follows:

$$\Delta U = H_k^T (J_k + \lambda_k L_k) H_k \tag{3.12}$$

$$\Delta V = H_k^T J_k T_k \tag{3.13}$$

For each new training partition received, the matrices U_k and V_k are updated as in Elastic ELM, in an incremental (I), decremental (D) or correctional (C) manner, as follows:

$$U_k = \begin{cases} U_{k-1} + \Delta U & (I) \\ U_{k-1} - \Delta U & (D) \\ U_{k-1} + \Delta U_I - \Delta U_D & (C) \end{cases} \tag{3.14}$$

$$V_k = \begin{cases} V_{k-1} + \Delta V & (I) \\ V_{k-1} - \Delta V & (D) \\ V_{k-1} + \Delta V_I - \Delta V_D & (C) \end{cases} \tag{3.15}$$

The updating formula of β_k is:

$$\beta_k = (\alpha I + U_k)^{-1} V_k \tag{3.16}$$

It is worth mentioning that the β calculation described in (3.10) and (3.16) can only be performed at the end of the training process, thus avoiding the computational cost of the inversion of the matrix U in each training partition. This makes SSOE-ELM faster than SOS-ELM, which calculates β recursively by inverting a reduced matrix at each training step. When used in large databases, or with data streams, this feature becomes a major advantage of SSOE-ELM. The SSOE-ELM pseudocode is described in Algorithm 7. Appendix A provides information on the computational time complexity of the SSOE-ELM algorithm.

3.1.1 Impacts of the size of training partition (N_k) in SSOE-ELM

Like Elastic ELM, the SSOE-ELM algorithm can receive training partitions of any size. The calculation of the U and V matrices does not require a specific number of training samples, nor does a training partition have the same number of samples as the previous one.

However, matrices L and J are calculated for each training partition, considering the number of samples N_k. In addition, the numbers of labeled and unlabeled samples from each training partition (N_{k_L} and N_{k_U}, respectively) are used to calculate the λ parameter. Therefore, it is necessary to establish minimum intervals for N_k values.

The μ parameter of the SSOE-ELM specifies how many neighbors will be considered when calculating the Laplacian matrix L. If the number of samples from the training

Algorithm 7 SSOE-ELM training

Input: Initial training partition with labeled training set $(X_0, T_0) = \{(x_i, t_i)|x_i \in R^d, t_i \in R^m, i = 1, \ldots, N_L\}$ and unlabeled training set $(X'_0) = \{(x'_i)|x'_i \in R^d, i = 1, \ldots, N_U\}$;
Activation function $g(x)$;
Number of hidden neurons n;
Regularization parameter α;
Penalty coefficient C;
Number os neighbors μ;

Output: input weights W, bias b, intermediate matrices U_k and V_k and output weights β

Initialization phase:

1: Initialize input weights W and bias b randomly drawn from an uniform distribution in [-1, 1];
2: With initial labeled training set (X_0, T_0) and initial unlabeled training set (X'_0), compute feature matrix H_0 similarly to ELM;
3: Compute tradeoff parameter λ_0 according to (3.11);
4: Compute Laplacian matrix L_0 according to (3.3);
5: Compute penalty matrix J_0 according to (3.6);
6: Compute matrix U_0 according to (3.8);
7: Compute matrix V_0 according to (3.9);

Sequential phase:

8: **for** each new training partition k, where $k \geq 1$, with labeled training set (X_k, T_k) and unlabeled training set (X'_k) **do**
9: Compute feature matrix H_k similarly to ELM;
10: Compute tradeoff parameter λ_k according to (3.11);
11: Compute Laplacian matrix L_k according to (3.3);
12: Compute penalty matrix J_k according to (3.6);
13: Compute matrix U_k according to (3.14);
14: Compute matrix V_k according to (3.15);
15: **end for**
16: Compute β_k according to (3.16);
17: **Return**

partition N_k is less than or equal to μ, all training samples will be neighbors, and will be considered close to each other. Thus, the Laplacian matrix will lose its ability to distinguish between near and distant samples, disregarding the smoothness assumption.

On the other hand, if N_k is greater than the number of neurons n, the calculation of the intermediate matrices becomes the operation with the highest computational cost, with complexity $O(nN_k^2)$. This operation is performed for each training partition, different from the β calculation, which is performed only at the end of the training process. This makes the SSOE-ELM training process slower, even if the number of neurons is not very large.

A good option for N_k is a value between μ and n. And although it is possible to train SSOE-ELM with partitions of different sizes, in the experiments performed in this thesis a fixed value for the size of the partition N_k was used. The partitions arriving from

the data stream can be adjusted to the N_k value, dividing a larger partition into several smaller ones, or accumulating the data that arrives until reaching the N_k value.

3.1.2 Limits of labeled partition of SSOE-ELM

As a semi-supervised algorithm that receives labeled and unlabeled partitions sequentially to train a classification model, SSOE-ELM is among supervised and unsupervised learning. To evaluate the limits of the SSOE-ELM labeled training partition, it is necessary to analyze three distinct cases where the algorithm could be applied: fully supervised learning; fully unsupervised learning; and semi-supervised learning.

3.1.2.1 Fully supervised case

In a fully supervised case, λ_k of SSOE-ELM becomes zero according to (3.11). V computation described by (3.9) and (3.13) remains the same, and U computation described by (3.8) and (3.12) becomes (3.17) and (3.18), respectively.

$$U_0 = H_0^T J_0 H_0 \tag{3.17}$$

$$\Delta U = H_{k+1}^T J_{k+1} H_{k+1} \tag{3.18}$$

As the training dataset is fully labeled, there are training samples of all classes and T in (3.9) and (3.13) is a matrix without zero-columns. In this situation, SSOE-ELM turns into a particular case of Elastic ELM with a penalty coefficient matrix J, suitable to classification tasks.

3.1.2.2 Fully unsupervised case

In a fully unsupervised case, T in (3.9) and (3.13) is a matrix with only zeros, turning V and ΔV into zero-matrices. Therefore, according to (3.10) and (3.16), β become a zero-matrix and SSOE-ELM is unable to classify any sample.

3.1.2.3 Semi-supervised case

SSOE-ELM is suitable for semi-supervised learning cases. The most important issue is to establish the limits of the labeled training partition in batch and online cases.

Batch case

A batch case is a special case of SSOE-ELM where all training samples are grouped in one partition. In this situation, only the initialization phase is required. This phase is described by (3.8)-(3.10).

Consider $T_0 = \{t_1, \ldots, t_N\}^T$ the output matrix of a training partition X_0 with N samples, where t_i is a row-vector with 1 in the column equivalent of sample label and zeros

otherwise. If there are no labeled samples of a class c_i in X_0, the i-th columns of T_0 will get only zeros. According to (3.9) and (3.10), the i-th column of V_0 and β_0 will also be zero vectors, and the model will be unable to correctly classify samples of label c_i.

On the other hand, if there are at least one labeled sample of a class c_i in X_0, the i-th column of β_0 will have non-zero values, making the model capable to classify samples of label c_i.

Condition 1: In batch cases (with only one training partition), the minimal condition to SSOE-ELM work properly is the presence of at least **one** labeled sample for each class (label) in training partition.

Online case

Online case of SSOE-ELM occurs when training samples arrive one-by-one or in small partitions. In this situation, initialization and sequential phases of SSOE-ELM are required.

Consider $T_k = \{t_1, \ldots, t_N\}^T$ the output matrix of a training partition X_k with N_k samples, where t_i is a row-vector with 1 in the column equivalent of sample label and zeros otherwise. Like in batch case, if there are no labeled samples of a class c_i in X_k, the i-th columns of T_k will get only zeros, and the i-th column of V_k and β_k will also be zero vectors. In this case, the model will be unable to correctly classify samples of label c_i.

As V matrix is incrementally updated, if a training partition X_{k+1} with at least one labeled sample of class c_i arrive, the model will be update according to (3.13) and (3.15). So, the i-th column of V_{k+1} and also β_{k+1} will have non-zero values, making the model capable to classify samples of label c_i.

Therefore, in online case it is not necessary that all training partitions includes samples of each class, as V_k is incrementally updated. If one training partition includes labeled samples for all classes, all other partitions can be fully unlabeled, and SSOE-ELM still works.

Condition 2: In online cases (with more than one training partition), the minimal condition to SSOE-ELM to work properly is the presence of at least **one** labeled sample of each class in the complete training dataset, regardless of the number or size of training partitions.

3.2 The algorithm SSOE-FP-ELM

In many real-world data stream problems, the situation known as concept drift is common. There are many reasons for concept drift to occur: changes or malfunction in data collection devices (sensors, cameras and others), changes in data pre-processing (filters, audio or video compression and others), transmission problems (truncated or incomplete

data, for example) or even changes in the data itself (changes in the problem domain; changes in the objects being monitored; inclusion, alteration or exclusion of features or classes, and many others). In problems like these, an algorithm that can detect and treat the concept drift is necessary.

The second proposed algorithm of this thesis is the Semi-Supervised Online Elastic ELM with hybrid Forgetting Parameter (SSOE-FP-ELM) (SILVA; KROHLING, 2019), an extension of SSOE-ELM with a semi-supervised forgetting parameter mechanism for data streams with concept drift.

SSOE-FP-ELM is divided into three main process: semi-supervised learning, forgetting parameter calculation and online update process. The first and third processes are similar to SSOE-ELM. The idea behind the SSOE-FP-ELM operation is to use the same semi-supervised mechanism of SSOE-ELM in each training partition separately, while monitoring changes in data. For each training partition, supervised and unsupervised forgetting factors are calculated to detect concept drift and adapt the model. These factors are combined in a semi-supervised forgetting parameter, which when applied to the intermediate matrices U and V, changes the way the model learns from the training samples. Figure 5 shows the schematic diagram illustrating SSOE-FP-ELM with one training partition k.

Like SSOE-ELM, SSOE-FP-ELM has two phases: initialization and sequential. Consider a training set arriving sequentially in partitions. Each partition k is represented by labeled samples (x_i, t_i), $x_i \in R^d$, $t_i \in R^m, i = 1, \ldots, N_{k_L}$ and unlabeled samples (x'_i), $x'_i \in R^d, i = 1, \ldots, N_{k_U}$, where N_{k_L} and N_{k_U} are the number of labeled and unlabeled samples in partition k respectively, d is the number of features and m is the number of labels. The initialization phase of SSOE-FP-ELM, where $k = 0$, is similar to SSOE-ELM, as follows:

$$U_0 = H_0^T(J_0 + \lambda_0 L_0)H_0 \tag{3.19}$$

$$V_0 = H_0^T J_0 T_0 \tag{3.20}$$

$$\beta_0 = (\alpha I + U_0)^{-1} V_0 \tag{3.21}$$

where T is the output matrix (labels), H is the feature matrix of input data, L is the graph Laplacian of the labeled and unlabeled training data, J is the penalty coefficient matrix for unbalanced datasets, β is the network output weigths matrix, α is the regularization factor to avoid singular matrices and increase the stability of the solution and λ is the tradeoff factor between labeled and unlabeled data.

The SSOE-ELM algorithm, in sequential phase, allows incremental, decremental, and correctional learning. A correctional learning occurs in situations when a model adds a set of training samples and removes another set at the same time. The sequential phase

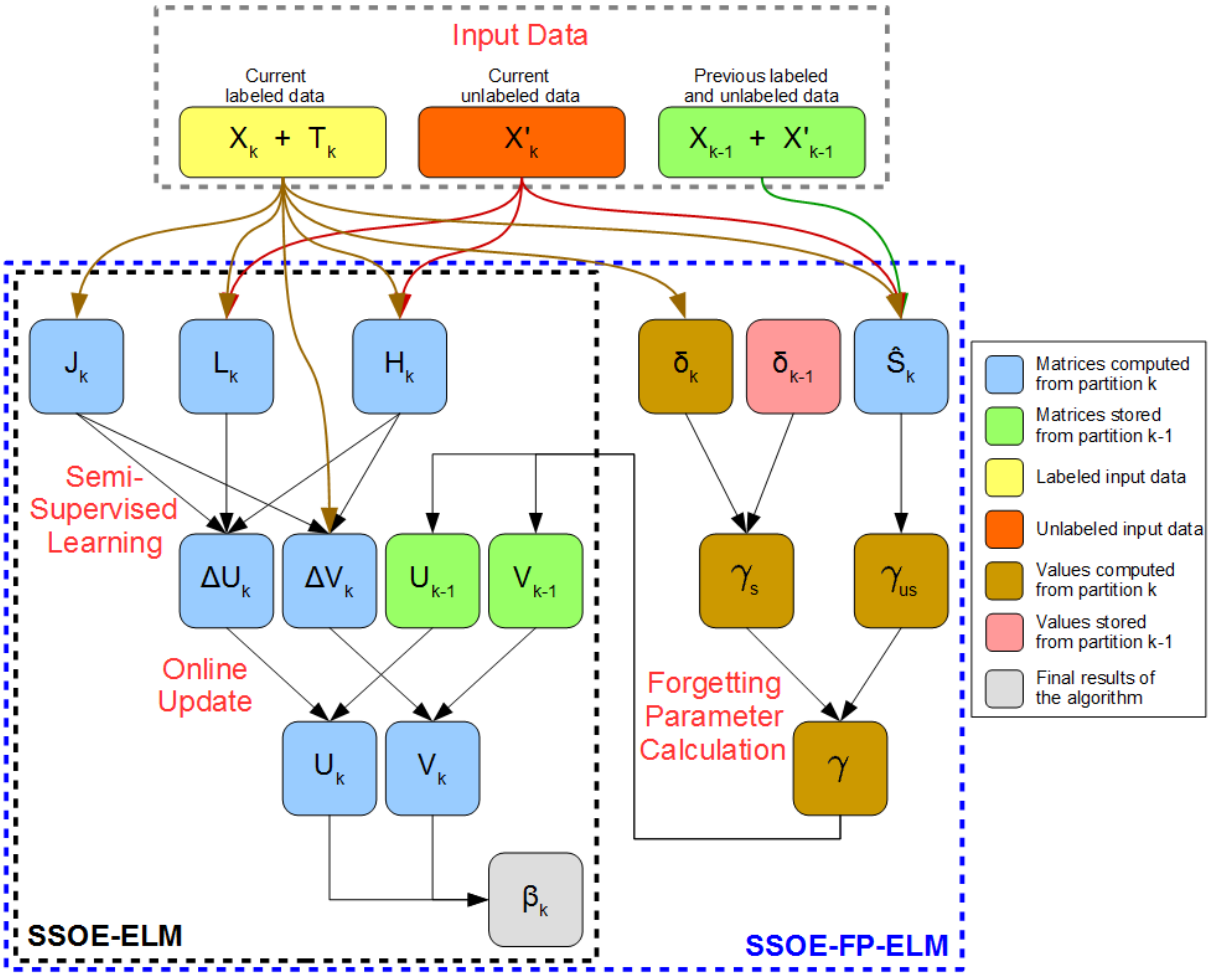

Figure 5 – Schematic diagram of SSOE-FP-ELM with semi-supervised learning, forgetting parameter calculation and online update steps for one training partition k

of SSOE-ELM with correctional learning is described as:

$$U_k = U_{k-1} + \Delta U_I - \Delta U_D \tag{3.22}$$

$$V_k = V_{k-1} + \Delta V_I - \Delta V_D \tag{3.23}$$

where ΔU_I and ΔV_I are the new training partition matrices, and ΔU_D and ΔV_D are the portion of U_{k-1} and V_{k-1} matrices that needs to be removed.

When ΔU_D and U_{k-1} are the same matrix, a "forgetting effect" occurs and the data in U_{k-1} is removed from the new matrix U_k. Applying a factor γ in ΔU_D, it is possible to control the rate of this forgetting effect. Thus, by rewriting (3.22) and (3.23), we obtain the forgetting mechanism formulation as follows:

$$U_k = (1 - \gamma)U_{k-1} + \Delta U_I \tag{3.24}$$

$$V_k = (1 - \gamma)V_{k-1} + \Delta V_I \tag{3.25}$$

Considering (3.24) and (3.25), the sequential phase of SSOE-FP-ELM for a training

partition k, where $k \geq 1$, can be described as:

$$\Delta U = H_k^T (J_k + \lambda_k L_k) H_k \tag{3.26}$$

$$\Delta V = H_k^T J_k T_k \tag{3.27}$$

$$U_k = (1 - \gamma) U_{k-1} + \Delta U \tag{3.28}$$

$$V_k = (1 - \gamma) V_{k-1} + \Delta V \tag{3.29}$$

$$\beta_k = (\alpha I + U_k)^{-1} V_k \tag{3.30}$$

where γ is the forgetting parameter in the range [0, 1].

It is interesting to note that the forgetting mechanisms of SSOE-FP-ELM can be seen as a special case of correctional learning with a forgetting parameter γ. When $\gamma = 0$, no forgetting mechanism is applied, and SSOE-FP-ELM work similar to SSOE-ELM. When $\gamma = 1$, all prior knowledge of the model is discarded, and a new model is trained from the beginning with the new training partition k.

It is important to properly determine the γ parameter. Values close to zero may not deal with the concept drift, while values close to one may force the model to retrain unnecessarily. Zhao, Wang & Park (2012) and Liu, Wu & Jiang (2016) propose supervised approaches to calculate the forgetting parameter, evaluating the accuracy of each training partition and using the obtained error to adapt the forgetting parameter. Although this approach demonstrates good results, it requires a large amount of labeled samples for evaluation and does not take into account unlabeled samples. In SSOE-FP-ELM, a semi-supervised forgetting parameter γ is proposed as follows:

$$\gamma = \ max(\gamma_s, \gamma_{us}) \tag{3.31}$$

where γ_s is a supervised factor calculated from the labeled samples of partition k, and γ_{us} is an unsupervised factor, calculated from all samples of partition k.

3.2.1 Calculation of the semi-supervised forgetting parameter

To calculate the forgetting parameter γ, SSOE-FP-ELM needs to keep in memory the previous and current training partition, as well as a measure of accuracy of the previous partition.

3.2.1.1 Supervised factor of forgetting parameter

The supervised factor γ_s is calculated by comparing the accuracy of the supervised classification of the model in two consecutive training partitions. A significant reduction in accuracy may indicate that concept drift occurred.

When a partition k arrives, where $k \geq 1$, all labeled samples of k are first used to evaluate the accuracy δ_k of the model, where δ_k is in the range [0, 1]. This accuracy is

compared with the previous accuracy δ_{k-1}, and the factor γ_s is given by:

$$\gamma_s = \begin{cases} \delta_{k-1} - \delta_k, & \delta_k < \delta_{k-1} \\ 0, & \delta_k \geq \delta_{k-1} \end{cases} \tag{3.32}$$

where $\gamma_s = 0$ means no supervised forgetting mechanism is needed, and $\gamma_s = 1$ means all prior knowledge of the supervised portion of the model is discarded.

3.2.1.2 Unsupervised factor of forgetting parameter

The unsupervised factor γ_{us} is calculated by comparing the probability distributions of each feature in two consecutive training partitions. A change in the probability distributions of a significant number of features may indicate a concept drift.

To measure this changes, a nonparametric statistical test is needed. For this task, the Wilcoxon test was choosen, but there are other methods (SIMPSON, 2015). According to Barros, Hidalgo & Cabral (2018), the Wilcoxon test can be used to determine whether two independent samples come from populations with the same distribution. In SSOE-FP-ELM, the Wilcoxon test is used for each feature of the dataset to discriminate those whose distribution has changed (PREL et al., 2010). If the values of a feature in a k training partition show significant changes in its probabilistic distribution in relation to the values in the k-1 partition, that feature is marked as "changed".

It is important to note that the features of a dataset do not have equal importance. Some features have a greater impact on the classification result than others. Therefore, in addition to identifying which features have changed, it is necessary to quantify the importance of these features selected by the Wilcoxon test. For this task, the Laplacian Scores were used as a feature selection technique, as proposed by He, Cai & Niyogi (2006). The Laplacian Scores are based on Laplacian Eigenmaps and Locality Preserving Projection, and measures the power of locality preserving of each feature, considering that data from the same class are often close to each other. For a training partition k of a dataset $X \in R^d$, where d is the number of features, there are Laplacian scores $S = (S_i)$, where $i = 1, \ldots, d$, indicating the relevance of each feature. A score near zero indicates a good feature. For each feature i of the training dataset, S_i is computed as follows:

$$f_i' = [f_{i,1}, \ldots, f_{i,N_k}]^T, \quad 1 = [1, \ldots, 1]^T,$$

$$\tilde{f}_i = f_i' - \frac{f_i'^T D_k 1}{1^T D_k 1} 1 \tag{3.33}$$

$$S_i = \frac{\tilde{f}_i^T L_k \tilde{f}_i}{\tilde{f}_i^T D_k \tilde{f}_i} \tag{3.34}$$

where $f_{i,j}$ is the j-th sample of the i-th feature, $j = 1, \ldots, N_k$; 1 is a column vector of ones; L is the graph Laplacian of training dataset and D is the diagonal matrix of dimension N_kxN_k with the degree of the vertices used to compute L.

In this work, the Laplacian scores computed as in (3.34) are then normalized in the range [0, 1] and reversed, so that the most important features have the highest values. We denote as $S' = (S'_i)$, where $i = 1, \ldots, d$, the normalized and reversed Laplacian scores of a training partition k, computed as follows:

$$S'_i = |\frac{S_i}{max(S)} - 1| \tag{3.35}$$

The unsupervised factor γ_{us} is calculated as follows:

$$\hat{S}_i \begin{cases} S'_i, & \text{if feature } i \text{ has changed according to Wilcoxon test} \\ 0, & \text{otherwise} \end{cases} \tag{3.36}$$

$$\gamma_{us} = \frac{\sum_{i=1}^{d} \hat{S}_i}{\sum_{i=1}^{d} S'_i} \tag{3.37}$$

where $\hat{S} = (\hat{S}_i)$, where $i = 1, \ldots, d$, is the normalized and reversed Laplacian scores only of the features that were changed according to the Wilcoxon test and d is the number of features.

Algorithm 8 Calculating Forgetting Parameter γ

Input: Training partition k with labeled training set $(X_k, T_k) = \{(x_i, t_i) | x_i \in R^d, t_i \in R^m, i = 1, \ldots, N_L\}$ and unlabeled training set $(X'_k) = \{(x'_i) | x'_i \in R^d, i = 1, \ldots, N_U\}$;
Previous training partition k-1 with labeled training set (X_{k-1}, T_{k-1}) and unlabeled training set (X'_{k-1});
Accuracy δ_{k-1} of previous partition;
Output: Forgetting Parameter γ
Supervised factor γ_s:
Compute accuracy δ_k of training partition k using labeled training set (X_k, T_k);
Compute supervised factor γ_s according to (3.32);
Unsupervised factor γ_{us}:
Compute graph Laplacian L_k and diagonal matrix D_k of training partition k using labeled training set (X_k) and unlabeled training set (X'_k);
for each feature i of training partition k **do**
    Compute Laplacian Scores S_i according to (3.34);
end for
for each feature i of training partition k **do**
    Compute normalized and reversed Laplacian Scores S'_i according to (3.35);
    Perform Wilcoxon test using training partition k and training partition k-1;
    Compute $\hat{S}_i$ according to (3.36);
end for
Compute unsupervised factor γ_{us} according to (3.37);
Forgetting Parameter γ:
Compute γ according to (3.31);
Return

The γ_{us} factor is given by the sum of the Laplacian scores of the features that were changed divided by the sum of the Laplacian scores of all features, where $\gamma_{us} = 0$ means

no unsupervised forgetting mechanism is needed (no features have been changed), and γ_{us} = 1 means all prior knowledge of the unsupervised portion of the model is discarded (all features have been changed). After computing γ_s and γ_{us}, the forgetting parameter γ is given by (3.31). Algorithm 8 details the calculation of γ. The SSOE-FP-ELM pseudocode is described in Algorithm 9. Appendix B provides information on the computational time complexity of the SSOE-FP-ELM algorithm.

Algorithm 9 SSOE-FP-ELM training

Input: Initial training partition with labeled training set $(X_0, T_0) = \{(x_i, t_i)|x_i \in R^d, t_i \in R^m, i = 1, \ldots, N_L\}$ and unlabeled training set $(X'_0) = \{(x'_i)|x'_i \in R^d, i = 1, \ldots, N_U\}$;
Activation function $g(x)$;
Number of hidden neurons n;
Regularization parameter α;
Penalty coefficient C;

Output: input weights W, bias b, intermediate matrices U_k and V_k and output weights β

Initialization phase:

1: Initialize input weights W and bias b randomly drawn from an uniform distribution in [-1, 1];
2: With initial labeled training set (X_0, T_0) and initial unlabeled training set (X'_0), compute feature matrix H_0 similarly to ELM;
3: Compute tradeoff parameter λ, graph Laplacian L_0 and penalty matrix J_0 similarly to SSOE-ELM;
4: Compute matrix U_0 according to (3.19);
5: Compute matrix V_0 according to (3.20);
6: Compute matrix β_0 according to (3.21);
7: Compute and store δ_0 as the accuracy of labeled training set (X_0, T_0), and store initial training partition;

Sequential phase:

8: **for** each new training partition k, where $k \geq 1$, with labeled training set (X_k, T_k) and unlabeled training set (X'_k) **do**
9: Compute feature matrix H_k similarly to ELM;
10: Compute tradeoff parameter λ, graph Laplacian L_k and penalty matrix J_k similarly to SSOE-ELM;
11: Compute forgetting parameter γ according to Algorithm 8;
12: Compute matrix U_k according to (3.28);
13: Compute matrix V_k according to (3.29);
14: Compute matrix β_k according to (3.30);
15: Compute and store δ_k as the accuracy of labeled training set (X_k, T_k), and store training partition k;
16: **end for**
17: **Return**

3.2.2 Impacts of the training partitions in SSOE-FP-ELM

The SSOE-FP-ELM algorithm has the same advantages and restrictions as the SSOE-ELM with respect to the size of the training partitions (N_k).

