Algorithms for ab intio and large scale prediction and classification of ncRNAs

Ludovic Platon

université PARIS-SACLAY

BNT: 2019SACLE003

Thèse de doctorat

Algorithms for *ab initio* identification and classification of ncRNAs

Thèse de doctorat de l'Université Paris-Saclay,
préparée à l'université d'Evry-Val-d'Essonne

École doctorale n° 580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité thèse: Informatique

Thesis presented at 2 pm, the 30 January 2019, by

Ludovic Platon

Jury composition:

Christine Gaspin Research director — INRA - MIA	Referee
Pierre Geurts Associate professor — Université de Liège	Referee
Younes Bennani Professor — Université Paris XIII	Examinator
Daniel Gautheret Professor — Université d'Orsay, Paris Saclay	President
Jean-Daniel Zucker Research director — IRD	Examinator
Fariza Tahi Associate professor - HDR — Université d'Evry, Paris Saclay	Supervisor
Abdelhafid Bendahmane Research director — INRA - IPS2	Co-supervisor
Farida Zehraoui Associate professor — Université d'Evry, Paris Saclay	Co-supervisor

Je tiens à remercier tout d'abord ma directrice de thèse Madame Fariza Tahi, maître de conférences à l'Université d'Évry, qui m'a encadré et conseillé durant cette thèse. Durant ces trois années, j'ai pu profiter de son expérience en bio-informatique et sur les ARN non-codants ainsi que de conseil pour gérer la charge de travail d'une thèse.

Je remercie mon co-directeur de thèse, Monsieur Abdelhafid Bendahmane, directeur de recherche à l'IPS2, qui m'a permis d'améliorer ma compréhension biologique des ARN non-codants, de la biologie des plantes ainsi que ses retours pertinents sur les travaux accomplis.

Je tiens également à remercier madame Farida Zehraoui, maître de conférences à l'Université d'Évry. Son expérience des réseaux de neurones, des cartes-organisatrices et des techniques d'apprentissage automatique m'ont été utile pour le développement de nos méthodes.

Grâce à ces trois personnes que j'ai pu avoir ma première expérience de chercheur et mener ma thèse à terme. Durant ces trois années j'ai pu m'appuyer également sur les membres de l'IBISC et de l'IPS2.

Je tiens à remercier l'équipe AROBAS et les doctorants de l'IBISC pour leurs échanges instructifs et leur aide apporté. Je souhaite remercier également les administrateurs systèmes Laurent Poligny ainsi que Ludovic Ishiomin qui m'ont été d'une grande d'aide lors de l'implémentation de mes algorithmes.

Je remercie tous les membres de l'équipe FLOCAD ainsi que Thomas Blein, Jérémie Bazin et Martin Crespi de l'équipe REGARN qui m'ont permis d'enrichir mes connaissances biologiques des plantes et plus particulièrement des ARN non-codants. Je tiens à remercier les membres du bureau 3.11 et plus particulièrement Joseph Tran ainsi que Vivien Sommard qui m'ont accompagné durant ces trois ans et aidé durant mes travaux.

Je souhaite aussi remercier ma famille et mes amis de m'avoir soutenu durant ma thèse. Ils m'ont permis de tenir durant les moments de doutes ainsi que dans les moments de grands stress. Je remercie plus particulièrement mon Père et ma Mère pour tout ce qu'ils ont fait pour moi. Je remercie également mon grand-père qui m'a toujours poussé à donner le meilleur de moi-même.

Contents

Glossary

CNN Convolutional Neural Network (CNN) is a type of neural network mostly used in imagery. 25

epoch In machine learning, an epoch is complete when all the input data is presented to the learning algorithm. 87

exon An exon is a part of a protein coding gene that is present in the mature mRNA. 17

Fickett Score pH value for which the molecule is electrically neutral. 22, 23

intragenic Element of process that occurs within a gene. 16

isoeletric pH value for which the molecule is electrically neutral. 22

k-mer frequencies Frequencies of all the words of size k in a string. 22, 25, 84, 85, 96, 97

lncRNA The long non-coding RNAs are a class of non-coding RNAs that are longer than 200 nucleotides. 9

MCL Markov Cluster Algorithm (MCL) is an unsupervised learning method based on random walk simulation in a graph. 25

MFE Minimum Free Energy (MFE) of an RNA is the free energy value of an RNA structure with the lowest free energy. 24

MKL Multiple Kernels Learning approach aims to optimize an objective function and a linear (or non-linear) combination of kernels. 82

MLP The Multiple Layer Perceptron (MLP) is a class of neural network with at least three layers, the input layer, one or more hidden layers and the output layer. 10, 11, 81–84, 87, 96, 97, 101

nucleolus The nucleolus is the largest structure of the nucleus in eukaryotic cells and are made of proteins, DNA and RNA. 16

ORF Open Reading Frame (ORF) is part of an RNA that can be translated into a protein. 22, 23, 96, 97

phenotype The phenotype of a living organism is its set of observable characteristics such as its length, color and size. 101

RF The Random Forest is an ensemble method that relies on decision trees. In this approach, multiple decision trees are trained on different subsets of the training dataset with different features, and the prediction is done by performing an averaging of a majority vote on the prediction trees. 23, 25

SVM Support Vector Machine (SVM) is a wide used supervised classifier. This classifier computes an hyperplane, which maximizes the largest separation "margin" of two classes. 23, 25, 82

transposon A transposable element is a DNA sequence capable to move in a genome. 16, 97

Introduction

General introduction

RNAs (also called transcripts) are the results of the DNA transcription. We can distinguish two classes of RNAs, coding and non-coding RNAs. The coding RNAs or messenger RNAs (mRNAs) are processed during the translation process to produce proteins. The non-coding RNAs (ncRNAs) usually don't encode proteins but have important roles in biology. They are involved in multiple biological processes such as the transcription and the translation. They are also implied in the sex determinism [105, 66, 3] or in disease such as cancers [8, 58]. The identification of ncRNAs and their biological functions help the comprehension of the biological processes where ncRNAs are involved.

The identification and classification of RNAs is a difficult task due to two major issues. The first one is the unclear separation between coding and non-coding RNAs. In [94], the authors show the existence of long non-coding RNAs (lncRNAs) that encode small peptides. The second issue is the unsettled classification of ncRNAs. In [17, 112], the authors enumerate respectively a lot of ncRNA classes. These class definitions raise two classification problems. The first one is related to the data available to create a reliable model. For a given species, certain classes might not contain enough known elements. The second problem is the ambiguous definition of the classes. In [112], the authors give the example of the RNA ANRIL which is a lncRNA, a NAT (antisense transcript) and a circular RNA. Moreover, the current classification is subject to modifications, because we don't know all the ncRNA classes. With further analysis of the ncRNAs with new sources of data, the definition and classification of the ncRNAs will be improved.

There are two levels of ncRNA classification (see Figure 1). The first one is the discrimination between coding and non-coding RNAs. The second one is the identification of the ncRNA sub-classes.

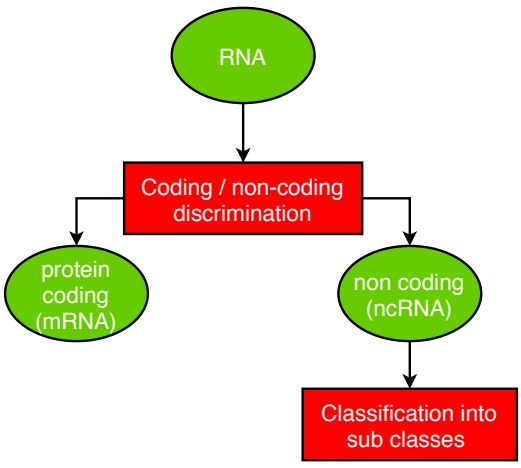


Figure 1: Classification levels of ncRNAs.

At the first level there are tools such as CONC [82], CPC [69], PORTRAIT [7], COME [54], CNCI [115], CPAT [127], iSeeRNA[113], PhyloCSF [115], PLEK [77], lncRNA-MFDL [33], lncRScan-svm [114], DeepLNC [120], CPC2 [62], PLncPRO [109] and BASiNET [57]. At the second level, we can identify two types of tools. The first ones focus on the identification of one ncRNA class. For example, there are miRNAFold [119, 117], mirBoost [118] or deepTarget [73] which predict miRNAs and piRPred [16] or IpiRId [14] which predict piRNAs. The second ones try to identify multiple classes of ncRNAs. There are ALPS [31], BlockClust [122], CoRAL [76], and DARIO [34] which use deep-sequencing data (RNA-seq data) and GraPPLe [20], nRC [35], RNAcon [93] that use secondary structure prediction data.
In the context of this thesis, we are interested in the ones that discriminate coding and non-coding RNAs and the ones that classify ncRNAs into sub classes. But these tools have two drawbacks currently. Their first drawback is their learning algorithm. They all use supervised learning (except for BlockClust) to compute their model and thus are designed to identify only the learned classes. The second one is their sources of data also called features. Most of them use only one source of data and thus have low prediction performance for some classes of ncRNAs that need multiple sources of data. In [112], the authors show how the ncRNAs can be classified by their association with other biological components, their resemblance with mRNA, the biological pathways or the secondary structures for example.
The ncRNAs classification state-of-art raises two major methodological issues. The first one is the classification of known classes and the identification of potential new classes. Currently the identification of new ncRNA classes is not covered by the state-of-art that focuses on the most common ncRNA classes. In this thesis, we combine supervised and unsupervised learning methods. With the supervised learning we can identify the learned classes and with the unsupervised learning we can identify potential new classes. The second methodological issue is related to the data. We need to extract valuable information from heterogeneous data sources and combine them. Due to the amount of data and their heterogeneity, the model needs to handle vectorial data and complex data. So a good classifier of ncRNAs needs to classify the known classes and discover potential new classes by using multiple heterogeneous data sources. Moreover, the method needs to be scalable and gives an useful representation of the data for further biological analysis.
To answer these issues, we developed a method relying on Self-Organizing Maps (SOM), perceptrons and rejection options. We use the SOM algorithm to analyse and represent the ncRNAs by a map of clusters where the topology of the data is preserved and so the related proximity of the ncRNAs is preserved. The developed approach is a neural network with at least 4 layers such that:

1. The first and second layers are composed of a specific SOM for each data source.
2. The third layer is a SOM that's fully connected to the SOMs in the second layer.
3. The last layers represent our Multi Layer Perceptrons (MLP) that is fully connected to the third layer.

We extended the output of the MLP with a rejection option which can reject the ambiguous prediction and identify the potential new classes of ncRNAs.
This neural network can be decomposed in two blocks, the unsupervised block and the supervised block. The unsupervised block, which we call Multiple Sources SOM (MSSOM) [100], is composed of the first, second and third layers. In MSSOM, we have a specific map for each data source. By doing so, MSSOM is able to handle complex data and numerical data without transformation. The different sources are combined in a final map where each cluster has its own data source weights vector.
The supervised block, which we call Supervised Layer SOM (SLSOM) [101] is composed of the third layer to the last layer and the rejection option. SLSOM is a combination of SOM and MLP where the perceptrons are

fully connected to a SOM. The MLP classifies the input data with the new representation produced by the SOM. Moreover, with this representation and the rejection option, we show how to identify ambiguous input data and also identify potential new classes.
Based on these methods, we implemented a first tool called IRSOM [103] which aims to discriminate coding and non-coding RNAs. By using a simple set of features, we show that IRSOM is able to separate the coding and non-coding RNAs effectively in a wide range of species coming from different reigns. With the rejection option, we also highlighted the ambiguous RNAs in a study case. The global tool which we call CRSOM for classifying the ncRNAs combines MSSOM and SLSOM in order to integrate all the data necessary for the classification of ncRNAs in subclasses.
To summarize, we propose a new method to identify and classify the ncRNAs. This method is able to identify known classes of ncRNAs and also to discover potential new classes by using different neural network algorithms and rejection options. We have shown and validated the results of the method in the case of the discrimination of coding and non-coding RNAs (IRSOM) and the classification of ncRNAs (CRSOM).

Plan

In this thesis, we first introduce the ncRNAs, their classes and their biological functions and then the Self-Organizing Maps (SOMs). Then we present in Chapter 1 the state-of-art of the tools used for the identification and classification of ncRNAs. We have organized the works according to the classification goal. The first part focuses on the identification of ncRNAs by discriminating coding and non-coding RNAs: we present SLSOM in Chapter 2 and then IRSOM in Chapter 3. We present in the second part the classification: we present MSSOM in Chapter 4 and then in Chapter 5, we present CRSOM, a classifier of ncRNAs, based on a combination of MSSOM and SLSOM.

What is a ncRNA?

DNA is the support of all the genetic information of a living organism. In Figure 2 we can see how the DNA is processed to produce RNA and potentially a protein (called also polypeptide) from a gene. During the transcription process, the strand coding RNA (coding strand) is duplicated by using its complementary strand (template strand). In certain cases, the produced transcript goes into a maturation process. For example, the transcription of messenger RNA (mRNA) gene produces a pre-mRNA that will be matured into a mRNA. The same process can happen for non-coding RNAs (ncRNAs), for example the transcription of a microRNA (miRNA) gene produces a pri-miRNA that will be matured into a miRNA.

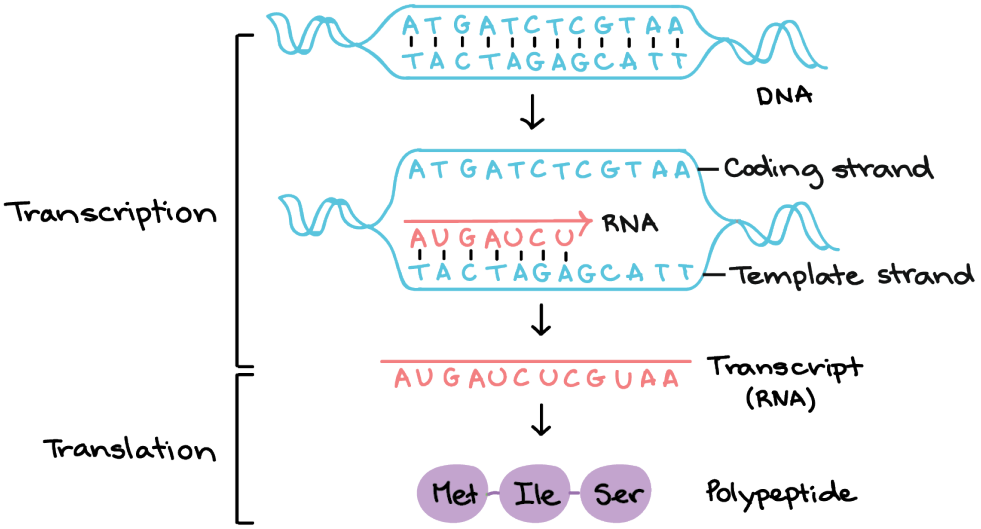


Figure 2: Transcription and translation process (Source: https://www.khanacademy.org/science/biology/gene-expression-central-dogma/transcription-of-dna-into-rna/a/stages-of-transcription).

The produced RNAs can be separated into two classes, the coding and the non-coding RNAs. The coding RNAs also called messenger RNAs (mRNAs) are the RNAs used during a translation process to produce proteins. The non-coding RNAs (ncRNAs) on contrary, are not processed into proteins but are used in other biological processes for their interactions with other biological components. For example the miRNAs (which are a sub class of ncRNA) bind to mRNA in order to avoid the translation process.
Figure 3 shows the main classes of ncRNAs studied in the literature and in the context of this thesis. We can distinguish three major categories of class types of ncRNAs: housekeeping ncRNAs, small ncRNAs and long ncRNAs.

Housekeeping ncRNAs

Housekeeping ncRNAs are one of the first classes of discovered ncRNAs. They are represented mostly by transfer RNAs (tRNAs) and ribosomal RNAs (rRNAs). These ncRNAs are involved in the translation process. The rRNAs are a part of the ribosomal complex and the tRNAs perform the amino acid transport. The rRNAs show high sequence conservation between species. It has also been shown that these ncRNAs have a conserved secondary structure and can be identified from their secondary structure. The sequence and secondary structure
The ribosomal complex is composed of two parts, the large subunit and the small subunit. In both subunits

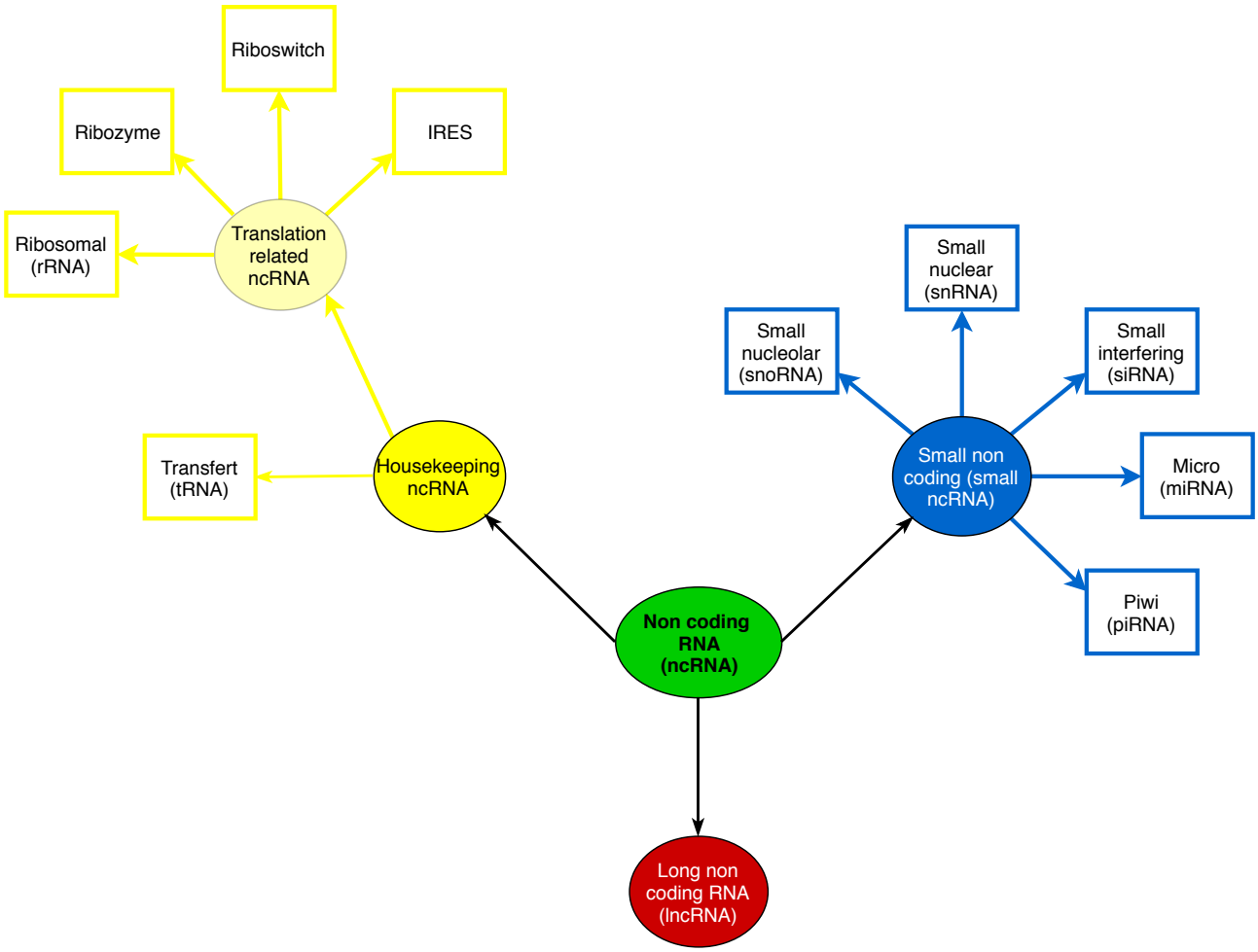


Figure 3: Principal classes of ncRNA.

we have different subtypes of rRNAs. For example, in prokaryotic organisms (bacteria and archaea), the large subunit is composed of 5S and 23S rRNA. The rRNAs have a well know secondary structure (see Figure 4). Moreover, the organization of the rRNAs in the ribosomal complex is conserved between species. As you see in Figure 4, the rRNAs have the same organization in the three species that are from three different reigns (bacteria, fungi and animal).

The tRNA is responsible of the amino acid transport. In order to achieve its function, the tRNAs have two major parts, the anticodon and the acceptor stem (see Figure 5). The anticodon is defined such that it recognizes a particular motif of three nucleotides that encodes the amino acid present on the acceptor stem.

Small ncRNAs

The five most studied small ncRNAs are the micro RNAs (miRNAs), the piwi RNAs (piRNAs), the small interfering RNAs (siRNAs), the small nuclear RNAs (snRNAs) and the small nucleolar RNAs (snoRNAs). The miRNAs, siRNAs and piRNAs are the smallest ones with a size around 20-24 nt. The small ncRNAs have a size ranging approximatively from 20 nucleotides (nt) to 300 nt.

The miRNAs are the most studied small ncRNAs. The canonical genesis of a mature miRNA is done in three stages. The first one is the creation of a pri-miRNA by the transcription process. The second one is the processing of the pri-miRNA by Drosha in order to produce a pre-miRNA. The third one is the cleavage of the pre-miRNA by Dicer which results in a mature miRNA. MiRNA precursors can be recognized thanks to their

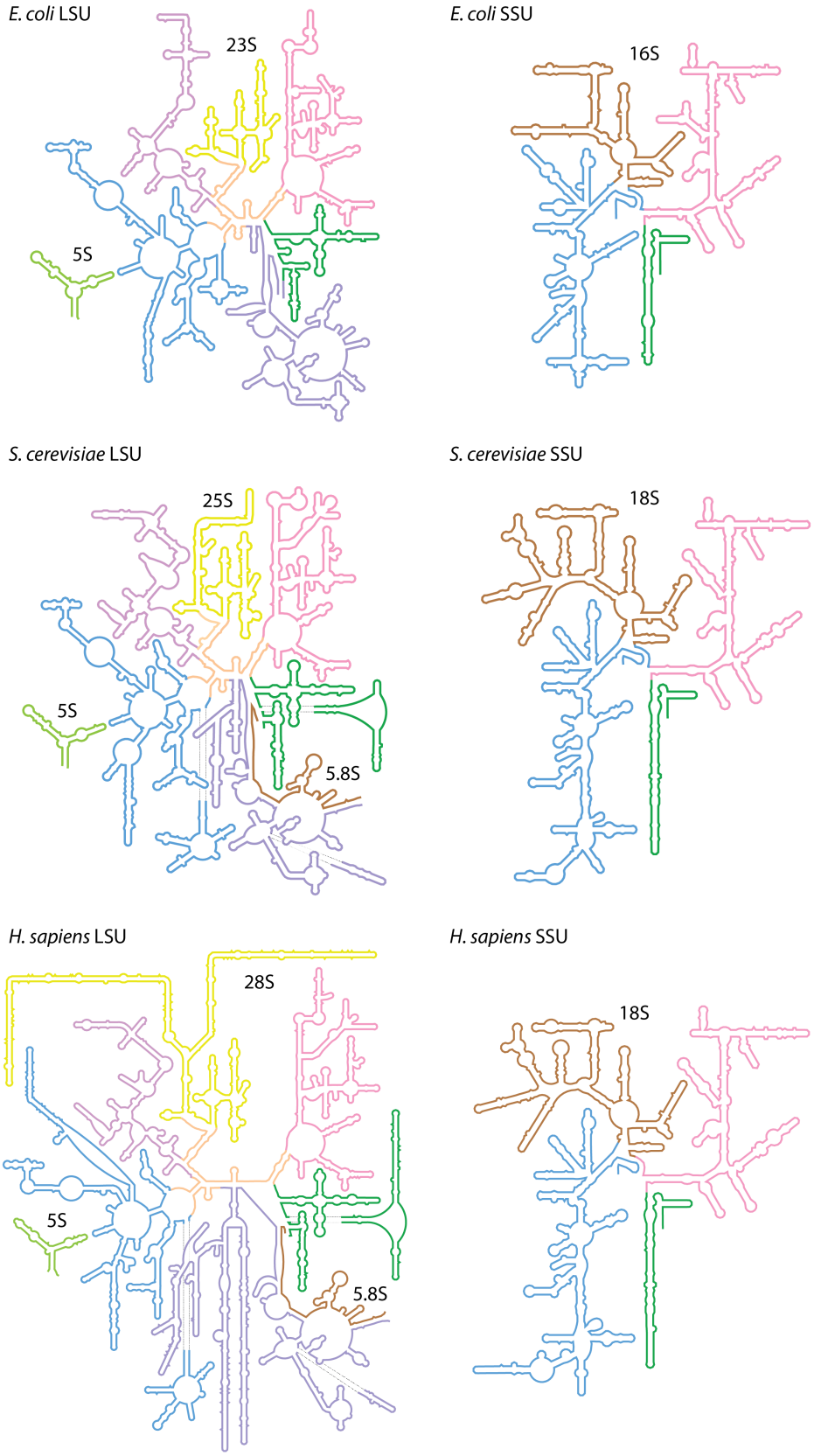


Figure 4: Secondary structure of rRNAs of *E. coli*, *S. cerevisiae* and Human. LSU stands for large subunit and SSU stands for small subunit. The different colors represent different domains in the rRNAs subunits (Source: [98]).

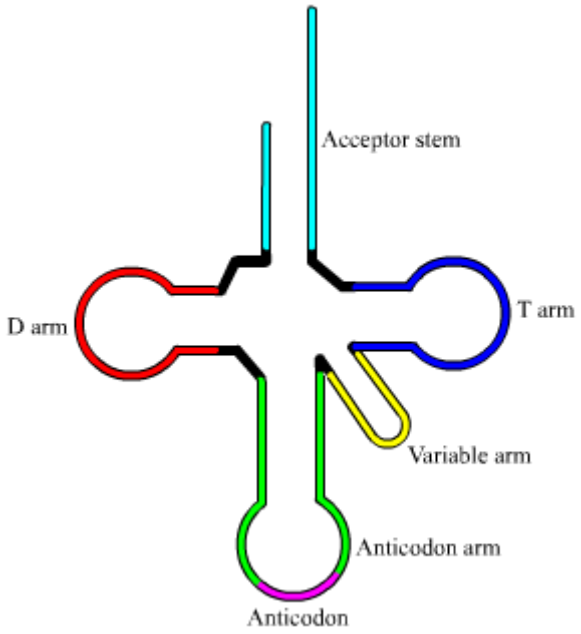


Figure 5: Secondary structure schema of a tRNA (Source: https://academickids.com/encyclopedia/index.php/Transfer_RNA).

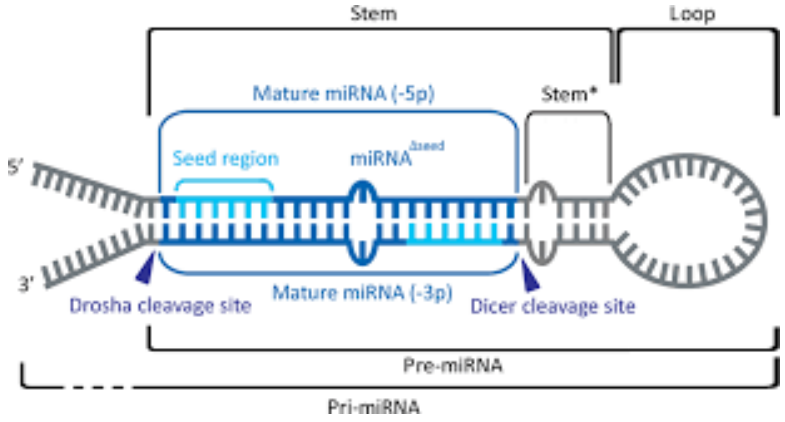


Figure 6: Secondary structure schema of a miRNA (Source: [59]).

secondary structure in the form of hairpin (see Figure 6). The miRNAs play a regulating role of the transcripts. By binding with these transcripts, they inhibit their translation into proteins.

The siRNAs and miRNAs are two classes of ncRNAs that share a lot of common characteristics. Both classes have approximately the same size, they are both processed by the Dicer proteins and have the same functions. Some scientists would say that both classes are just one class, but they show some noticeable differences. The siRNAs are brought in the cell by external sources such as viruses into the form of a double stranded RNA when the miRNAs represent a genuine component of the genome. Moreover, a siRNA binds to a specific transcript when a miRNA can bind to multiple transcripts and their precursors show different secondary structures.

The piRNAs, miRNAs and the siRNAs have comparable length (approximately 26 to 31 nt for the piRNAs and 21 to 24 for the miRNAs and the siRNAs) but have different roles. Their name is derived from then interaction with the PIWI proteins. The interaction of the piRNAs with PIWI proteins forms a complex which aims to repress the expression of transposon. The piRNAs have three noticeable characteristics. The first one is that their sequences commonly start with an uridine. The second one is that they are organized

into clusters in the genome. The third one is that they have a 2'-O-methyl modification at their end. However the piRNAs don't have secondary structure.
The snRNAs and snoRNAs are the longest small RNAs with a size of approximately 150 nt for the snRNAs and 60-300 nt for the snoRNAs. Both classes show specific secondary structure and are present in the nucleus (the snoRNAs are more precisely in the nucleolus). These classes have different biological functions, the snRNAs are mostly involved in the processing of pre-messenger RNAs when the snoRNAs guide the chemical modifications of the RNAs (methylation and pseudouridylation). Both classes don't show a specific secondary structure but the snRNAs are associated with particular proteins and the snoRNAs are associated with the 2'-O-methylation or pseudouridylation of RNAs.

Long ncRNAs

The long ncRNAs (lncRNAs) represent the ncRNA classes that are longer than 200 nt. The study of these classes is on going and thus, the classification of the lncRNAs undergoes some modifications. One of the first classifications of the lncRNAs was focused on their position relative to proteins coding genes [85, 6] (see Figure 7). This classification is composed of 7 classes of lncRNAs: the sense, antisense, intronic, promoter, intergenic, bidirectional and enhancer.
The sense, antisense, intronic and promoter lncRNAs are intragenic RNAs (RNAs that are transcribed from a gene DNA). Sense lncRNAs are transcribed from the coding strand in the same sense as the coding gene. The particularity of the sense lncRNAs is their overlap with at least one exon of the protein coding gene. Antisense lncRNAs are transcribed on the conjugate strand in both ways. The antisense lncRNAs can interact with the mRNA gene transcribed from the coding strand. This interaction is possible because the antisens lncRNAs are transcribed from the conjugate strand of the mRNA gene and thus possess a complementary sequence to the exon of the mRNA gene. The intronic lncRNAs are transcribed from the intronic region of the gene and contain only intronic sequences.
The promoter and enhancer lncRNAs are transcribed respectively from the promoter and enhancer regions of the gene. The bidirectional lncRNAs use the same promoter as the protein-coding genes. But the bidirectional lncRNAs are transcribed in the opposite direction. The intergenic lncRNAs are the only lncRNAs that are not close to a mRNA gene. In [85], the authors cite multiple works showing that the intergenic lncRNAs are conserved across multiple vertebrate species. The DNA regions coding the intergenic lncRNAs show the same transcription marker as the DNA regions coding mRNA. In order to be an intergenic lncRNA, a lncRNA needs to be at least 1kbp away from the closest gene. The 1kbp number is an arbitrary limit to separate the promoter and the intergenic lncRNA. This separation imply that the genes promoters are at most 1kbp away from the gene. But we know that some promoters such as the distal promoters can be located farther from the genes.
Currently, the classification of lncRNAs has changed. In [112], the authors present a review of lncRNA classes and categories of lncRNA classes. They define ten categories based on discriminative characteristics such as:

- length,
- association with annotated protein coding genes,
- association with DNA components (different than the genes),
- resemblance with RNA,
- association with repeats (repeated genes or repeated regions for example),
- association with biological pathway or stability,

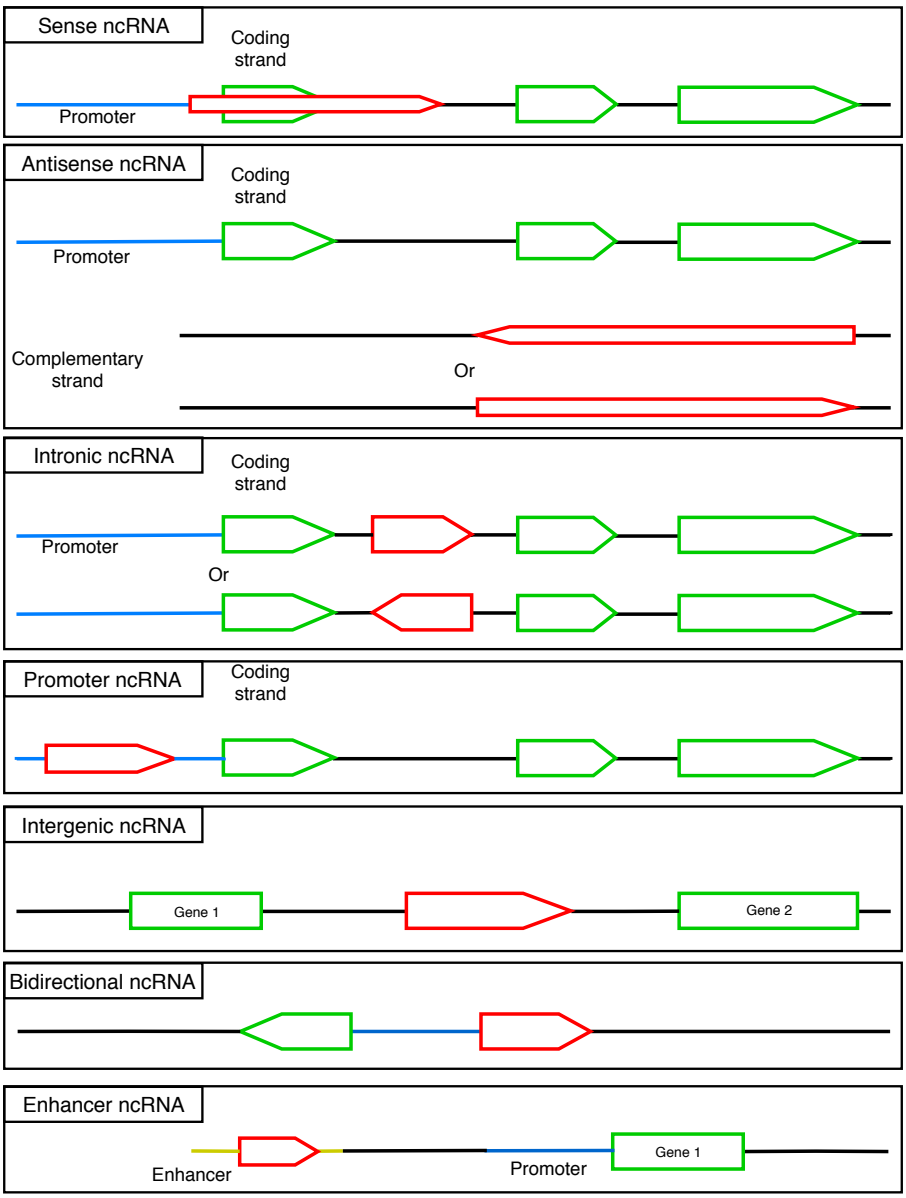


Figure 7: LncRNA classes based on genome position. The red boxes represent the non-coding genes coding for lncRNAs. The protein exons are represented by green boxes. The blue lines represent the promoters of the protein coding gene. The yellow line represents the enhancer of the protein coding genes. (Source: [6])

- sequence and structure conservation,
- expression in different biological states,
- association with sub cellular structures,
- function.

Each category of classes is composed of multiple lncRNA classes, with a total of 52 classes. The authors of [112] state that the actual classification is ambiguous. To support their point, they take the example of the ANRIL transcript. The ANRIL is a lncRNA, a NAT (antisense transcript) and a circular RNA. Moreover, some classes described in the literature do not give insight of the biological purpose of the lncRNAs. Currently, the lncRNAs classes represent a small group of ncRNA showing a common biological function or mechanism.

Why Self-Organizing Maps (SOM)?

Self-Organizing Map (SOM) [67] is a neural network that is able to cluster and visualize high dimensional data. SOM is a method related to the vector quantization (VQ) method such as the k-means or the Learning Vector Quantization (LVQ)[1]. The difference between the SOM and the other VQ like methods is its neurons organization. By using an unsupervised competitive learning algorithm, SOM is able to produce a map representing the input space. The produced map can have different structures but the most common one is a grid (rectangular or hexagonal). Other structures are possible such as the tree and the cylinder. The topology of the input space is preserved by using a neighborhood function during the learning phase of the SOM. In our experimentation, we use the rectangular grid instead of the other topology for its simple implementation and visualization. For each experimentation, we choose a grid size in function of our need. In the cases where we need a low granularity, we use small grids and for high granularity, we use large grid.

SOM learning algorithm

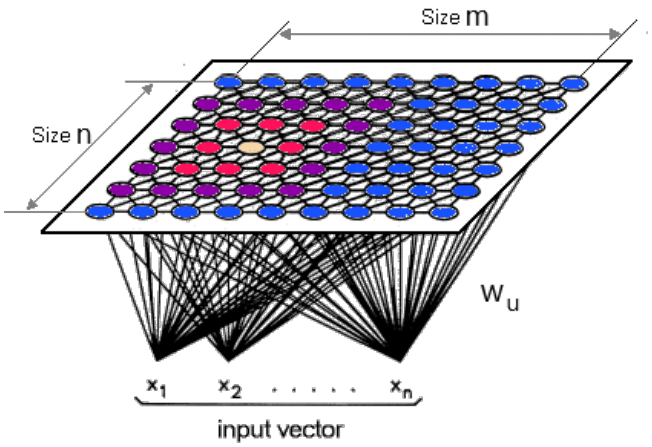


Figure 8: SOM layer representation (Source: http://www.lohninger.com/helpcsuite/kohonen_network_-_background_information.htm).

Let be an input dataset X such that $x_i \in X$ is the feature vector of the i^{th} input data. We can define a SOM composed of U neuron units where $U = m \times n$ such that m and n represent respectively the width and the height of the rectangular grid. A SOM is most of the time a two layer neural network (see Figure 8). The first

[1]LVQ is a supervised version of VQ.

layer represents the input data and the second layer is the computation layer composed of the U neuron units. Each neuron unit is a cluster represented by a weight vector $w_u \in \mathbb{R}^d$ such that $w_u = [w_{u,1}, w_{u,2}, ..., w_{u,d}]$.
The learning algorithm of SOM is composed of two steps, which are:

- Assignation: the Best Matching Unit (BMU) of a given input x_i is computed by using a distance measure (Euclidean distance, for example) between the input and weight vectors:

$$BMU(x_i) = argmin_{u \in U} \parallel w_u - x_i \parallel \tag{1}$$

  where $\parallel . \parallel$ represents the L2-norm.

- Update: the BMU and its neighbors in the map have their weights $w_u(t)$ updated towards the current input x_i:

$$w_u(t+1) = w_u(t) + \alpha(t) h_t(BMU(x_i), u)(x_i - w_u) \tag{2}$$

  where $\alpha(t) = 1 - \frac{t}{T}$ (with T the total number of iterations) is the learning rate and $h_t(BMU(x_i), u) = exp(-\frac{d(BMU(x_i),u)^2}{r \times \left(1 - \frac{t}{T}\right)})$ is the neighbourhood function (r is the radius of the map and T is the maximal number of iterations) with $d(BMU(x_i), u)$ the Manhattan distance between the winning neuron and the neuron u in the SOM structure.