Due to the calculation of the forgetting parameter, the SSOE-FP-ELM needs to check the accuracy of the model with each new training partition. If the measured accuracy is far below the accuracy observed in the previous partition, a forgetting parameter close to 1 will be generated, forcing the model to forget the previous knowledge obtained. In situations where the training partition has few labeled instances, the algorithm may incorrectly detect a concept drift due to this drop in accuracy, leading the model to forget its prior knowledge. To prevent this undesired behavior, it is important that each training partition has labeled samples, ensuring the calculation of γ.

It is important to note that the operations for calculating the unsupervised factor of γ, in particular the calculation of Laplacian scores, are performed on the X matrix of the input data, being performed at each training partition processed. For datasets with a large number of columns (for example the Gisette dataset, used in the experiments that will be presented), the calculation of Laplacian scores becomes a computationally expensive operation, increasing the training time.

3.3 Remarks

In this chapter, two new algorithms have been proposed, based on ELM variants. The first one, called Semi-Supervised Online Elastic Extreme Learning Machine (SSOE-ELM) is a semi-supervised algorithm, with online update and low training time, developed to handle large and unbalanced datasets and data streams. The SSOE-ELM is trained sequentially with labeled and unlabeled samples, and the complete training dataset is not required at the beginning of the training process.

The second algorithm proposed in this chapter is called Semi-Supervised Online Elastic Extreme Learning Machine with hybrid Forgetting Parameter (SSOE-FP-ELM). It extends the SSOE-ELM, incorporating a concept drift treatment mechanism. In data streams, the data received may change over time. These variations can be due to many factors: problems with data collection, transmission or pre-processing; changes in the monitored environment; among others. To address these drifts, a semi-supervised forgetting parameter was added to the SSOE-ELM to detect changes and adapt the model to the new dataset. This parameter is composed of a supervised factor, which periodically checks the model for accuracy drops as an indicator of concept drifts, and an unsupervised factor, which verifies changes in the input data distribution as an indicator of concept drifts. These two factors are combined, generating a forgetting parameter that is responsible for adapting the model to the detected changes, either updating with the new information, or discarding all the acquired knowledge and retraining from the beginning.

4 Experimental Results

In order to assess the performance of the two proposed algorithms, four sets of experiments were performed. They are described in the following.

In the first experiment, the proposed algorithms were tested in public benchmarks in a fully supervised way, comparing the obtained results with another semi-supervised and online algorithm named SOS-ELM and a well-known supervised algorithm, the Support Vector Machine (SVM). In the second experiment, the proposed algorithms were tested in the same benchmarks, in a semi-supervised way, comparing the results with the SOS-ELM algorithm. In the third experiment, the proposed algorithms were tested in a fully supervised way in concept drift situations, comparing the obtained results with SOS-ELM and FP-ELM. In the fourth experiment, the proposed algorithms were tested in a semi-supervised way in concept drift situations, comparing the results with the SOS-ELM algorithm.

It is important to note that the proposed algorithms are aimed at data streams, but the experiments carried out were performed with static datasets. This was done because simulating the experiments using real data streams poses an additional difficulty: the long-term availability of the same data used in the experiments presented. To allow the reproducibility of the experiments, static datasets were used, divided into small partitions (chunks), and each partition is presented to the algorithm separately, sequentially, as if the data had arrived through a data stream. If other researchers wish to repeat the experiments, they only need to use the same datasets, which are in the public domain, and follow the same process of creating the partitions.

All code was implemented in Python and Tensorflow, and was tested on an Intel computer with a 2.60GHz Core i7-6 processor and 16GB RAM memory, with an NVidia GeForce GTX 960M graphics card. For SVM, the standard implementation of Scikit-Learn library (PEDREGOSA et al., 2011) was used. The source codes are available online at https://bitbucket.org/carlosalexandress/ssoe-elm-ssoe-fp-elm.

4.1 Benchmarks

For these experiments, 12 public domain classification benchmarks were used. For better organization, these benchmarks have been separated into image datasets and textual datasets, and are described in the following sections. In all datasets, inputs were normalized between 0 and 1. Table 1 presents the number of features, number of labels, number of training and test samples of each dataset and an indication of whether or not the dataset

has a test partition. To show the degree of unbalance of the datasets, the number of samples from the majority and minority classes are reported. For datasets that have a test partition, the number of samples from the majority and minority class were collected from the training partition. For datasets that do not have a test partition, these values were collected from the complete dataset.

Table 1 – Benchmarks

Dataset	Features	Labels	Train Samples	Test Samples	Test Partition	Minority Class	Majority Class
COIL20	1024	20	1008	432	No	72	72
COIL100	1024	100	5040	2160	No	72	72
CrowdMap	28	6	10545	300	Yes	53	7431
DNA	180	3	2000	1186	Yes	464	1051
Gisette	5000	2	6000	1000	Yes	3000	3000
Isolet	617	26	6238	1559	Yes	238	240
KDEF_Front	900	7	686	294	No	140	140
Musk	166	2	4618	1980	No	1017	5581
Spam	57	2	3220	1381	No	1813	2788
StatImgSeg	19	7	1617	693	No	330	330
StatLandSat	36	6	3104	1331	Yes	415	1072
Waveform	21	3	3500	1500	No	1647	1696

4.1.1 Image datasets

In these datasets, inputs are image pixels, organized in a one-dimensional vector.

The Columbia Object Image Library datasets (COIL-20 and COIL-100) are available from the Columbia University Computer Vision Laboratory. COIL-20 (NENE; NAYAR; MURASE, 1996b) is a database of 1400 grayscale images of 20 objects, placed on a motorized turntable against a black background. The objects have a wide variety of complex geometric and reflectance characteristics. The turntable was rotated through 360 degrees to vary object pose with respect to a fixed camera. Each object has 72 images of size 640x480, with pose intervals of 5 degrees. COIL-100 dataset (NENE; NAYAR; MURASE, 1996a) contains 7200 color images of 100 objects (72 images per object) of size 640x480, acquired in a similar way to the COIL-20 dataset. In this thesis, COIL-100 dataset was converted to grayscale, and both COIL-20 and COIL-100 were rescaled to 32x32 pixels.

The Karolinska Directed Emotional Faces (KDEF) dataset (LUNDQVIST; FLYKT; ÖHMAN, 1998) is available at the Emotion Lab at Karolinska Institutet. KDEF is a set of 4900 562x762 images of human facial expressions. The set of pictures contains 70 individuals displaying 7 different emotional expressions (neutral, happy, angry, afraid, disgusted, sad and surprised) in 2 different photo sessions. Each expression is viewed from 5 different angles (full left profile, half left profile, straight, half right profile, full right

profile). In this thesis, a subset of KDEF dataset containing only the front-face images (980 samples) rescaled to 30x30 pixels was used to classify facial expression. This subset was named "KDEF_FRONT". The classification task is to recognize the facial expression of each sample.

4.1.2 Textual datasets

In these datasets, the inputs are handcrafted or calculated features, organized in a one-dimensional vector. Even on the datasets where the samples are images, the pixels of the image are not used as inputs.

The CrowdSourcedMapping dataset is available at the University of California Irvine (UCI) Machine Learning Repository (DUA; GRAFF, 2019). This dataset was derived from geospatial data from two sources: 1) Landsat time-series satellite imagery from the years 2014-2015; 2) crowdsourced georeferenced polygons with land cover labels obtained from OpenStreetMap. The CrowdSourcedMapping dataset brings two major difficulties to the classification task: the test set is very small in relation to the training set, and there is a large presence of noise and class labeling errors in the training set.

The DNA dataset is available at the OpenML Machine Learning Datasets (VANSCHOREN et al., 2013) and LIBSVM Datasets Repository (FAN; LIN, 2019). This dataset consists of 3186 splice junctions of primate gene sequences (DNA). Splice junctions are points on a DNA sequence at which 'superfluous' DNA is removed during the process of protein creation in higher organisms. The classification task in this dataset is to recognize, given a sequence of DNA, the boundaries between exons (the parts of the DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out). The 3 classes are: exon/intron boundaries (EI); intron/exon boundaries (IE); or Neither.

The Gisette dataset (DUA; GRAFF, 2019) and LIBSVM Datasets Repository (FAN; LIN, 2019). This dataset was used in the NIPS 2003 feature selection challenge. It is based on the MNIST dataset and aims to classify the digits "4" and "9". 2500 real features were generated from the data, and another 2500 random features without predictive power were added to increase the complexity of the problem.

The Isolet dataset (DUA; GRAFF, 2019) was generated from the sounds of the letters of the alphabet. 150 subjects spoke the name of each letter of the alphabet twice. Of these 150 subjects, 120 were separated in the training set and the remaining 30 in the test set.

The Musk dataset (DUA; GRAFF, 2019) describes a set of 102 molecules (and all possible rotations of these molecules) of which 39 are judged by human experts to be musks and the remaining 63 molecules are judged to be non-musks.

The Spambase dataset (DUA; GRAFF, 2019) contains statistical values calculated

from the textual content of emails marked by human agents as "spam" or "non-spam", indicating whether or not an email is spam.

The Statlog Image Segmentation (DUA; GRAFF, 2019) instances are samples of a 3x3 region randomly drawn from a database of 7 outdoor images. The images were handsegmented to create a classification for every pixel.

The Statlog Landsat Satellite dataset (DUA; GRAFF, 2019) consists of the multi-spectral values of pixels in 3x3 neighbourhoods in a satellite image, and the classification associated with the central pixel in each neighbourhood. The aim is to predict this classification, given the multi-spectral values.

The Waveform dataset (DUA; GRAFF, 2019) contains 3 classes of waves, each one generated from a combination of 2 or 3 "base" waves. All features include noise (mean 0, variance 1).

4.2 Hyperparameter optimization

For a fair comparison, each of the algorithms was executed using a set of hyperparameters close to the optimum. For this, it was necessary to use a hyperparameter tuning technique. There are several techniques for this, and three of the most known and simple are Manual Search, Grid Search and Random Search.

Manual Search is the adjustment of hyperparameters through manual configuration, based on the researcher's prior knowledge of the problem domain. The researcher must define restricted search spaces for each hyperparameter, making small adjustments. The main disadvantage of this method is the need to know each problem in depth to adjust the hyperparameters.

Grid search, according to Ensor & Glynn (1997), is an exhaustive search using a manually specified subset of the hyperparameter decision space of an objective function that you want to maximize (accuracy) or minimize (error rate). Grid Search has the disadvantage of having a high computational cost, as it evaluates each combination of hyperparameter values, even those in less promising search spaces.

Random Search is a technique based on Grid Search in which only a few random combinations of hyperparameters are evaluated. According to Bergstra & Bengio (2012), Random Search has all the practical advantages of Grid Search (conceptual simplicity, ease of implementation, trivial parallelism) and trades a small reduction in efficiency in low-dimensional spaces for a large improvement in efficiency in high-dimensional search spaces.

For these reasons, a Random Search technique was used in this thesis in hyperparameter optimization in all experiments performed.

4.3 Experimental setup

For the datasets that had a test partition, this partition was used. For the datasets without a test partition, data was shuffled and divided into 70% for training and 30% for test. In semi-supervised experiments, the training partition was divided into labeled and unlabeled samples according to the ratios described in the specific section of the experiment.

The parameters of the algorithms for each dataset were determined by Random Search method, using the training partition as validation. For SSOE-ELM, SSOE-FP-ELM and SOS-ELM, values in the range $[10^{-5}, 10^{-4}, \ldots, 10^{4}, 10^{5}]$ were tested for the parameter C. The size of the chunks N_k for online training of the algorithms was initially tested with values in the interval [50, 52, ..., 200], and was later set to 200 for all algorithms to facilitate the reproduction of the experiments. For the SOS-ELM algorithm, the tradeoff parameter λ was tested with values between 0 and 1. Note that for parameters C, N_k and λ, the same ranges presented by Jia et al. (2016) were adopted. The values tested for the number of neighbors μ, required to compute graph Laplacian, were [5, 10, 15, 20]. Huang et al. (2012) chooses the regularization parameter α of ELM in the range $[2^{-24}, 2^{-23}, \ldots, 2^{24}, 2^{25}]$, so the same interval was used in this thesis for SSOE-ELM and SSOE-FP-ELM. For FP-ELM, the parameters θ and η were tested with values in the range [0.0, 0.1, ..., 19.9, 20.0], as proposed by Liu, Wu & Jiang (2016). For the regularization parameter α, Liu, Wu & Jiang (2016) set a fixed value of 0.02, and in this thesis this value was also used. The number of neurons n for all algorithms was tested with values in the interval [50, 100, ..., 3000]. For SVM, Hsu, Chang & Lin (2003) suggests the choice of regularization parameter α in the range $[2^{-5}, 2^{-3}, \ldots, 2^{15}]$ and parameter γ in the range $[2^{-15}, 2^{-13}, \ldots, 2^{3}]$. In this thesis, the suggested intervals have been extended to $[2^{-5}, 2^{-4}, \ldots, 2^{15}]$ for α and $[2^{-15}, 2^{-14}, \ldots, 2^{3}]$ for γ. In all experiments, the RBF kernel was used. For each experiment, an hyperparameter tuning was performed for all algorithms using the intervals described above.

It is worth mentioning that the size of chunks N_k does not need to be fixed, being able to change with each training partition. In this thesis, the same value of N_k was used for all training partitions of a given experiment just to facilitate the reproducibility of the experiment.

4.4 First experiment - supervised comparison

In this first experiment, SSOE-ELM and SSOE-FP-ELM were evaluated in 12 public benchmarks, in a fully supervised way (all training samples are labeled), comparing the obtained results with SOS-ELM, FP-ELM and SVM. An hyperparameter tuning with Random Search method was performed in all algorithms. Table 2 presents these

parameters.

Table 2 – Algorithm parameters for supervised experiment

	SSOE-ELM				SSOE-FP-ELM				SOS-ELM				FP-ELM			SVM	
Dataset	n	α	C	μ	n	α	C	μ	n	λ	C	μ	n	θ	η	γ	α
COIL20	2000	2^{-5}	10^{3}	20	2550	2^{3}	10^{3}	15	2700	0.24	10^{3}	20	2700	2.5	5.5	2^{-5}	2^{5}
COIL100	2300	2^{4}	10^{3}	5	2500	2^{-5}	10^{-1}	15	2000	0.26	10^{3}	10	2100	8.3	6.3	2^{-7}	2^{10}
CrowdMap	1300	2^{-5}	10^{-3}	15	2250	2^{-8}	10^{-4}	5	300	0.15	10^{2}	10	300	13.7	1.3	2^{0}	2^{8}
DNA	3000	2^{-15}	10^{-5}	5	1450	2^{-16}	10^{-5}	20	600	0.45	10^{5}	15	350	14.9	3.7	2^{-5}	2^{2}
Gisette	2200	2^{9}	10^{2}	5	2950	2^{14}	10^{3}	20	1250	0.89	10^{0}	5	1800	10.3	15.6	2^{-9}	2^{4}
Isolet	2950	2^{7}	10^{1}	5	2450	2^{-15}	10^{-5}	20	2700	0.14	10^{4}	20	2300	17.2	13.6	2^{-6}	2^{6}
KDE_Front	3000	2^{15}	10^{5}	5	3000	2^{13}	10^{4}	15	2700	0.15	10^{-4}	5	3000	15.4	2.1	2^{-8}	2^{10}
Musk	2950	2^{-15}	10^{-2}	20	2750	2^{3}	10^{3}	5	2150	0.27	10^{4}	10	2800	9.8	10.8	2^{-3}	2^{10}
Spam	600	2^{-22}	10^{-3}	20	2800	2^{-10}	10^{2}	20	2250	0.27	10^{4}	15	1950	12.0	17.0	2^{-2}	2^{8}
StatImgSeg	2450	2^{3}	10^{5}	5	1000	2^{-22}	10^{-1}	15	2100	0.34	10^{5}	20	2650	9.2	7.0	2^{2}	2^{11}
StatLandSat	650	2^{-10}	10^{4}	10	2100	2^{-10}	10^{-1}	10	2250	0.66	10^{4}	10	2300	7.6	19.0	2^{2}	2^{8}
Waveform	2150	2^{-9}	10^{-1}	20	2650	2^{-2}	10^{0}	5	1800	0.08	10^{3}	10	650	6.7	18.0	2^{-15}	2^{14}

For each dataset, the experiment was repeated 30 times, using the mean and standard deviation of the classification accuracy as performance metrics. In Table 3, the mean accuracy of each algorithm is presented.

Table 3 – Algorithms accuracy (%) in supervised experiment

Dataset	SSOE-ELM	SSOE-FP-ELM	SOS-ELM	FP-ELM	SVM
COIL20	99.8 ± 0.2	**99.9 ± 0.2**	99.8 ± 0.3	99.8 ± 0.2	99.8 ± 0.2
COIL100	96.2 ± 0.4	96.6 ± 0.5	95.9 ± 0.5	96.2 ± 0.5	**99.7 ± 0.1**
CrowdMap	**51.6 ± 1.4**	**51.6 ± 1.5**	34.4 ± 3.5	27.8 ± 2.9	26.0 ± 0.0
DNA	93.6 ± 0.3	92.9 ± 0.4	88.2 ± 1.0	90.2 ± 0.6	**95.7 ± 0.0**
Gisette	95.9 ± 0.4	96.3 ± 0.3	93.5 ± 0.7	94.0 ± 0.7	**98.1 ± 0.0**
Isolet	95.2 ± 0.3	95.0 ± 0.3	94.4 ± 0.4	94.5 ± 0.5	**96.9 ± 0.0**
KDEF_Front	80.8 ± 2.0	80.7 ± 2.3	77.0 ± 2.8	72.0 ± 2.6	**82.7 ± 2.2**
Musk	99.1 ± 0.3	98.7 ± 0.3	98.7 ± 0.2	99.2 ± 0.3	**99.6 ± 0.2**
Spam	92.4 ± 0.7	92.5 ± 0.6	92.8 ± 0.6	92.8 ± 0.6	**93.8 ± 0.5**
StatlogImgSeg	95.9 ± 0.9	95.6 ± 0.7	96.2 ± 0.7	94.6 ± 1.0	**96.9 ± 0.7**
StatlogLandSat	87.5 ± 0.4	87.8 ± 0.2	87.4 ± 0.2	85.2 ± 0.1	**90.4 ± 0.0**
Waveform	86.8 ± 0.7	**87.2 ± 0.7**	86.3 ± 0.7	86.1 ± 0.7	87.0 ± 0.6

The Wilcoxon test was performed to a pairwise comparison of the performance of the algorithms for each dataset. In Table 4 the p-values between SSOE-ELM and the other algorithms for each dataset are presented. In values marked with '*', p-value ≤ 0.01 and, according to Derrac et al. (2011), there is a significant difference between the two algorithms.

For COIL20 dataset, there is no significant differences between the algorithms. There are 7 significant differences between SSOE-ELM and SOS-ELM, and in all of them the proposed algorithm has the highest accuracy. One of the causes of this behavior is the presence of a regularization parameter α in the SSOE-ELM. According to Huang et al. (2012), the use of a regularization factor in the ELM formulation makes the obtained solution more stable and with greater generalization ability.

Table 4 – p-value of the Wilcoxon test in comparison between SSOE-ELM and other algorithms in supervised experiment

Datasets	SSOE-ELM SOS-ELM	SSOE-ELM FP-ELM	SSOE-ELM SVM	SSOE-ELM SSOE-FP-ELM
COIL20	0.4035	0.2612	0.2254	0.3147
COIL100	0.0176	0.9528	0.0000*	0.0011*
CrowdMap	0.0000*	0.0000*	0.0000*	0.6952
DNA	0.0000*	0.0000*	0.0000*	0.0000*
Gisette	0.0000*	0.0000*	0.0000*	0.0009*
Isolet	0.0000*	0.0000*	0.0000*	0.0555
KDEF_Front	0.0000*	0.0000*	0.0030*	0.8016
Musk	0.0000*	0.2550	0.0000*	0.0001*
Spam	0.0399	0.0428	0.0000*	0.4553
StatlogImgSeg	0.2550	0.0000*	0.0000*	0.0679
StatlogLandSat	0.6681	0.0000*	0.0000*	0.0003*
Waveform	0.0061*	0.0007*	0.6520	0.0963

Between SSOE-ELM and SVM, there are 10 significant differences, and only in the CrowdSourcedMapping dataset the proposed algorithm have greater accuracy than SVM. This result was already expected, since the differential of the proposed algorithm is in semi-supervised cases. But even with performance below SVM in these 9 datasets, the difference between the two algorithms was small, ranging from 1% to 3% in the average accuracy.

Between SSOE-ELM and FP-ELM, there are 8 significant differences, and in all of them the proposed algorithm has the highest accuracy.

In Table 5 the p-values between SSOE-FP-ELM and the other algorithms for each dataset are presented. Values marked with '*' represent a p-value ≤ 0.01, indicating that there is a significant difference between the two algorithms.

There are 9 significant differences between SSOE-FP-ELM and SOS-ELM, and only in the StatlogImageSegmentation dataset the proposed algorithm has lower accuracy. The reason for this advantage is the same as for SSOE-ELM, since the two algorithms have a very similar structure. The same occurs when comparing SSOE-FP-ELM and SVM, in which the algorithms have significant differences in the same 10 datasets, but only in the CrowdSourcedMapping dataset does SSOE-FP-ELM perform better.

Between SSOE-FP-ELM and FP-ELM, there are 10 significant differences, and only in the Musk dataset the proposed algorithm has lower accuracy.

Table 6 presents the training time of the algorithms for each dataset. It is important to note that the SSOE-ELM, SSOE-FP-ELM and SOS-ELM algorithms have partially parallel implementations, using GPU processing, while the FP-ELM and SVM have implementations entirely in CPU. In addition, for the comparison of training time, the

Table 5 – p-value of the Wilcoxon test in comparison between SSOE-FP-ELM and other algorithms in supervised experiment

Datasets	SSOE-FP-ELM SOS-ELM	SSOE-FP-ELM FP-ELM	SSOE-FP-ELM SVM	SSOE-FP-ELM SSOE-ELM
COIL20	0.9058	0.9058	0.8476	0.3147
COIL100	0.0000*	0.0054*	0.0000*	0.0011*
CrowdMap	0.0000*	0.0000*	0.0000*	0.6952
DNA	0.0000*	0.0000*	0.0000*	0.0000*
Gisette	0.0000*	0.0000*	0.0000*	0.0009*
Isolet	0.0000*	0.0002*	0.0000*	0.0555
KDEF_Front	0.0000*	0.0000*	0.0028*	0.8016
Musk	0.2772	0.0000*	0.0000*	0.0001*
Spam	0.1474	0.1646	0.0000*	0.4553
StatlogImgSeg	0.0004*	0.0005*	0.0000*	0.0679
StatlogLandSat	0.0000*	0.0000*	0.0000*	0.0003*
Waveform	0.0000*	0.0000*	0.1646	0.0963

same number of neurons was used in the ELM-based algorithms (1500 neurons), since this parameter has a great influence on the operations performed. Therefore, the comparison of training times is only a reference for the experiments carried out.

Table 6 – Algorithms training time (in seconds) in supervised experiment

Dataset	SSOE-ELM	SSOE-FP-ELM	SOS-ELM	FP-ELM	SVM
COIL20	1.23	2.67	1.10	1.85	2.48
COIL100	1.39	13.10	4.33	10.12	23.99
CrowdMap	2.19	5.51	7.75	19.26	3.07
DNA	0.48	1.75	1.78	3.56	1.16
Gisette	2.50	52.97	5.86	13.26	90.52
Isolet	1.55	12.10	5.11	11.75	23.85
KDEF_Front	0.19	1.53	0.81	1.12	0.59
Musk	1.08	3.91	3.73	7.97	1.07
Spam	0.64	1.82	2.60	5.35	0.52
StatlogImgSeg	0.39	0.82	1.50	2.74	0.08
StatlogLandSat	0.97	2.40	3.55	7.57	0.66
Waveform	0.75	1.68	2.82	5.88	0.33

4.5 Second experiment - semi-supervised comparison

In this second experiment, SSOE-ELM and SSOE-FP-ELM were evaluated in the same 12 public benchmarks, in a semi-supervised way, comparing the obtained results with SOS-ELM. For each dataset, the training partition was divided into labeled and unlabeled instances, starting with 10% of labeled samples and 90% unlabeled, and gradually increasing to 90% of labeled samples and 10% unlabeled. The training was then performed in an online manner using a chunk of the training partition each time. For each division,

the experiment was repeated 30 times, using the mean and standard deviation of the classification accuracy as performance metrics. An hyperparameter optimization with Random Search method was performed in all algorithms, using only 10% of labeled samples (worst case in this experiment). Table 7 presents these parameters.