SOM example

Here, we present a simple example of SOM learned on colors to show how we can use the SOM to analyze data. In this example we use 15 colors. Each color is defined by a vector of size 3 using the RGB code. The RGB code represents the primary color composition. The first value is for red, the second for green and the third for blue.
In order to obtain the Figure 9, we trained a SOM on the 15 colors. The SOM uses a rectangular grid of dimension 10x10 and thus 100 neuron units. The SOM has been trained with 300 iterations. At each iteration, a color has been randomly selected. Figure 9 represents the neuron unit weights of the SOM using the RGB code. The labelled neuron units represent the color of the dataset.
With this example we can understand easily the interest of SOM. The close colors such as blue, cyan, sky blue or grey blue are neighbors in the grid. The colors resulting of primary color mix such as green or purple are between the yellow and blue for the green and between red and blue for the purple. Moreover, the SOM is able to create different gradients of colors in agreement with the neighborhood. Those gradients are induced by the weights update rule where the BMU and its neighbors are updated together.
The property highlighted by the color example can be applied for ncRNAs. With SOM we are able to characterize the ncRNAs and identify the similarities between the ncRNA classes. Moreover, we can identify the ncRNAs that are in the overlapping area between ncRNA classes by combining a SOM approach with a perceptron layer including a rejection option. The supervised learning layer identifies the area corresponding to a ncRNA class. By using a rejection option, we are able to control the confidence of this area. For example, on the colors SOM, we can reject the colors that are too far from the color center (reject the color gradient that are too far from the color) or the ones that are ambiguous (between two or more known colors).

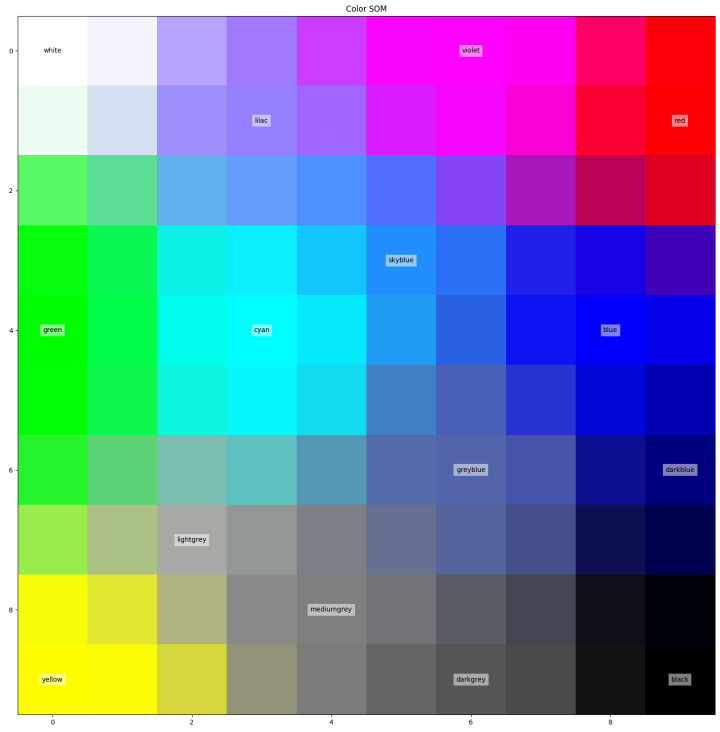


Figure 9: Colors repartition in a SOM using a 10x10 rectangular grid. The labels represent the 15 colors of the dataset.

Chapter 1

State of art

The tools used for the identification and the classification of ncRNAs can be organized according to two points of view. The first one is their classification approach. With this point of view, we can separate the tools in two classes, which are the homology based methods and the de novo approaches. The difference between the homology and the de novo approaches is how they are using available data. In the case of homology based methods, the RNAs are compared to known RNAs in order to find their classes. On contrary, the de novo approaches extract discretionary features from the known RNAs. The homology approaches have two major drawbacks: their speed and their lack of flexibility. These methods can only predict RNAs that are highly similar to the ones available in their databases. The flexibility can be improved by having more RNAs in databases but the larger is the database, the slower is the approach due to the number of comparisons.

The second point of view is the classification goal. With this point of view, the methods are separated in three groups, which represent the possible goal (see Figure 1): discrimination between coding and non-coding RNAs, identification of one class of ncRNAs and the classification of multiple classes of ncRNAs. In this thesis, we refer the methods that discriminate the coding and non-coding RNAs as the ones that identify ncRNAs and the ones that classify the ncRNAs in one or multiple classes as the ones that classify the ncRNAs.

For this state of art, we chose to organize the tools according to their classification goal. This organization is

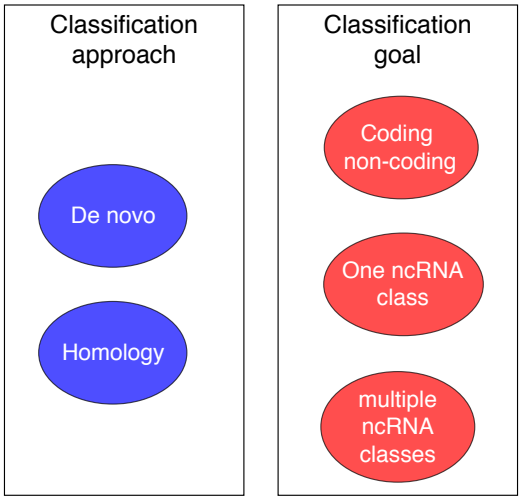


Figure 1.1: Categories of ncRNA tools.

more appropriate to understand the limitations of the related work in the identification and the classification of ncRNAs. The state of art is composed of two parts. The first part can be considered as the starting point of a ncRNA analysis which is the identification of ncRNAs. In the second part, we present the tools used to classify the ncRNAs into sub classes.

1.1 ncRNA identification

The historical difference between coding and non-coding RNAs is the lack of coding potential of the ncRNAs. Their identification can be achieved using two different types of data: experimental data or RNA sequences. In the review [22], the authors present the tools using experimental data. The most used experimental data is the one resulting of ribosome profiling [55]. As a quick reminder, the ribosome is the complex translating an RNA into a protein. Ribosome profiling aims to estimate the translation level of an RNA. In this thesis, we focus on the ones that use RNA sequences.
Most of the tools that discriminate coding and non-coding RNAs focus on the identification of ncRNAs longer than 200 nucleotides due to a belief that RNAs smaller than 200 nucleotides can not be translated into a protein. As a consequence, the reviews on the subject present tools that are mostly designed for the identification of lncRNAs [125, 2, 48, 53, 63, 136, 22]. But we must take care of this assumption because in [5], the authors stated that ribosome profiling shows translation sites on small RNAs.
The first methods identifying ncRNAs used sequence homology approach. The most basic approach is to BLAST an RNA sequence against a protein database. If the RNA sequence shows reliable similarities to one or more proteins, the RNA is predicted as coding, else it is predicted as non-coding. This basic approach has the drawback of the homology methods. The first one is that its results depend on the known proteins and ORF. For example, the mRNA of unknown proteins will be predicted as non-coding while it is a coding RNA. In order to improve the prediction performances, different tools have been developed such as COME [54], CONC [82], CPC [69], iSeeRNA[113], lncRScan-svm [114], PhyloCSF [115], PLncPRO [109] and RNAcode [129]. These tools combine homology data and features extracted from the RNA sequences. Most of the tools use features derived from the Open Reading Frame (ORF) contained in the RNA. Certain tools use the k-mer frequencies or features extracted from the nucleotide composition. Among all of these tools, the most known and most used is CPC. CPC uses a combination of features extracted from BLAST results and features related to ORF. But CPC shows a major drawback which is its time consumption.
An updated version of CPC, called CPC2 [62], has been proposed. This new version avoids the big time consumption by using a de novo approach. The authors avoid the use of homology data and use only features extracted from the RNA sequences. As in the first version, CPC2 uses features related to ORF (length, coverage and isoeletric point) and the Fickett Score [36]. By using these features, CPC2 gives better average performances than CPC with a computation time approximatively 1000 times smaller than CPC. CPC2 is not the only de novo tool, there are also PORTRAIT [7], CNCI [115], CPAT [127], PLEK [77], lncRNA-MFDL [33], DeepLNC [120], CPC2 [62] and BASiNET [57]. As CPC and CPC2, CNCI, CPAT and PLEK are among the most referenced tools in benchmark and reviews. CPAT, which has been proposed before CPC2, is the first tool that uses the Fickett score. In addition to the Fickett Score, CPAT uses also features related to ORF and an hexamer score (log-likelihood ratio between the frequencies of each Hexamer in the sequence with a model computed on coding sequences). CNCI and PLEK are both based on sequence motifs and on SVM to build the model from the computed features. CNCI uses the frequency of adjoining nucleotide triplets (ANT) to find most-like coding DNA sequences (MLCDS). Five features are extracted from the MLCDS, such as

sequence codon bias and Sequence-score (S-score). PLEK bases its predictions on an improved k-mer scheme where the frequencies are weighted by the k-mer length.
Beside these tools, COME is an interesting tool which shows good results on Arabidopsis thaliana, which is one of the base plant model. COME combines multiple data sources such as features coming from the RNA sequences, homology data and experimental data.
The majority of the tools that discriminate coding and non-coding RNAs based on their sequences use Support Vector Machine (SVM) (CONC, CNCI, CPC, CPC2, iSeeRNA, lncRScan-svm, PLEK, PORTRAIT). Besides SVM, there are also tools using Random Forest (RF) algorithms (COME and PLncPRO). Both algorithms (SVM and RF) are ones of the most used machine learning approaches. They can be easily understood and easily used thanks to packages or libraries such as libsvm (an SVM library) or the python package scikit-learn. Other classical algorithms have been used. For example, linear regression is used in CPAT and expected maximization (EM) in PhyloCSF. More recently, neural networks and deep-learning have been introduced (BASiNET, DeepLNC and lncRNA-MFDL) and give better performance than the older methods according to their respective article.
As we can see, a wide range of machine learning algorithms has been used for the identification of ncRNAs. But these methods have all a drawback, they separate coding and non-coding RNAs into two different classes without identifying the RNAs that are at the border of these two classes. As presented before, we know there are lncRNAs that are coding for small peptides and thus, there exist ncRNAs that are labeled as "non-coding" but are "coding" for a peptide. In this thesis we address this issue by proposing a tool called IRSOM which uses neural networks and a rejection option. By using the rejection option, we are able to identify a third class of RNAs containing the RNAs at the border of the coding and non-coding classes.

1.2 ncRNA classification

NcRNAs can be separated into multiple classes as we presented previously. Each class of ncRNAs is associated to specific biological mechanisms. For example, the miRNAs are responsible of the translation inhibition of mRNAs. So, in order to understand the impact of the ncRNAs in a biological study, we need to classify them. There are two types of tools to classify the ncRNAs, the ones that classify one class of ncRNAs and the ones that classify multiple classes of ncRNAs. Both types of tools have different purposes.
The first one aims to detect the ncRNAs related to one biological mechanism. This type of tools focus mostly on the classification of miRNAs and piRNAs due to their impact in diseases like cancer [97, 19]. Multiple studies have highlighted the differential expression of these ncRNAs in cancers. In [84], the authors show that the expression profile of miRNAs can be used to classify cancer. This kind of study can be achieved by using RNA-seq data with tools that classify miRNAs or piRNAs. These tools use features that are highly specific to the ncRNA class they predict. For example, the tools miRNAfold [117] and miRBoost [118] classify the miRNAs by using features derived from their specific secondary structure in hairpin. And in the case of the piRNAs, IpiRId [14] classifies the piRNAs by using epigenetics markers, sequence features and binding data specific to the piRNAs. However, the tools that predict one class of ncRNAs show good performances but in a more general biological study on ncRNAs, it can be useful to have tools that predict the most common classes of ncRNAs.
Existing tools for multiple ncRNA class prediction tend to be less specific and slower than the ones that predict one class of ncRNAs. Even if they are less specific, they show good performances. In the current state of art, we can distinguish two types of approaches to classify multiple classes of ncRNAs. The first one such as

ALPS [31], DARIO [34], CoRAL [76] and BlockClust [122] uses deep-sequencing data (RNA-seq for example). The second one such as GraPPLE [20], RNAcon [93] and nRC [35] uses mostly features extracted from the secondary structure.

The tools that use deep-sequencing data create a transcription profile. The transcription profile of a given RNA is the distribution of its reads along the RNA. The difference between ALPS, DARIO, CoRAL and BlockClust lies in the features extracted from the transcription profiles and how they are used.

In ALPS, the authors represent a transcription profile by a pattern matrix where an input $N[l, i]$ represents the number of reads of length l at position i. With this pattern matrix and the genome annotation, ALPS computes a p-value for each ncRNA class. The class with the smallest p-value is assigned to the RNA.

DARIO, CoRAL, BlockClust represent the transcription profils in a different way. Instead of having a pattern matrix, which contains the number of reads at each position, they separate the transcript reads in multiple blocks (called transcription locus in CoRAL). In the case of DARIO, they extract features such as the number of blocks assigned to a transcript, the total length of the blocks, the number of overlapping blocks, the number of nuclotides overlapped by at least two blocks, the maximum, minimum and mean of the number of reads in a block and the maximum, minimum and the mean distance between two consecutive blocks. BlockClust uses also features extracted from the blocks (block length, entropy of read length or entropy of read expressions for example) but also features extracted from the group of blocks (entropy of read start and end among all the blocks and median normalized read expressions for example) and features extracted from the contiguous blocks (distance between two consecutive blocks and the difference in median read expressions).

CoRAL uses the information of the different locus as in ALPS by creating a pattern matrix but in this case the value i represents a block instead of a genomic position. Then the pattern matrix is transformed into log-odds ratios. Moreover, CoRAL uses also features derived from the antisense abundance (number of reads mapped to the antisense strand), cleavage position entropy (positional entropy of $5'$ and $3'$), composition of the reads and the Minimum Free Energy (MFE).

The four tools use different types of classifiers. ALPS uses empirical p-values in order to classify the ncRNAs. CoRAl and DARIO use the RF as classifier. BlockClust can be used with a supervised and unsupervised classifiers; the unsupervised version uses the Markov Cluster Algorithm (MCL) and the supervised version uses the stochastic gradient descent SVM.

The tools using mostly secondary structures base their computations on a graph representing RNA secondary structure. GraPPLE and RNAcon extract features from the graph by computing graph properties such as the density, transitivity or nodes betweenness centrality for example. GraPPLE and RNAcon use the same 20 graph properties. The difference between these tools is that GraPPLE uses an SVM classifier when RNAcon uses a RF. With this little differences, RNAcon shows better performances than GraPPLE. Moreover, RNAcon can use k-mer frequencies to discriminate coding and non-coding RNAs.

The most recent tool nRC, uses a different approach. From the RNA secondary structure graph, nRC computes and selects the most discriminative substructures. For each RNA, it creates a binary vector representing the presence or absence of the substructures. Then with these vectors, it trains a Convolutional Neural Network (CNN) to classify the RNAs. With these new features and the use of CNN, nRC shows better performances than RNAcon.

Both tools using deep-sequencing data and the ones relying mostly on secondary structure show good performances in certain cases. The BlockClust (the best tool using RNA-seq data) supervised version shows better precision than DARIO on three ncRNA classes which are the miRNA, tRNA and snoRNA. On the contrary, nRC (the best tool using secondary structure) has poor results for the snoRNA (HACA-box and

CD-box) and the miRNA, but good results on intron RNA, rRNA and tRNA. The bad results on the snoRNAs and miRNAs can be explained by the fact that both classes contain an hairpin structure. BlockClust shows good results on the ncRNA classes that are highly expressed when nRC shows good results for ncRNA classes with a high conserved secondary structure (the intron group 1 and 2 or the rRNAs for example). These performance differences support the idea that we must use different sources of data to classify ncRNAs.

Part I

Identification of ncRNAs

Chapter 2

SLSOM, a reliable classifier

Clustering is the most popular tool for exploratory data analysis. It can help guide the analysts to understand the data, when the goal of the analysis is well defined and captured by the clustering model. Most of the time, this goal is expressed by identifying categories in a given application. In this thesis, we are interested in the classification of ncRNAs. We show previously the principal classes of ncRNAs. Even if these classes represent the vast majority of the studied ncRNAs they are not all the ncRNAs. Moreover, we know there exist more ncRNA classes to discover.

In this chapter, we propose a multilayer supervised SOM called SLSOM, where the first layer corresponds to the input layer, the second layer consists of the classical unsupervised SOM and the last layers are fully-connected and linked to the SOM units. The unsupervised SOM layer provides clusters and data visualization, and the supervised fully-connected layers are used for data classification. The clustering is not influenced by the supervised layers, since the label information does not modify the cluster organization in SOM. The map in the SOM layer is trained with the aim of locating the positions of the training samples on the map such that the relative distance between them in the original data space is preserved as much as possible.

We associated two kinds of rejection to the output layer. The first one rejects the classification of samples which are not associated to one of the classes used in the output layer. The second one rejects the ambiguous classification of the examples located in class intersection regions. We also compared, for the two rejection types, the effect of using global or local rejection. In global rejection, we associate one threshold for all the classes and in the local rejection, we use several thresholds, one for each class.

The aim of SLSOM is to propose an analysis approach that is able to exploit available labels by using a classification technique and to take advantage of the exploratory property of clustering. In addition, our approach combines clustering and classification with rejection options, in order to consider partial information on some labels (some classes must be discovered). Our approach has the advantage of classifying samples of known categories (classes) with high confidence and rejecting the classification of samples for which the classification confidence is not sufficient. Those rejected samples can then be analyzed and visualized using the clustering method.

2.1 Supervised SOM related work

In this section, we present the two main strategies proposed in the literature to introduce label information into the SOM. We also present works related to supervised learning models with rejection.

2.1.1 Supervised classification with SOM

Several efforts have been made to introduce the label information in the SOM [65] [68] [87]. We can distinguish two strategies: the first one uses label information during the SOM training step. The second one uses this information after the map construction.
The first strategy is proposed by Kohonen in [68]. Its principle is to combine the label vector x_l with the input vectors x_v to form an augmented vector $x = [x_v, x_l]$. A SOM trained on augmented vectors produces a supervised SOM. For each unit u, the corresponding weight vector $w_u = [w_u^v, w_u^l]$ is updated using a classical SOM algorithm. The augmented vectors are used during the assignment step (finding the Best Matching Unit (BMU)) and in the update step (update of the neurons weight vectors).
In the second strategy, two main approaches are proposed in the literature. In the first approach, proposed in [65], the authors use the same weight vector as in the first strategy, but don't use the label part during the assignment step. With this modification, the supervised SOM is built as an unsupervised SOM, where the weight vectors of neurons contain the label information. In the second and more trivial approach, proposed in [71] and [110], labels are associated to the units (clusters) after the SOM training step using the training data assigned to each unit. The units are labelled by majority voting of their corresponding input data. In [71], authors present two other methods based on the distances between the unit prototypes and their related training data. The first method assigns to a neuron the label of the closest training example. The second method associates to each neuron the label of the closest class to the neuron prototype. Each class l is represented by the mean vector of the training examples of the class l that belongs to the unit.
All the supervised SOM strategies proposed in the literature use the information of one SOM prototype to classify new examples. By doing this, they use only the local information of the prototypes. To overcome this problem, we propose to add fully-connected layers on top of an unsupervised SOM and to classify new inputs by taking into account the distances between the inputs and all the map prototypes.

2.1.2 Classification with reject option

The performance of a classifier depends on its ability to correctly predict the class of a given input data. An accurate prediction improves the performance of the classifier when a misclassification reduces its performance. Thus, to improve the performance of a classifier, the number of misclassifications should be reduced. A misclassification happens when an input data is in an ambiguous region (region where two or more classes are overlapping) or is far from the learned classes. One way to reduce the amount of misclassifications is to use a rejection option. The rejection option will improve the performance of the classifier by rejecting some wrong predictions. By increasing the number of rejected input data, the number of rejected misclassifications increases but the number of rejected accurate predictions increases as well. So there is a tradeoff to find between the performance of the classifier and the amount of rejected predictions.
We can distinguish two types of rejection: the distance rejection and the ambiguity rejection. The distance rejection rejects samples that are far from the learned classes. The ambiguity rejection rejects samples in regions where classes are overlapping.

Ambiguity rejection

In one of the pioneer works about the rejection [23], the author introduced the bases of a rejection option which is called Chow's rule. According to the Chow's rule, the prediction of the input data x_i for a given model θ is rejected if:

$$\max_c P(x_i|\theta_c) < \beta_{chow} \tag{2.1}$$

where c and θ_c represent respectively the class and the model parameter for the class c, β_{chow} is a given threshold and $P(x_i|\theta_c)$ is the probability that x_i belongs to the class c.

The optimal value of β_{chow} can be determined as follows. Let be π the cost matrix of the classification such that:

$$\pi(a,b) = \begin{cases} \omega & \text{if rejected} \\ 0 & \text{if } a = b \\ 1 & \text{if } a \neq b \end{cases}$$

where $0 < \omega < 1$ is the reject cost. We can define two loss functions E_P and E_R which correspond respectively to the expected loss of the prediction and the rejection such that:

$$E_P(x_i) = 1 - P(x_i|\theta_{y_i})$$
$$E_R(x_i) = \omega$$

where y_i is the class label of the input data x_i. From these loss functions, Chow proposes to solve the equation:

$$\begin{aligned} E_P(x_i) = E_R(x_i) &\Leftrightarrow 1 - P(x_i|\theta_{y_i}) = \omega \\ &\Leftrightarrow P(x_i|\theta_{y_i}) = 1 - \omega \end{aligned}$$

The optimal value of β_{chow} is thus given by: $\beta_{chow} = 1 - \omega_r$.

Chow's method allows to obtain an error and a rejection tradeoff relation using the Bayesian decision theory. This approach is based on the hypothesis that we have a complete knowledge of the priori probability distribution of the classes and the posterior probabilities. This assumption is, in general, not satisfied in real problems where these probabilities are often unknown.

In [40], authors propose to associate local thresholds (one for each class) to the Chow's method. They show that local thresholds work better than a global one when the class probabilities are estimated.

Some works associate rejection to neural networks. In [56] and [28], the authors define an ambiguity rejection option from the outputs of a neural network $o = [o_1, o_2, ..., o_p]$. The authors use the sigmoid function as the activation function of the output neurons. The sigmoid function is defined by: $sigmoid(z) = \frac{1}{1+\exp(z)}$, where z is the activation of a neuron. The neural network rejects the classification of an input x_i if the difference between the largest neuron output for predicting the class o^* and the second largest neuron output for predicting the class o^{**} is lower than a threshold β_{diff} such as:

$$\mid o^* - o^{**} \mid < \beta_{diff} \tag{2.2}$$

Distance rejection

In [56] and [28], the authors also define a distance rejection option from the outputs of a neural network. In the distance rejection, the neural network rejects the classification of an input x if the probability of the predicted class o^* is lower than a defined threshold β_{max} such as:

$$o^* = \max\{o_1, o_2, ..., o_p\} < \beta_{max} \tag{2.3}$$

The rejected input data may belong to a potential new class.
We add rejection options to our model in order to improve the reliability of the obtained results. We introduce rejections using both the distance and the ambiguity options. The distance rejection is used to discover potential new classes and the ambiguity rejection will improve the prediction in case of overlapping classes.

2.2 Supervised SOM with rejection

We propose a new supervised SOM which uses a multi layer of perceptrons for the classification. We associate two rejection options (distance and ambiguity rejections) to our proposed approach in order to improve the classification results and to discover new potential classes. We also propose to use several thresholds (one for each class) for both rejection options.

2.2.1 Supervised SOM

Let a set of examples (x_i, y_i), $i = 1, ..., n$, such that x_i is the feature vector of the i^{th} example and y_i is a vector representing its label (the dimension of y_i is equal to the number of classes). If the class of x_i is c, the c^{th} component of y_i is equal to 1 and the other components are equal to 0. The goal of the supervised SOM is to learn a function $\psi : \mathrm{X} \rightarrow \mathrm{Y}$, where X is the input space and Y is the output space using an extension of SOM.
We propose a multilayer architecture containing the SOM (see Figure 2.1): an input layer that represents the inputs, a SOM layer, which corresponds to a classical SOM and fully-connected layers, which consist of a multilayer perceptron (MLP) that is connected to the SOM neurons with forward connections.

The SOM layer

Each neuron unit u in the SOM is associated with a weight vector $w_u = [w_u^1, w_u^2, ..., w_u^m]^T \in R^m$ with the same dimension as the input vector $x = [x_1, x_2, ..., x_m]^T \in R^m$.
Through an unsupervised learning process, the output neurons are tuned and organized after several presentations of the data. The learning algorithm that leads to a self-organization can be summarized in two steps:

- A winning or best-matching unit of the map, denoted $s(x)$, is found by using a distance measure (Euclidean distance, for example) between the input and weight vectors:

$$s(x) = \arg\min_{r \in A} d(x, w_u) \tag{2.4}$$

  where A is the set of neurons and $d()$ is the Euclidean distance.

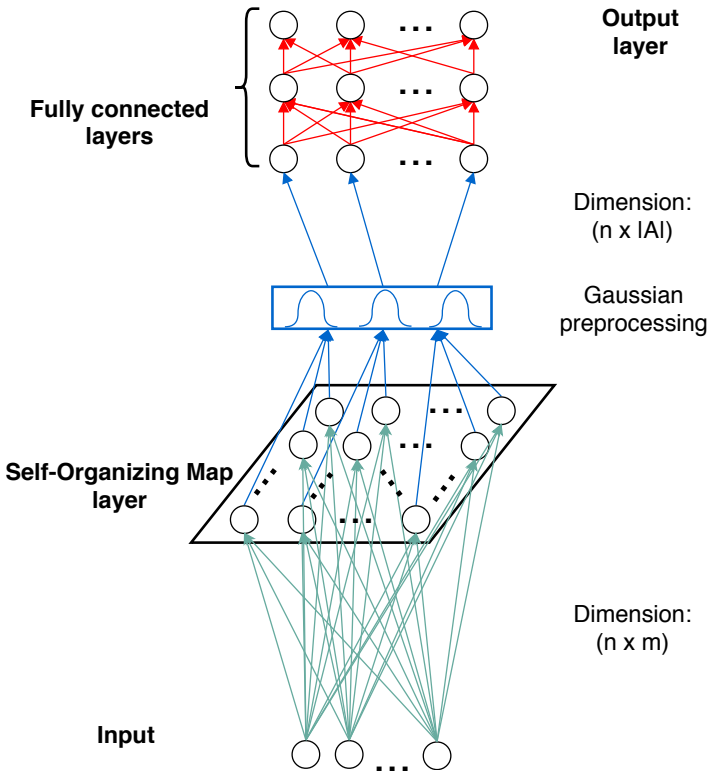


Figure 2.1: Example of SLSOM architecture with a two hidden layers MLP.

- The winner and its neighbors in the map have their weights $w_u(t)$ updated towards the current input x:

$$w_u(t+1) = w_r(t) + \alpha(t) H_t(s(x), u)[x - w_u(t)] \tag{2.5}$$

where $\alpha(t) = 1 - \frac{t}{T}$ (with T the total number of iterations) is the learning rate and $H_t(s(x), u) = exp(-\frac{d(s(x),u)^2}{2\sigma^2(t)})$ is the neighborhood function ($\sigma(t)$ the neighborhood function's radius).

The SOM layer output function

The activation of the unit u in the SOM map for the input data x_i can be computed as follows:

$$a_{u,i} = \| x_i - w_u^h \|_2$$

We define an output activity $a_{u,i}^{out}$ as the activity seen from the fully-connected layers between the unit u of SOM and the input data x_i. It is defined as a Gaussian similarity, as proposed in [123]:

$$a_{u,i}^{out} = exp^{\frac{-a_{u,i}^2}{2\sigma^2}}$$

where the parameter σ^2 is the variance of the Gaussian distribution.

This SOM activation function consider the activities of all neurons in the map for each input. It corresponds to overlapping receptive fields that are close to what we observe in biological visual areas. In addition, in [51], authors show that it gives better results than just considering the best matching unit of the SOM for regression, and we have verified and confirmed this for classification on different datasets (see Table 2.8).

The fully-connected layers (MLP)

The MLP contains B hidden layers and receives the activation pattern provided by the SOM.
The first layer receives the activity output a^{out} from the SOM layer. The activation of the neuron l in the hidden layer j $(1 \leq j \leq B)$ is defined by:

$$h_l^j = f\left(\sum_k w_{kl}^j \; h_k^{j-1} + b_l^j\right) \tag{2.6}$$

$$\tag{2.7}$$

where w_{kl}^j is the weight of the connection linking the neuron l of the layer j to the neuron k of the layer $(j-1)$, b_l is the bias of the neuron l and h_k^{j-1} the activation of the neuron k in the layer $(j-1)$ ($h_k^0 = a_k^{out}$ with a_k^{out} the activity of the unit k of SOM). $f()$ is the activation function. We use the Rectified Linear Unit (ReLU) activation function that has become very popular in the last few years. It is computed as follows:

$$ReLU(x) = max(0, x) \tag{2.8}$$

The output of a neuron c in the last layer is obtained as:

$$o_c = f^{out}\left(\sum_k w_{kc}^{out} \; h_k^B + b_c^{out}\right) \tag{2.9}$$

$$\tag{2.10}$$

where w^{out} and b_l^{out} are respectively the weights and the biases of the output layer and h^B the activations of the last hidden layer.
The class-membership probabilities can be obtained by using the softmax activation function defined as follows:

$$f^{out}(z_i) = softmax(z_i) = \frac{exp(z_i)}{\sum_i exp(z_i)}. \tag{2.11}$$

To train the MLP, we used the cross-entropy cost function C, which is among the most popular choices in the state of the art implementations for classification problems. It is defined by:

$$C(Y, O) = -\sum_i \sum_c y_{ic} \ln o_{ic} \tag{2.12}$$

where Y is the vector of true labels and O the output vector of our approach.
We have used the Adam [64] optimizer since it outperforms the other optimizers in practice.

Regularization

Regularization is a technique which aims to improve the generalization of a learned model and to avoid overfitting [43]. It introduces a penalty term in order to explore some regions of the function space, which are used to build the model. This leads to the improvement of the generalization capacities of the model. In our approach, we use the L2 norm regularization. The cost function can be written as:

$$Loss(Y, O) = C(Y, O) + \lambda \sum_c || \; w_c^{out} \; ||^2 \tag{2.13}$$

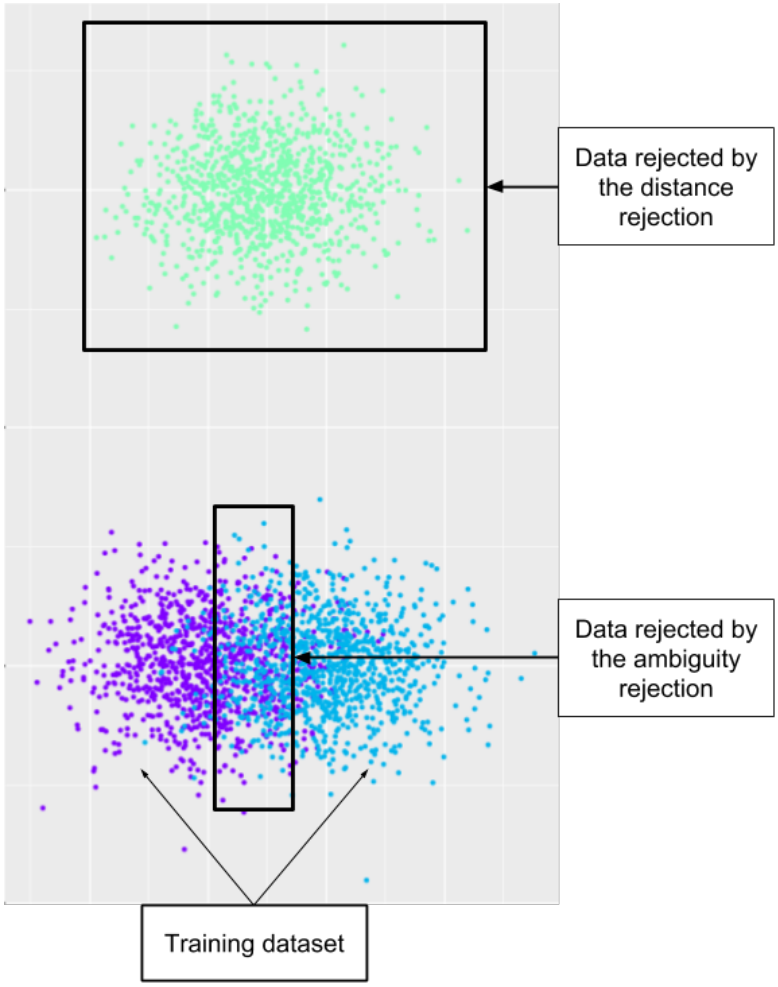


Figure 2.2: Schema representing the distance and ambiguity rejection rules.

where λ is the parameter which controls the importance of the regularization term.

2.2.2 Supervised SOM with reject option

In order to improve the reliability of the classification provided by our approach, a rejection decision is plugged to the output layer of the neural network. The rejection decision is based on the estimation of the posteriori probabilities computed from the output layer. We propose a new classifier with rejection based on two rejection rules (see Figure 2.2). The first one, called distance rejection, rejects the samples that are far from the known classes and thus, can be potential new classes. The second one, called ambiguity rejection, rejects the samples that are in ambiguous areas (where two or more classes are overlapping). Both rules rely on the output of the perceptrons, where the output o_{ic} of the neuron c in the output layer for the input x_i corresponds to the estimation of the posterior probability $P(c|x_i)$ of an input x_i to be an element of the class c. In order to distinguish the two types of rejection (distance and ambiguity), the output o_{ic} of the neuron c in the output layer is defined using a sigmoid activation function given by:

$$o_{ic} = sigmoid(\sum_k w_{kc}^{out}\ h_{ic}^{B} + b_c^{out})$$
$$= \frac{1}{1 + \exp\left(\sum_k w_{kc}^{out}\ h_{ic}^{B} + b_c^{out}\right)}$$

where w_{kc}^{out} is the connection between the neurons c and k in the output and the last hidden layer and b_c^{out} is the bias of the neuron c.

Distance rejection rule

In the distance rejection rule, the prediction is rejected if the largest output o_{ic} is lower than a threshold β_{max}. Thus we can define the distance rejection rule $rule_{dist}$ by:

$$rule_{dist}(x_i) = \begin{cases} 1 & \text{if } o_{ic^*} < \beta_{max} \\ 0 & \text{otherwise} \end{cases} \tag{2.14}$$

where $o_{ic^*} = \max_c\{o_{ic}\}$ and $0 \leq \beta_{max} \leq 1$.

Ambiguity rejection rule

In the ambiguity rejection rule, the prediction of the input i is rejected if the difference between the two largest outputs o_{ic^*} and $o_{ic^{**}}$ ($o_{ic^{**}} = \max_{c \neq c^*}\{o_{ic}\}$ and $c^* = \arg\max_c\{o_{ic}\}$) is lower than a threshold β_{diff}. We can define the ambiguity rejection rule $rule_{diff}$ by:

$$rule_{diff}(x_i) = \begin{cases} 1 & \text{if } o_{ic^*} - o_{ic^{**}} < \beta_{diff} \\ 0 & \text{otherwise} \end{cases} \tag{2.15}$$

where $0 \leq \beta_{diff} \leq 1$.

Classifier with rejection

In this work, we associate the distance rejection rule and the ambiguity rejection rule to the classifier. If an input data is far from its predicted class, the classifier rejects its prediction and labels it as rejected by the distance rejection rule. If an input data is in an ambiguous area, the classifier rejects its prediction and labels it as rejected by the ambiguity rejection rule. In certain cases, an input data can be rejected by both rejection rules. For example, we can have an input data that is far from two classes and equidistant from them. In this case, we label the input data as rejected by the distance rejection rule because in our opinion, this input data may belong to a potential new class.
We define our classifier with rejection option ψ^{global} such that:

$$\psi^{global}(x_i) = \begin{cases} -1 & \text{if } rule_{dist}(x_i) = 1 \\ -2 & \text{else if } rule_{diff} = 1 \\ \arg\max_c \ o_{ic} & \text{otherwise} \end{cases} \tag{2.16}$$

Classifier with local rejection threshold

The previously defined classifier with rejection is defined in a global way, i.e there is one threshold for the distance rejection rule and another one for the ambiguity rejection rule for all the classes. This classifier with rejection suffers from a lack of flexibility. This suggests the use of local reject thresholds, one for each class, in order to obtain optimal decisions and reject regions. We thus extend the distance rejection rule and the

ambiguity rejection rule by adding a threshold for each class in order to have a local rejection such that:

$$rule_{dist}^{local}(x_i) = \begin{cases} 1 & \text{if } o_{ic^*} < \beta_{max}^{c^*} \\ 0 & \text{otherwise} \end{cases}$$

$$rule_{diff}^{local}(x_i) = \begin{cases} 1 & \text{if } o_{ic^*} - o_{ic^{**}} < \beta_{diff}^{c^*} \\ 0 & \text{otherwise} \end{cases}$$

where $\beta_{max}^{c^*}$ and $\beta_{diff}^{c^*}$ represent the thresholds associated to the predicted class c^* using the local distance rejection and the ambiguity rejection respectively. We call these new rejection option rules the local distance rejection rule and the local ambiguity rejection rule.

Thresholds computation

The optimal thresholds can be determined using the Chow's method [23] presented in the Section 2.1. But this method suggests that there is a high confidence in the posteriori probabilities. In our case, we may have small datasets, and so the posteriori probabilities are not reliable. In order to evaluate our rejection options, we test different values of thresholds using a grid search.

2.3 Experimentation

Our method, called SLSOM, is implemented using the TensorFlow [1] Python API. TensorFlow is an open-source software library for machine learning developed by Google. It is designed to be scalable, flexible and distributed.
SLSOM was tested on several datasets, and compared to different other classification methods existing in the literature.

2.3.1 Datasets

To evaluate the proposed method, we considered three artificial datasets and nine real world datasets. The real world datasets are extracted from the UCI database [81]. In the UCI datasets, each instance with missing values has been removed. The different datasets are briefly described in Table 2.1.

Artificial datasets description The first artificial dataset (Artificial 1) is composed of four Gaussian clusters (see Figure 2.3a), each representing a class. In this dataset, we have two overlapping classes (classes 0 and 1). It consists of two dimensional data instances generated from bi-dimensional Gaussian distributions. We define for each class a bi-dimensional Gaussian distribution $N(\mu_c, \sigma)$, where μ_c and σ represent respectively the center vector and the covariance matrix of the distribution for the class c. The covariance matrix is the same for each distribution and is defined by:

$$\sigma = \begin{bmatrix} 0.04 & 0 \\ 0 & 0.04 \end{bmatrix} \tag{2.17}$$

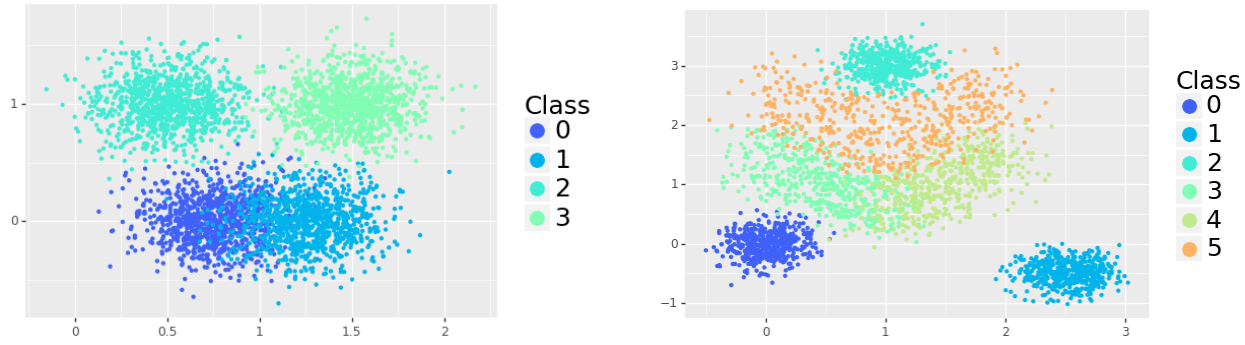
The coordinates of the distribution centers μ_c are given in Table 2.2.