Table 7 – Algorithm parameters for semi-supervised experiment

	SSOE-ELM				SSOE-FP-ELM				SOS-ELM			
Dataset	n	α	C	μ	n	α	C	μ	n	λ	C	μ
COIL20	2950	2^{13}	10^2	10	2100	2^{15}	10^3	5	200	0.81	10^5	15
COIL100	2250	2^{12}	10^5	5	2850	2^{18}	10^4	5	800	0.45	10^2	20
CrowdMap	1000	2^{15}	10^3	20	1000	2^{15}	10^3	20	3000	0.01	10^1	20
DNA	1350	2^{17}	10^4	5	2550	2^{16}	10^5	20	300	0.02	10^{-1}	5
Gisette	2300	2^{20}	10^3	5	2850	2^{18}	10^4	15	200	0.90	10^4	15
Isolet	2750	2^{17}	10^4	5	2250	2^{20}	10^4	15	350	0.73	10^2	5
KDEF_Front	1450	2^{20}	10^5	10	1250	2^{16}	10^4	15	350	0.10	10^{-4}	5
Spam	2100	2^{11}	10^5	5	1300	2^1	10^3	10	400	0.09	10^3	15
Musk	1150	2^{10}	10^5	15	1650	2^{14}	10^5	20	200	0.60	10^4	5
StatImgSeg	1450	2^1	10^4	5	2550	2^3	10^5	5	700	0.94	10^4	10
StatLandSat	2000	2^{-5}	10^3	10	1050	2^{-5}	10^2	5	1650	0.65	10^3	10
Waveform	1200	2^7	10^1	5	2800	2^{12}	10^5	10	2100	0.02	10^1	20

In Table 8, the mean accuracy of each algorithm is presented for three configurations of labeled training samples. Figure 6 shows the accuracy of the algorithms for all configurations of labeled samples in training partition.

SSOE-ELM and SSOE-FP-ELM presented greater mean accuracy than the SOS-ELM in 10 out of 12 datasets. According to plots in Figure 6, SSOE-ELM presents an inferior performance than SOS-ELM in Spam and Waveform datasets, while SSOE-FP-ELM presents an inferior performance than SOS-ELM in Spam and StatlogLandSat datasets. This behavior is similar to that presented in the fully supervised comparison.

Norm of output weights

Acording to Huang, Zhu & Siew (2006), ELM tends to reach both the smallest training error and the smallest norm of weights. Bartlett (1998) states that the smaller the norm of the weights, the better generalization ability the neural network tend to have. Han, Yao & Ling (2013) uses the norm of output weights (β matrix of ELM) as a measure of the generalization ability of the network to compare ELM models.

Table 8 – Algorithms accuracy (%) in semi-supervised experiment

Dataset	SSOE-ELM	SSOE-FP-ELM	SOS-ELM
10 % labeled			
COIL20	78.3 ± 2.6	**79.4 ± 3.7**	60.6 ± 2.6
COIL100	**66.1 ± 1.3**	63.8 ± 2.3	59.6 ± 1.6
CrowdMap	34.0 ± 3.0	**38.7 ± 4.5**	30.1 ± 4.6
DNA	**85.7 ± 1.1**	82.4 ± 1.8	58.2 ± 1.7
Gisette	90.4 ± 0.6	**93.2 ± 0.6**	80.9 ± 1.6
Isolet	**89.8 ± 0.7**	87.1 ± 1.4	76.1 ± 1.3
KDEF_Front	**49.7 ± 4.0**	49.5 ± 4.5	32.6 ± 3.5
Musk	91.9 ± 0.9	**92.4 ± 1.1**	86.0 ± 1.4
Spam	**89.8 ± 1.0**	89.0 ± 1.4	89.4 ± 0.9
StatlogImgSeg	**91.0 ± 2.1**	90.0 ± 1.9	**91.0 ± 1.5**
StatlogLandSat	83.6 ± 0.7	84.0 ± 0.5	**84.2 ± 0.7**
Waveform	83.3 ± 1.3	**84.6 ± 1.1**	84.3 ± 0.6
50 % labeled			
COIL20	95.5 ± 1.2	**96.0 ± 1.1**	91.5 ± 1.7
COIL100	**88.5 ± 0.8**	88.4 ± 0.8	86.6 ± 0.7
CrowdMap	**48.5 ± 1.7**	46.5 ± 3.0	38.3 ± 3.5
DNA	91.3 ± 0.6	**91.5 ± 0.6**	77.5 ± 0.9
Gisette	90.4 ± 0.5	**95.4 ± 0.5**	86.2 ± 1.1
Isolet	**94.4 ± 0.4**	93.4 ± 0.4	88.2 ± 0.6
KDEF_Front	64.5 ± 2.8	**64.8 ± 2.9**	57.7 ± 3.1
Musk	**96.6 ± 0.5**	95.2 ± 0.6	90.8 ± 1.0
Spam	91.5 ± 0.6	91.5 ± 0.7	**91.6 ± 0.8**
StatlogImgSeg	94.7 ± 0.9	**94.9 ± 0.9**	94.4 ± 0.7
StatlogLandSat	**87.0 ± 0.4**	86.2 ± 0.4	86.5 ± 0.3
Waveform	84.8 ± 0.9	**86.5 ± 0.7**	**86.5 ± 0.8**
90 % labeled			
COIL20	96.4 ± 0.9	**97.1 ± 0.9**	95.3 ± 1.0
COIL100	**95.6 ± 0.5**	93.1 ± 0.6	91.2 ± 0.7
CrowdMap	49.0 ± 1.4	**50.0 ± 1.6**	39.7 ± 2.3
DNA	92.0 ± 0.4	**92.6 ± 0.6**	81.8 ± 0.9
Gisette	90.3 ± 0.5	**95.8 ± 0.4**	86.6 ± 1.2
Isolet	**95.2 ± 0.2**	94.1 ± 0.3	89.5 ± 0.6
KDEF_Front	67.7 ± 2.3	**69.3 ± 2.7**	66.2 ± 2.1
Musk	**97.3 ± 0.3**	95.6 ± 0.5	91.0 ± 0.8
Spam	91.7 ± 0.7	**92.2 ± 0.7**	91.8 ± 0.7
StatlogImgSeg	95.0 ± 0.7	**95.8 ± 0.6**	94.8 ± 1.0
StatlogLandSat	**87.4 ± 0.2**	86.9 ± 0.2	86.9 ± 0.2
Waveform	85.6 ± 0.9	**87.0 ± 0.8**	86.6 ± 0.8

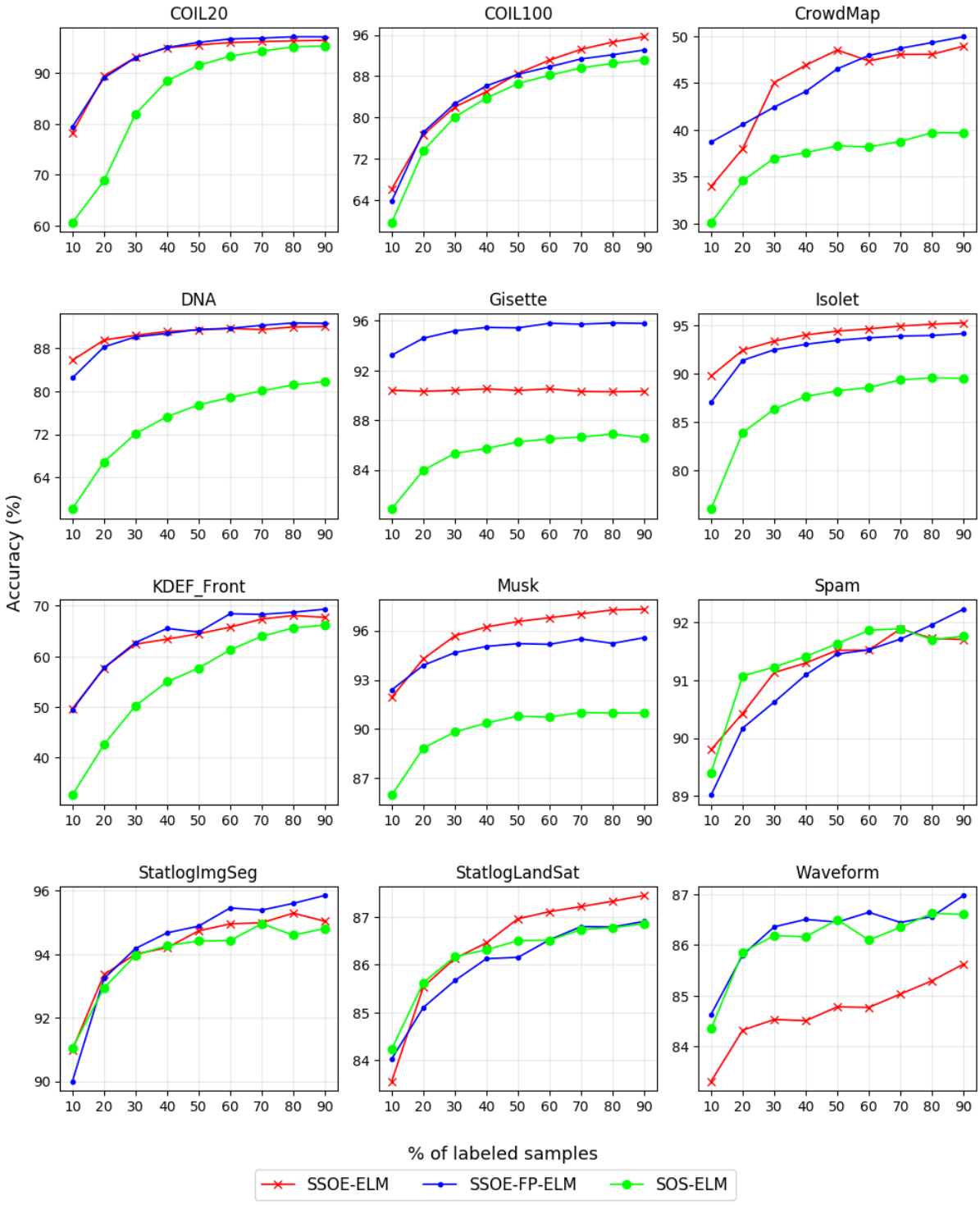

Figure 6 – Algorithms accuracy (%) in semi-supervised experiment

To compare the generalization ability of SSOE-ELM and SOS-ELM, we compute the norm of the output weights at the end of training phase in 30 executions of each algorithm for each dataset. Figure 7 shows the plots of the average norm of the output weights of each algorithm for all configurations of labeled/unlabeled samples.

The proposed algorithms presented the norm of output weights equal to or less

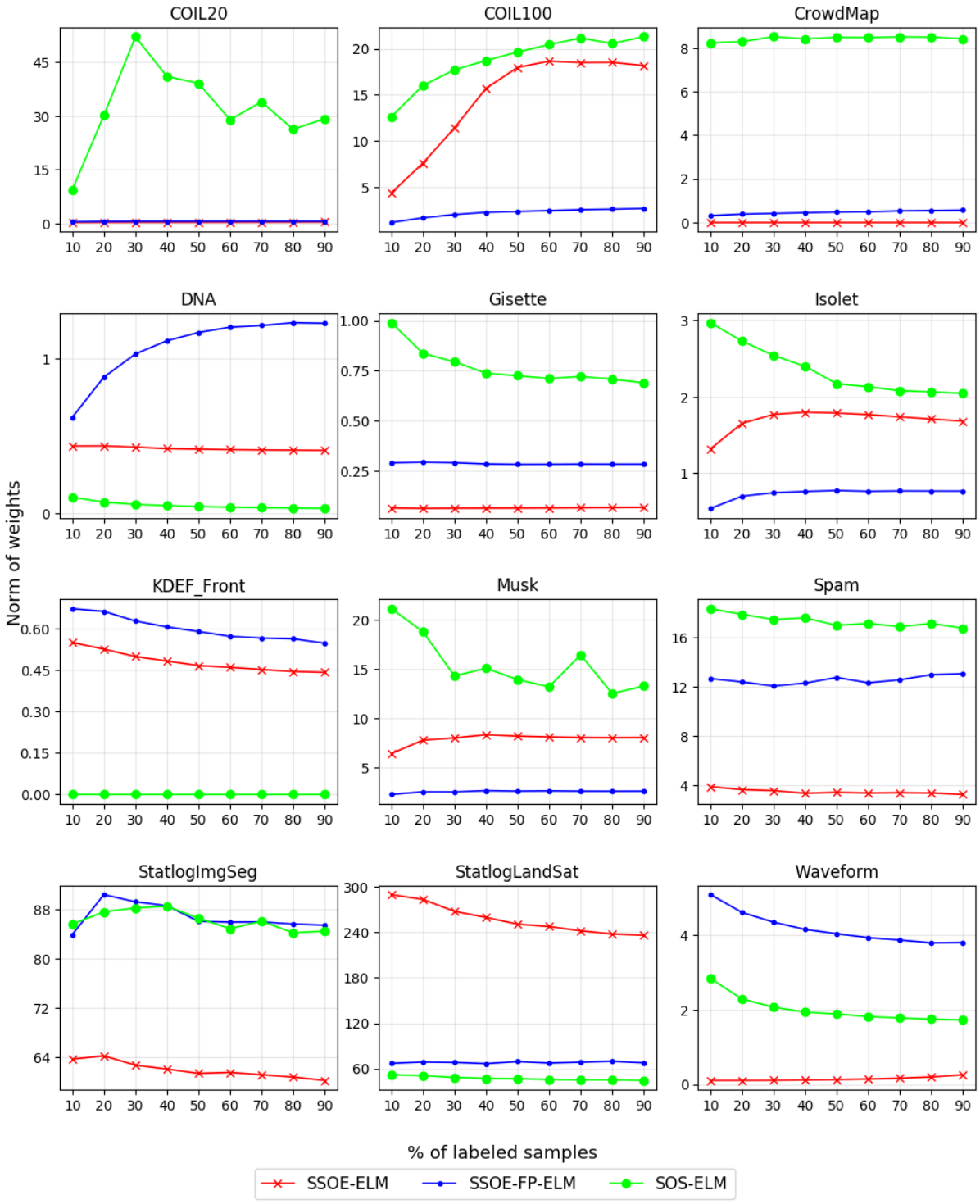

Figure 7 – Algorithms norm of output weights in supervised experiment

than the SOS-ELM in at least 8 of the 12 datasets. SSOE-ELM had a higher norm than SOS-ELM in the DNA, KDEF_Front and StatlogLandSat datasets. SSOE-FP-ELM presented a higher norm than SOS-ELM in the DNA, KDEF_Front, StatlogLandSat and Waveform datasets. This confirms the approach presented by Huang et al. (2012) that the regularization parameter in the ELM formalization leads to better generalization ability.

4.6 Third experiment - supervised comparison with concept drift

In this third experiment, SSOE-ELM and SSOE-FP-ELM were evaluated in public benchmarks with generated concept drift, in a fully supervised way (all training samples are labeled), comparing the obtained results with SOS-ELM and FP-ELM. It is worth mentioning that the SSOE-FP-ELM and FP-ELM algorithms are adapted to deal with concept drifts, while the SSOE-ELM and SOS-ELM algorithms are not.

Two different concept drifts were tested. In the first experiment, a Label Concept Drift (LCD) was generated in the middle of training dataset, changing the labels. In the second experiment, a Feature Concept Drift (FCD) was generated in the middle of training dataset, changing the inputs. For each type of concept drift, two tests were performed: generation of an abrupt concept drift (100% of the features or labels were changed), and generation of a gradual concept drift (30% of the features or labels were changed).

To measure the drift and update the forgetting parameter in SSOE-FP-ELM algorithm, the labeled set of each training partition was first used as a test partition to get the model accuracy, and then used as a training partition.

The same parameters of the first experiment (fully supervised classification without concept drift) were used for all algorithms. For each dataset, the experiment was repeated 30 times, using the mean and standard deviation of the classification accuracy as performance metrics.

In the learning evolution plots, COIL20 and KDEF_Front datasets were removed because they had few training partitions, which made the plot uninformative.

Abrupt Concept Drift

To check the behavior of the proposed algorithms in the face of an abrupt concept drift, FCD and LCD with 100% changes were generated. In the FCD, all the features were shuffled, and in the LCD, all the labels were switched.

Table 9 presents the mean accuracy of the algorithms for each dataset with abrupt FCD. In this experiment, SSOE-ELM presents greater accuracy than SOS-ELM in 7 out of 12 datasets, and outperformed FP-ELM in 8 out of 12 datasets. The SSOE-FP-ELM algorithm had better results than SOS-ELM in 10 out of 12 datasets, and was better than FP-ELM in 11 out of 12 datasets.

Figure 8 shows the learning evolution of the algorithms during the training process, in situations where an abrupt FCD occurs. For each training partition that arrives, the models were trained and the accuracy was measured using the test partition of the dataset. The drop in accuracy of all models is noticeable when the concept drift occurs, but with the exception of the CrowSourcedMapping and DNA datasets, all algorithms showed a good recovery after the concept drift. This occurs because in FCD, there is a change in the

Table 9 – Algorithms accuracy (%) in supervised experiment with Abrupt FCD

Dataset	SSOE-ELM	SSOE-FP-ELM	SOS-ELM	FP-ELM
COIL20	96.0 ± 1.4	96.9 ± 1.2	96.7 ± 1.0	**97.4 ± 1.3**
COIL100	88.3 ± 1.0	**90.6 ± 0.7**	87.6 ± 0.8	88.5 ± 0.9
CrowdMap	45.7 ± 3.2	**49.7 ± 1.9**	32.6 ± 3.4	28.5 ± 2.9
DNA	85.0 ± 1.4	**85.7 ± 1.0**	76.4 ± 1.8	78.6 ± 1.6
Gisette	93.8 ± 0.6	**94.7 ± 0.6**	90.6 ± 0.8	90.9 ± 0.6
Isolet	93.7 ± 0.4	**94.1 ± 0.5**	92.1 ± 0.5	92.4 ± 0.5
KDEF_Front	68.5 ± 2.4	**69.9 ± 2.6**	61.3 ± 3.2	60.1 ± 2.9
Musk	97.1 ± 0.4	**97.3 ± 0.4**	95.3 ± 0.6	96.3 ± 0.4
Spam	89.5 ± 1.1	**90.8 ± 0.8**	90.5 ± 1.2	89.8 ± 0.9
StatlogImgSeg	94.3 ± 0.8	94.9 ± 1.0	**95.1 ± 0.8**	93.9 ± 0.8
StatlogLandSat	86.6 ± 0.4	**87.1 ± 0.6**	86.9 ± 0.3	84.7 ± 0.3
Waveform	82.9 ± 1.3	83.6 ± 1.1	84.1 ± 0.9	**85.6 ± 0.8**

values of the inputs of the dataset, which are multiplied by the input weights of the neural network and later changed by an activation function. This process softens the effects of the concept drift.

Table 10 – Algorithms accuracy (%) in supervised experiment with Abrupt LCD

Dataset	SSOE-ELM	SSOE-FP-ELM	SOS-ELM	FP-ELM
COIL20	24.4 ± 3.5	49.1 ± 4.1	32.7 ± 4.4	**97.3 ± 1.0**
COIL100	30.6 ± 1.7	**79.0 ± 3.4**	29.6 ± 1.8	45.5 ± 1.3
CrowdMap	30.1 ± 14.8	**49.7 ± 14.5**	23.9 ± 11.1	27.0 ± 8.3
DNA	68.0 ± 19.7	**84.4 ± 8.0**	62.7 ± 16.0	75.6 ± 12.2
Gisette	71.3 ± 16.7	**94.0 ± 3.9**	61.6 ± 18.2	83.1 ± 9.7
Isolet	19.4 ± 6.1	**89.1 ± 4.4**	26.3 ± 3.3	45.1 ± 3.0
KDEF_Front	34.1 ± 9.7	**42.8 ± 11.0**	30.4 ± 12.1	41.2 ± 7.5
Musk	85.0 ± 19.0	**94.1 ± 5.5**	79.3 ± 21.0	89.6 ± 16.2
Spam	80.6 ± 10.6	86.8 ± 6.5	78.9 ± 11.8	**89.9 ± 5.2**
StatlogImgSeg	39.9 ± 10.9	**80.7 ± 10.9**	38.8 ± 14.7	63.6 ± 10.6
StatlogLandSat	41.1 ± 17.4	78.4 ± 8.3	31.0 ± 15.4	**88.3 ± 4.1**
Waveform	52.6 ± 24.2	77.8 ± 5.3	54.1 ± 19.2	**91.8 ± 4.8**

Table 10 presents the mean accuracy of the algorithms for each dataset with abrupt LCD. In this experiment, the effect of the concept drift on the classification model is more visible. In LCD, the concept drift is applied to the labels of the training partition, so there is not the same smoothing effect that occurs on the FCD. After the concept drift, the algorithms that do not have a forgetting mechanism (SSOE-ELM and SOS-ELM) are unable to completelly recover their accuracy. In contrast, the algorithms that have the forgetting factor (SSOE-FP-ELM and FP-ELM) are able to recover quickly from the drop in accuracy.

In datasets with abrupt LCD, SSOE-ELM presents greater accuracy than SOS-ELM in 9 out of 12 datasets, but it presents a better result than FP-ELM only in the

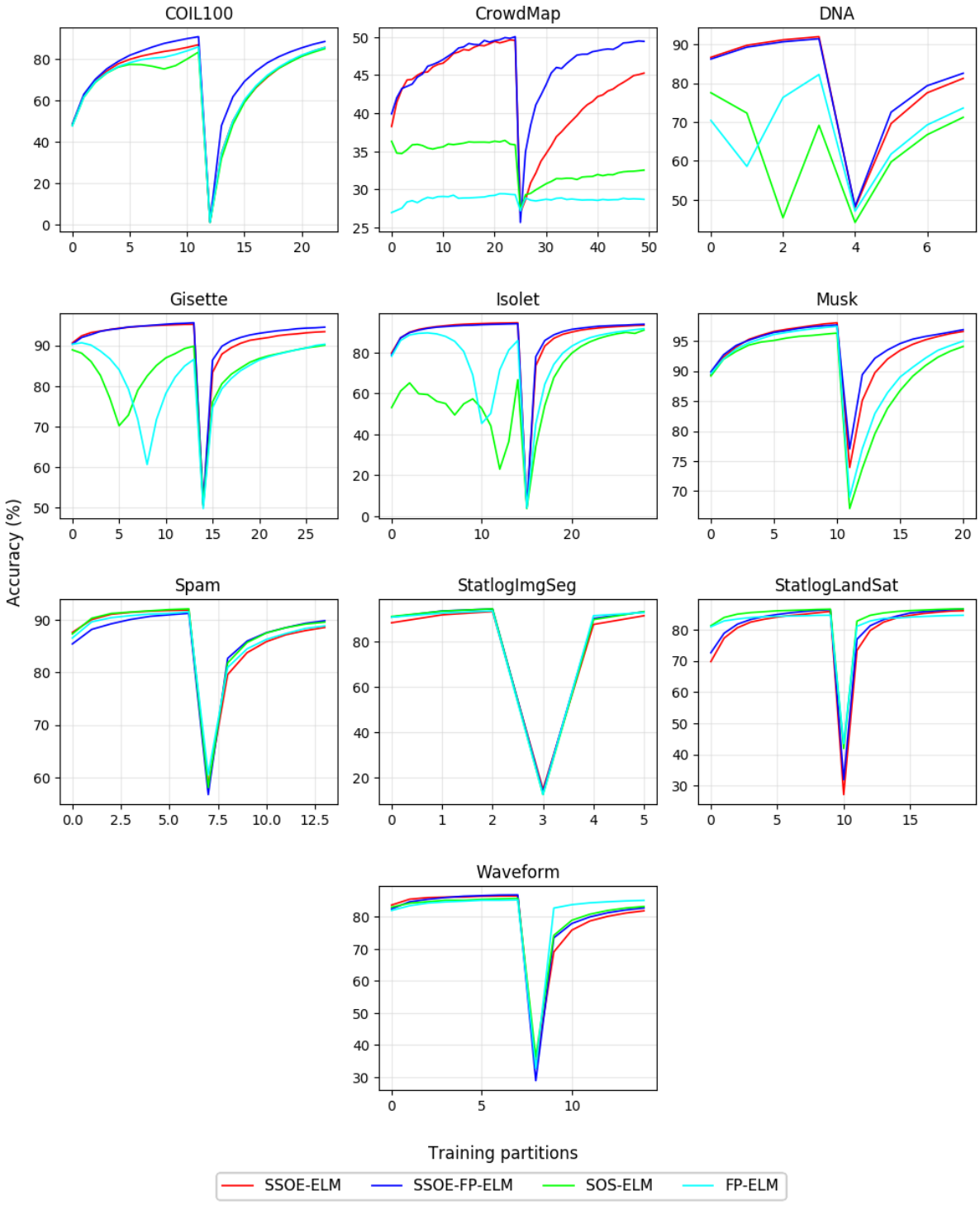

Figure 8 – Learning evolution of algorithms in supervised experiment with Abrupt FCD

CrowdSourcedMapping dataset. The SSOE-FP-ELM algorithm had better results than SOS-ELM in all datasets, and was better than FP-ELM in 8 out of 12 datasets.

Figure 9 shows the learning evolution of the algorithms during the training process, in situations where an abrupt LCD occurs.