Dataset	Instances	Attributes	Classes	Description
Artificial 1	4 000	2	4 classes of 1 000 points	2D generated points
Artificial 2	3 000	2	6 classes of 500 points	2D generated points
Artificial 3	40 000	10	4 classes of 10 000 points	10D generated points
MNIST	70 000	784 (28x28 images)	10 balanced classes	hand written digits
Cardiotocography	2026	23	3 classes 0: 1 655 1: 295 2: 176	fetal heart rate (FHR) and uterine contraction (UC) features on cardiotocograms
Yeast	1 299	8	4 classes 0: 463 1: 429 2: 244 3: 163	Protein localization site Keep only the four first classes which have the most instances
WDBC	683	10	2 classes 0: 444 239: 239	Wisconsin Diagnostic Breast Cancer
Dermatology	358	34	6 classes 0: 111 1: 60 2: 71 3: 48 4: 48 5:20	Clinical and histopathological information of patient with erythemato-squamous diseases
Ionosphere	351	34	2 classes 0: 126 1: 225	Radar data
E.coli	336	7	4 classes 0: 143 1: 116 2: 52 3: 25	Protein localization sites
SPECTF	267	45	2 classes 0: 55 1: 212	Single Proton Emission Computed Tomography (SPECT) images
Iris	150	4	3 classes 50 instances in each class	Sepal and petal length and width

Table 2.1: Overview of the datasets used in the benchmark.

Cluster id	X	Y
0	0.75	0
1	1.25	0
2	0.5	1
3	1.5	1

Table 2.2: Center vectors of the cluster distributions for Artificial 1 dataset.



(a) Artificial 1 (b) Artificial 2

Figure 2.3: Plot representing the 2D points of Artificial 1 and Artificial 2 datasets.

Cluster id	X	Y
0	0.0	0.0
1	2.0	-0.5
2	1.0	3.0

Table 2.3: Center vectors of the Gaussian clusters for Artificial 2 dataset

The second artificial dataset (Artificial 2) is composed of three Gaussian clusters (labeled 0, 1 and 2) and three clusters generated with uniform distributions (labeled 3, 4 and 5) (see Figure 2.3b). As for the first artificial dataset, the Gaussian clusters are generated with bi-dimensional Gaussian distributions. The parameter σ is the same but the center vectors are different. The coordinates of the centers μ_c for the Gaussian clusters are given in Table 2.3. For each point $(x, y) \in X \times Y$ of clusters 3, 4 and 5, the intervals of the uniform distributions are given in Table 2.4. After the generation of points with the uniform distribution, we add a Gaussian noise to clusters 4, 5 and 6. The purpose of this second artificial dataset, which is more complex than the first one, is to highlight the necessity of the use of local rejection in some situations. The clusters are generated in order to have different degrees of ambiguity. Clusters 3, 4 and 5 have a high ambiguity in their overlapping areas. Cluster 5 overlaps with cluster 2. Clusters 0 and 3 are close but still separable. Cluster 1 is well separated from others.

The third artificial dataset (Artificial 3) is composed of four Gaussian clusters, each representing a class. It consists of ten dimensional data instances generated from ten dimensional Gaussian distributions. We define for each class a ten-dimensional Gaussian distribution. The covariance matrix σ is the same for each distribution and is defined such that $\sigma_{ii} = 0.01$ for $i = 1, .., 10$ and $\sigma_{ij} = 0$ for $i, j = 1, .., 10$, $i \neq j$. The coordinates of the Gaussian distribution centers are given in the Table 2.5. The purpose of building this dataset is to show the ability of our approach to handle large datasets (this dataset contains 40 000 points).

Cluster id	X	Y
3	$[0.0, 1.0]$	$[1.0 - x, 2.0 - x]$
4	$[1.0, 2.0]$	$[x - 1.0, x]$
5	$[0.0, 2.0]$	$[2.0 - x, 3.2 - x]$ if $x < 1$ $[0.0 + x, 1.2 + x]$ otherwise

Table 2.4: Intervals of the uniform distributions in Artificial 2 dataset. For each point $(x, y) \in X \times Y$, the value of y depends on the value of x.

Cluster id	Dimension									
	0	1	2	3	4	5	6	7	8	9
0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
2	0.5	0.45	0.35	0.2	0	-0.25	-0.55	-0.9	-1.3	-1.75
3	-0.5	-0.45	-0.35	-0.2	0	0.25	0.55	0.9	1.3	1.75

Table 2.5: Center vectors of the Gaussian clusters for Artificial 3 dataset.

2.3.2 Tested methods

We compared our method SLSOM to several classical supervised learning existing methods as well as to existing supervised SOM methods. These supervised learning methods received two kinds of input. The first one corresponds to the initial dataset. The second one represents the activity outputs of SOM.
We used the scikit-learn [95] implementation of five classical classifiers:

- Support Vector Machines (SVM) [26]. SVM is a widely used supervised classifier. This classifier computes a hyperplane, which maximizes the largest separation "margin" of two classes.
- Random Forest (RF) [52]. The Random Forest is an ensemble method that relies on decision trees. In this approach, multiple decision trees are trained on different subsets of the training dataset with different features, and the prediction is done by performing an averaging of a majority vote on the prediction trees.
- Gaussian Naïve Bayes (GNB) [49]. The Gaussian naïve Bayes is a classifier based on the Bayes's theorem with strong independence assumptions between the features. In order to deal with continuous data, the assumption is made that the continuous values associated to each class are distributed according to a Gaussian distribution.
- K-Nearest Neighbors (KNN) [27]. The K-Nearest Neighbors algorithm classifies a new input by using a majority vote of the k-closest training examples in the feature space. The new input is assigned to the most common class among its k-nearest neighbors.
- Logistic Regression (LR) [41]. The logistic regression finds the best fitting model to describe the relationship between the feature of interest (dependent variable that represents the class) and a set of independent explanatory variables (features). Logistic regression predicts the probability of presence of the class.

For the supervised SOM approaches, we implemented (with TensorFlow) five variants of SOM proposed in the literature:

- In the first one, that we call SSOM, the input vectors and labels are combined into an augmented vector for the SOM training [68].
- In the second one, that we call ESSOM, only the inputs that constitute a part of the augmented vectors are used in the training step [65].
- In the third one, that we call VSSOM, the neurons are labelled using a majority vote among the classes of the training examples associated to the neurons [87].
- In the fourth one, that we call MDSSOM, the neurons are labelled by the class of their closest input example to the neuron prototype [71].

- In the fifth one, that we call ADSSOM, the neurons are labelled by the class of examples for which the mean is the closest to the neuron prototype (from the training examples represented by a neuron, we compute the mean of the examples representing each class) [71].

2.3.3 Results

The comparison was performed on the twelve datasets presented above (see Table 2.1). We first evaluate the proposed supervised SOM without reject option. After that, we show the strength of the reject option using the artificial datasets.

Results of SLSOM without reject option

We measure the performance of each method by using the F1-score, MCC and accuracy measures:

$$\text{F1-score} = \frac{1}{ncl} \times \sum_c \frac{2TP_c}{2TP_c + FP_c + FN_c} \tag{2.18}$$

$$\text{MCC} = \frac{1}{ncl} \times \sum_c \frac{TP_cTN_c - FP_cFN_c}{\sqrt{(TP_c + FP_c)(TP_c + FN_c)(TN_c + FP_c)(TN_c + FN_c)}} \tag{2.19}$$

$$\text{Accuracy} = \frac{1}{ncl} \times \sum_c \frac{TP_c + TN_c}{TP_c + TN_c + FP_c + FN_c} \tag{2.20}$$

where TP_c, TN_c, FP_c and FN_c represent respectively the true positive, true negative, false positive and false negative when the class c is considered as the positive class and the other as the negative one and ncl is the number of classes.

The datasets are normalized (to obtain values between 0 and 1) in order to reduce measurement biases. We evaluated the different methods using a 5-fold cross-validation. To evaluate the contribution of the SOM, we benchmark the classical classifier (SVM, RF, GNB, KNN and LR) with the raw data and the activation pattern of the SOM. In each fold, we varied the parameters in order to find the best combination of parameters for each method and for each dataset. For the SVM, we varied the penalty term of the error (by a factor of 10 from 10^{-5} to 10^5) and the kernel type (linear, polynomial, rbf and sigmoid). For the Random Forest, we varied the number of trees (from 10 to 50 by a step of 10) and the function measuring the strip quality (entropy or the Gini impurity). For the Logistic Regression, we varied the regularization norm (l1 or l2) and the regularization parameter (from 1 to 10^5 by a factor of 10). For the KNN, we varied the number of neighbors (from 5 to 30 by a step of 5) and the distance weight function (uniform or the inverse of the distance). For the SSOM, we varied the label weights from 0.5 to 5 by a step of 0.5. The selected parameters set is the one that gives the best accuracy mean on a given dataset.

In the SOM layer of SLSOM, we have used a rectangular grid. The number of units in this layer will influence the ability of the SOM to generalize from training examples to unknown samples. We have tested different map sizes for each dataset and we have chosen those which give the best generalization results (see Table 2.6).

Map size	Dataset
4x4	Iris
5x5	Dermatology, E.coli, Ionosphere, SPECTF
6x6	WDBC, Yeast
8x8	Artificial 1
7x7	Artificial 2, Cardiotocography
10x10	Artificial 3
12x12	MNIST

Table 2.6: Size of the map used for each dataset.

Organisation id	Layer 1	Layer 2	Layer 3
1	$ncl + \frac{(\|A\|-ncl)}{2}$	None	None
2	$ncl + \frac{2(\|A\|-ncl)}{3}$	$ncl + \frac{(\|A\|-ncl)}{3}$	None
3	$ncl + \frac{(\|A\|-ncl)}{3}$	$ncl + \frac{2(\|A\|-ncl)}{3}$	None
4	$ncl + \frac{3(\|A\|-ncl)}{4}$	$ncl + \frac{(\|A\|-ncl)}{2}$	$ncl + \frac{1(\|A\|-ncl)}{4}$
5	$ncl + \frac{3(\|A\|-ncl)}{4}$	$ncl + \frac{1(\|A\|-ncl)}{4}$	$ncl + \frac{(\|A\|-ncl)}{2}$

Table 2.7: Number of perceptrons in the hidden layer of the tested MLP architectures. ncl is the number of classes and $|A|$ is the number of unit prototypes.

For SLSOM, we varied three parameters: the learning rate, the regularization parameter, and different architectures of the MLP. We tested three values of the learning rate: 0.001, 0.1 and 0.3, three values of regularization parameter: 0.001, 0.1, 0.3 and 0.5, and six different MLP architectures that are described in Table 2.7. The number of hidden layers in the MLP varied from one to three. We set the parameter σ of the output activity such that the term $2\sigma^2$ corresponds to the average minimal distance between the neuron prototypes of the SOM.

Dataset	F1-score		MCC		Accuracy	
	GA	BA	GA	BA	GA	BA
Artificial 1	0.930	0.931	0.907	0.909	0.965	0.965
Artificial 2	0.928	0.9	0.914	0.882	0.976	0.967
Artificial 3	0.796	0.708	0.999	1.0	0.999	0.938
Cardiotocography	0.796	0.708	0.709	0.575	0.924	0.899
Dermatology	0.945	0.930	0.937	0.922	0.985	0.982
E. coli	0.895	0.918	0.870	0.893	0.951	0.962
Ionosphere	0.945	0.872	0.893	0.755	0.951	0.888
Iris	0.95	0.955	0.928	0.941	0.969	0.973
MNIST	0.924	0.870	0.916	0.856	0.985	0.97
SPECTF	0.640	0.496	0.405	0.190	0.781	0.773
WDBC	0.966	0.961	0.933	0.923	0.969	0.964
Yeast	0.633	0.592	0.495	0.445	0.805	0.786

Table 2.8: Average performances of SLSOM with the Gaussian activity (GA) and the BMUs activity (BA).

We have also compared the Gaussian activity that we have defined to a classical SOM activity (See Table 2.8). Table 2.8 shows that the Gaussian activity gives the best classification results for our datasets.
The performance of the methods on the different considered datasets described above (artificial datasets and real datasets (except MNIST)) are given in Figure 2.4, Figure 2.5 and Figure 2.6. Our method SLSOM gives competitive results compared to the other existing supervised SOM methods on all considered datasets. Furthermore, our method is very competitive with the classical supervised methods like Gaussian Naïve Bayes (GNB), K-Nearest Neighbors (KNN), Logical Regression (LR) and SVM. SLSOM gives the best classification results on Artificial 2 dataset. Moreover, our approach gives comparable or better results than the variants of the classical supervised methods that receive the SOM activity outputs. It should be noted that the SOM does not improve the prediction results of these classifiers but decreases their performances, except for the Ionosphere dataset using GNB, KNN and LR. For RF, the performance drop is higher than with other classifiers.
Figure 2.7 gives the results on the MNIST dataset. It shows the performance of SLSOM with the Gaussian activity (SLSOM), SLSOM with the BMUs activity (SLSOM_bmus) and the other supervised SOM. Our method, which is the one that gives the best result, has an accuracy of 0.984, a MCC of 0.915 and a F1 score of 0.923. Moreover, with a test error rate of 1.76%, SLSOM is better than some classical supervised learning algorithms like linear classifiers, some neural networks architectures, KNN with Euclidean distance, etc. (see the benchmark in [72]).

Results of SLSOM with reject option

We evaluated the performance and the utility of the classifier with rejection option on Artificial 2 dataset. Firstly, we show how the accuracy of SLSOM can be improved by using the rejection options. Then, we demonstrate the ability of our approach (i) to find potential new classes using the distance rejection option

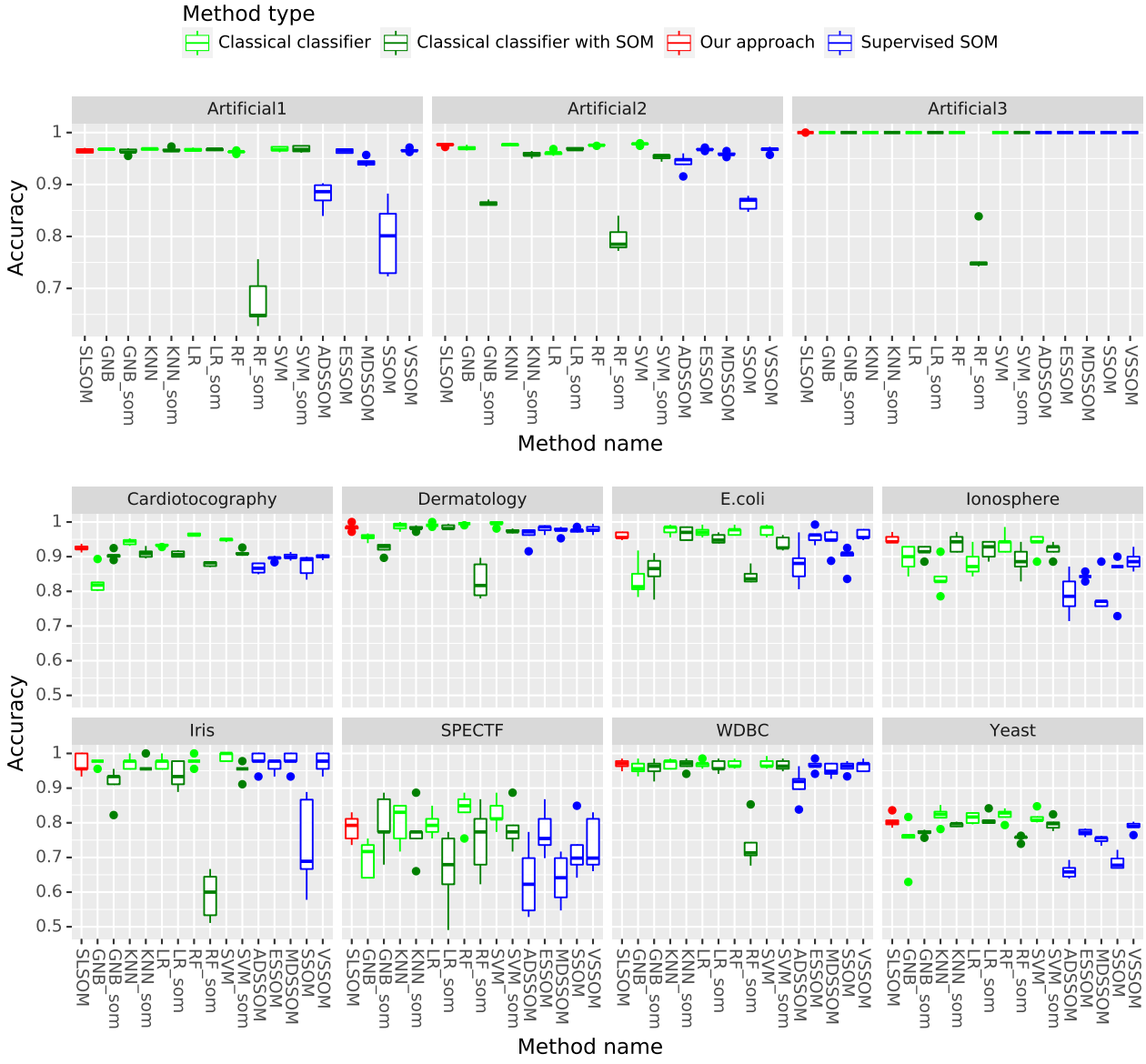


Figure 2.4: Accuracy results of SLSOM and existing methods on artificial datasets and real datasets.

Figure 2.5: F1-score results of SLSOM and existing methods on artificial datasets and real datasets.

Figure 2.6: MCC results of SLSOM and existing methods on artificial datasets and real datasets.

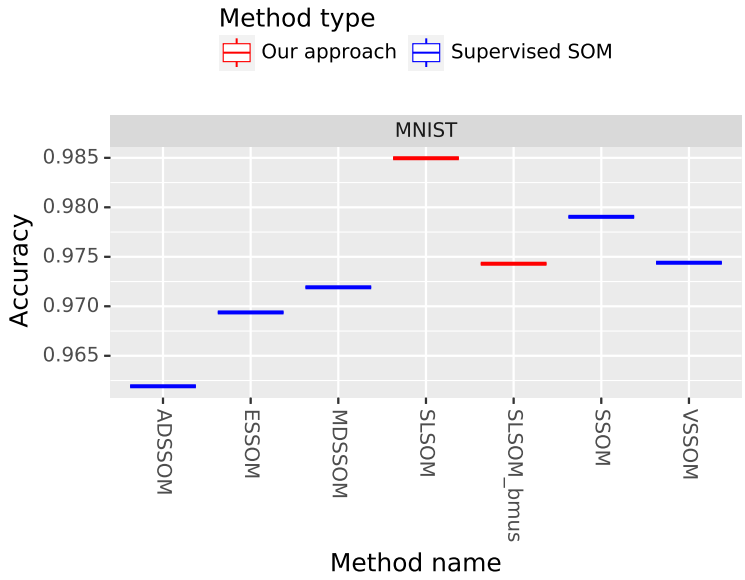


Figure 2.7: Accuracy of SLSOM and the supervised SOM on the MNIST dataset.

and (ii) to identify ambiguous areas using the ambiguity rejection.

Improving the performance

We divide the classification outputs into two decision regions G and E such that G represents the good predictions and E represents the misclassified predictions (errors). The use of rejection allows us to also divide the output space in a different way by defining two regions A and R such that A represents the accepted predictions and R the rejected ones. From these four regions, we can define two performance measurements for classification with rejection. These two measurements are the accurate prediction rate in the accepted predictions (Accuracy) and the misclassification rate in the rejected predictions (Misclassification). They are defined as follows:

$$\text{Accuracy} = \frac{|G \cap A|}{|A|} \quad \text{Misclassification} = \frac{|E \cap R|}{|E|} \tag{2.21}$$

Figure 2.8 shows the performance of the classifier with rejection option on Artificial 2 dataset. We varied the thresholds of the distance and ambiguity rules between 0 and 1. We can see that the accuracy of the prediction increases when the rejection rate increases. It has to be noticed that by increasing the reject rate, we increase the rate of misclassified predictions which are rejected. The figure shows that for a given accuracy or rejected misclassification, the local rejection options can produce the same or better result with less rejected predictions. For example, the global rejection option rejects 20% of the dataset (with 90% of misclassified data) in order to have an accuracy of 0.991, whereas the local rejection has an accuracy of 0.996 with 20% of rejected predictions (with 95 % of misclassified data rejected). This difference can be explained by the different distances between the clusters and their organization hence the need to have a different rejection threshold for each cluster.

Case study

The interest of SLSOM is its ability to identify and visualize potential new classes and ambiguous areas in the input data using local rejection. To show these properties, we split the second artificial dataset into training

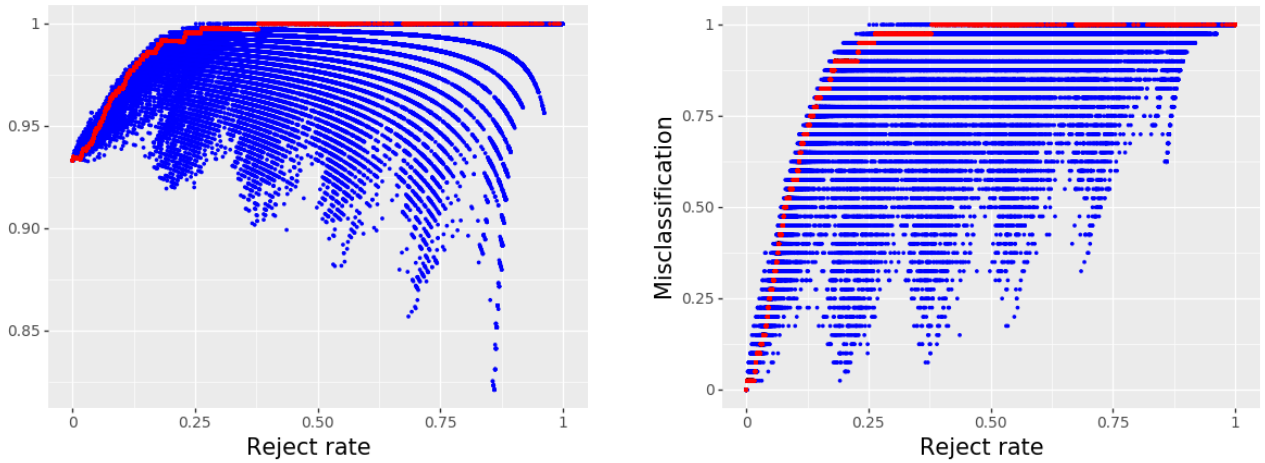
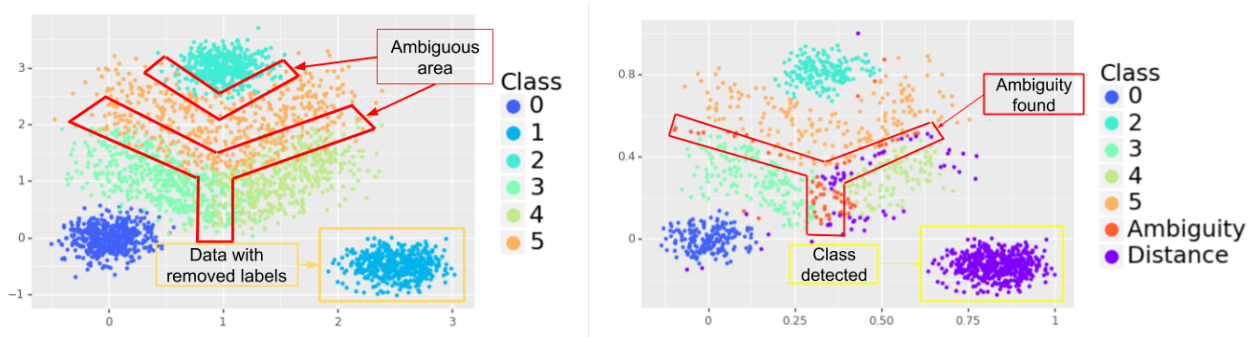


Figure 2.8: Rejection performance on Artificial 2 dataset. Left: Accuracy of the classifier with distance rejection option. Right: Misclassification rejected rate of the classifier with rejection option. The red points represent the values for the global rejection option and the blue ones represent the values for the local rejection option.

and test sets. The training set contains input data composed of all the classes and the class labels for all the classes except class 1. In the training step, the SOM layer was trained with input data coming from all the classes (without using labels) and the MLP is trained using labelled data (the class 1 is not represented in the training set). In this experiment, the class 1 represents the new class that our algorithm must discover with the distance rejection option. The set of threshold values used is described in Table 2.9. The threshold values were determined by analyzing the outputs of SLSOM. For the distance rule, we set the threshold values to the lower bound of the largest output values for each class. For the ambiguity rule, we set the threshold values to the lower bound of the difference between the largest and the second largest output values for each class.

ID set	Value of β_{max} for each class					Value of β_{diff} for each class				
	0	2	3	4	5	0	2	3	4	5
local threshold values	0.9	0.65	0.65	0.95	0.65	0.1	0.1	0.15	0.15	0.15

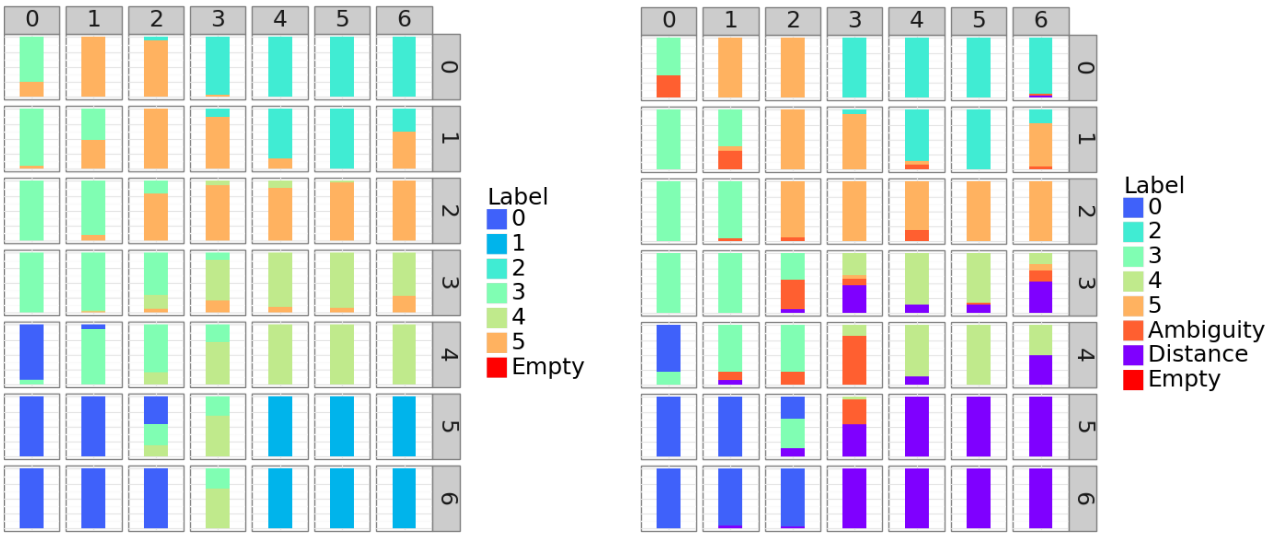
Table 2.9: Thresholds values used in the case study.



(a) With the true labels.

(b) With the rejected labels.

Figure 2.9: Plot of the 2D points.



(a) With the true labels.

(b) With the rejected labels.

Figure 2.10: Plot of the label repartition in the SOM for the Artificial 2 dataset.

Figure 2.9a and Figure 2.10a show respectively the 2D points and the true label repartition in the SOM computed on the whole training set of Artificial dataset 2. On Figure 2.9a we can see the overlapping areas induced by classes 3, 4 and 5. We can also see how far class 1 is from the other ones. On Figure 2.10a we show the true label repartition of the input data in the SOM. As expected, the input data of classes 0, 1 and 2 are located at the corners of the map. On the contrary, input data of classes 3, 4 and 5 are in the center of the SOM which also show the ambiguities identified in Figure 2.9a.
Figure 2.9b and Figure 2.10b show respectively the 2D points and the true label repartition in the SOM computed on the test set using the threshold values of the local rejection. In Figure 2.10b we can see on the bottom right of the map that some input data are rejected with the distance rejection. The proximities of these examples in the SOM suggest that they belong to the same class. We can verify that the rejected examples correspond to the class 1 represented in Figure 2.10a. We can also see in Figure 2.9b that the input data in ambiguous areas are rejected correctly with the ambiguity rule. Moreover, the rejection option didn't find noticeable ambiguity around classes 0 and 2. This result can be explained by the definition of classes 0 and 2 which are separable.

2.4 Conclusion

In this chapter, we presented a new approach of supervised SOM, where we take into account all the map prototypes to classify new patterns. In this approach, we combine a multilayer perceptron with a SOM. Experiments performed on several datasets of different sizes show very good results compared to other supervised SOM methods existing in the literature. These results are competitive (or better) with those obtained using efficient classification methods like SVM. To our knowledge, only one approach (called LISSOM) proposes to associate a supervised layer to SOM [21]. This approach is built for a specific application which is the recognition of hand-written digit from images. Our approach is more general since it is based on standard SOM and can be used in various applications. In addition, we propose to use a cross-entropy loss function that represents one of the most popular choices in state-of-the art implementations

and to add regularization term to the loss function in order to ovoid overfitting.
We also evaluated our supervised SOM with a rejection option. This new classifier with rejection option combines the distance and the ambiguity rejection options. The rejection allows us to improve classification results, to discover new classes and to analyze the samples belonging to the discovered classes. In addition, we have proposed to use local thresholds (one threshold for each class) in order to have more flexibility and precision in the rejection process. This allows us to improve the classification results especially for difficult classification problems. The only supervised SOM approach which uses rejection options is described in [110]. To achieve this, authors trained an unsupervised SOM and associated labels to the neurons by a majority vote. The vote results allowed them to compute the posteriori classes probabilities for each neuron. By using only the global ambiguity rejection based on Chow's method [23], they re-labelled the prototypes of the map. In this approach, a simple supervised SOM based on the majority vote is used with a global rejection option (we have shown that our approach outperforms this SOM variant).

Chapter 3

IRSOM, a reliable identifier of ncRNAs based on SOM

Here we present IRSOM [103], a new alignment-free method for discriminating non-coding and coding RNAs. IRSOM is a variant of SLSOM for ncRNAs. As SLSOM it is a supervised classifier composed of a Self-Organizing Map (SOM) and a perceptron layer which is fully connected to the SOM. The difference between IRSOM and SLSOM is that IRSOM uses backpropagation to improve the quality of the SOM. In IRSOM, we can use the backpropagation because we are not interested in the discovery of new classes and thus do not need the unsupervised classification of the SOM. IRSOM uses several features that are related to the sequence statistics (k-mers motifs frequencies, codon position biases, nucleotide frequencies and GC content) and the putative ORFs (coverage of the longest ORF, ORFs coverage distribution, start and end codon distribution, ORF frequency, ORF length and the frame bias).

The rejection in IRSOM allows to keep reliable predictions and to abstain in the situations where the predictions are unreliable. In IRSOM, we use the ambiguity rejection in order to identify the ambiguous transcripts that are on the boundaries between the coding and the non-coding RNAs. Moreover, by combining the rejection option with the SOM, we are able to visualize and analyse the rejected transcripts. For example, analysing the ORF feature profiles in the SOM allows to highlight the known differences between the coding and the non-coding RNAs and also shows the ambiguous characteristics of the rejected transcripts.

IRSOM was tested on datasets of several and different species (Human, Mouse, Arabidopsis thaliana, Zebrafish, Escherichia coli, Saccharomyces cerevisae and Drosophila), and compared to different existing tools: CPC2 [62], CPAT [125], CNCI [115] and PLEK [79]. It shows better or equivalent performances than the other tools in prediction results and comparable time consumption with the fastest tools (CPAT and CPC2). It gives an accuracy greater than 0.95 for almost all species, and reaches for some of them more than 0.99. It also demonstrates its capacity to handle very large datasets thanks to its low running time in prediction as well as in training (respectively less than 2 minutes and less than 4 minute on a Human dataset). In addition, IRSOM is able to visualize the data repartition into clusters for both the coding and the non coding RNA classes and to analyse the rejected classifications. The cluster profiles for each feature can also be visualized and analysed thanks to IRSOM.

3.1 IRSOM, an adaptation of SLSOM for ncRNAs

IRSOM is a three layers neural network (see Figure 3.1) composed of an input layer that represents the input data, a hidden layer which corresponds to a SOM [67], and an output layer (supervised layer), that consists of two perceptrons which are fully connected to the neurons of the SOM with forward connections.

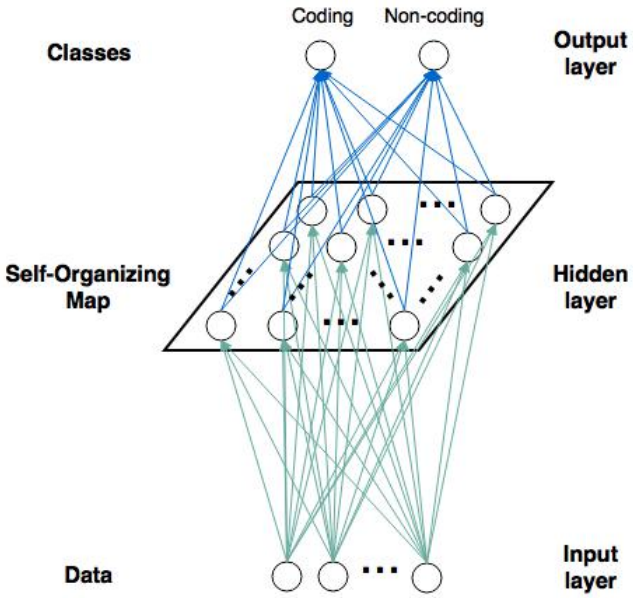


Figure 3.1: IRSOM architecture

3.1.1 IRSOM algorithm

The SOM computes a new representation of the transcripts using a map of neurons. The perceptron layer is able to assign correctly a class to a transcript by using its new representation. A backpropagation modifies the SOM organisation, in order to match with the classes of the transcripts. Moreover, we extend the perceptron layer with a rejection option where the ambiguous predictions are rejected. This rejection option highlight the transcripts which are between the coding and the non-coding RNAs.
Let a set of input data $X = \{x_1, x_2, ..., x_n\}$ and their corresponding labels $Y = \{y_1, y_2, ..., y_n\}$ such that $y_i \in \{0, 1\}^2$ is a vector representing the label of x_i. An element $y_i \in Y$ is defined such that:

$$y_i = \begin{cases} [1, 0] & \text{for coding RNAs} \\ [0, 1] & \text{for non-coding RNAs} \end{cases} \tag{3.1}$$

Learning step

The learning of the weights is divided into two parts, the forward propagation and the backpropagation. During the forward propagation step, the activation of the neurons in the different layers is propagated in order to compute the output and its related errors. The errors are then back propagated during the backpropagation step using gradient descent in order to update the network weights.

Forward propagation In this phase, at each iteration (one iteration corresponds to one batch), the units activations are propagated through the network to generate the output values. These outputs are then

compared to the input classes and the error is computed.

In order to keep the organization property of the map, the activation $a_{u,i}$ of the unit u depends on u and its neighbors u' such that:

$$a_{u,i} = \sum_{u' \in U} exp\left(-\frac{1}{2} \parallel x_i - w_{u'} \parallel^2\right) \sigma_t(u, u')$$
$$\sigma_t(u, u') = exp\left(-\frac{d(u, u')^2}{\tau \times \left(1 - \frac{t}{T}\right) \times r}\right)$$

where $d(u, u')$ is the Manhattan distance between the neuron u and u' and τ is a constant. The output $o_l \; \forall l \in \{0, 1\}$ of the two perceptrons are computed as:

$$o_{l,i} = sigmoid(\sum_u w_{l,u}^{out} a_{u,i} + b_l) \tag{3.2}$$

where w_{ul}^{out} is the connection between the perceptron l and the map unit u and b_l is the bias of the perceptron l and the sigmoid function is defined by:

$$sigmoid(x) = \frac{1}{1 + exp(-x)} \tag{3.3}$$

We use a loss function $L()$, which consists of the cross-entropy cost function $C()$ and a L2-norm regularization term. The regularization aims to improve the generalization of a learned model and to avoid overfitting. This liss function is calculated as follow:

$$L(Y, O) = C(Y, O) + \lambda \sum_u \parallel w_u^{out} \parallel^2 \tag{3.4}$$

where O is a vector containing the ouput of the perceptrons, λ is the parameter which controls the importance of the regularization term, and

$$C(Y, O) = -\frac{1}{N} \sum_i \sum_l y_{i,l} \ln(o_{l,i}) \tag{3.5}$$

Backpropagation This phase allows to calculate the error contribution of each unit after a batch is processed. During the backpropagation, all the weights of the network (the weights of the SOM and the perceptrons) are optimized to reduce the loss function $L()$. The gradient descent optimization algorithm is used to adjust the output weights and the SOM weights by computing the gradient of the loss function $L(Y, O)$. After the computation of the error, it is distributed back through the layers of the supervised SOM network. The weights of our neural network are optimized using the momentum optimizer such that:

$$w(t+1) = w(t) - \mu_1 \times \left(\mu_2 \times acc_w + \frac{\partial L(Y, O)}{\partial w}\right) \tag{3.6}$$

where w is a weight of the neural network, acc_w represents the sum of the gradient for this weight over the iterations, and μ_1 and μ_2 are constants controlling respectively the learning rate and the importance of the accumulation. We use the momentum optimizer instead of the gradient descent in order to avoid local optima. The local optima are avoided by using an accumulator term (acc_w) which represent the gradient descent direction over the previous iterations.

The gradients of the output weights are given by:

$$\frac{\partial L(Y,O)}{\partial w_{l,u}^{out}} = -\frac{1}{N}\sum_{i}\sum_{l} y_{i,l} \times (1 - sigmoid(\sum_{u} w_{l,u}^{out} a_{u,i} + b_l)) \times a_{u,i} + 2\lambda w_{l,u}^{out}$$

The gradient of the SOM weights is computed by:

$$\begin{aligned}\frac{\partial L\,(Y,O)}{\partial w_u} = -\frac{1}{N}\sum_{i}\sum_{l} y_{il} \times (1 - sigmoid(\sum_{u} w_{l,u}^{out} a_{u,i} + b_l)) \\ \times \sigma_t(BMU(x_i), u) \times (x_i - w_u) \times exp\left(-\frac{1}{2} \parallel x_i - w_u \parallel^2\right)\end{aligned}$$

Prediction step

During the prediction, we compute the output of the perceptrons using a slightly different activation function. For a given unit, the activation does not rely on its neighbors. We compute the activation a'_{iu} of the unit u for the transcript x_i by:

$$a_{u,i} = exp\left(-\frac{1}{2} \parallel x_i - w_u \parallel^2\right) \tag{3.7}$$

The activation of the unit u depends only on itself and the input x_i because during the prediction step we assume that the training step is finished. So the effect of the neighbors in the activation function is null ($\sigma_t(u, u') = 0$ for $u \neq u'$).

The output of the perceptrons $o_l \; \forall l \in \{0, 1\}$ is computed as follows:

$$o_{l,i} = sigmoid(\sum_{u} w_{ul}^{out} a'_{iu} + b_l) \tag{3.8}$$

The class of a transcript is determined by the maximal output of the perceptrons:

$$class(x_i) = \begin{cases} \text{coding RNA} & \text{if } o_{0,i} > o_{l,i} \\ \text{non coding RNA} & \text{otherwise} \end{cases} \tag{3.9}$$

Reject option

To improve the reliability of our method and identify the ambiguous transcripts, we use one of the rejection approaches proposed in [56]. The greater the difference between $o_{0,i}$ and $o_{1,i}$ is, the greater is the confidence in the prediction. By following the second rejection method in the article [56], we can improve the reliability of the prediction by rejecting the ambiguous classifications. We are able to define a classifier with rejection option called $\psi(x_i)$ such that:

$$\psi(x_i) = \begin{cases} -1 & \text{if } |o_{0,i} - o_{1,i}| < \beta \\ \arg\max_l \; o_{l,i} & \text{otherwise} \end{cases} \tag{3.10}$$

where β is the rejection threshold. When the absolute difference value between $o_{0,i}$ and $o_{1,i}$ is lower than a threshold β, the prediction is rejected and set to -1.