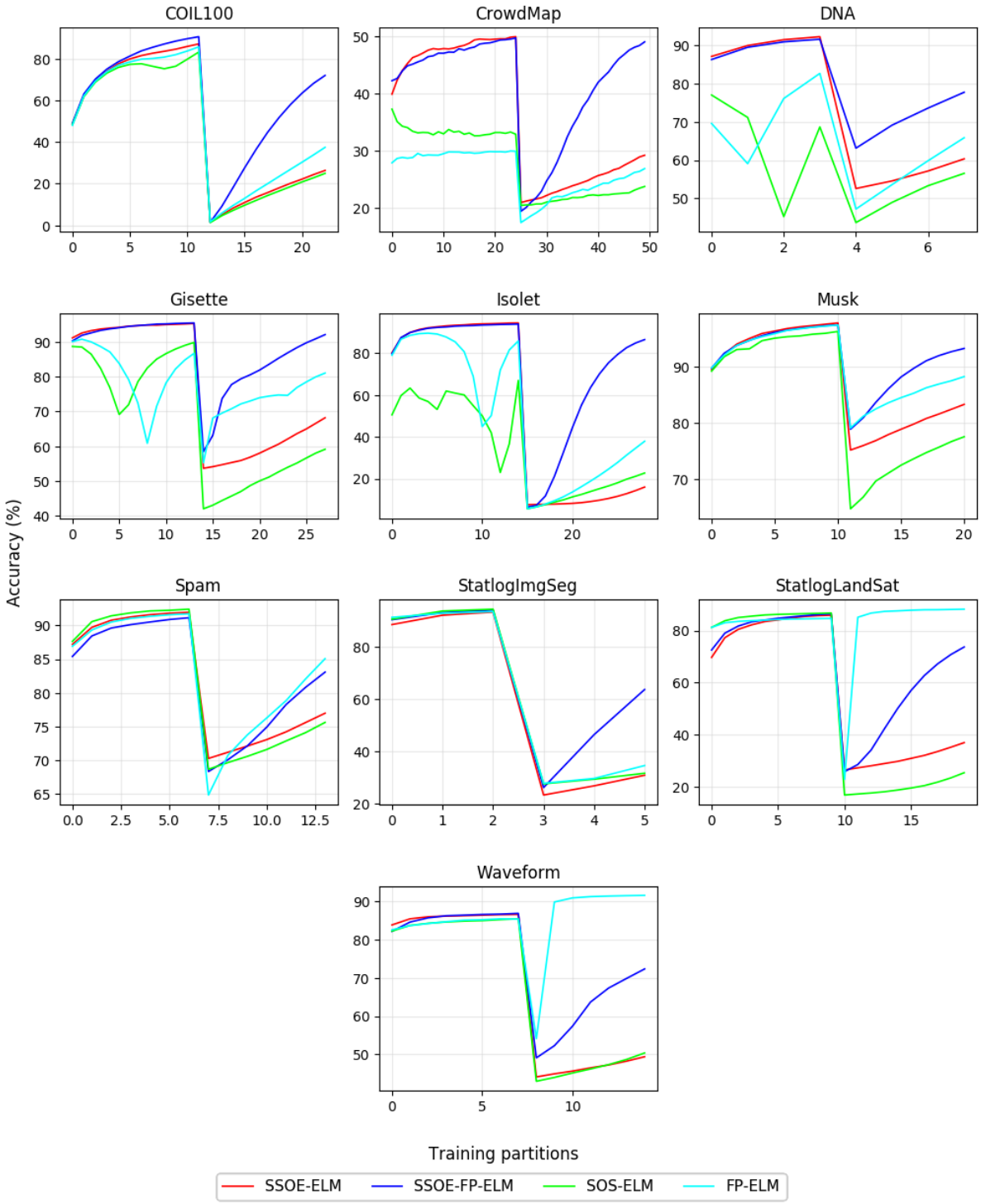

Figure 9 – Learning evolution of algorithms in supervised experiment with Abrupt LCD

Gradual Concept Drift

To check the behavior of the proposed algorithms in the face of a gradual concept drift, FCD and LCD with 30% changes were generated.

Table 11 presents the mean accuracy of the algorithms for each dataset with gradual FCD. Figure 10 shows the learning evolution of the algorithms during the training process, in situations where an gradual FCD occurs. For each training partition that arrives, the models were trained and the accuracy was measured using the test partition of the dataset.

Table 11 – Algorithms accuracy (%) in supervised experiment with Gradual FCD

Dataset	SSOE-ELM	SSOE-FP-ELM	SOS-ELM	FP-ELM
COIL20	98.2 ± 0.6	98.7 ± 0.7	98.8 ± 0.5	**99.2 ± 0.4**
COIL100	91.9 ± 0.5	**92.4 ± 0.5**	91.2 ± 0.8	91.9 ± 0.8
CrowdMap	51.1 ± 1.7	**51.9 ± 2.7**	32.9 ± 2.9	30.1 ± 2.9
DNA	**91.6 ± 0.5**	91.0 ± 1.0	85.2 ± 1.3	86.8 ± 1.5
Gisette	95.3 ± 0.5	**95.9 ± 0.5**	92.9 ± 0.7	92.7 ± 0.7
Isolet	**94.3 ± 0.5**	94.1 ± 0.3	92.7 ± 0.4	93.3 ± 0.5
KDEF_Front	75.3 ± 2.7	**75.5 ± 2.6**	67.9 ± 2.8	64.7 ± 2.9
Musk	**97.9 ± 0.3**	97.8 ± 0.3	96.4 ± 0.5	97.3 ± 0.3
Spam	91.6 ± 0.8	92.0 ± 0.6	**92.2 ± 0.7**	92.0 ± 0.8
StatlogImgSeg	95.0 ± 1.0	**95.6 ± 0.7**	95.1 ± 2.4	94.9 ± 0.9
StatlogLandSat	86.8 ± 0.3	**87.5 ± 0.5**	87.1 ± 0.2	84.8 ± 0.3
Waveform	85.7 ± 1.1	**86.1 ± 1.1**	85.5 ± 1.2	85.2 ± 1.1

Table 12 – Algorithms accuracy (%) in supervised experiment with Gradual LCD

Dataset	SSOE-ELM	SSOE-FP-ELM	SOS-ELM	FP-ELM
COIL20	68.6 ± 13.3	**82.4 ± 5.0**	82.1 ± 8.2	82.0 ± 8.2
COIL100	77.3 ± 1.6	79.7 ± 1.3	76.4 ± 1.8	**81.5 ± 0.9**
CrowdMap	**52.0 ± 1.5**	51.0 ± 1.7	35.4 ± 3.6	29.6 ± 2.9
DNA	**93.6 ± 0.4**	93.0 ± 0.3	88.2 ± 0.6	90.2 ± 0.7
Gisette	96.1 ± 0.5	**96.3 ± 0.5**	93.7 ± 0.7	94.0 ± 0.7
Isolet	79.1 ± 3.8	81.5 ± 3.1	78.5 ± 2.5	**83.7 ± 2.9**
KDEF_Front	74.5 ± 4.9	**74.9 ± 4.0**	73.1 ± 4.7	67.8 ± 4.2
Musk	**99.1 ± 0.2**	98.8 ± 0.2	98.7 ± 0.3	**99.1 ± 0.3**
Spam	92.4 ± 0.5	92.6 ± 0.5	**92.9 ± 0.4**	92.6 ± 0.6
StatlogImgSeg	90.1 ± 5.9	90.3 ± 5.7	87.5 ± 4.9	**92.1 ± 4.0**
StatlogLandSat	81.3 ± 6.3	83.7 ± 5.8	83.7 ± 6.0	**84.1 ± 2.5**
Waveform	86.6 ± 0.8	**87.2 ± 0.6**	86.2 ± 0.6	86.1 ± 0.8

In the plots of learning evolution in situations of gradual FCD, it is possible to observe an effect similar to that observed in the graphs of abrupt FCD. The algorithms suffer a loss in accuracy, but in most cases they manage to recover. This is due to the concept drift smoothing effect caused by the nonlinear activation function of neural

networks. The difference observed between abrupt and gradual FCD is mainly in the percentage of lost accuracy, which, as expected, is smaller in the gradual concept drift.

Table 12 presents the mean accuracy of the algorithms for each dataset with gradual LCD. Figure 11 shows the learning evolution of the algorithms during the training process, in situations where a gradual LCD occurs.

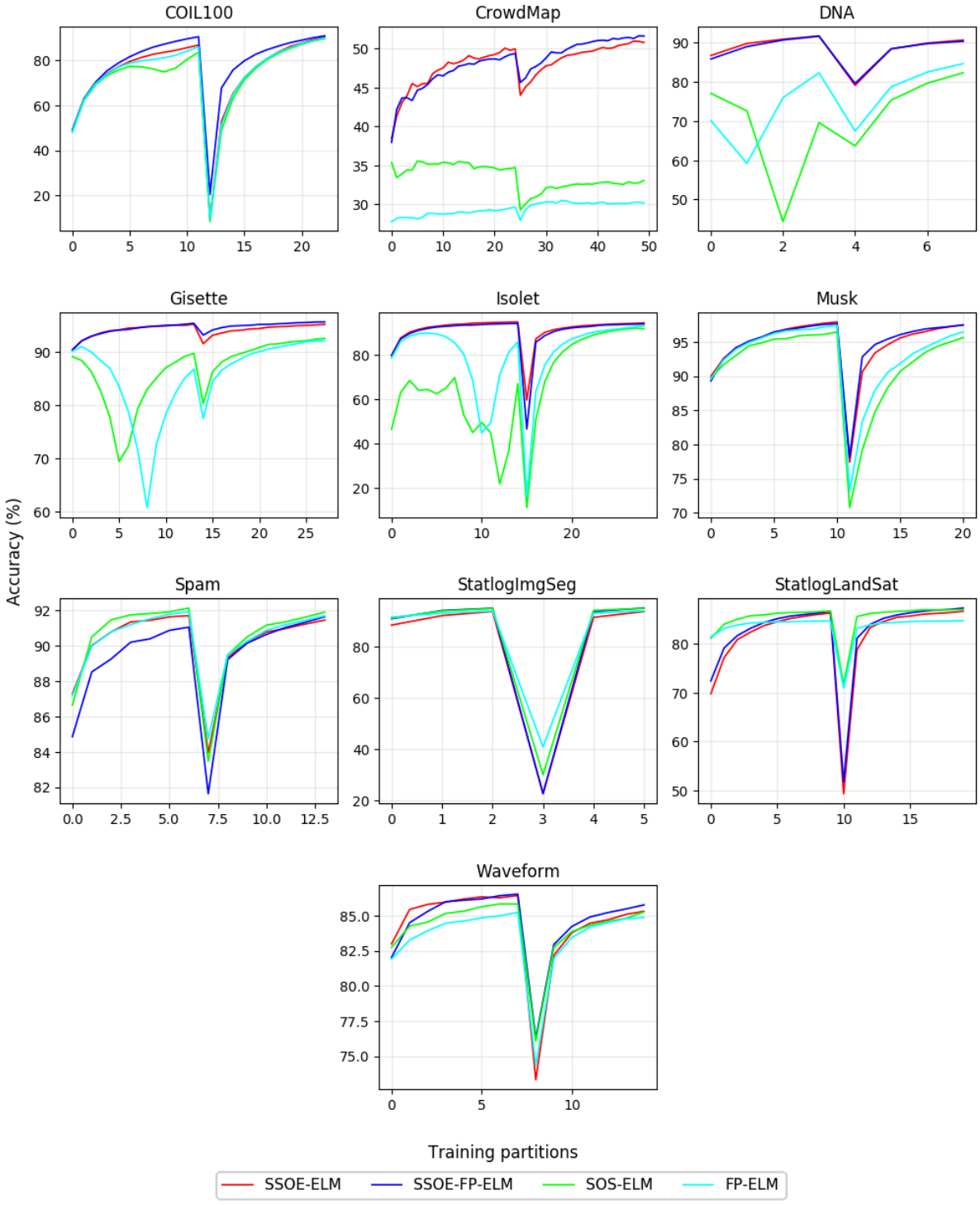

Figure 10 – Learning evolution of algorithms in supervised experiment with Gradual FCD

In this experiment, a curious effect can be seen on the gradual LCD. In the learning evolution plots, it is possible to perceive for some datasets that there was no impact on the accuracy of the model when the gradual concept drift occurred. This is due to the fact that the dataset has few classes (for example, DNA dataset with 3 classes and Gisette dataset with 2 classes), so that the change caused by the gradual LCD becomes barely noticeable to the classification models.

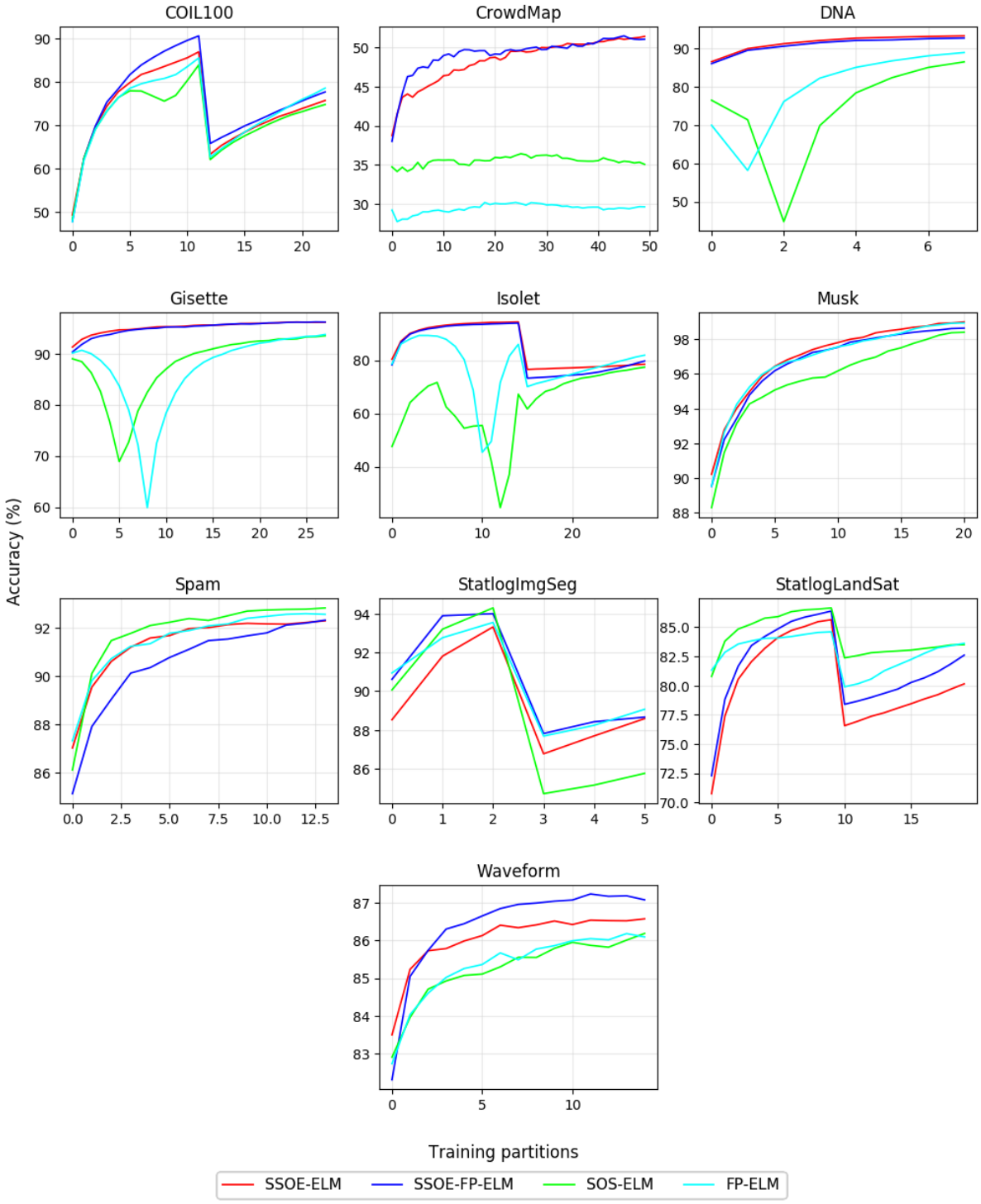

Figure 11 – Learning evolution of algorithms in supervised experiment with Gradual LCD

4.7 Forth experiment - semi-supervised comparison with concept drift

In this forth experiment, SSOE-ELM and SSOE-FP-ELM were evaluated in the same 12 public benchmarks, in a semi-supervised way, in the presence of concept drift, comparing the obtained results with SOS-ELM. For each dataset, the training partition was divided into labeled and unlabeled instances, starting with 10% of labeled samples and 90% unlabeled, and gradually increasing to 90% of labeled samples and 10% unlabeled. The training was then performed in an online manner using a chunk of the training partition each time. For each division, the experiment was repeated 30 times, using the mean and standard deviation of the classification accuracy as performance metrics.

As in the previous experiment, two different concept drifts were tested: Label Concept Drift (LCD) and Feature Concept Drift (FCD). For each type of concept drift, two tests were performed: generation of an abrupt concept drift (100% of the features or labels was changed), and generation of a gradual concept drift (30% of the features or labels was changed).

To measure the drift and update the forgetting parameter in SSOE-FP-ELM algorithm, the labeled set of each training partition was first used as a test partition to get the model accuracy, and then used as a training partition.

The same parameters of second experiment (semi-supervised classification without concept drift) were used for all algorithms. For each dataset, the experiment was repeated 30 times, using the mean and standard deviation of the classification accuracy as performance metrics.

Abrupt Concept Drift

Table 13 presents the mean accuracy of each algorithm for three configurations of labeled training samples in datasets with occurrence of FCD. Figure 12 shows the accuracy of the algorithms for all configurations of labeled samples in training partition.

Table 14 presents the mean accuracy of each algorithm for three configurations of labeled training samples in datasets with occurrence of LCD. Figure 13 shows the accuracy of the algorithms for all configurations of labeled samples in training partition.

From the plots presented, it is possible to observe that the SSOE-FP-ELM algorithm presents results equal to or superior to SOS-ELM in all data sets with LCD occurrence, and in all except the StatlogLandSat dataset when FCD occurs. SSOE-ELM shows equal or better results than SOS-ELM in 9 out of 12 datasets with FCD. In addition, it is possible to observe that the SSOE-FP-ELM algorithm has much higher accuracy than SOS-ELM in all datasets with LCD occurrence. On the other hand, SSOE-ELM presents results very close to SOS-ELM in LCD cases, emphasizing that neither algorithm offers

concept drift treatment mechanisms.

Table 13 – Algorithms accuracy (%) in semi-supervised experiment with Abrupt FCD

Dataset	SSOE-ELM	SSOE-FP-ELM	SOS-ELM
10 % labeled			
COIL20	**61.8 ± 4.8**	61.5 ± 4.6	44.6 ± 4.9
COIL100	**51.7 ± 1.6**	51.3 ± 2.1	47.2 ± 1.8
CrowdMap	31.1 ± 3.6	**36.5 ± 5.3**	30.6 ± 4.4
DNA	**71.7 ± 2.6**	69.2 ± 3.2	51.3 ± 2.2
Gisette	88.2 ± 1.0	**91.3 ± 0.7**	75.2 ± 1.8
Isolet	**82.6 ± 1.2**	80.9 ± 1.7	64.0 ± 1.8
KDEF_Front	36.0 ± 4.5	**37.2 ± 3.9**	25.7 ± 3.5
Musk	87.2 ± 2.1	**89.4 ± 1.6**	81.4 ± 1.8
Spam	83.9 ± 1.8	**85.9 ± 1.5**	84.6 ± 2.1
StatlogImgSeg	87.0 ± 1.9	84.4 ± 2.9	**87.6 ± 1.8**
StatlogLandSat	82.2 ± 0.9	82.9 ± 0.9	**83.2 ± 0.9**
Waveform	76.4 ± 2.8	**82.1 ± 1.4**	79.5 ± 1.7
50 % labeled			
COIL20	88.7 ± 2.4	**88.8 ± 2.2**	77.5 ± 2.6
COIL100	74.1 ± 1.1	**79.7 ± 1.3**	75.0 ± 0.8
CrowdMap	41.9 ± 4.8	**43.9 ± 2.5**	37.2 ± 4.0
DNA	**81.5 ± 1.4**	**81.5 ± 1.3**	67.6 ± 1.8
Gisette	89.2 ± 0.9	**93.9 ± 0.5**	80.7 ± 1.8
Isolet	**92.0 ± 0.5**	91.9 ± 0.5	82.6 ± 0.8
KDEF_Front	53.1 ± 3.1	**57.0 ± 2.9**	44.0 ± 3.5
Musk	93.4 ± 0.9	**93.6 ± 0.7**	86.0 ± 1.5
Spam	87.0 ± 1.3	**89.0 ± 1.1**	87.6 ± 0.9
StatlogImgSeg	**93.6 ± 0.9**	**93.6 ± 1.1**	93.1 ± 0.9
StatlogLandSat	**86.1 ± 0.4**	85.8 ± 0.4	**86.1 ± 0.5**
Waveform	77.1 ± 2.8	**84.3 ± 1.1**	81.8 ± 1.8
90 % labeled			
COIL20	92.2 ± 1.6	**92.7 ± 1.7**	85.4 ± 2.2
COIL100	**87.0 ± 0.8**	86.4 ± 0.9	81.0 ± 0.7
CrowdMap	41.0 ± 3.8	**48.4 ± 2.8**	38.8 ± 4.1
DNA	83.0 ± 1.5	**84.5 ± 1.5**	72.4 ± 1.8
Gisette	88.8 ± 0.7	**94.5 ± 0.5**	81.0 ± 1.5
Isolet	**93.4 ± 0.5**	92.9 ± 0.5	84.9 ± 0.8
KDEF_Front	57.5 ± 2.6	**59.5 ± 3.0**	50.2 ± 4.4
Musk	**94.7 ± 0.5**	93.9 ± 0.7	86.3 ± 1.2
Spam	87.4 ± 1.4	**89.7 ± 1.0**	88.1 ± 1.2
StatlogImgSeg	94.3 ± 0.8	**94.4 ± 1.0**	93.9 ± 0.9
StatlogLandSat	**87.0 ± 0.3**	86.5 ± 0.3	86.4 ± 0.4
Waveform	77.6 ± 2.6	**85.1 ± 0.7**	82.3 ± 1.3

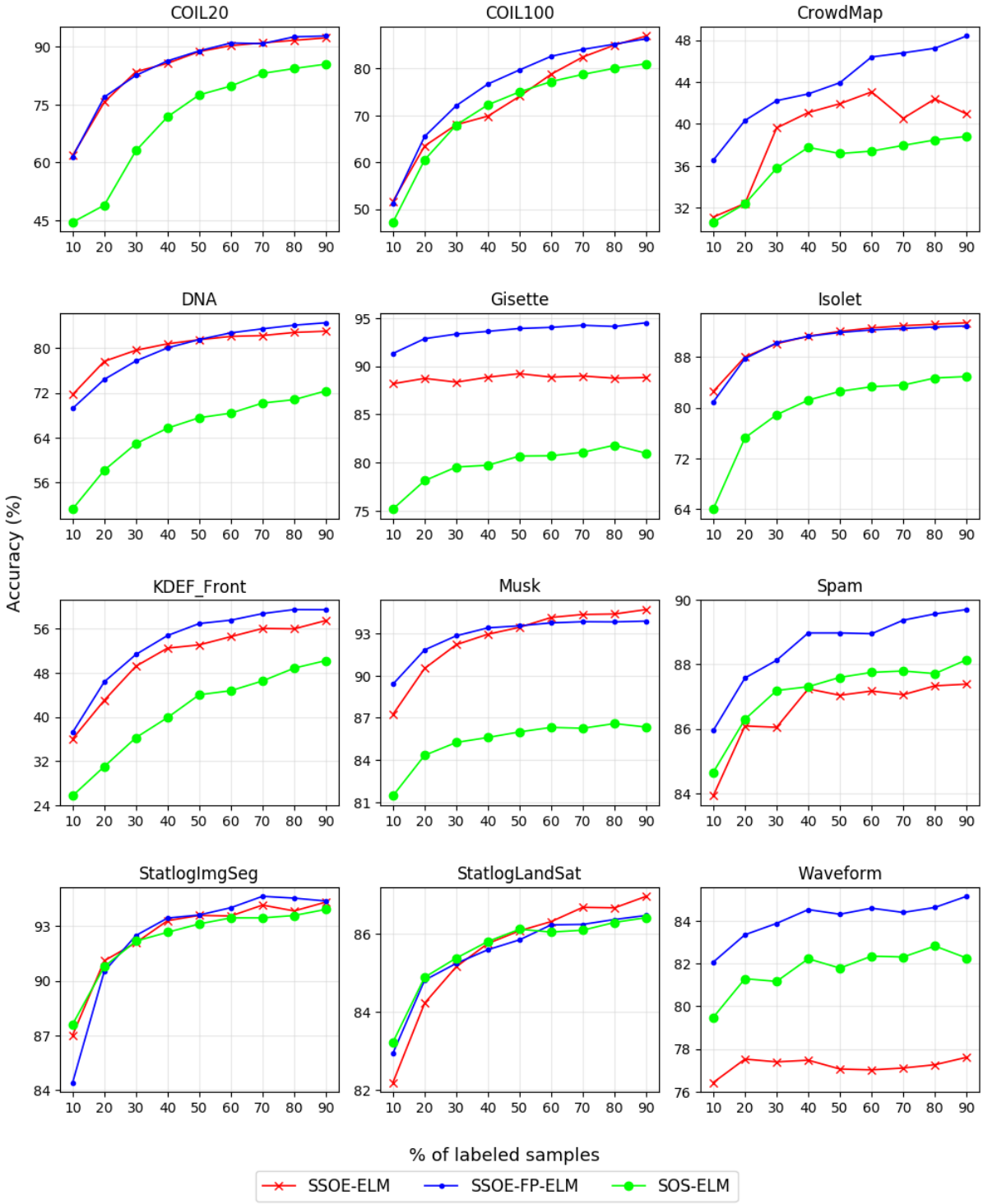

Figure 12 – Algorithms accuracy (%) in semi-supervised experiment with Abrupt FCD