The parameter β is application dependent. For certain applications we may want a high β in order to have the most reliable predictions but in an exploratory analysis, we may use a smaller β in order to keep more

predictions even if they are potentially misclassified examples.

3.1.2 Features used in IRSOM

IRSOM algorithm is based on three types of features which are sequence bias, ORF statistics and K-mer motifs:

- Sequence bias: we used the codon position bias, the frequencies of each nucleotide and the GC frequency. The purpose of the codon position bias is to measure if there is nucleotide position bias in codons. It is computed as follows:

$$X_{pos} = \frac{min(X_1, X_2, X_3)}{max(X_1, X_2, X_3)}$$

  where for a given base X: X_1 is the number of X in positions 0, 3, 6, ...; X_2 is the number of X in positions 1, 4, 7, ...; and X_3 is the number of X in positions 2, 5, 8, ...

- ORF: we computed the length and coverage of the maximal ORF that are useful to access the information of the most probable coding sequence of the transcript. In order to rescale the ORF length, we defined the transformed ORF length such that:

$$\text{ORF length} = log_{10}(x)$$

  where x is the raw ORF length.

  We consider the mean and standard deviation of the length and coverage of all the possible ORFs. Moreover, we compute the mean and standard deviation of the start and end codon of all the possible ORFs in the transcript. We add also in our model the frame bias of the ORFs and the ORF frequency such that:

$$\text{Frame bias} = 1 - \frac{\min_{i\in\{0,1,2\}} |ORF_i|}{\max_{i\in\{0,1,2\}} |ORF_i|}$$

$$\text{ORF frequency} = \frac{\sum_{i\in\{0,1,2\}} |ORF_i|}{\text{Number of start codon}}$$

  where ORF_i represent the ORFs in the frame i.

- K-mer: the k-mers are all the words of size K that are contained in a string. Here we select the k-mers of size 3, 4 and 5 and compute their frequencies. We did not use k-mers with higher k because the computation coast induced was not worth the performance improvement.

3.2 IRSOM validation

3.2.1 Datasets

We evaluated our method with coding and non-coding RNAs coming from several species. In order to cover a large spectrum of species from different reigns, we selected RNAs from Human, Mouse, Oryza sativa, Arabidopsis thaliana, Zebrafish, Escherichia coli, Saccharomyces cerevisae and Drosophila. The number of transcripts and their origins are given in Table 3.1 and the distribution of the transcripts length is shown in Figure 3.2.

Species	Coding		Non-coding	
	Origin	Number	Origin (RNAcentral)	Number
Human	Ensembl 92	45 956	Gencode	30 171
Mouse	Ensembl 92	23 715	Gencode	17 582
Zebrafish	Ensembl 92	41 760	Rfam	13 885
O. sativa	Ensemble plants 38	42 362	Rfam	6 076
A. thaliana	Ensemble plants 38	19 228	Rfam, RefSeq, TAIR	7 036
S. cerevisiae	Ensembl 92	6 684	Rfam	1 355
Drosophila	Ensembl 92	13 928	Flybase	3 610
E. coli (K-12)	Ensembl bacteria 37	4 083	Rfam	1 058

Table 3.1: Benchmark datasets.

The sequences of coding transcripts were extracted from the Ensembl databases [135]. We selected only the transcripts available in the Swiss-Prot database [25] in order to have manually curated transcripts except for the Zebrafish and the Drosophila. Due to their low amount of transcripts, we selected the transcripts with ID in the UniParc[1]. The non-coding transcripts come from the RNAcentral database [24]. RNAcentral combines the information of multiple ncRNA databases (Ensembl ([135]), Rfam [60], RefSeq [90], GENCODE [29]).
In the case of the Human and Mouse datasets, we selected the ncRNAs coming from the GENCODE database due to their manual curation. For the Drosophila dataset, we selected the ncRNAs available in the Flybase database which is a reference database for the Drosophila. For the Arabidopsis thaliana, we got transcripts from 3 databases in order to have enough data: Rfam, RefSeq and TAIR [10] (a database specialized on Arabidopsis thaliana). We added the transcripts from the TAIR database because it is specialized on Arabidopsis thaliana. Finally, for the other datasets, we selected the ncRNAs coming from the Rfam database.

3.2.2 Experimental protocol

We show the performance of our tool by comparing it to four classical ncRNA identification tools which are CPAT [125], CPC2 [62], CNCI [115] and PLEK [79].
The benchmark was executed on a 50 cores virtual machine (VM) under debian with 128 Gb available memory. Each tool was launched separately on the VM with the sequences in fasta format.
We performed two types of performance analysis: a cross-validation analysis and a prediction analysis. For this purpose, each of the different datasets presented above, noted D, has been divided into two subsets (of the same size) D_{sub1} and D_{sub2}, D_{sub1} used in the cross-validation and D_{sub2} in the prediction.
During the cross-validation of IRSOM, we evaluated the impact of the rejection threshold on the different datasets. We therefore performed a 10-fold cross-validation with IRSOM, and then a prediction with the different tools, CPAT, CPC2, CNCI, PLEK and IRSOM.
Among the tested tools, only CPAT was retrained on all datasets. Unfortunately, the sources to build a new model for CPC2 and CNCI are not available, and PLEK is too slow to create new models (more than 2 days for a training on a dataset of 7 500 sequences with 10 threads). We did not run CNCI on the datasets of the Drosophila, Escherichia coli and Saccharomyces cerevisae because the available models were designed for

[1]UniParc is a non-redundant protein database.

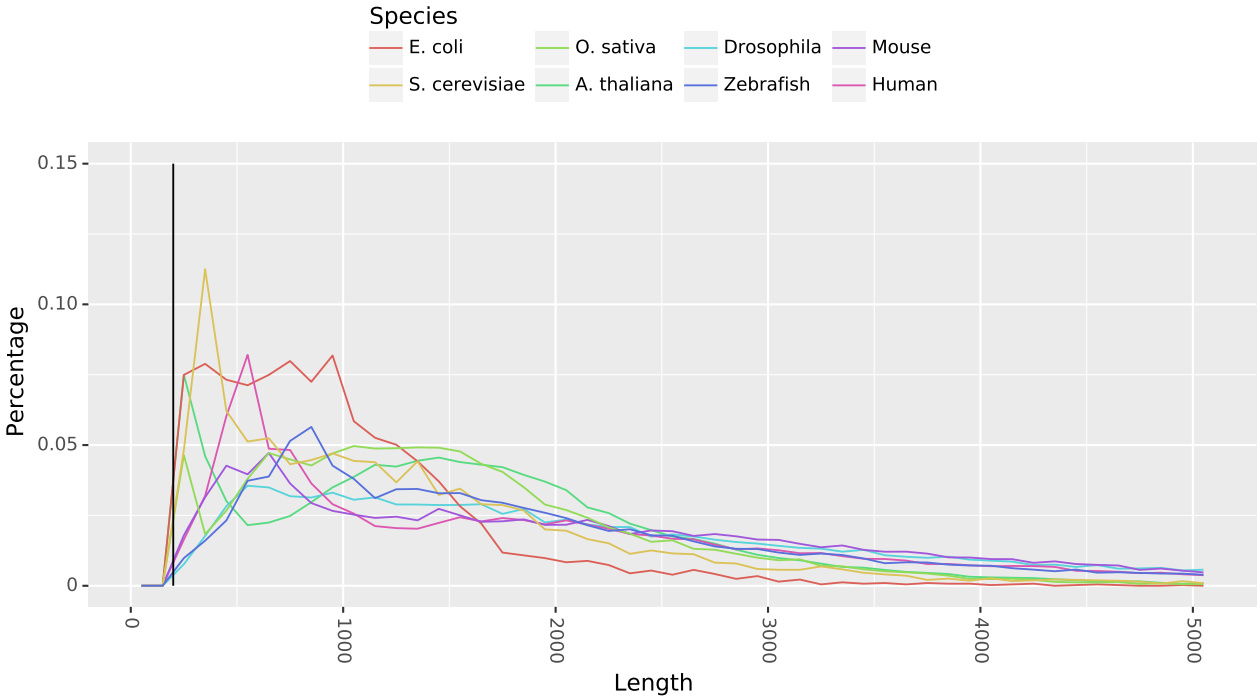


Figure 3.2: Distribution of transcripts lengths for all considered datasets. The black vertical line separates the transcripts smaller than 200 from the longest ones.

vertebrate or plant organisms only. PLEK authors built their model with a Human dataset but in their article [79], they show results on vertebrate species. We then run PLEK on the vertebrates datasets. Finally, IRSOM and CPAT are trained on each of the considered species, as well as on all species together (cross-species model). In this last case we note the two tools as IRSOM_cross and CPAT_cross respectively.
The default parameters of the tested tools have been used during the training and prediction as explained by the authors in their respective documentation. For each model computed with CPAT, we determined the threshold separating the coding and the non-coding RNAs by maximizing the sensitivity and specificity. For PLEK we set the minimal length of an accepted transcript to 20 in order to keep the small transcripts (which represent a small part of the datasets).
In case of our tool, there are different parameters. The SOM dimension which is a grid of size 10×10. The parameter τ in the σ function is set to 0.5. The regularization factor is set to 0.001. The size of the batches is set to 100. The learning constants μ_1 is set to 0.05 and μ_2 is set to 0.25. The upper limit of the number of iterations is set to 10 000, but the training step can end earlier if the difference between the loss function of two consecutive iterations is smaller than 10^{-6}. The values assigned to the different parameters are computed on the training set used for the cross-validation. We initialized the SOM in the hidden layer by the SOM learning algorithm.

3.2.3 Results

We measured the classification performance using three measures:

- Accuracy: represents the percentage of correctly classified RNAs:

$$Acc = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.11}$$

- Sensitivity: measures the rate of true positives:

$$Sensitivity = \frac{TP}{TP + FN} \tag{3.12}$$

- Specificity: measures the rate of true negatives:

$$Specificity = \frac{TN}{TN + FP} \tag{3.13}$$

where TP are the true positives, TN are the true negatives, FP are the false positives and FN are the false negatives. Here the positive class represents the non-coding RNAs and the negative one the coding RNAs. In the case of IRSOM, the TP, TN, FP and FN are computed on the non rejected data.

Cross-validation results

As mentioned above, we performed a 10-fold cross-validation with our tool IRSOM. For each species dataset D, we applied IRSOM on the subset D_{sub1} (as described in subsection 3.2.2). The obtained results are given in Figure 3.3 and show the performance of IRSOM for different rejection thresholds.

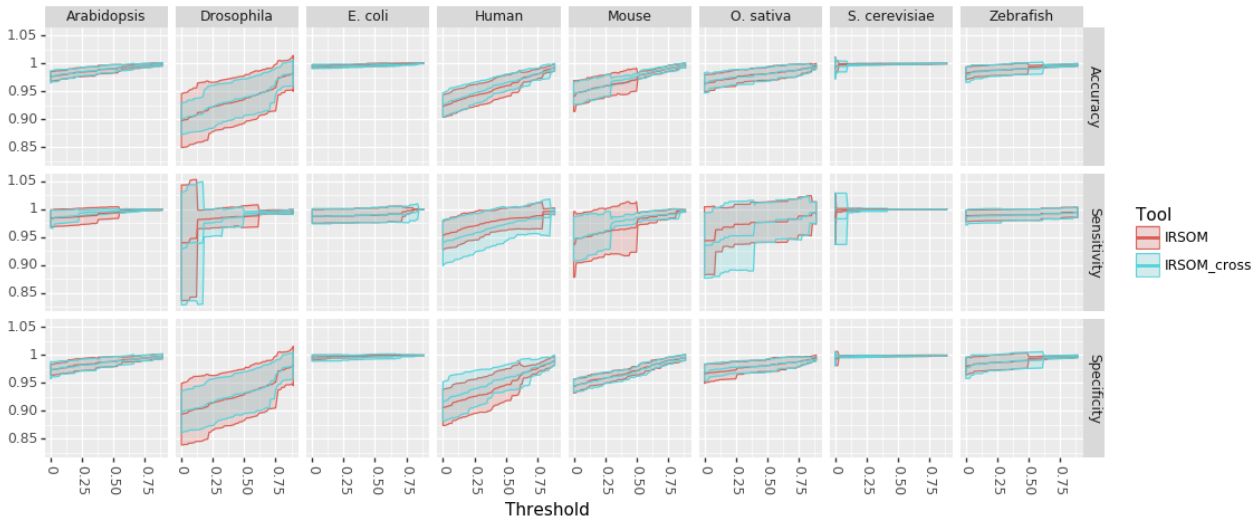


Figure 3.3: IRSOM performance (mean ± standard deviation) in regard to the rejection threshold for all the datasets.

As we can see on the Figure 3.3, the performance of IRSOM increases when we increase the rejection threshold. But as we can see in Figure 3.4, the number of rejected predictions increases when the rejection threshold increases.

In order to define a good rejection threshold, we need to find a trade off between the performance and the rejection rate. For the cross-species model of IRSOM, we set the rejection threshold at 0.7. This threshold gives good performance for all species (Accuracy greater than 0.975) with a reasonable rejection rate (less than 20% for all species). For the species specific model, we set a threshold for each species. We set the thresholds to values that show the highest performances. For the S. cerevisiae and E. coli, we set a threshold of 0.1 and 0.2 respectively. For the plants, we set a threshold of 0.6 for the Arabidopsis and 0.8 for the Oryza sativa. The eucaryotes species have a wider range of threshold. We set a threshold of 0.5 for the Zebrafish, 0.75 for the Drosophila and Mouse and 0.8 for the Human. For all the species, we have at most 30% of the

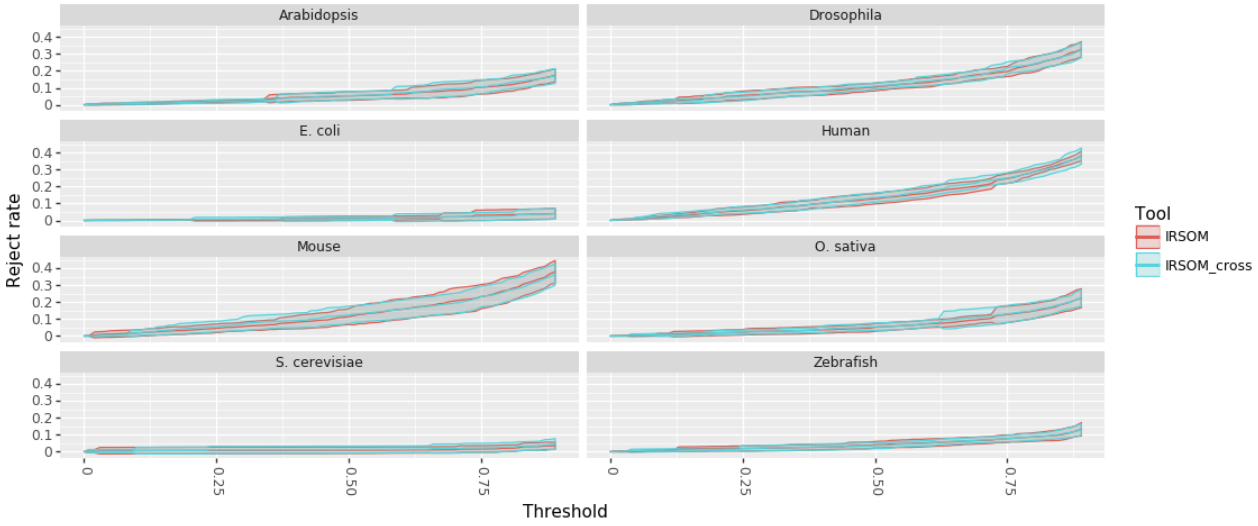


Figure 3.4: IRSOM rejection rate (mean ± standard deviation) in regard to the rejection threshold for all the datasets.

predictions that are rejected. For most of them, we have a rejection rate lower than 20% (A. thaliana thaliana and Oryza sativa) or even 10% (E. coli, Zebrafish and S. cerevisiae). It has to be noticed that the thresholds and rejection rate increase with the complexity of the species. This variation can be due to the potential higher ambiguity between the coding and the non-coding RNAs in complex organisms.

Prediction results

On each species dataset D, we performed predictions on the subset D_{sub2}, using the different tools IRSOM, CPC2, CPAT, CNCI and PLEK. The predictions were processed using the provided models in case of CPC2, CNCI and PLEK and with the models obtained after a training on the subset D_{sub1} in case of IRSOM and CPAT. With IRSOM and CPAT, we performed two types of tests, one where the corresponding model is used for the considered species, and one where the cross-species model is used for each of the species. Table 3.2 gives the models used by each of the tested tools.

Tools	CNCI	CPC2	PLEK	CPAT IRSOM	CPAT_cross IRSOM_cross
Models	Vertebrates and Plants	Coding: Human Non-coding: Human and mouse	Human	Retrained on each species	Retrained on cross-species

Table 3.2: Models used in the prediction performed by each of the tools CNCI, CPAT, CPC2, PLEK and IRSOM. CPAT_cross and IRSOM_cross designate respectively CPAT and IRSOM when used with the cross-species model.

Figure 3.5 shows the obtained results. For IRSOM, we use the rejection thresholds defined previously on the subset D_{sub1}. The tools retrained on our datasets, i.e. IRSOM and CPAT, are represented by full lines and the tools that we could not retrain (we used the provided models) are represented by dotted lines.
The obtained results show a good performance of our tool IRSOM compared to the other tools (see Table 3.3 and Figure 3.5). IRSOM exceeds 0.95 in accuracy for all the species. Compared to CPAT, the only tool we succeeded to retrain on our data and the second best tool, IRSOM shows slightly better results for all

Figure 3.5: Performance results obtained by CNC2, CNCI, CPAT, PLEK and IRSOM on each of Human, Mouse, A. thaliana thaliana, Oryza saliva, Zebrafish, Escherichia coli, Saccharomyces cerevisiae and Drosophila species.

considered species. Furthermore, the two models of CPAT show the same performance on all datasets as for the two models of IRSOM (except on the Human).

Method	Accuracy	Sensitivity	Specificity
CPAT	0.97	0.97	0.97
CPAT_cross	0.97	0.97	0.97
CPC2	0.91	0.97	0.90
IRSOM	**0.98**	**0.99**	**0.98**
IRSOM_cross	**0.98**	**0.99**	**0.98**

Table 3.3: Prediction mean performance of CPAT, CPC2 and IRSOM

CPC2 shows an accuracy greater than 0.9 except on the Zebrafish (0.88) and the O. sativa (0.8). These results are explained by the lower specifitciy on these datasets (0.84 for the Zebrafish and 0.77 for O. sativa). CNCI was used only on the vertebrates (Human, Mouse and Zebrafish) and plants (A. thaliana and O. sativa) datasets according to their models. CNCI shows lower results than CPAT, CPC2 and IRSOM on all the datasets, except on the Zebrafish where it gives better results than CPC2.

The tool that gives the worst results is PLEK. As mentioned above, the model provided by the authors is the result of a training on Human dataset. On the Human and Mouse datasets, PLEK shows suitable results. As prospected, the best results are those on Human (0.93). On the Zebrafish, we obtained an accuracy of 0.78 when the authors have shown in [79] an accuracy of 0.91.