Table 14 – Algorithms accuracy (%) in semi-supervised experiment with Abrupt LCD

Dataset	SSOE-ELM	SSOE-FP-ELM	SOS-ELM
10 % labeled			
COIL20	28.1 ± 5.7	**38.8 ± 7.1**	15.2 ± 2.5
COIL100	27.5 ± 1.5	**46.7 ± 2.8**	27.0 ± 1.3
CrowdMap	22.0 ± 13.8	**49.0 ± 11.7**	27.3 ± 10.8
DNA	**73.3 ± 13.4**	69.3 ± 12.6	48.6 ± 8.3
Gisette	70.1 ± 30.3	**92.4 ± 4.6**	71.9 ± 10.2
Isolet	40.0 ± 2.9	**75.2 ± 2.9**	27.3 ± 2.8
KDEF_Front	**30.5 ± 10.7**	**30.5 ± 7.8**	20.5 ± 4.5
Musk	81.0 ± 15.9	**92.7 ± 2.6**	80.7 ± 11.6
Spam	81.2 ± 8.1	**90.6 ± 3.6**	83.8 ± 6.6
StatlogImgSeg	48.7 ± 13.2	**60.0 ± 7.3**	46.3 ± 12.4
StatlogLandSat	43.8 ± 12.7	**82.1 ± 7.4**	40.7 ± 17.1
Waveform	66.4 ± 19.4	**79.5 ± 6.3**	58.5 ± 19.4
50 % labeled			
COIL20	16.0 ± 7.0	**67.5 ± 9.8**	23.1 ± 4.0
COIL100	35.2 ± 1.8	**72.7 ± 2.2**	26.0 ± 2.3
CrowdMap	28.3 ± 13.1	**52.7 ± 12.3**	22.9 ± 14.0
DNA	70.4 ± 18.8	**78.9 ± 7.0**	64.0 ± 11.8
Gisette	70.0 ± 31.9	**94.9 ± 2.7**	65.0 ± 20.8
Isolet	24.5 ± 3.9	**83.1 ± 3.0**	17.8 ± 4.5
KDEF_Front	24.2 ± 8.9	**35.1 ± 10.7**	25.2 ± 7.1
Musk	81.6 ± 18.9	**94.3 ± 4.6**	85.9 ± 9.9
Spam	83.5 ± 8.9	**91.3 ± 4.2**	83.1 ± 8.8
StatlogImgSeg	36.0 ± 14.6	**78.8 ± 11.6**	35.8 ± 13.1
StatlogLandSat	35.6 ± 14.4	**82.6 ± 8.8**	33.3 ± 16.2
Waveform	65.8 ± 18.1	**79.7 ± 8.5**	64.8 ± 19.0
90 % labeled			
COIL20	12.5 ± 6.3	**82.9 ± 5.2**	28.9 ± 4.5
COIL100	31.9 ± 1.8	**80.3 ± 2.1**	19.0 ± 1.8
CrowdMap	27.4 ± 9.6	**47.7 ± 9.2**	22.7 ± 12.2
DNA	66.5 ± 17.0	**82.3 ± 5.0**	61.0 ± 17.2
Gisette	65.0 ± 35.6	**94.5 ± 2.3**	67.0 ± 20.3
Isolet	17.0 ± 6.0	**84.5 ± 2.5**	15.1 ± 3.8
KDEF_Front	26.8 ± 12.7	**36.0 ± 8.7**	25.1 ± 9.1
Musk	79.4 ± 19.5	**94.4 ± 4.5**	82.6 ± 12.3
Spam	81.1 ± 8.7	**89.0 ± 5.0**	81.4 ± 7.8
StatlogImgSeg	31.6 ± 15.0	**81.2 ± 11.8**	34.7 ± 18.6
StatlogLandSat	33.3 ± 19.4	**81.2 ± 9.9**	39.5 ± 18.7
Waveform	65.0 ± 20.8	**76.2 ± 6.7**	55.1 ± 19.9

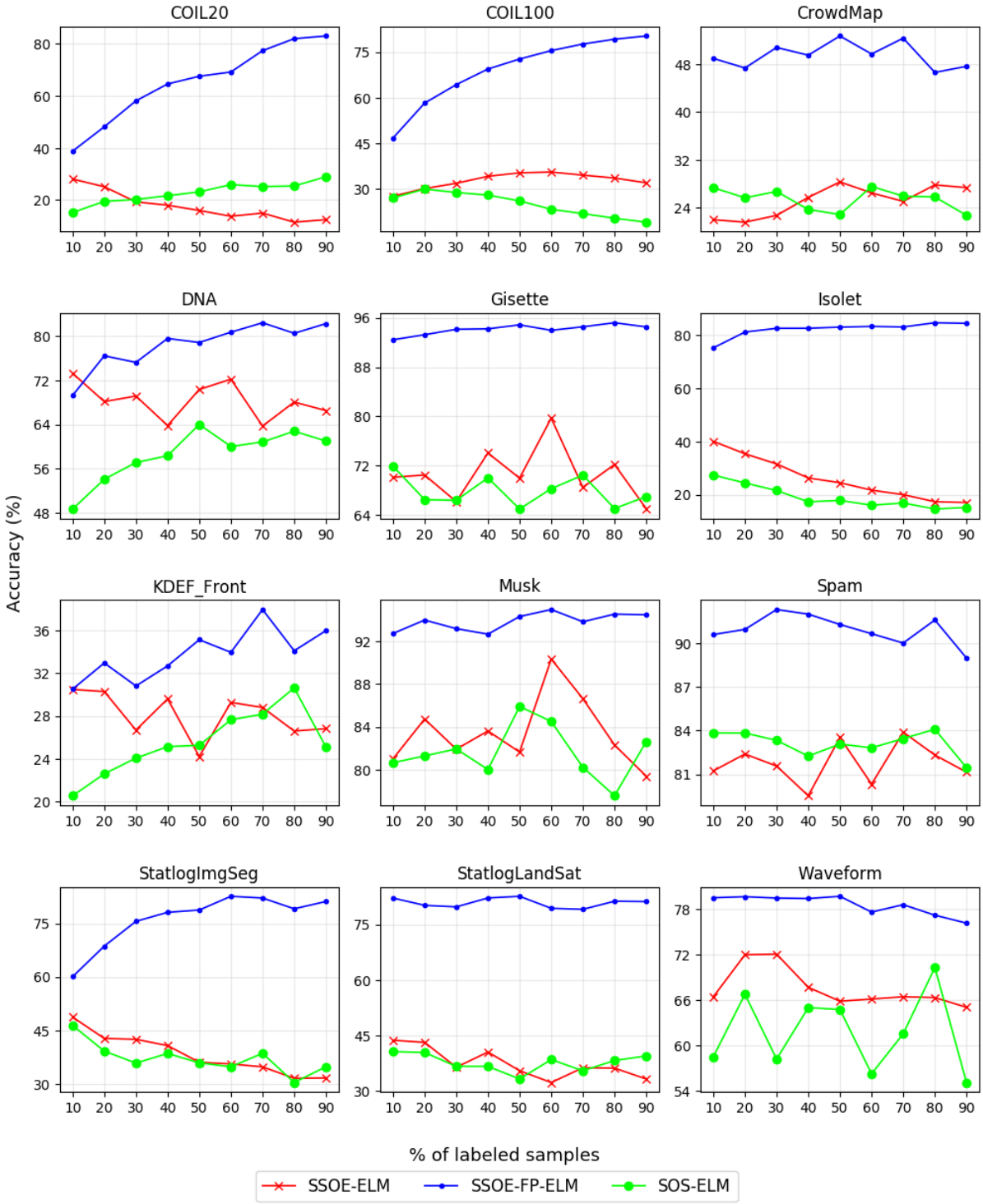

Figure 13 – Algorithms accuracy (%) in semi-supervised experiment with Abrupt LCD

Gradual Concept Drift

Table 15 presents the mean accuracy of each algorithm for three configurations of labeled training samples in datasets with occurrence of FCD. Figure 14 shows the accuracy of the algorithms for all configurations of labeled samples in training partition.

Table 16 presents the mean accuracy of each algorithm for three configurations of labeled training samples in datasets with occurrence of LCD. Figure 15 shows the accuracy of the algorithms for all configurations of labeled samples in training partition.

In the plots presented, it is possible to identify a similar behavior of the algorithms between abrupt and gradual FCD. The results are very close, with a reduction in the difference in accuracy between the algorithms in the gradual concept drift.

With the occurrence of gradual LCD, the results change in relation to the abrupt LCD. SSOE-FP-ELM presents better results than SOS-ELM in 11 out of 12 datasets, losing in the Spam dataset. The SSOE-ELM, on the other hand, has improved results compared to the abrupt LCD, having better accuracy than the SOS-ELM in 8 out of 12 datasets.

Table 15 – Algorithms accuracy (%) in semi-supervised experiment with Gradual FCD

Dataset	SSOE-ELM	SSOE-FP-ELM	SOS-ELM
10 % labeled			
COIL20	**70.5 ± 3.9**	69.2 ± 5.6	52.2 ± 3.8
COIL100	**56.9 ± 1.4**	55.5 ± 2.5	52.0 ± 1.5
CrowdMap	33.2 ± 2.1	**39.0 ± 3.8**	32.0 ± 4.7
DNA	**83.0 ± 1.8**	79.2 ± 2.6	56.7 ± 2.6
Gisette	89.5 ± 0.7	**92.8 ± 0.6**	78.6 ± 1.0
Isolet	**86.6 ± 0.7**	83.4 ± 1.5	69.3 ± 1.3
KDEF_Front	45.1 ± 3.3	**46.4 ± 2.8**	28.2 ± 2.4
Musk	88.9 ± 1.5	**90.6 ± 2.2**	83.5 ± 1.0
Spam	87.9 ± 1.0	87.7 ± 1.2	**88.7 ± 1.6**
StatlogImgSeg	88.3 ± 2.5	87.2 ± 2.6	**90.5 ± 1.7**
StatlogLandSat	83.3 ± 1.2	83.4 ± 1.0	**84.1 ± 0.9**
Waveform	82.2 ± 1.2	**83.9 ± 1.0**	83.4 ± 1.0
50 % labeled			
COIL20	93.1 ± 1.4	**93.8 ± 2.1**	83.7 ± 1.7
COIL100	79.4 ± 1.3	**82.6 ± 0.8**	80.4 ± 0.7
CrowdMap	**47.1 ± 2.2**	44.9 ± 2.1	36.1 ± 4.7
DNA	**89.0 ± 1.8**	88.5 ± 1.1	75.4 ± 2.0
Gisette	89.8 ± 0.5	**95.2 ± 0.5**	84.9 ± 1.5
Isolet	**93.2 ± 0.3**	92.5 ± 0.4	85.4 ± 0.5
KDEF_Front	**63.5 ± 2.6**	61.7 ± 4.0	49.8 ± 3.0
Musk	**94.7 ± 0.3**	94.3 ± 0.8	88.1 ± 0.9
Spam	90.4 ± 0.9	90.4 ± 1.5	**90.8 ± 1.2**
StatlogImgSeg	93.7 ± 0.9	**94.5 ± 0.8**	94.1 ± 0.8
StatlogLandSat	**86.4 ± 0.4**	86.1 ± 0.6	86.3 ± 0.3
Waveform	83.5 ± 1.5	**85.9 ± 0.9**	85.1 ± 0.9
90 % labeled			
COIL20	95.1 ± 1.0	**95.2 ± 1.3**	90.6 ± 1.7
COIL100	**90.7 ± 0.4**	88.9 ± 0.9	85.9 ± 0.8
CrowdMap	47.4 ± 1.3	**48.5 ± 1.9**	38.7 ± 5.7
DNA	90.0 ± 0.8	**90.6 ± 1.4**	80.2 ± 1.0
Gisette	90.1 ± 0.3	**95.3 ± 0.5**	84.7 ± 1.2
Isolet	**94.1 ± 0.3**	92.9 ± 0.4	87.7 ± 0.5
KDEF_Front	**65.7 ± 3.8**	65.5 ± 2.9	57.8 ± 3.3
Musk	**95.8 ± 0.8**	94.7 ± 0.6	89.4 ± 0.9
Spam	90.2 ± 0.8	**91.2 ± 0.6**	90.9 ± 0.8
StatlogImgSeg	**95.1 ± 0.9**	**95.1 ± 1.0**	94.1 ± 0.8
StatlogLandSat	**87.2 ± 0.4**	86.8 ± 0.2	86.6 ± 0.3
Waveform	83.9 ± 1.3	**86.0 ± 1.1**	85.4 ± 0.8

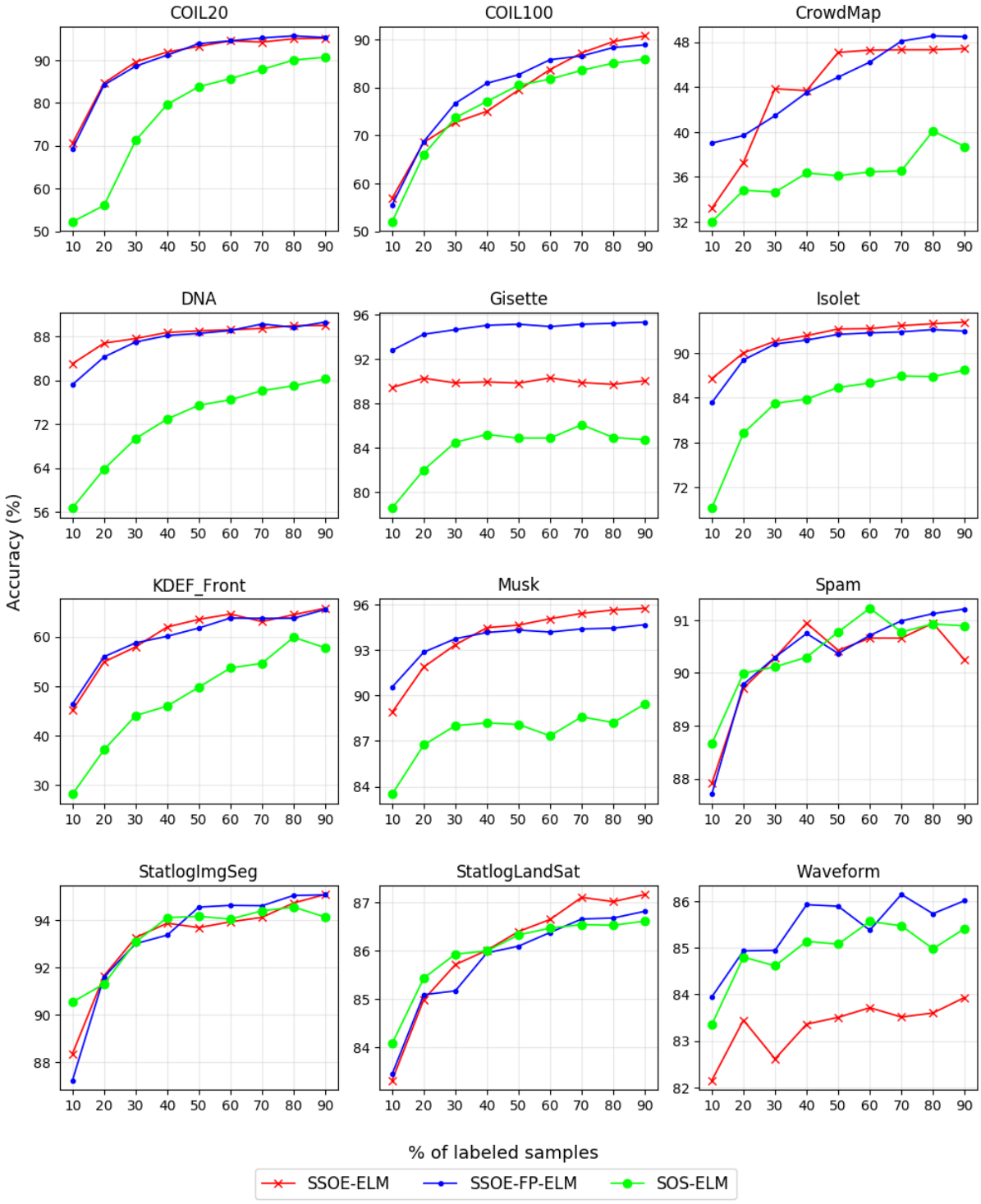

Figure 14 – Algorithms accuracy (%) in semi-supervised experiment with Gradual FCD

Table 16 – Algorithms accuracy (%) in semi-supervised experiment with Gradual LCD

Dataset	SSOE-ELM	SSOE-FP-ELM	SOS-ELM
10 % labeled			
COIL20	**68.5 ± 3.2**	65.8 ± 4.5	21.8 ± 2.7
COIL100	52.7 ± 2.4	**58.1 ± 1.8**	50.0 ± 2.4
CrowdMap	33.9 ± 4.2	**38.4 ± 4.8**	32.1 ± 3.1
DNA	**86.1 ± 1.2**	82.3 ± 1.8	58.1 ± 2.8
Gisette	89.8 ± 0.7	**93.6 ± 0.7**	80.8 ± 1.2
Isolet	80.4 ± 3.9	**82.1 ± 2.8**	64.0 ± 2.6
KDEF_Front	45.6 ± 5.8	**49.1 ± 3.1**	29.8 ± 3.9
Musk	91.7 ± 0.7	**93.0 ± 1.0**	86.1 ± 1.0
Spam	89.1 ± 1.0	**89.3 ± 1.3**	**89.3 ± 1.1**
StatlogImgSeg	**88.9 ± 3.8**	85.3 ± 4.2	87.2 ± 5.1
StatlogLandSat	81.1 ± 4.5	**82.3 ± 2.8**	80.1 ± 6.4
Waveform	82.9 ± 1.0	84.2 ± 0.8	**84.5 ± 0.7**
50 % labeled			
COIL20	77.3 ± 7.1	**77.6 ± 4.5**	45.7 ± 3.7
COIL100	72.3 ± 2.0	**74.9 ± 1.5**	69.3 ± 1.3
CrowdMap	**47.7 ± 2.5**	46.3 ± 2.7	36.6 ± 3.4
DNA	**91.5 ± 0.5**	91.4 ± 0.6	77.6 ± 1.4
Gisette	90.0 ± 0.6	**95.6 ± 0.4**	86.3 ± 1.0
Isolet	**81.3 ± 3.9**	81.1 ± 3.7	74.4 ± 4.9
KDEF_Front	61.4 ± 4.1	**65.2 ± 4.2**	54.3 ± 5.8
Musk	**96.6 ± 0.4**	95.2 ± 0.4	90.8 ± 0.8
Spam	91.3 ± 0.9	91.4 ± 0.7	**91.8 ± 0.7**
StatlogImgSeg	89.0 ± 5.8	**89.1 ± 4.4**	88.2 ± 6.2
StatlogLandSat	81.2 ± 7.0	**84.5 ± 4.5**	82.3 ± 7.0
Waveform	84.4 ± 0.7	**86.6 ± 0.6**	86.3 ± 0.8
90 % labeled			
COIL20	73.8 ± 4.2	**78.4 ± 5.3**	69.2 ± 6.0
COIL100	**77.6 ± 1.3**	76.1 ± 1.6	70.5 ± 1.7
CrowdMap	49.3 ± 1.0	**50.9 ± 1.4**	40.8 ± 2.4
DNA	91.9 ± 0.4	**92.9 ± 0.8**	82.0 ± 0.4
Gisette	90.1 ± 0.3	**95.8 ± 0.3**	86.2 ± 0.9
Isolet	**81.8 ± 3.5**	80.7 ± 4.4	75.1 ± 5.8
KDEF_Front	**66.1 ± 5.1**	64.6 ± 5.3	63.0 ± 5.0
Musk	**97.4 ± 0.3**	95.3 ± 0.5	90.9 ± 1.0
Spam	91.6 ± 0.8	**92.2 ± 0.6**	91.5 ± 0.6
StatlogImgSeg	91.5 ± 5.1	**93.7 ± 4.4**	89.8 ± 6.5
StatlogLandSat	**83.9 ± 5.6**	82.5 ± 3.8	76.9 ± 8.9
Waveform	85.8 ± 1.0	**87.3 ± 0.6**	86.5 ± 1.0

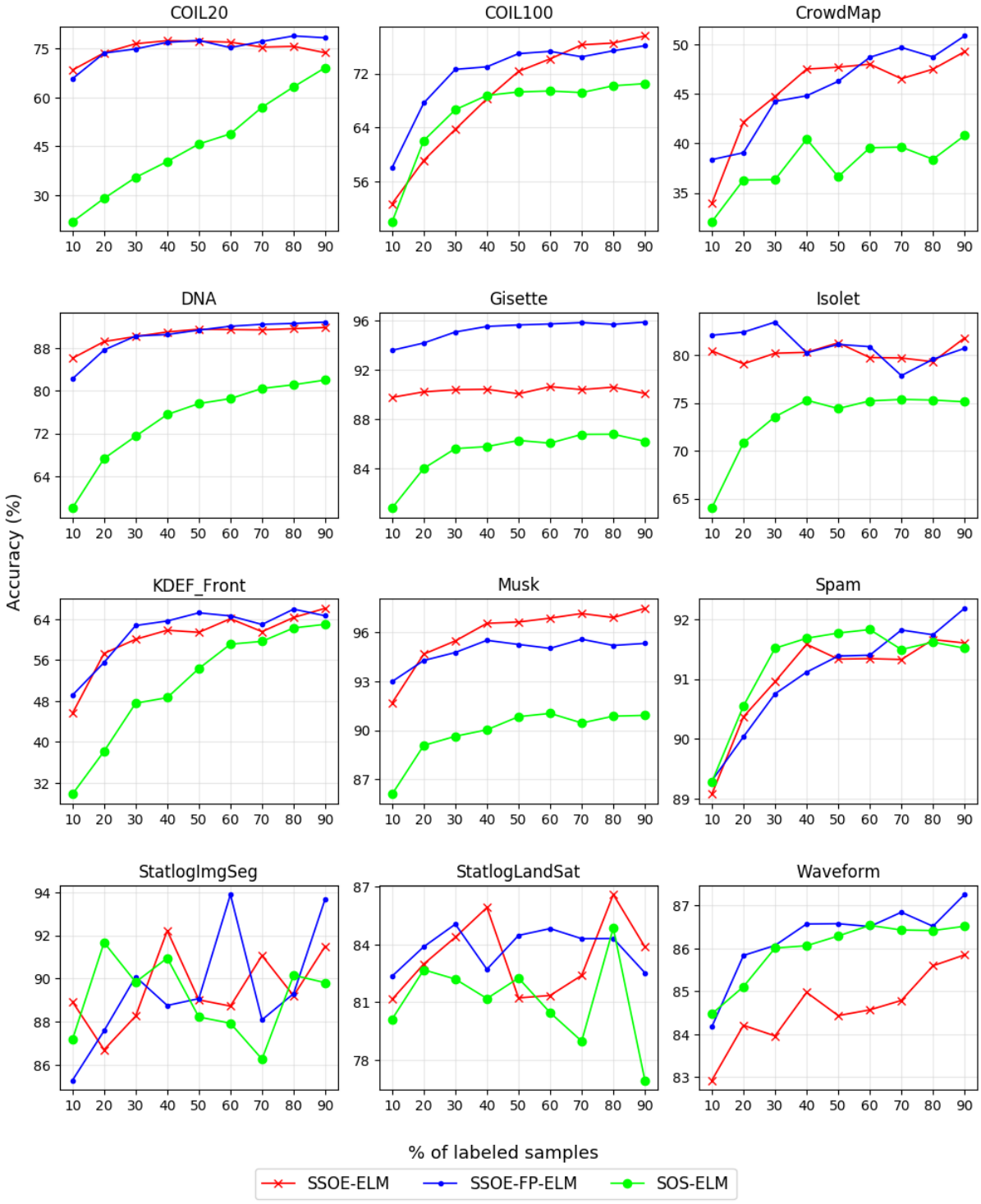

Figure 15 – Algorithms accuracy (%) in semi-supervised experiment with Gradual LCD

4.8 Remarks

In the tests performed, the SSOE-ELM showed accuracy slightly lower than the SVM algorithm in fully supervised situations, but still competitive. In semi-supervised situations, the SSOE-ELM was compared to the SOS-ELM, an algorithm with the same characteristics as the proposed one (semi-supervised, online and ELM-based). Regarding accuracy, SSOE-ELM has shown to be superior to SOS-ELM in most tests, in datasets with and without concept drift. And with regard to training time, the online update mechanism of SSOE-ELM, based on Elastic ELM, has shown to be more efficient than SOS-ELM, which was based on OS-ELM. In SSOE-ELM, two intermediate matrices are updated for each training partition processed, but the network output weights do not need to be calculated and can be updated at the end of the training. So, the calculation of the β output weights matrix, which involves a matrix inversion and is the most expensive operation of the training, can be performed only once. In contrast, the SOS-ELM performs an update on the β matrix at each training partition, by inverting a smaller matrix.

Additionally, the SSOE-ELM has shown a greater generalization ability than the SOS-ELM, observing the norm of its output weights. This fact is due to the α regularization factor, which also makes the solution found more stable. Despite not being directly related to greater accuracy, greater generalization capacity makes the model adaptable in relation to test samples not found during training.

In the tests performed, the SSOE-FP-ELM showed accuracy close to the SSOE-ELM in fully supervised situations, overcoming it in performance in some cases. This means that the use of a forgetting parameter did not negatively impact the performance of the algorithm in the base case (supervised). In semi-supervised situations, SSOE-FP-ELM was also close to SSOE-ELM, presenting inferior performance in a few cases. The difference between the two algorithms is more visible in concept drift situations, in which the SSOE-FP-ELM is more accurate than the SSOE-ELM in practically all cases, especially in cases of LCD. In comparison with the FP-ELM, an algorithm also adapted to deal with concept drifts, the SSOE-FP-ELM is superior in accuracy in fully supervised situations without concept drifts, and competitive in the presence of concept drift. The main difference between SSOE-FP-ELM and FP-ELM is the ability of the proposed algorithm to work also in semi-supervised situations. Regarding the training time, the SSOE-FP-ELM is slower than the SSOE-ELM, due to the calculation of the γ forgetting parameter. The supervised factor γ_s requires the matrix of β output weights to be calculated for each training partition, and the unsupervised factor γ_{us} requires the calculation of Laplacian scores for each training partition as well. Both operations are computationally expensive, greatly increasing the training time of the algorithm.

In some semi-supervised plots, it is possible to observe a decrease in the accuracy of the algorithms as the percentage of labeled samples increases. This is due to the fact

that the process of parameter optimization of the algorithms was performed with 10% of the training set labeled, and 90% unlabeled (worst case in the experiments). Therefore, the best parameters found refer to this dataset configuration, and are not necessarily the same when the number of labeled samples increases.

Regarding the type of dataset most suitable for each algorithm, it is important to note that SSOE-ELM and SSOE-FP-ELM are ELM-based algorithms, and for this reason they have no usage restrictions for any type of dataset and present satisfactory results in different situations. In addition, the experiments performed showed that in datasets with a large number of features, the proposed algorithms indicate better results. Despite this, some factors can reduce the accuracy of the proposed algorithms, such as:

- The proposed algorithms do not have any mechanism for acquiring features in image datasets that take into account the position of the pixels, such as the convolutions of a CNN. Therefore, the proposed algorithms tend to obtain lower results than CNNs in fully supervised situations in datasets where the inputs are image pixels.
- SSOE-ELM and SSOE-FP-ELM do not use any technique to deal with noisy datasets, such as Waveform and CrowdsourcedMapping (DUA; GRAFF, 2019), and suffer a decrease in accuracy in these situations. SSOE-FP-ELM is even more susceptible to high noise levels, which affect the concept drift detection mechanism.

5 Case Study

To confirm the feasibility of the previously proposed algorithms, a case study was investigated. In this case study we use as benchmark the PAMAP2 (Physical Activity Monitoring for Aging People) dataset, available at the University of California Irvine (UCI) Machine Learning Repository (DUA; GRAFF, 2019).

In these experiments, the same source code and the same test environment as the previous experiments was used.

5.1 Data processing

The PAMAP2 dataset contains data of 18 different physical activities (such as running, ascending stairs, house cleaning, etc.), performed by 9 subjects wearing a heart rate monitor and 3 inertial measurement units (IMU). The IMUs are positioned on the dominant arm wrist, dominant leg ankle, and chest. All sensor data were collected at 100Hz, and some values are missing. For more information about the PAMAP2 dataset, please refer to Reiss & Stricker (2012).

For this work, the dataset was processed following the activity recognition chain presented in Roggen et al. (2011). The steps of data processing are described in the following sections.

5.1.1 Preprocessing

In the preprocessing step, some information present in the dataset was discarded. According to Reiss & Stricker (2012), for different tasks in physical activity monitoring, accelerometers outperform gyroscopes. Therefore, data from gyroscopes and magnetometers were discarded, and only accelerometers were used.

For missing data, simple methods of value imputation were used, following Saar-Tsechansky & Provost (2007).

5.1.2 Segmentation

Segmentation of the data stream is required to obtain a section of data that probably contains a gesture or movement. A common type of segmentation technique is the sliding window with a predefined size.

In Reiss & Stricker (2012), a sliding window of 5.12 seconds, corresponding to 512 samples, was used. In this work, the same window size is used, with a 1 second shift

between two windows.

For classification purposes, only windows containing samples labeled with the same class have been retained.

5.1.3 Feature extraction

In Reiss & Stricker (2012), 137 features were extracted from each data window of 512 samples: 133 features from IMU acceleration data and 4 features from the heart rate data.

In this work, a smaller set of 62 features were extracted from the heart rate, IMU acceleration and temperature data. Feature extraction followed the techniques presented in Figo et al. (2010) and Preece et al. (2008). Sensor data were normalized before feature extraction.

For heart rate and body temperatures, mean and standard deviation were computed. For 3D-acceleration data, mean and standard deviation were calculated for each axis, and the correlation between each pair of axes is computed.

The techniques used, although simple, showed good results in the experiment. The choice for such techniques, rather than more complex ones, was motivated by the computational effort required to process the samples. Since ELM-based algorithms have as their main feature their short training time, feature extraction techniques that can be processed faster become more appropriate.

5.2 Experimental setup

In this work, three experiments with PAMAP2 dataset were performed. In the first experiment, the proposed algorithms were tested in the same classification problems defined in Reiss & Stricker (2012), in a fully supervised way, comparing the obtained results with those of the reference paper. In the second experiment, the proposed algorithms were tested in these four classification problems in a semi-supervised way, comparing the obtained results with SOS-ELM algorithm. In the third experiment, the proposed algorithms were tested in concept drift situations, showing the learning evolution and classification recovery capability, comparing the obtained results with SOS-ELM and FP-ELM algorithms.

The training of all algorithms was performed in an online manner using a chunk of the training partition each time. Each experiment was repeated 30 times, using the mean and standard deviation of the classification accuracy as performance metrics.

The parameters of the algorithms for each PAMAP2 problem were determined by the Random Search method, using the training partition as validation. For SSOE-ELM, SSOE-FP-ELM and SOS-ELM, values in the range $[10^{-5}, 10^{-4}, \ldots, 10^{4}, 10^{5}]$ were tested

for the parameter C. The size of the chunks for online training of the algorithms was set in the interval [50, 52, ..., 200], and was later set to 200 for all algorithms to facilitate the reproduction of the experiments. For the SOS-ELM algorithm, the tradeoff parameter λ was tested with values between 0 and 1. Note that for parameters C, size of chunks and λ, the same ranges presented by Jia et al. (2016) were adopted. The values tested for the number of neighbors μ, required to compute graph Laplacian, were [5, 10, 15, 20]. Huang et al. (2012) chooses the regularization parameter α of ELM in the range [2^{-24}, 2^{-23}, ..., 2^{24}, 2^{25}], so the same interval was used in this thesis for SSOE-ELM and SSOE-FP-ELM. For the parameters θ and η of FP-ELM, the values tested were in the range [0.0, 0.1, ..., 19.9, 20.0], as proposed by Liu, Wu & Jiang (2016). For the regularization parameter α, Liu, Wu & Jiang (2016) set a fixed value of 0.02, and in this thesis this value was also used. The number of neurons n for all algorithms was tested with values in the interval [50, 3000]. For SVM, Hsu, Chang & Lin (2003) suggests the choice of regularization parameter α in the range [2^{-5}, 2^{-3}, ..., 2^{15}] and parameter γ in the range [2^{-15}, 2^{-13}, ..., 2^{3}]. In this thesis, the suggested intervals have been extended to [2^{-5}, 2^{-4}, ..., 2^{15}] for α and [2^{-15}, 2^{-14}, ..., 2^{3}] for γ. In all experiments, the RBF kernel was used. For each experiment, an hyperparameter optimization was performed for all algorithms using the intervals previously described.

5.3 Classification problems of PAMAP2 dataset

In Reiss & Stricker (2012) there are four classification problems, with the 12 activities presented in Table 17 performed by 9 subjects. The classification problemas are presented in Table 18 with number of features, number of labels, training and testing samples, and are briefly described in the following sections.

Table 17 – 12 activities used by Reiss & Stricker (2012)

1	Lying	2	Sitting	3	Standing
4	Walking	5	Running	6	Cycling
7	Nordic walking	12	Ascending stairs	13	Descending stairs
16	Vacuum cleaning	17	Ironing	24	Rope jumping

Table 18 – PAMAP2 Classification Problems

Dataset	Features	Labels	Train Samples	Test Samples	Minority Class	Majority Class
Intensity Estimation	62	3	13257	5682	2522	8502
Basic Activity Recognition	62	5	7340	3146	955	3678
Background Activity Recognition	62	6	13257	5682	955	8453
All Activities Recognition	62	12	13257	5682	468	2353

5.3.1 Intensity estimation task

In this problem, 3 classes are defined: activities of light, moderate and vigorous effort based on the metabolic equivalent (MET) of the different activities according to Ainsworth et al. (2000). The 3 classes are defined as following:

- Activities of light effort (< 3.0 METs): lying, sitting, standing and ironing;
- Activities of moderate effort (3.0-6.0 METs): vacuum cleaning, descending stairs, normal walking, Nordic walking and cycling;
- Activities of vigorous effort (> 6.0 METs): ascending stairs, running and rope jumping.

5.3.2 Basic activity recognition task

In this problem, 5 activities are selected as basic classes: lying, sitting/standing, walking, running and cycling. All other activities are discarded for this task.

It is important to note that the activities sitting and standing are forming one class in this problem, since an extra IMU on the thigh would be needed for a reliable differentiation of these postures (REISS; STRICKER, 2012).

5.3.3 Background activity recognition task

In this problem, the same 5 classes of basic activity recognition task are used, and a new class called "Others" is included, with the samples of remaining activities. The goal of this test is to correctly classify the 5 basic activities from previous test, ignoring any other activity performed in the meantime. Since the user can perform countless activities and just a few of them are important to monitoring algorithms, the use of "Others" class is justified.

5.3.4 All activity recognition task

In this problem, all 12 defined activities presented in Table 17 are used.

5.4 First experiment - supervised comparison

In this first experiment, SSOE-ELM and SSOE-FP-ELM were evaluated in the same four classification problems defined by Reiss & Stricker (2012), in a fully supervised way (all training samples are labeled), comparing the results with the reference paper. An hyperparameter optimization with Random Search method was performed in all algorithms. Table 19 presents these parameters.

Table 19 – Algorithm parameters for supervised experiment

	SSOE-ELM				SSOE-FP-ELM				SOS-ELM				FP-ELM			SVM	
Dataset	n	α	C	μ	n	α	C	μ	n	λ	C	μ	n	θ	η	γ	α
Intensity Estimation	2950	2^{-1}	10^3	10	2950	2^{-3}	10^3	20	2450	0.84	10^4	10	2550	1.4	2.0	-6	14
Basic Activity Recognition	2350	2^{-7}	10^2	15	2700	2^{-3}	10^2	5	2500	0.62	10^3	20	2900	10.9	8.4	-6	14
Background Activity Recognition	2950	2^{-16}	10^{-1}	20	2800	2^4	10^5	5	2850	0.08	10^3	5	3000	14.3	16.3	-5	13
All Activities Recognition	2950	2^6	10^5	5	2700	2^{20}	10^3	15	2200	0.35	10^4	10	2550	10.2	15.5	-7	15

Table 20 presents the accuracy results of the proposed algorithms, comparing with the results presented by Reiss & Stricker (2012). In addition, the results obtained with FP-ELM, SOS-ELM and SVM for the same problems were also compared with the proposed algorithms. Reiss & Stricker (2012) does not provide any information about the execution time. Therefore in this experiment no comparison of training time and algorithm execution was performed.

Table 20 – Algorithms accuracy (%) in supervised experiment

Classifier	Intensity Estimation	Basic Activity Recognition	Background Activity Recognition	All Activity Recognition
Decision tree (C4.5)	97.89%	99.70%	97.09%	95.46%
Boosted C4.5	**99.86%**	99.95%	**99.80%**	**99.69%**
Bagging C4.5	98.31%	99.71%	97.87%	96.66%
Naive Bayes	88.45%	99.23%	85.08%	94.38%
kNN	**99.86%**	**100.00%**	99.57%	99.25%
SVM	99.3% ± 0.1%	99.3% ± 0.2%	98.9% ± 0.1%	98.6% ± 0.1%
FP-ELM	92.3% ± 0.8%	97.6% ± 0.3%	86.1% ± 1.5%	78.6% ± 1.8%
SOS-ELM	99.1% ± 0.1%	99.3% ± 0.2%	98.5% ± 0.2%	97.9% ± 0.2%
SSOE-ELM	99.2% ± 0.1%	99.2% ± 0.2%	98.6% ± 0.1%	98.1% ± 0.2%
SSOE-FP-ELM	99.1% ± 0.1%	99.3% ± 0.2%	98.6% ± 0.1%	97.8% ± 0.2%

In all four problems, SSOE-ELM and SSOE-FP-ELM obtained accuracy below the best result presented by Reiss & Stricker (2012). Considering that the proposed algorithms were developed to deal with semi-supervised situations, and that the difference in accuracy for the best result was below 2% in all four datasets, the results presented are still competitive in the fully supervised case. In addition, SSOE-ELM and SSOE-FP-ELM had higher accuracy than FP-ELM in all datasets, and better results than SOS-ELM in 3 out of 4 datasets.

5.5 Second experiment - semi-supervised comparison

In the second experiment, the proposed algorithms were tested in the same four classification problems of PAMAP2 dataset in a semi-supervised way, comparing the obtained results with SOS-ELM algorithm. In this experiment, the training dataset was divided in labeled and unlabeled partitions, starting with 10% of labeled samples and 90% of unlabeled samples, and gradually increasing to 90% labeled and 10% unlabeled samples. The training was then performed in an online manner using a chunk of the training partition each time. An hyperparameter optimization with Random Search method was performed in all algorithms, using only 10% of labeled samples (worst case in this experiment). Table 21 presents these parameters.

Table 21 – Algorithms parameters for semi-supervised experiment

	SSOE-ELM				SSOE-FP-ELM				SOS-ELM			
Dataset	n	α	C	μ	n	α	C	μ	n	λ	C	μ
IntensityEstimation	2950	2^{13}	10^2	10	2100	2^{15}	10^3	5	200	0.81	10^5	15
BasicActivityRecognition	2250	2^{12}	10^5	5	2850	2^{18}	10^4	5	800	0.45	10^2	20
BackgroundActivityRecognition	2500	2^{14}	10^{-5}	15	1000	2^{15}	10^3	20	3000	0.01	10^1	20
AllActivitiesRecognition	1350	2^{17}	10^4	5	2550	2^{16}	10^5	20	300	0.02	10^{-1}	5

Table 22 presents the mean accuracy of the algorithms for three configurations of labeled training samples. Figure 16 shows the mean accuracy of the algorithms for all configurations of labeled training samples.

Table 22 – Algorithms accuracy (%) in semi-supervised experiment

Dataset	SSOE-ELM	SSOE-FP-ELM	SOS-ELM
10% labeled			
IntensityEstimation	96.4 ± 0.3	**96.5 ± 0.4**	95.4 ± 0.5
BasicActivityRecognition	97.2 ± 0.4	**98.0 ± 0.3**	96.5 ± 0.5
BackgroundActivityRecognition	**95.7 ± 0.4**	95.3 ± 0.4	94.2 ± 0.4
AllActivitiesRecognition	**93.3 ± 0.5**	92.9 ± 0.4	92.7 ± 0.4
50% labeled			
IntensityEstimation	97.8 ± 0.2	**98.6 ± 0.2**	96.8 ± 0.3
BasicActivityRecognition	**98.9 ± 0.2**	98.6 ± 0.2	98.7 ± 0.2
BackgroundActivityRecognition	97.5 ± 0.2	**97.8 ± 0.2**	96.6 ± 0.3
AllActivitiesRecognition	95.6 ± 0.2	96.6 ± 0.2	**96.7 ± 0.2**
90% labeled			
IntensityEstimation	98.0 ± 0.1	**98.9 ± 0.1**	97.0 ± 0.3
BasicActivityRecognition	**99.2 ± 0.1**	98.7 ± 0.2	99.1 ± 0.1
BackgroundActivityRecognition	97.7 ± 0.2	**98.3 ± 0.2**	96.9 ± 0.3
AllActivitiesRecognition	95.9 ± 0.2	**97.2 ± 0.3**	97.1 ± 0.2

In this semi-supervised experiment, the SSOE-ELM algorithm performed better than SOS-ELM in 3 out of 4 datasets, losing in the "All Activity Recognition" task. SSOE-FP-ELM presents better results than SOS-ELM in two datasets, and a very close

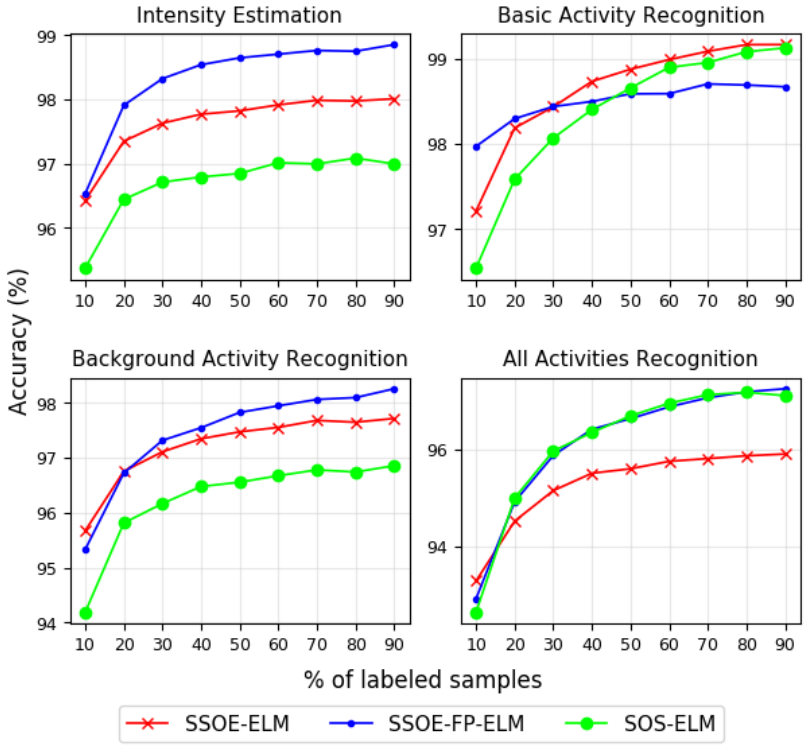

Figure 16 – Algorithms accuracy (%) in semi-supervised experiment

performance in the task "All Activity Recognition". In the "Basic Activy Recognition" task, the performance of SSOE-FP-ELM is better than that of SOS-ELM with few labeled samples, but the situation is reversed as more labeled samples are used.

5.6 Third experiment - supervised comparison with concept drift

In this third experiment, SSOE-ELM and SSOE-FP-ELM were evaluated in the four classification problems of PAMAP2 dataset with generated concept drift, in a fully supervised way (all training samples are labeled), comparing the obtained results with SOS-ELM and FP-ELM.

Two different concept drifts were tested. In the first experiment, a Label Concept Drift (LCD) was generated in the middle of training dataset, changing the labels. In the second experiment, a Feature Concept Drift (FCD) was generated in the middle of training dataset, changing the inputs.

To measure the drift and update the forgetting parameter in SSOE-FP-ELM algorithm, the labeled set of each training partition was first used as a test partition to get the model accuracy, and then used as a training partition.

The same parameterss of first experiment (fully supervised classification without concept drift) was used for all algorithms. For each dataset, the experiment was repeated 30 times, using the mean and standard deviation of the classification accuracy as performance

metrics.

Abrupt Concept Drift

Table 23 presents the accuracy of the proposed algorithms in supervised situations with abrupt FCD and LCD, comparing with SOS-ELM and FP-ELM.

Table 23 – Algorithms accuracy (%) in supervised experiment with Abrupt FCD and Abrupt LCD

Dataset (Abrupt FCD)	SSOE-ELM	SSOE-FP-ELM	SOS-ELM	FP-ELM
Intensity Estimation	98.4 ± 0.1	**98.6 ± 0.2**	98.2 ± 0.2	92.1 ± 0.7
Basic Activity Recognition	98.5 ± 0.2	**98.9 ± 0.2**	98.8 ± 0.2	97.5 ± 0.4
Background Activity Recognition	97.4 ± 0.2	**97.8 ± 0.2**	97.3 ± 0.2	87.1 ± 2.1
All Activities Recognition	96.4 ± 0.3	**96.7 ± 0.3**	96.1 ± 0.2	78.6 ± 1.6
Dataset (Abrupt LCD)	SSOE-ELM	SSOE-FP-ELM	SOS-ELM	FP-ELM
Intensity Estimation	54.0 ± 22.6	**96.2 ± 3.6**	57.8 ± 20.5	93.2 ± 2.5
Basic Activity Recognition	47.4 ± 17.2	91.2 ± 7.9	46.0 ± 17.9	**97.8 ± 0.6**
Background Activity Recognition	46.9 ± 14.7	**94.3 ± 5.1**	46.5 ± 15.3	88.2 ± 2.8
All Activities Recognition	36.3 ± 8.0	**93.7 ± 2.0**	33.6 ± 7.5	78.4 ± 2.3

In abrupt FCD, the SSOE-FP-ELM algorithm performs better than all other algorithms in the four classification problems. On the abrupt LCD, SSOE-FP-ELM does better than SOS-ELM and the other proposed algorithm, SSOE-ELM, on all datasets, and has better performance than FP-ELM on 3 out of 4 datasets, losing in the "Basic Activity Recognition" task.

Gradual Concept Drift

Table 24 presents the accuracy of the proposed algorithms in an experiment with gradual FCD and LCD, comparing with SOS-ELM and FP-ELM.

Table 24 – Algorithms accuracy (%) in supervised experiment with Gradual FCD and Gradual LCD

Dataset (Gradual FCD)	SSOE-ELM	SSOE-FP-ELM	SOS-ELM	FP-ELM
Intensity Estimation	98.6 ± 0.2	**98.7 ± 0.1**	98.5 ± 0.3	92.1 ± 1.2
Basic Activity Recognition	98.8 ± 0.2	98.9 ± 0.1	**99.0 ± 0.2**	97.5 ± 0.4
Background Activity Recognition	97.8 ± 0.3	**98.0 ± 0.1**	97.7 ± 0.3	86.1 ± 1.3
All Activities Recognition	**96.9 ± 0.2**	96.8 ± 0.2	96.7 ± 0.4	78.5 ± 2.2
Dataset (Gradual LCD)	SSOE-ELM	SSOE-FP-ELM	SOS-ELM	FP-ELM
Intensity Estimation	**99.2 ± 0.1**	**99.2 ± 0.1**	99.0 ± 0.1	92.3 ± 0.8
Basic Activity Recognition	95.9 ± 6.5	**97.5 ± 3.3**	92.9 ± 8.7	97.4 ± 0.3
Background Activity Recognition	93.0 ± 5.2	**97.5 ± 1.9**	91.4 ± 9.6	86.6 ± 1.3
All Activities Recognition	84.0 ± 9.3	**93.5 ± 3.0**	82.3 ± 8.0	78.9 ± 2.1

In the gradual FCD, the algorithms are more balanced. The two proposed algorithms perform better in 3 out of 4 datasets, and SOS-ELM obtains better results in one dataset, all 3 algorithms with very close results. In the gradual LCD, the proposed algorithms

obtain better results in all datasets, with SSOE-FP-ELM winning in 3 of them, and the two proposed algorithms tied in the "Intensity Estimation" task.

5.7 Fourth experiment - semi-supervised comparison with concept drift

In this forth experiment, SSOE-ELM and SSOE-FP-ELM were evaluated in the four classification problems of PAMAP2, in a semi-supervised way, in the presence of concept drift, comparing the obtained results with SOS-ELM. For each dataset, the training partition was divided into labeled and unlabeled instances, starting with 10% of labeled samples and 90% unlabeled, and gradually increasing to 90% of labeled samples and 10% unlabeled. The training was then performed in an online manner using a chunk of the training partition each time. For each division, the experiment was repeated 30 times, using the mean and standard deviation of the classification accuracy as performance metrics.

As in the previous experiment, two different concept drifts were tested: Label Concept Drift (LCD) and Feature Concept Drift (FCD).

To measure the drift and update the forgetting parameter in SSOE-FP-ELM algorithm, the labeled set of each training partition was first used as a test partition to get the model accuracy, and then used as a training partition. The same parameters of second experiment (semi-supervised classification without concept drift) were used for all algorithms. For each dataset, the experiment was repeated 30 times, using mean and standard deviation of classification accuracy as performance metrics.

Abrupt Concept Drift

Table 25 presents the mean accuracy of the algorithms for three configurations of labeled training samples with Abrupt FCD. Figure 17 shows the mean accuracy of the algorithms for all configurations of labeled training samples in presence of Abrupt FCD.

The SSOE-FP-ELM algorithm has shown better accuracy in all four datasets in the presence of abrupt FCD. The SSOE-ELM algorithm was better than the SOS-ELM in 3 out of 4 datasets, having the worst result in the task "All Activity Recognition".

Table 26 presents the mean accuracy of the algorithms for three configurations of labeled training samples with Abrupt LCD. Figure 18 shows the mean accuracy of the algorithms for all configurations of labeled training samples in presence of Abrupt LCD.

In abrupt LCD situations, the SSOE-FP-ELM algorithm had much higher accuracy than the others, while the SSOE-ELM and SOS-ELM algorithms had very similar results. This result highlights the fact that only the SSOE-FP-ELM algorithm has mechanisms to deal with concept drift.

Table 25 – Algorithms accuracy (%) in semi-supervised experiment with Abrupt FCD

Dataset	SSOE-ELM	SSOE-FP-ELM	SOS-ELM
10 % labeled			
Intensity Estimation	93.6 ± 0.4	**94.4 ± 0.5**	91.2 ± 0.9
Basic Activity Recognition	95.4 ± 0.6	**97.3 ± 0.4**	94.8 ± 0.8
Background Activity Recognition	92.8 ± 0.6	**93.5 ± 0.4**	89.9 ± 0.7
All Activities Recognition	**90.0 ± 0.7**	89.9 ± 0.7	88.9 ± 0.7
50 % labeled			
Intensity Estimation	96.1 ± 0.3	**98.0 ± 0.2**	93.5 ± 0.6
Basic Activity Recognition	98.1 ± 0.3	**98.2 ± 0.3**	97.7 ± 0.3
Background Activity Recognition	95.9 ± 0.3	**96.8 ± 0.2**	93.5 ± 0.4
All Activities Recognition	93.8 ± 0.4	**95.4 ± 0.3**	94.8 ± 0.4
90 % labeled			
Intensity Estimation	96.5 ± 0.3	**98.4 ± 0.2**	93.8 ± 0.5
Basic Activity Recognition	**98.5 ± 0.2**	98.2 ± 0.3	98.4 ± 0.3
Background Activity Recognition	96.3 ± 0.3	**97.5 ± 0.2**	94.0 ± 0.5
All Activities Recognition	94.3 ± 0.3	**96.2 ± 0.3**	95.5 ± 0.3

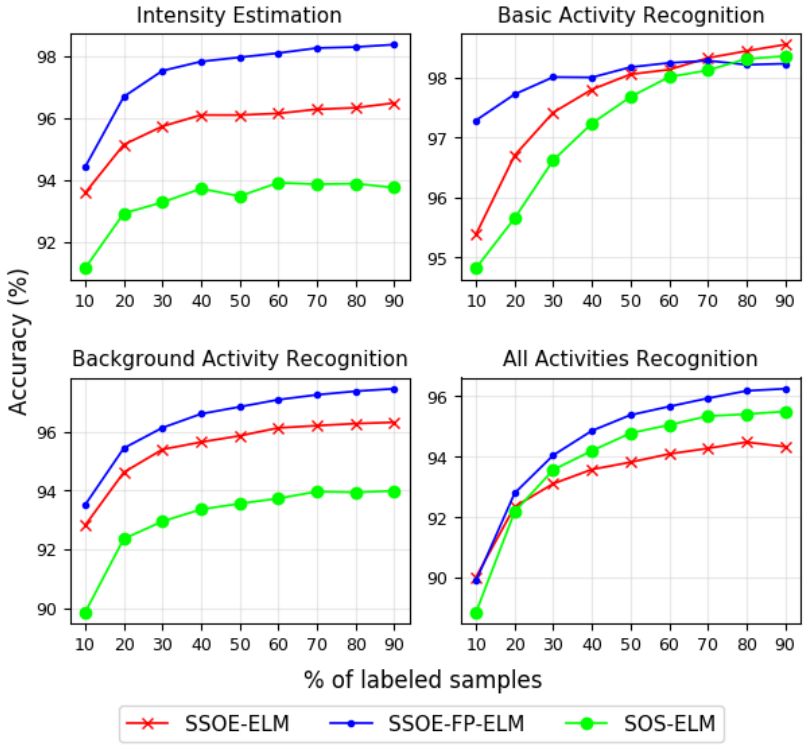

Figure 17 – Algorithms accuracy (%) in semi-supervised experiment with Abrupt FCD

Table 26 – Algorithms accuracy (%) in semi-supervised experiment with Abrupt LCD

Dataset	SSOE-ELM	SSOE-FP-ELM	SOS-ELM
10 % labeled			
Intensity Estimation	61.7 ± 22.0	**93.9 ± 1.9**	58.1 ± 18.3
Basic Activity Recognition	53.1 ± 10.4	**96.1 ± 3.1**	53.3 ± 14.8
Background Activity Recognition	49.1 ± 11.3	**90.4 ± 3.0**	48.7 ± 14.1
All Activities Recognition	39.4 ± 6.1	**88.3 ± 2.2**	41.5 ± 5.3
50 % labeled			
Intensity Estimation	53.0 ± 25.1	**96.8 ± 3.1**	50.0 ± 24.4
Basic Activity Recognition	49.3 ± 16.5	**96.2 ± 3.4**	47.2 ± 12.6
Background Activity Recognition	42.1 ± 16.2	**95.3 ± 2.9**	43.0 ± 16.0
All Activities Recognition	29.9 ± 8.4	**95.2 ± 0.9**	37.8 ± 9.3
90 % labeled			
Intensity Estimation	51.4 ± 22.2	**97.6 ± 2.1**	53.6 ± 24.1
Basic Activity Recognition	44.9 ± 20.0	**94.8 ± 7.8**	49.0 ± 13.4
Background Activity Recognition	40.3 ± 17.0	**94.6 ± 4.8**	46.9 ± 18.3
All Activities Recognition	28.3 ± 9.7	**95.8 ± 1.1**	30.6 ± 8.3

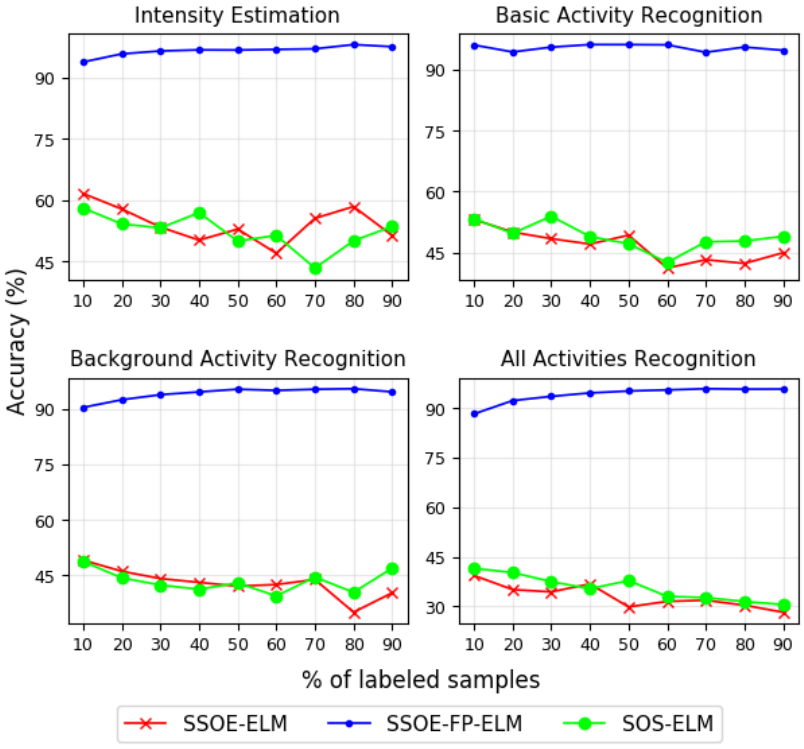

Figure 18 – Algorithms accuracy (%) in semi-supervised experiment with Abrupt LCD

Gradual Concept Drift

Table 27 presents the mean accuracy of the algorithms for three configurations of labeled training samples with Abrupt FCD. Figure 19 shows the mean accuracy of the algorithms for all configurations of labeled training samples in presence os Abrupt FCD.

In the plots presented, a similar behavior of the algorithms between abrupt and

gradual FCD is observed. The results are very close, with a reduction in the difference in accuracy between the algorithms in the gradual FCD.

Table 28 presents the mean accuracy of the algorithms for three configurations of labeled training samples with Abrupt LCD. Figure 20 shows the mean accuracy of the algorithms for all configurations of labeled training samples in presence os Abrupt LCD.

In situations of gradual LCD, the SSOE-FP-ELM algorithm continues to have the best performance in all datasets, but the difference in accuracy for the other algorithms is smaller.

Table 27 – Algorithms accuracy (%) in semi-supervised experiment with Gradual FCD

Dataset	SSOE-ELM	SSOE-FP-ELM	SOS-ELM
10 % labeled			
Intensity Estimation	94.7 ± 0.6	**94.9 ± 0.5**	93.2 ± 0.8
Basic Activity Recognition	96.3 ± 0.4	**97.6 ± 0.4**	95.6 ± 0.5
Background Activity Recognition	**94.0 ± 0.5**	93.7 ± 0.5	91.9 ± 0.5
All Activities Recognition	**91.9 ± 0.6**	90.5 ± 0.5	90.4 ± 0.9
50 % labeled			
Intensity Estimation	97.0 ± 0.4	**98.3 ± 0.2**	95.5 ± 0.6
Basic Activity Recognition	**98.4 ± 0.3**	98.3 ± 0.2	98.1 ± 0.4
Background Activity Recognition	96.5 ± 0.3	**97.1 ± 0.2**	94.7 ± 0.5
All Activities Recognition	94.8 ± 0.4	**95.7 ± 0.3**	95.4 ± 0.3
90 % labeled			
Intensity Estimation	97.4 ± 0.3	**98.5 ± 0.1**	95.7 ± 0.5
Basic Activity Recognition	**98.9 ± 0.2**	98.3 ± 0.2	98.8 ± 0.2
Background Activity Recognition	96.8 ± 0.3	**97.6 ± 0.2**	95.4 ± 0.3
All Activities Recognition	95.0 ± 0.4	**96.5 ± 0.3**	96.2 ± 0.3

Table 28 – Algorithms accuracy (%) in semi-supervised experiment with Gradual LCD

Dataset	SSOE-ELM	SSOE-FP-ELM	SOS-ELM
10 % labeled			
Intensity Estimation	96.4 ± 0.3	**96.6 ± 0.3**	95.3 ± 0.4
Basic Activity Recognition	91.4 ± 7.2	**95.2 ± 3.0**	91.0 ± 6.0
Background Activity Recognition	93.8 ± 2.5	**94.1 ± 1.7**	92.6 ± 3.0
All Activities Recognition	80.4 ± 6.1	**89.2 ± 1.5**	80.2 ± 5.1
50 % labeled			
Intensity Estimation	97.9 ± 0.2	**98.6 ± 0.1**	96.9 ± 0.2
Basic Activity Recognition	95.2 ± 5.5	**96.2 ± 4.1**	90.5 ± 8.9
Background Activity Recognition	93.7 ± 4.5	**95.8 ± 2.5**	93.5 ± 4.3
All Activities Recognition	81.9 ± 8.9	**93.2 ± 3.1**	83.0 ± 7.6
90 % labeled			
Intensity Estimation	98.1 ± 0.2	**98.8 ± 0.1**	97.0 ± 0.3
Basic Activity Recognition	94.7 ± 7.7	**96.8 ± 4.4**	91.7 ± 8.7
Background Activity Recognition	94.0 ± 5.6	**96.0 ± 3.4**	95.6 ± 2.5
All Activities Recognition	79.1 ± 8.0	**93.6 ± 2.7**	82.4 ± 8.4

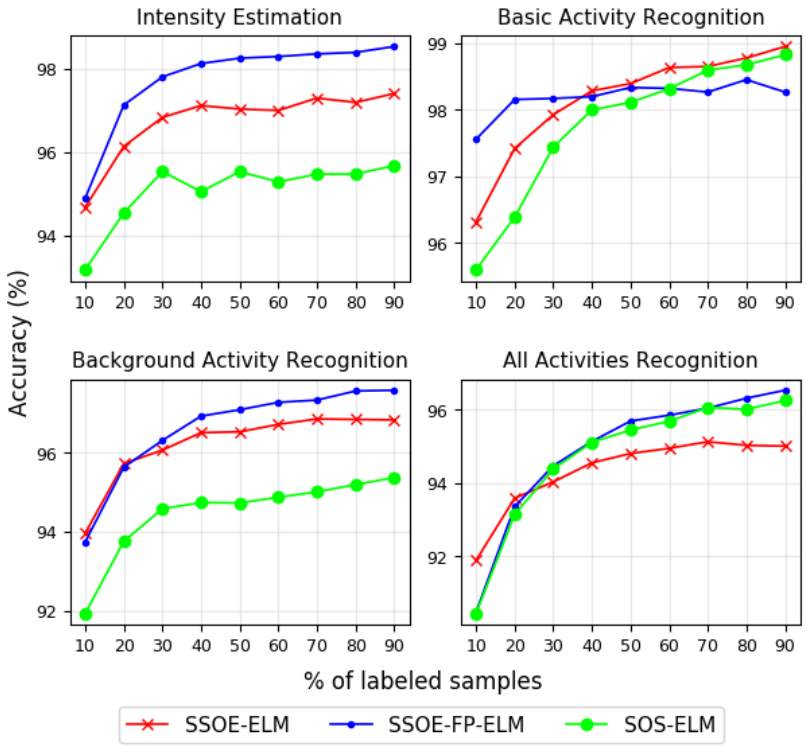

Figure 19 – Algorithms accuracy (%) in semi-supervised experiment with Gradual FCD

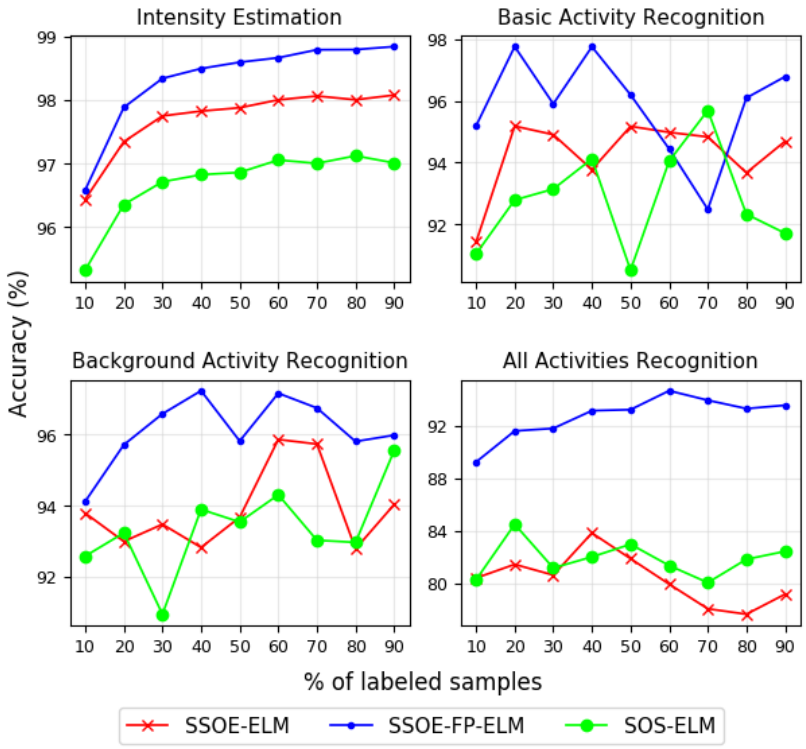

Figure 20 – Algorithms accuracy (%) in semi-supervised experiment with Gradual LCD

6 Conclusions

In this thesis, two elastic online semi-supervised algorithms were proposed for data classification. The first algorithm, called Semi-Supervised Online Elastic Extreme Learning Machine (SSOE-ELM), is a semi-supervised graph-based algorithm, with online update and low training time, developed to handle large and unbalanced datasets and data streams, able to solve problems in three big data dimensions: Velocity, Volume and Veracity. With respect to the Velocity dimension, the SSOE-ELM algorithm is based on ELM, being able to quickly process incoming training samples continuously. Regarding the Volume dimension, SSOE-ELM decomposes the output weight matrix calculation into two matrices U and V, being able to handle both large datasets and data streams. With regard to the Veracity dimension, two major challenges are dealing with partially labeled datasets and unbalanced datasets. The SSOE-ELM algorithm is capable of handling both problems using semi-supervised learning methods and a penalty coefficient for unbalanced datasets.

The second algorithm, called Semi-Supervised Online Elastic Extreme Learning Machine with hybrid Forgetting Parameter (SSOE-FP-ELM), consists of an extension of SSOE-ELM to address another major challenge of the big data, i.e., the Veracity dimension: the concept drift. In data streams, received data may change over time. These variations can be due to many factors: problems with data collection, transmission or pre-processing; changes in the monitored environment; among others. To address these drifts, a hybrid forgetting parameter was added to the SSOE-ELM to detect changes and adapt the model. This parameter is composed of a supervised factor, which periodically checks the model for accuracy drops as an indicator of concept drifts, and an unsupervised factor, which verifies changes in the input data distribution as an indicator of concept drifts. These two factors are combined, generating a forgetting parameter that is responsible for adapting the model to the detected changes, either updating with the new information, or discarding all the acquired knowledge and retraining from the beginning.

The algorithms were tested in supervised and semi-supervised situations, with and without concept drifts, comparing with the SOS-ELM algorithm, which has the same characteristics as the proposed algorithms (semi-supervised, online and ELM-based). Additionally, the proposed algorithms were compared with SVM, a well-known supervised classification algorithm, and FP-ELM, an ELM-based supervised algorithm that deals with concept drifts, to verify if the proposed algorithms would perform well compared to specialized algorithms in the situations of supervised classification and concept drifts, respectively.

At first, SSOE-ELM and SSOE-FP-ELM were tested in fully supervised situations to ensure that they would work in the basic case. In this experiments, public domain datasets with different amount of features and samples were used, partitioning the training dataset into small chunks to simulate data streams. The proposed algorithms were compared with SOS-ELM, SVM and FP-ELM. Supervised tests were conducted with and without the presence of concept drifts. The following experiments were performed in semi-supervised situations, comparing the results obtained with the SOS-ELM. The semi-supervised tests were also conducted with and without the presence of concept drifts.

In the experiments performed, SSOE-ELM and SSOE-FP-ELM showed good overall results. Compared with SVM, the proposed algorithms showed a slightly lower accuracy in most of the datasets used. This difference is due to the fact that they were developed to operate mainly in semi-supervised situations. The inferior but competitive performance of the algorithms compared to SVM shows that they are working in the basic (supervised) case, in addition to treating their main focus, the semi-supervised situation. When compared with SOS-ELM, another online and semi-supervised algorithm, the proposed algorithms outperform SOS-ELM in accuracy and generalization ability in most cases, and SSOE-ELM still presents shorter training time. This indicates that the proposed algorithms achieve their objective in solving semi-supervised and online problems. And when tested in concept drift situations compared to FP-ELM, the proposed algorithms presents higher accuracy in most supervised situations, and SSOE-FP-ELM presents better performance and better recovery ability in the case of two different concept drifts - Feature Concept Drift (FCD) and Label Concept Drift (LCD) - both gradual and abrupt.

The results obtained indicate that the proposed algorithms are viable alternatives to treat partially labeled, unbalanced and very large datasets, including data streams. Moreover, the SSOE-FP-ELM has shown to be able to handle different concept drifts well, despite having a longer training time.

Comparing the two proposed algorithms, the experiments showed that in fully supervised situations without the occurrence of concept drifts, both algorithms present close accuracy in most datasets. In semi-supervised cases without concept drift, a similar result is observed, with the two algorithms having a very close accuracy in most datasets. In the supervised or semi-supervised case with the occurrence of a concept drift, the SSOE-FP-ELM presents better results than the SSOE-ELM, especially on the LCD (gradual or abrupt) and the abrupt FCD.

Therefore, the following usage guidelines can be established: for classification of datasets that do not have a concept drift, it is better to use the SSOE-ELM, as it has an accuracy close to the SSOE-FP-ELM in these cases, and its training is faster because it is not necessary to calculate the forgetting parameter. In cases with abrupt FCD, or any type of LCD, the SSOE-FP-ELM is the best option, as it tends to present better accuracy

than the SSOE-ELM. In cases of gradual FCD, the SSOE-FP-ELM has slightly higher accuracy than the SSOE-ELM, and it should be analyzed for each dataset whether the accuracy gain is worth the increase in training time. In situations where it is not possible to identify if there is a concept drift, the SSOE-FP-ELM should be used, as it is a more general solution.

The main contributions presented in this thesis are:

- A new ELM-based semi-supervised online classification algorithm for unbalanced datasets that presents accuracy comparable to other algorithms in the literature and shorter training time.
- An extension of the previous ELM-based semi-supervised online classification algorithm for data streams in the presence of virtual and real concept drifts.
- A new hybrid forgetting parameter computation for semi-supervised algorithms that takes into account labeled and unlabeled samples.

Limitations and future directions

The proposed algorithms present some limitations, and there are many possibilities for improvements that may be addressed in future works:

- The λ tradeoff parameter of SSOE-ELM is calculated using a simple proportion of the number of unlabeled samples in the training base. This calculation considers that the more unlabeled samples are present in the training, the greater is the weight of these unlabeled samples in the final classification model. This premise is not always true, because in more specialized datasets (medical images, for example), the labeled samples have a much greater weight in the final result. An approach that weighs labeled and unlabeled samples, based on prior knowledge of experts, analysis of data distribution, clustering or entropy, could increase the performance of the algorithm.
- The γ forgetting parameter of the SSOE-FP-ELM is calculated as the maximum factor of change found (supervised or unsupervised). This calculation is based on the premise that once any type of concept drift is detected, the forgetting parameter must act and adapt the model. However, as noticed in the experiments, generally a FCD has less impact than a LCD on the model, due to the concept drift smoothing promoted by the neural network activation function. Therefore, it would not be necessary to apply the same forgetting parameter in both cases, even if the supervised and the unsupervised factors are numerically equal. In order to avoid unnecessary loss of knowledge by applying a higher forgetting parameter, a way of weighting the factors is necessary. One way (not yet studied) to minimize this problem is to

calculate Laplacian scores from the feature matrix H, and not from the input matrix X, but further studies are needed.

- To calculate the supervised factor of the forgetting parameter, the SSOE-FP-ELM needs to check the current accuracy of the model, which means calculating the β output weights matrix. As this operation is carried out with each training partition, the SSOE-FP-ELM loses one of the advantages presented by the SSOE-ELM: the speed of the training due to the lack of the output weights matrix calculation until the end of the training process. To avoid this, a mechanism that would inform the algorithm the best time to calculate the output weights matrix would be interesting. This mechanism could be based on the accuracy of the current model, only performing the recalculation when the difference in accuracy is greater than a threshold; or the number of samples processed, updating the output weights matrix after a certain number of training partitions. However, the first approach would not detect a gradual concept drift, as the accuracy could drop slowly, while the second approach would not detect abrupt concept drifts immediately, requiring a certain number of partitions until the forgetting parameter was triggered. Perhaps a hybrid mechanism would be the best option.

- The proposed algorithms use similarity information between labeled and unlabeled training samples, through the Laplacian matrix, to infer information about labels when there are not enough labeled samples to train the model. Works using Gabriel Graph, such as Takahashi, Torres & Braga (2019) and Torres et al. (2020), provide good results in obtaining information about the structure of the dataset. To obtain better results, the structural information from the training dataset could be used, generating the Gabriel Graph from labeled and unlabeled training samples.

Open Research Topics

- **Treatment of noisy data streams, detecting changes in the presence of different noise levels**. Since input data is changing over time, it is a challenging task to separate noise from data in concept drift situations.

- **Unsupervised learning**. This thesis presents two algorithms that demonstrate good results in supervised and semi-supervised learning situations. A research topic not yet covered is the adaptation of these algorithms to work in unsupervised learning cases. This can occur in two situations: when an abrupt concept drift is followed by training partitions that are completely unlabeled; or when the initial training partitions do not have labeled samples. A hybrid approach of classification and online clustering can be useful in cases where labels are not yet available.

- **Self-supervised learning**. In addition to the hybrid approach of classification and online clustering, an interesting development of the topic addressed would be to employ self-supervised learning to reduce the number of labels required for training the algorithms. A self-supervised approach would be particularly useful if adapted to assist in measuring the accuracy of the model for calculating the supervised factor of the forgetting parameter, removing an important limitation of the SSOE-FP-ELM, which is the need for all training partitions to have labeled samples.

- **Deep architectures**. This thesis presents improvements in ELM-based algorithms. Essentially, ELM is a "shallow" algorithm, as it is used to train neural networks with only one hidden layer. There are "deep" variations of the ELM, which stacks several ELM networks, such as the Multi-Layer ELM (ML-ELM) (KASUN et al., 2013), Stacked ELM (S-ELM) (Zhou et al., 2015), Semi-supervised Deep ELM (SDELM) (GU et al., 2015) and Stacked Sparse Denoising Autoencoder - Ridge Regression (SSDAE-RR) (CAO; HUANG; SUN, 2016). A possible evolution of SSOE-ELM and SSOE-FP-ELM is an adaptation to a deep architecture, which would allow the algorithms to map more complex features.

Bibliography

ADIWARDANA, D. D. F.; MATSUKAWA, A.; WHANG, J. Using generative models for semi-supervised learning. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention–MICCAI*. [S.l.: s.n.], 2016.

AINSWORTH, B. E. et al. Compendium of physical activities: an update of activity codes and MET intensities. *Medicine and Science in Sports and Exercise*, v. 32, n. 9; SUPP/1, p. S498–S504, 2000.

AKUSOK, A. et al. High-performance Extreme Learning Machines: a complete toolbox for big data applications. *IEEE Access*, v. 3, p. 1011–1025, 2015.

AN, X. et al. Sample selected extreme learning machine based intrusion detection in fog computing and mec. *Wireless Communications and Mobile Computing*, v. 2018, 2018.

ANDERSON, R. et al. Recurring concept meta-learning for evolving data streams. *Expert Systems with Applications*, v. 138, 2019.

ARAÚJO, L. R. G. et al. Regularization of extreme learning machines with information of spatial relations of the projected data. In: *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*. [S.l.: s.n.], 2019. p. 593–597.

ARTAC, M.; JOGAN, M.; LEONARDIS, A. Incremental PCA for on-line visual learning and recognition. In: *Object recognition supported by user interaction for service robots*. [S.l.: s.n.], 2002. v. 3, p. 781–784 vol.3.

BARBAKH, W.; FYFE, C. Online clustering algorithms. *International Journal of Neural Systems*, v. 18, n. 03, p. 185–194, 2008.

BARROS, R. S. M.; SANTOS, S. G. T. C. A large-scale comparison of concept drift detectors. *Information Sciences*, v. 451-452, p. 348 – 370, 2018.

BARROS, R. S. M. de; HIDALGO, J. I. G.; CABRAL, D. R. de L. Wilcoxon rank sum test drift detector. *Neurocomputing*, v. 275, p. 1954 – 1963, 2018.

BARTLETT, P. L. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, v. 44, n. 2, p. 525–536, 1998.

BEN-ISRAEL, A.; GREVILLE, T. N. *Generalized inverses: theory and applications*. [S.l.]: Springer Science & Business Media, 2003. v. 15.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, v. 13, n. Feb, p. 281–305, 2012.

BLUM, A. On-line algorithms in machine learning. In: *Online algorithms*. [S.l.]: Springer, 1998. p. 306–325.

BUDIMAN, A.; FANANY, M. I.; BASARUDDIN, C. Adaptive online sequential ELM for concept drift tackling. *Computational Intelligence and Neuroscience*, v. 2016, 2016.

CAO, L.; HUANG, W.; SUN, F. Building feature space of extreme learning machine with sparse denoising stacked-autoencoder. *Neurocomputing*, v. 174, p. 60–71, 2016.

CAO, W. et al. Fuzziness-based online sequential extreme learning machine for classification problems. *Soft Computing*, v. 22, n. 11, p. 3487–3494, 2018.

CHAPELLE, O.; SCHÖLKOPF, B.; ZIEN, A. *Semi-Supervised Learning*. [S.l.]: MIT Press, 2006.

CHAWLA, N. V. et al. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, v. 16, p. 321–357, 2002.

CHEN, X.; WANG, T. Combining active learning and semi-supervised learning by using selective label spreading. In: *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. [S.l.: s.n.], 2017. p. 850–857.

CORMEN, T. H. et al. *Introduction to algorithms*. [S.l.]: MIT press, 2009.

COSTA, F. G. da; RIOS, R. A.; MELLO, R. F. de. Using dynamical systems tools to detect concept drift in data streams. *Expert Systems with Applications*, v. 60, p. 39–50, 2016.

DAGHER, I. Incremental PCA-LDA algorithm. In: *2010 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*. [S.l.: s.n.], 2010. p. 97–101.

DERRAC, J. et al. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, v. 1, n. 1, p. 3–18, 2011.

DING, S. et al. Kernel based online learning for imbalance multiclass classification. *Neurocomputing*, v. 277, p. 139 – 148, 2018.

DUA, D.; GRAFF, C. *UCI Machine Learning Repository*. 2019. Disponível em: <http://archive.ics.uci.edu/ml>.

DUPRE, R. et al. Improving dataset volumes and model accuracy with semi-supervised iterative self-learning. *IEEE Transactions on Image Processing*, p. 1–1, 2019.

ENSOR, K. B.; GLYNN, P. W. Stochastic optimization via grid search. *Lectures in Applied Mathematics*, v. 33, p. 89–100, 1997.

FAN, R.-E.; LIN, C.-J. *LIBSVM Data: Classification, Regression, and Multi-label*. 2019. Disponível em: <https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>.

FIGO, D. et al. Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, v. 14, n. 7, p. 645–662, 2010.

GAMA, J. a. et al. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, v. 46, n. 4, p. 1–37, 2014.

GANDOMI, A.; HAIDER, M. Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, v. 35, n. 2, p. 137 – 144, 2015.

GOLUB, G. H.; LOAN, C. F. van. *Matrix Computations*. 3. ed. [S.l.]: The Johns Hopkins University Press, 1989. (Johns Hopkins Series in the Mathematical Sciences).

GU, Y. et al. Semi-supervised deep extreme learning machine for wi-fi based localization. *Neurocomputing*, v. 166, p. 282–293, 2015.

GUO, W. et al. Online sequential extreme learning machine with generalized regularization and adaptive forgetting factor for time-varying system prediction. *Mathematical Problems in Engineering*, v. 2018, 2018.

HAN, F.; YAO, H.-F.; LING, Q.-H. An improved evolutionary Extreme Learning Machine based on particle swarm optimization. *Neurocomputing*, v. 116, p. 87–93, 2013.

HE, X.; CAI, D.; NIYOGI, P. Laplacian score for feature selection. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2006. p. 507–514.

HORTA, E. G.; CASTRO, C. L. d.; BRAGA, A. P. Stream-based extreme learning machine approach for big data problems. *Mathematical Problems in Engineering*, v. 2015, 2015.

HSU, C.-W.; CHANG, C.-C.; LIN, C.-J. A practical guide to support vector classification (technical report). *Department of Computer Science and Information Engineering, National Taiwan University, Taipei*, 2003. Disponível em: <www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>.

HUANG, G. et al. Semi-supervised and unsupervised Extreme Learning Machines. *IEEE Transactions on Cybernetics*, v. 44, n. 12, p. 2405–2417, 2014.

HUANG, G.-B. et al. Extreme Learning Machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, v. 42, n. 2, p. 513–529, 2012.

HUANG, G.-B.; ZHU, Q.-Y.; SIEW, C.-K. Extreme Learning Machine: a new learning scheme of feedforward neural networks. In: *IEEE International Joint Conference on Neural Networks*. [S.l.: s.n.], 2004. v. 2, p. 985–990.

HUANG, G.-B.; ZHU, Q.-Y.; SIEW, C.-K. Extreme Learning Machine: theory and applications. *Neurocomputing*, v. 70, n. 1, p. 489–501, 2006.

HUYNH, H. T.; WON, Y. Regularized online sequential learning algorithm for single-hidden layer feedforward neural networks. *Pattern Recognition Letters*, v. 32, n. 14, p. 1930–1935, 2011.

HYDE, R.; ANGELOV, P.; MACKENZIE, A. Fully online clustering of evolving data streams into arbitrarily shaped clusters. *Information Sciences*, v. 382-383, p. 96–114, 2017.

JIA, X. et al. A semi-supervised online sequential Extreme Learning Machine method. *Neurocomputing*, v. 174, p. 168–178, 2016.

JING, L.; TIAN, Y. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

KANG, Q. et al. A noise-filtered under-sampling scheme for imbalanced classification. *IEEE Transactions on Cybernetics*, v. 47, n. 12, p. 4263–4274, 2017.

KASUN, L. L. C. et al. Representational learning with extreme learning machine for big data. *IEEE Intelligent Systems*, v. 28, n. 6, p. 31–34, 2013.

KINGMA, D. P. et al. Semi-supervised learning with deep generative models. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2014. p. 3581–3589.

KRISHNASAMY, G.; PARAMESRAN, R. Hessian semi-supervised Extreme Learning Machine. *Neurocomputing*, v. 207, p. 560–567, 2016.

KROHLING, R. A. Private communication, PPGI/UFES, 2020.

LANEY, D. *3D Data Management: Controlling Data Volume, Velocity and Variety*. 2001. Disponível em: <http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf>.

LAW, M. H. C.; JAIN, A. K. Incremental nonlinear dimensionality reduction by manifold learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 28, n. 3, p. 377–391, 2006.

LI, Y. Deep Reinforcement Learning. *arXiv e-prints*, 2018.

LIANG, N.-Y. et al. A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, v. 17, n. 6, p. 1411–1423, 2006.

LIN, J. et al. Iterative incremental clustering of time series. In: *Advances in Database Technology - EDBT 2004*. [S.l.]: Springer Berlin Heidelberg, 2004. p. 106–122.

LIN, W.-C. et al. Clustering-based undersampling in class-imbalanced data. *Information Sciences*, v. 409-410, p. 17 – 26, 2017.

LIU, D.; WU, Y.; JIANG, H. FP-ELM: An online sequential learning algorithm for dealing with concept drift. *Neurocomputing*, v. 207, p. 322–334, 2016.

LUNDQVIST, D.; FLYKT, A.; ÖHMAN, A. *The Karolinska directed emotional faces (KDEF)*. [S.l.]: Karolinska Institutet, 1998. CD ROM from Department of Clinical Neuroscience, Psychology section, Karolinska Institutet.

MA, L. et al. Graph-based semi-supervised learning for spectral-spatial hyperspectral image classification. *Pattern Recognition Letters*, v. 83, p. 133–142, 2016.

MAALØE, L. et al. Improving semi-supervised learning with auxiliary deep generative models. In: *NIPS Workshop on Advances in Approximate Bayesian Inference*. [S.l.: s.n.], 2015.

MICHE, Y. et al. SOM-ELM - Self-organized clustering using ELM. *Neurocomputing*, v. 165, p. 238–254, 2015.

NARAYANASWAMY, S. et al. Learning disentangled representations with semi-supervised deep generative models. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2017. p. 5925–5935.

NENE, S. A.; NAYAR, S. K.; MURASE, H. Columbia Object Image Library (COIL-100). Technical report CUCS-006-96, 1996.

NENE, S. A.; NAYAR, S. K.; MURASE, H. Columbia Object Image Library (COIL-20). Technical report CUCS-005-96, 1996.

PACHECO, A. G.; KROHLING, R. A.; SILVA, C. A. da. Restricted boltzmann machine to determine the input weights for extreme learning machines. *Expert Systems with Applications*, v. 96, p. 77 – 85, 2018.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

PHAM, D. T.; DIMOV, S. S.; NGUYEN, C. D. An incremental k-means algorithm. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, v. 218, n. 7, p. 783–795, 2004.

POZZOLO, A. D.; CAELEN, O.; BONTEMPI, G. When is undersampling effective in unbalanced classification tasks? In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. [S.l.: s.n.], 2015. p. 200–215.

PRAKASH, V. J.; NITHYA, D. L. M. A Survey on Semi-Supervised Learning Techniques. *arXiv e-prints*, p. 1–5, 2014.

PREECE, S. J. et al. A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data. *IEEE Transactions on Biomedical Engineering*, v. 56, n. 3, p. 871–879, 2008.

PREL, J.-B. D. et al. Choosing statistical tests: part 12 of a series on evaluation of scientific publications. *Deutsches Ärzteblatt International*, v. 107, n. 19, p. 343, 2010.

PRESS, W. H. et al. *Numerical Recipes: The Art of Scientific Computing*. 3. ed. New York, NY, USA: Cambridge University Press, 2007.

QIAO, S. et al. Deep co-training for semi-supervised image recognition. In: *The European Conference on Computer Vision (ECCV)*. [S.l.: s.n.], 2018. p. 135–152.

QIU, J. et al. A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*, v. 2016, n. 1, 2016.

REISS, A.; STRICKER, D. Creating and benchmarking a new dataset for physical activity monitoring. In: *Proceedings of the 5th International Conference on Pervasive Technologies Related to Assistive Environments*. [S.l.]: ACM, 2012. (PETRA '12), p. 40:1–40:8.

ROGGEN, D. et al. Wearable computing: Designing and sharing activity-recognition systems across platforms. *IEEE Robotics and Automation Magazine*, v. 18, n. 2, p. 83–95, 2011.

ROH, Y.; HEO, G.; WHANG, S. E. A survey on data collection for machine learning: a big data-ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

ROMASZEWSKI, M.; GŁOMB, P.; CHOLEWA, M. Semi-supervised hyperspectral classification from a small number of training samples using a co-training approach. *ISPRS Journal of Photogrammetry and Remote Sensing*, v. 121, p. 60–76, 2016.

RUMELHART, D. E. et al. Learning representations by back-propagating errors.

Cognitive modeling, v. 5, n. 3, p. 1, 1988.

SAAR-TSECHANSKY, M.; PROVOST, F. Handling missing values when applying classification models. *Journal of Machine Learning Research*, v. 8, p. 1623–1657, 2007.

SÁEZ, J. A.; KRAWCZYK, B.; WOŹNIAK, M. Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets. *Pattern Recognition*, v. 57, p. 164–178, 2016.

SAMIAPPAN, S.; MOORHEAD, R. J. Semi-supervised co-training and active learning framework for hyperspectral image classification. In: *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. [S.l.: s.n.], 2015. p. 401–404.

SAWANT, S. S.; PRABUKUMAR, M. A review on graph-based semi-supervised learning methods for hyperspectral image classification. *The Egyptian Journal of Remote Sensing and Space Science*, 2018.

SETHI, T. S.; KANTARDZIC, M. On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications*, v. 82, p. 77–99, 2017.

SHALEV-SHWARTZ, S. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, v. 4, n. 2, p. 107–194, 2012.

SILVA, C. A. S. da; KROHLING, R. A. Classificação de grandes bases de dados utilizando algoritmo de Máquina de Aprendizado Extremo. In: SBC. *Proceedings of XLVIII SBPO - Brazilian Symposium on Operational Research*. [S.l.], 2016. p. 1806–1815.

SILVA, C. A. S. da; KROHLING, R. A. Semi-Supervised Online Elastic Extreme Learning Machine for Data Classification. In: *IEEE IJCNN - International Joint Conference on Neural Networks*. [S.l.: s.n.], 2018. p. 1511–1518.

SILVA, C. A. S. da; KROHLING, R. A. Semi-Supervised Online Elastic Extreme Learning Machine with Forgetting Parameter to deal with concept drift in data streams. In: *IEEE IJCNN - International Joint Conference on Neural Networks*. [S.l.: s.n.], 2019. p. 1–8.

SILVA, C. A. S. da; KROHLING, R. A. Semi-supervised classification of non-stationary data streams with extreme learning machine-based algorithms. *Expert Systems with Applications*, 2020. (Under review).

SIMPSON, S. H. Creating a data analysis plan: What to consider when choosing statistics for a study. *The Canadian Journal of Hospital Pharmacy*, v. 68, n. 4, p. 311, 2015.

SMALE, S.; YAO, Y. Online learning algorithms. *Foundations of Computational Mathematics*, v. 6, n. 2, p. 145–170, 2006.

SU, H. et al. Interactive cell segmentation based on active and semi-supervised learning. *IEEE Transactions on Medical Imaging*, v. 35, n. 3, p. 762–777, 2016.

SUBRAMANYA, A.; TALUKDAR, P. P. Graph-based semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, v. 8, n. 4, p. 1–125, 2014.

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: MIT press, 2018.

Takahashi, C. C.; Torres, L. C. B.; Braga, A. P. Gabriel graph transductive approach to dataset shift. In: *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*. [S.l.: s.n.], 2019. p. 1622–1627.

TANHA, J.; SOMEREN, M. van; AFSARMANESH, H. Semi-supervised self-training for decision tree classifiers. *International Journal of Machine Learning and Cybernetics*, v. 8, n. 1, p. 355–370, 2017.

Torres, L. C. B. et al. Large margin gaussian mixture classifier with a gabriel graph geometric representation of data set structure. *IEEE Transactions on Neural Networks and Learning Systems*, p. 1–7, 2020.

TRIGUERO, I.; GARCÍA, S.; HERRERA, F. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information systems*, v. 42, n. 2, p. 245–284, 2015.

TSYMBAL, A. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, v. 106, n. 2, p. 58, 2004.

VANSCHOREN, J. et al. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations*, v. 15, n. 2, p. 49–60, 2013.

WANG, H.; ABRAHAM, Z. Concept drift detection for streaming data. In: *International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2015. p. 1–9.

WANG, J.; ZHAO, P.; HOI, S. C. H. Cost-sensitive online classification. *IEEE Transactions on Knowledge and Data Engineering*, v. 26, n. 10, p. 2425–2438, 2014.

WU, D. et al. Self-training semi-supervised classification based on density peaks of data. *Neurocomputing*, v. 275, p. 180–191, 2018.

XIN, J. et al. Elastic Extreme Learning machine for big data classification. *Neurocomputing*, v. 149, p. 464–471, 2015.

XU, S.; WANG, J. Dynamic Extreme Learning Machine for data stream classification. *Neurocomputing*, v. 238, p. 433–449, 2017.

YAN, J. et al. A scalable supervised algorithm for dimensionality reduction on streaming data. *Information Sciences*, v. 176, n. 14, p. 2042 – 2065, 2006.

YANG, Z.; COHEN, W. W.; SALAKHUTDINOV, R. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016.

YAO, Y. On complexity issues of online learning algorithms. *IEEE Transactions on Information Theory*, v. 56, n. 12, p. 6470–6481, 2010.

YING, Y.; PONTIL, M. Online gradient descent learning algorithms. *Foundations of Computational Mathematics*, v. 8, n. 5, p. 561–596, 2008.

YING, Y.; ZHOU, D.-X. Online regularized classification algorithms. *IEEE Transactions on Information Theory*, v. 52, n. 11, p. 4775–4788, 2006.

ZHAN, W.; ZHANG, M.-L. Inductive semi-supervised multi-label learning with co-training. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. [S.l.]: ACM, 2017. (KDD '17), p. 1305–1314.

ZHAO, J.; WANG, Z.; PARK, D. S. Online sequential extreme learning machine with forgetting mechanism. *Neurocomputing*, v. 87, p. 79–89, 2012.

ZHENG, Z.; CAI, Y.; LI, Y. Oversampling method for imbalanced classification. *Computing and Informatics*, v. 34, n. 5, p. 1017–1037, 2016.

Zhou, H. et al. Stacked extreme learning machines. *IEEE Transactions on Cybernetics*, v. 45, n. 9, p. 2013–2025, 2015.

ZHU, X. Semi-supervised learning. *Encyclopedia of Machine Learning and Data Mining*, p. 1142–1147, 2017.

APPENDIX A – Time Complexity of SSOE-ELM

For many classification problems, the total number of training samples N and the number of neurons n are the highest values, generally greater than the number of features d or the number of classes c. Therefore, to compute the ELM time complexity, the order of importance of the parameters from the most relevant to the least relevant is: number of training samples and number of neurons; number of features; number of classes.

Considering the dimensions of the input matrices $X_{[N\times d]}$, $W_{[d\times n]}$ and $T_{[N\times c]}$, and the greater impact of parameters N and n on computational time, Table 29 presents the computational complexity of each operation in Standard ELM, and the final complexity, according to Cormen et al. (2009). It is important to note that in classification problems with a large number of training samples, the term n^2N gains greater importance in the calculation of the final time complexity. In problems where the number of neurons n is greater than the number of training samples N, the term n^3 stands out. Both operations are performed only once, during β computation.

Table 29 – Standard ELM computational complexity, based on Akusok et al. (2015)

Projection to hidden layer (H matrix computation)	
Operation	Time Complexity
$H_{[N\times n]} = f(XW + b)$	$O(Ndn)$
β matrix computation	
Operation	Time Complexity
$A_{[n\times n]} = H^T H$	$O(n^2 N)$
$B_{[n\times c]} = H^T T$	$O(nNc)$
$C_{[n\times n]} = \alpha I + A$	$O(n^2)$
$D_{[n\times n]} = C^{-1}$	$O(n^3)$
$\beta_{[n\times c]} = (\alpha I + H^T H)^{-1} H^T T = DB$	$O(n^2 c)$
Final Time Complexity	
$O(n^2 N + n^3)$	

In the SSOE-ELM algorithm, which performs the training process in an online manner, the total number of training samples N is replaced by the size of each training partition N_k in the input matrices dimensions and in the calculations performed. Generally, the size of training partitions is smaller than the number of neurons, making parameter n the most important when calculating computational time of SSOE-ELM.

Table 30 presents the computational complexity of each operation in SSOE-ELM, and the final complexity. In addition to the standard ELM operations, SSOE-ELM

computes the intermediate matrices L, J (for semi-supervised learning), U and V (for online update) for each training partition. It is important to note that the β computation (which is the operation with the highest computational cost) only needs to be performed once at the end of the training. Therefore, the computational time of the SSOE-ELM remains in the same order of magnitude as the Standard ELM.

Table 30 – SSOE-ELM computational complexity for each training partition k, based on Akusok et al. (2015)

Projection to hidden layer (H matrix computation)	
Operation	Time Complexity
$H_{k[N_k \times n]} = f(XW + b)$	$O(N_k dn)$
Computation of Laplacian matrix L_k and Penalty matrix J_k	
Operation	Time Complexity
Find μ nearest neighbors in H	$O(nN_k\mu)$
$A_{k[N_k \times N_k]}[i,j] = adj(x_i, x_j)$	$O(N_k^2)$
$D_{k[N_k \times N_k]}[i,i] = \sum_{j=1}^{N_k} A_k[i,j]$	$O(N_k^2)$
$L_{k[N_k \times N_k]} = D_k$ - A_k	$O(N_k^2)$
$J_{k[N_k \times N_k]}[i,i] = C/N_{t_i}$	$O(N_k)$
Computation of intermediate matrices U_k and V_k	
Operation	Time Complexity
$M_{1[N_k \times N_k]} = J_k + \lambda_k L_k$	$O(N_k^2)$
$M_{2[n \times N_k]} = H_k^T M_1$	$O(nN_k^2)$
$\Delta U_{[n \times n]} = H_k^T (J_k + \lambda_k L_k) H_k = M_2 H_k$	$O(n^2 N_k)$
$M_{3[n \times N_k]} = H_k^T J_k$	$O(nN_k^2)$
$\Delta V_{[n \times c]} = H_k^T J_k T_k = M_3 T_k$	$O(nN_k c)$
$U_{k[n \times n]} = U_{k-1} + \Delta U$	$O(n^2)$
$V_{k[n \times c]} = V_{k-1} + \Delta V$	$O(nc)$
β matrix computation	
Operation	Time Complexity
$M_{4[n \times n]} = \alpha I + U$	$O(n^2)$
$M_{5[n \times n]} = M_4^{-1}$	$O(n^3)$
$\beta_{[n \times c]} = (\alpha I + U)^{-1} V = M_5 V$	$O(n^2 c)$
Final Time Complexity	
Operation	Time Complexity
At each training partition	$O(n^2 N_k)$
At the end of the training process	$O(n^2 N_k + n^3)$

*M_1, M_2, M_3, M_4 and M_5 are temporary matrices

APPENDIX B – Time Complexity of SSOE-FP-ELM

The SSOE-FP-ELM algorithm is based on SSOE-ELM, so the operations of the semi-supervised learning and online update steps are the same for both algorithms. The difference between the two proposed algorithms, in terms of computational time, is the calculation of the semi-supervised forgetting parameter in the SSOE-FP-ELM.

Table 31 presents the computational complexity of each operation in SSOE-FP-ELM. SSOE-FP-ELM has a longer training time than the SSOE-ELM, due to the supervised factor of the forgetting parameter. To assess the accuracy of the model from one training partition to another and check whether a label concept drift has occurred, it is necessary to calculate the β output weights matrix, and this is the operation with the highest computational cost. Thus, at each training partition the SSOE-FP-ELM needs to carry out all operations (calculate the intermediate matrices L, J, U and V, compute the β matrix and calculate the forgetting parameter), different from the SSOE-ELM that does not need to calculate β.

Table 31 – SSOE-FP-ELM computational complexity for each training partition k, based on Akusok et al. (2015)

Projection to hidden layer (H matrix computation)	
Operation	Time Complexity
$H_{k[N_k \times n]} = f(XW + b)$	$O(N_k dn)$
Computation of Laplacian matrix L_k and Penalty matrix J_k	
Operation	Time Complexity
Find μ nearest neighbors in H	$O(nN_k\mu)$
$A_{k[N_k \times N_k]}[i,j] = adj(x_i, x_j)$	$O(N_k^2)$
$D_{k[N_k \times N_k]}[i,i] = \sum_{j=1}^{N_k} A_k[i,j]$	$O(N_k^2)$
$L_{k[N_k \times N_k]} = D_k$ - A_k	$O(N_k^2)$
$J_{k[N_k \times N_k]}[i,i] = C/N_{t_i}$	$O(N_k)$
Computation of Forgetting Parameter γ	
Operation	Time Complexity
Wilcoxon test in X_{k-1} and X_k	$O(dN_k)$
Laplacian scores $\rightarrow S_{[1\times d]}$	$O(dN_k^2)$
Normalized reversed Laplacian scores $\rightarrow S'_{[1\times d]}$	$O(d)$
Laplacian scores of changed inputs $\rightarrow \hat{S}_{[1\times d]}$	$O(d)$
Computation of intermediate matrices U_k and V_k	
Operation	Time Complexity
$M_{1[N_k \times N_k]} = J_k + \lambda_k L_k$	$O(N_k^2)$
$M_{2[n \times N_k]} = H_k^T M_1$	$O(nN_k^2)$
$\Delta U_{[n\times n]} = H_k^T(J_k + \lambda_k L_k)H_k = M_2 H_k$	$O(n^2 N_k)$
$M_{3[n \times N_k]} = H_k^T J_k$	$O(nN_k^2)$
$\Delta V_{[n\times c]} = H_k^T J_k T_k = M_3 T_k$	$O(nN_k c)$
$U_{k[n\times n]} = (1-\gamma)U_{k-1} + \Delta U$	$O(n^2)$
$V_{k[n\times c]} = (1-\gamma)V_{k-1} + \Delta V$	$O(nc)$
β matrix computation	
Operation	Time Complexity
$M_{4[n\times n]} = \alpha I + U$	$O(n^2)$
$M_{5[n\times n]} = M_4^{-1}$	$O(n^3)$
$\beta_{[n\times c]} = (\alpha I + U)^{-1} V = M_5 V$	$O(n^2 c)$
Final Time Complexity	
Operation	Time Complexity
At each training partition	$O(n^2 N_k + n^3)$
At the end of the training process	$O(n^2 N_k + n^3)$

*M_1, M_2, M_3, M_4 and M_5 are temporary matrices