Finally, IRSOM gives the best performances (accuracy mean: 0.98) compared to the other tools (CPAT (both models): 0.97, CPC2: 0.91). Every tool gives a nearly perfect sensitivity (IRSOM (both models): 0.99, CPAT (both models): 0.97 and CPC2: 0.97) but lower performance in term of specificity, except for IRSOM (0.98 for both models) and CPAT (0.97 for both models). These results suggest that all the tools are able to correctly

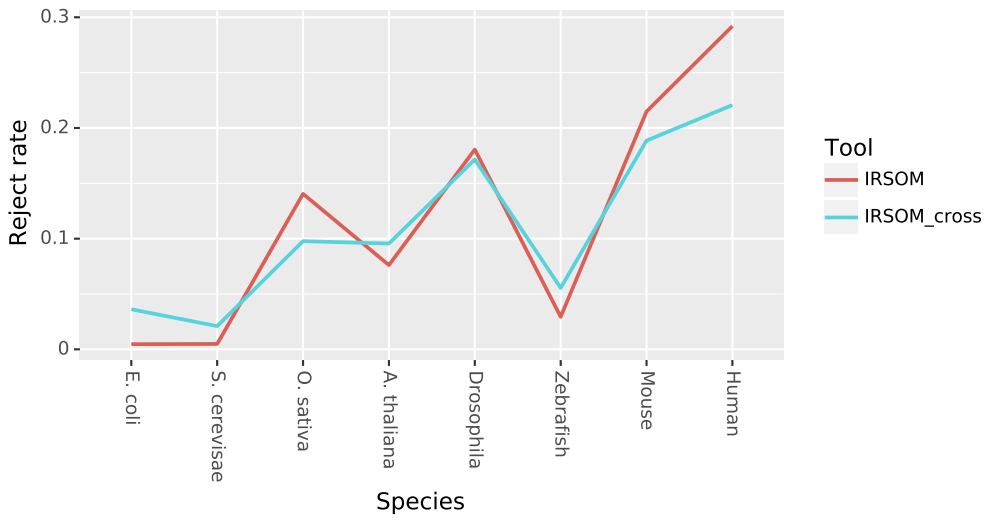
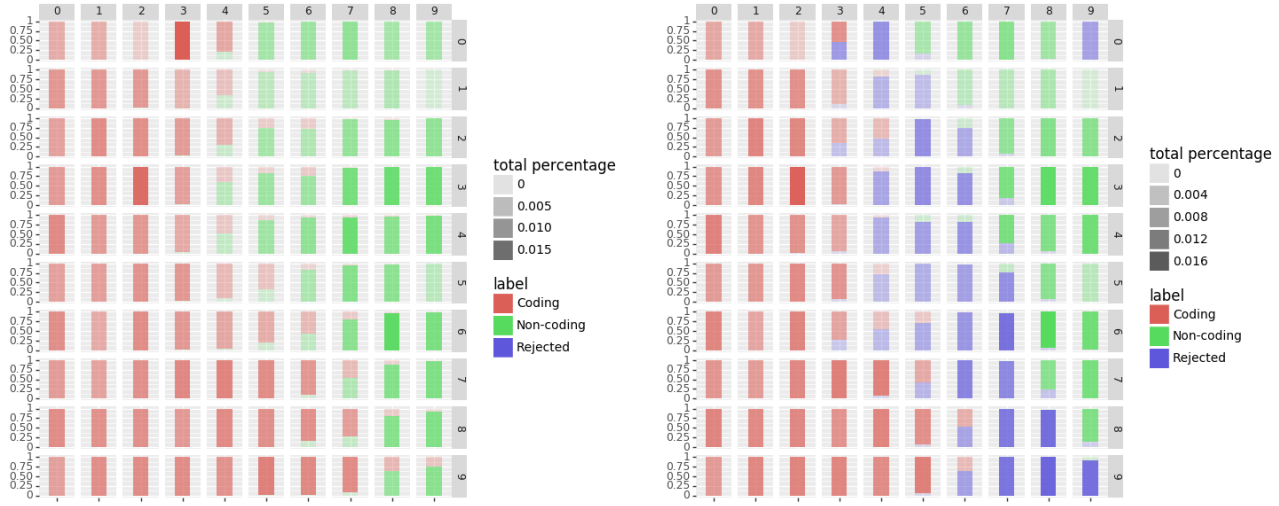


Figure 3.6: Rejection rate of both IRSOM models on all the datasets.

identify the non-coding RNAs, but predict wrongly a part of the coding RNAs. With our rejection option, we can identify and reject the prediction of the ambiguous transcripts and improve our specificity. Furthermore, the mean reject rate of both models is around 15% with a slightly higher reject rate for the species specific model (See Figure 3.6). These results confirm the ability of our method to gives accurate prediction and identify the ambiguous transcripts without rejecting too much data. In addition, the rejected transcripts can be visualized and analysed using the SOM.

Case study

Here we demonstrate how the rejection option improves the prediction results. To do so, we investigate the prediction by visualizing the SOM prototypes and the distribution of the labels in the SOM. One of the most interesting properties of SOM is its capacity to visualize the data by projecting it to a low dimensional space. By using this property, we can extract the profiles of the transcripts that are close to a neuron prototype by taking its representative.



(a) True label distribution.

(b) Rejected label distribution.

Figure 3.7: True and predicted label repartition in the SOM with the Human dataset

In our case, we look at the transcripts that are rejected in the Human dataset using the species specific model of IRSOM. The Figure 3.7a and Figure 3.7b show respectively the true label distribution and the predicted label (with rejection) distribution of the SOM in the hidden layer of IRSOM. We can see in Figure 3.7a that the map separates well the coding RNAs from the non-coding RNAs. In Figure 3.7b, we can see that the neurons assigned to the rejected predictions are at the boundary between the coding and the non-coding regions in the map, except for one neuron in the top right corner of the map. In the case of this neuron, the non-coding neuron output is too low and so the difference between the two output values is lower than the threshold used for the Human dataset (which is 0.8). As a reminder, a prediction is rejected if the difference between the coding and the non-coding neurons outputs is lower than a threshold β.

Figure 3.8 shows the profiles for the ORF features. We can see that the coding transcripts have a high ORF coverage while the non-coding transcripts have a lower or near zero ORF coverage as expected. Moreover, the coding transcripts have a low ORF frequency (close to 0) when the non-coding transcripts have a high ORF frequency (close to 1). These features mean that for a given number of start codon, there are more ORFs in the non-coding transcripts than in the coding ones.
The characteristics of the transcripts in the rejected area are more ambiguous. They show an ORF coverage of 0.5 with a high ORF frequency like the non-coding transcripts. The average ORF coverage with the high ORF frequency suggests that these transcripts have coding sequences that are not stable as the other coding transcripts. Moreover, these transcripts can potentially produce proteins as the coding RNAs and also have the same function as the non-coding RNAs.

3.2.4 Running time

We compare IRSOM's time performance the other tools. We measured the prediction times on each species dataset as well as on the cross-species dataset in order to see how the methods scale. The obtained running times on the different datasets, ordered from the smallest to the biggest one, are given in Table 3.4.

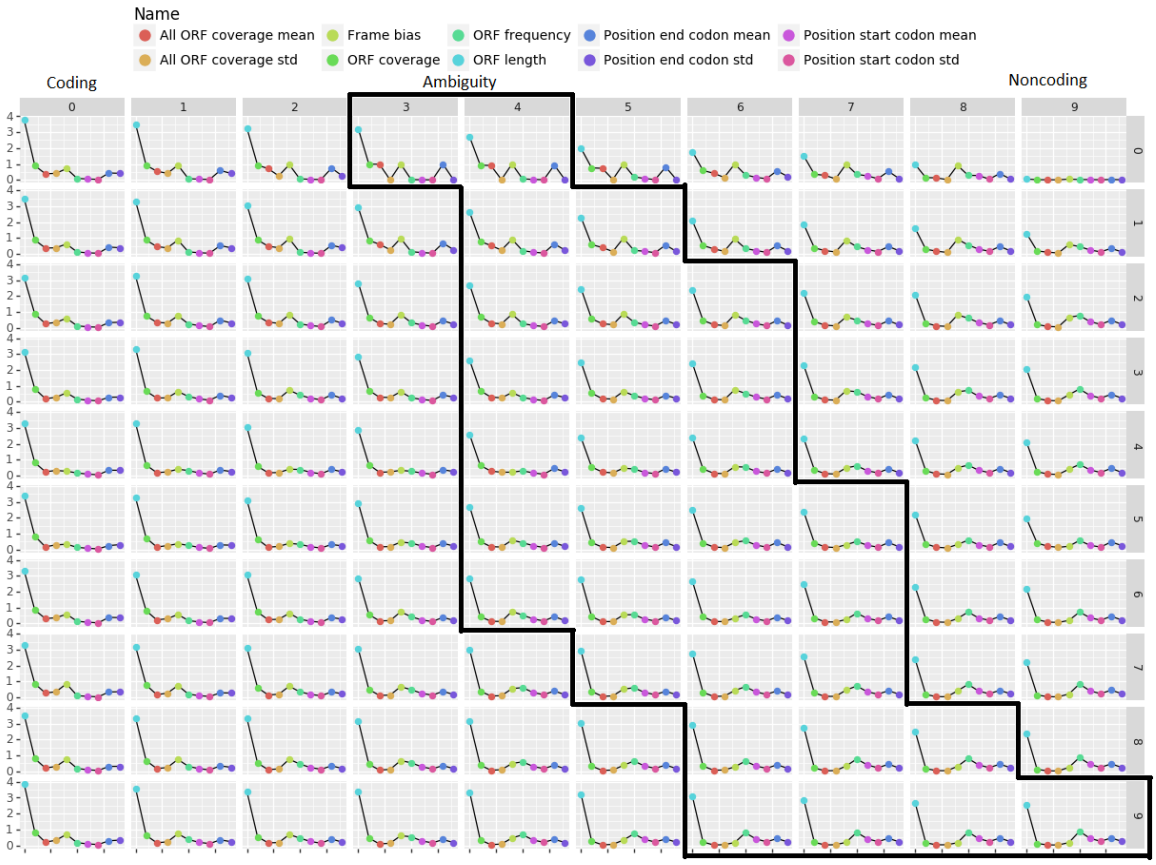


Figure 3.8: ORF profiles of the neurons for the human dataset where each point represents a value of the ORF feature.

Dataset	Dataset size	Running time (in seconds)				
		CNCI	CPAT	CPC2	PLEK	IRSOM
Cross-species	147 322		221	203	1315	254
Human	38 063	2 747	60	57	370	82
Zebrafish	27 782	1 074	41	36	240	64
O. sativa	24 187	520	30	28	202	59
Mouse	20 648	2 958	34	30	206	56
Drosophila	17 047		39	32	197	51
A. thaliana	13 112	316	20	17	120	43
S. cerevisiae	3 913		7	5	40	29
E. coli	2 570		4	3	33	27

Table 3.4: Prediction running times obtained on a 50-cores virtual machine (VM) under debian with 128 Gb of memory and 2.8 Ghz of CPU.

As we can see in Table 3.4, our tool IRSOM gives comparable running time compared to the existing tools. For instance, on the Human dataset, which is composed of around 38 000 sequences, it took less than four

minutes to generate the Human model and less than two minutes for the prediction.
The prediction time difference between IRSOM, CPAT and CPC2 (which are the two fastest) is due to the computation of the features. In our tool, the features are computed by a C++ executable and imported in a python script for the prediction (or training). By doing so, we are able to handle large volumes of data but induce an overheads. In CPAT and CPC2, everything is done in python and so there are no overhead. Finally, IRSOM is one of the fastest tool and gives comparable results to CPC2 and CPAT.

3.3 Conclusion

We presented here a new approach and tool for identifying rapidly and efficiently ncRNAs. Our tool, called IRSOM is able to accurately discriminate coding and non-coding RNAs. Furthermore, with our rejection option, we are able to identify the ambiguous transcripts and analyse them with the SOM. Compared to the state of art, our tool gives the best results on several species of different reigns. It gives also good time computing for small and large datasets.
By using the rejection option, we are able to increase the prediction accuracy. Moreover, we highlight the fact that the limit between coding and non-coding transcripts is not well defined. And so, the coding and the non-coding transcripts have to be seen as a range of transcripts instead of two separable types of transcripts.

Part II

Classification ncRNAs

Chapter 4

MSSOM, a multiple heterogeneous sources classifier

In many application areas, data of interest can be described using multiple heterogeneous sources. Each source can contain a partial information about the objects. With multiple information sources simultaneously available, it is a challenging task how to conduct integrated exploratory analysis. For example, on biological studies about genes, one objective can be the extraction of the sequences information but also their expression profiles and their related epigenetics markers. The main exploratory analysis approach of data is clustering. Clustering finds subgroups of objects that are similar.

The data sources can be of different types: numerical or complex (trees, graphs, sequences, etc.). The complex types can be represented using similarity or dissimilarity matrices. When dealing with heterogeneous mixed (numerical and complex) data sources, existing approaches transform all the data sources to one type (numerical vectors or similarity (dissimilarity) matrices) and apply adapted clustering algorithms. Few clustering algorithms address the problem of learning the weights associated to the sources [126], [137]. In addition, most of the clustering algorithms learn the same source combination for all the clusters.

In this work, we propose to address the clustering from heterogeneous mixed sources by representing the complex data sources using kernels and by keeping the numerical data sources without any transformation. In addition, we propose to learn the source weights at the level of a cluster. Instead of learning the same combination for the whole space, we learn a different kernel combination for each cluster. This makes sense in several applications since a cluster can represent a group or a class of objects that share the same characteristics (data sources) and different groups can have different characteristics. For this purpose, we use self-organizing maps (SOM) [67], which are able to perform clustering and to provide a useful visualisation of the clustered data.

This chapter is organized as follows: we first address the clustering problem using multiple heterogeneous sources and present the related works to multiple sources SOMs. Then we introduce our new multiple sources SOM algorithm called MSSOM. We show the efficiency of our algorithm by presenting results on artificial and real data.

4.1 Multiple sources clustering related work

The basic problem of exploiting multiple information sources for unsupervised learning approaches has been extensively studied in the literature [126] [96] [139]. This problem is often known as multi-view clustering which has been successfully applied in many applications. The information sources can represent heterogeneous data that are of different types: digital, texts, graphs, trees, categories, etc. Complex data can be represented using dissimilarity or similarity (kernel) matrices. A common approach to cluster these heterogeneous data is to convert all data types to the same type: numerical, dissimilarity or kernel matrices.

Classical clustering algorithms concatenate (or combine) all multiple sources into a single one [139] before performing clustering. This is not appropriate in many real world applications because the data sources may not have the same importance and some noisy sources can deteriorate the clustering results. To overcome such limitations, other approaches are proposed in the literature. These approaches can be classified into three groups as proposed in [131]: co-training, multiple kernel clustering, and subspace learning. Co-training approach [13] [11] [138] alternately maximizes the mutual agreement on two distinct sources of the unlabelled data. Multiple kernel clustering approach [126] [137] proposes to learn an optimal linear or non-linear combination of kernels for clustering. Subspace approach [18] aims to obtain a latent subspace with low dimension shared by multiple sources by assuming that the input sources are generated from this latent subspace.

Some approaches are proposed in the literature in order to deal with heterogeneous data sources using SOM. The most common approach is to combine the different sources before using the SOM. For numerical data sources, an augmented vector is formed by concatenating the vectors associated to each source. This vector is presented as input to classical SOM [124]. For complex data sources, each source is represented by a similarity matrix called kernel matrix. Symmetric positive definite kernel matrices encode the similarities between the objects of interest in their respective input space. The kernels allow to construct the same representation for all the sources in order to integrate them in the algorithms. The resulting kernels are combined using a fixed rule without any parameters (e.g., summation or multiplication of the kernels) and then presented to a kernel SOM.

To our knowledge, only one approach learns the combination parameters of the data sources. It is called Multiple Kernel SOM [91]. Each data source is represented by a kernel function and the algorithm learns the optimal linear combination of the kernels. A convex combination of the kernels is defined by:

$$K(x_i, x_j) = \sum_d \alpha_d K_d(x_i, x_j) \tag{4.1}$$

where $\alpha_d \in [0, 1]$ and $\sum_d \alpha_d = 1$. Following the general framework of kernel SOM, the prototypes can be written as a convex combination of the input data in a Hilbert space (called the feature space) H:

$$p_u = \sum_{u=1}^{U} \gamma_{ui} \phi(x_i) \tag{4.2}$$

where the application $\phi : G \to H$ is symmetric and positive. The squared distance between x_i and a prototype p_u is then computed by:

$$\| \phi(x_i) - p_u \|^2 = k(x_i, x_i) - 2 \sum_{j=1}^{n} \gamma_{uj} k(x_i, x_j) + \sum_{s,l=1}^{n} \gamma_{us} \gamma_{ul} k(x_l, x_s) \tag{4.3}$$

The multiple kernel SOM optimizes the weights in order to minimize the following energy function:

$$\varepsilon((\gamma_{ui})_{ui},(\alpha_d)_d) = \sum_{i=1}^{n}\sum_{u=1}^{U} h_{BMU_i,u}(t) \parallel \phi(x_i)^{\alpha} - P_u^{\alpha} \parallel_{\alpha}^2 \tag{4.4}$$

where $\phi(x_i)^{\alpha}$, P_u^{α} and $\| \, . \, \|_{\alpha}$ are used to emphasize that these quantities depend on α. The multiple kernel SOM learns the same kernel combination for each cluster. We propose to combine the different sources inside the algorithm and to represent the importance of each source locally for each cluster.

4.2 MSSOM algorithm

We propose a three layer architecture (see Figure 4.1).

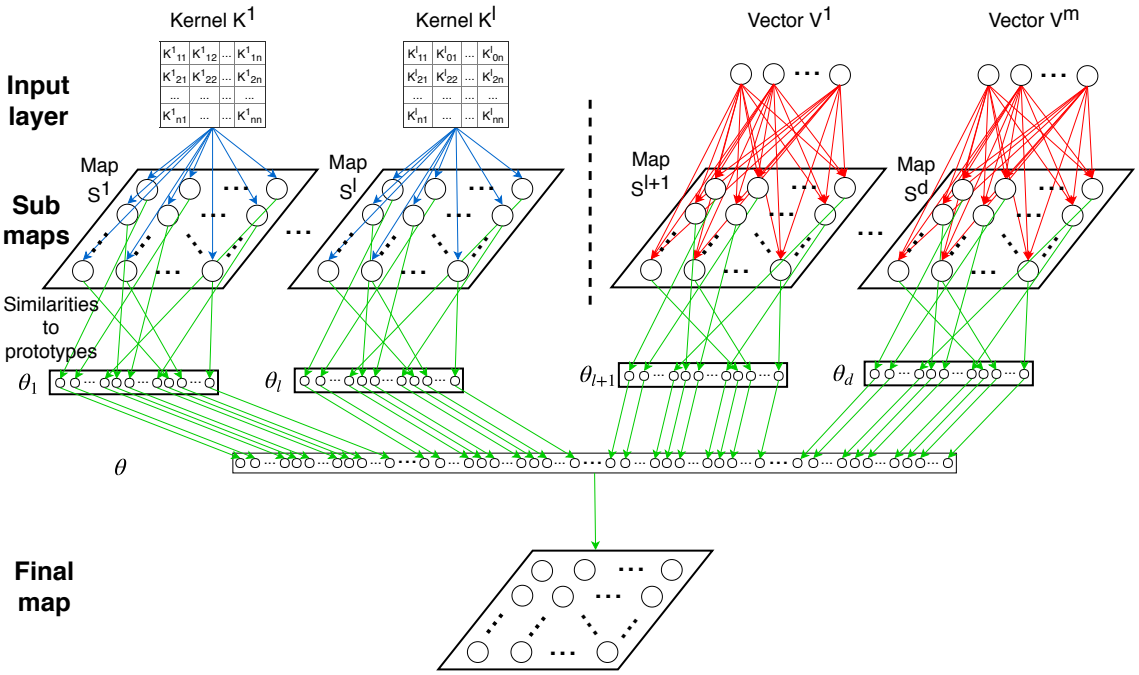


Figure 4.1: Multiple Sources SOM architecture

4.2.1 First layer

The first layer represents the input layer. Let be a dataset $X = \{x_1, x_2, ..., x_n\}$ where x_i is described using d sources $S = \{S^1, S^2, ..., S^d\}$. These sources can be represented by numerical vectorial data or by complex data. Let be $K = \{K^1, K^2, ..., K^l\}$ the kernels representing the complex sources and $V = \{V^1, V^2, ..., V^{d-l}\}$ the vectors representing the numerical sources where K^i represents the complex features of the source i and V^i the numerical features of the source $i + l$.

4.2.2 Second layer

The second layer is composed of SOMs computed from each source. Each SOM uses a algorithm that handle the source. For vectorial data we use a SOM that handles numerical data and for complex data we use a SOM that handle a kernel. In order to have better performances and reduce computation time consumption, we train the SOMs using the batch algorithm defined in [67]. Batch SOM learning algorithms are composed of two steps as the original algorithm called also iterative algorithm. The difference between the iterative and batch algorithms is that the iterative algorithms compute the assignation and update steps with one input data while the batch algorithms do it on all the input data. This difference makes the batch algorithms converge in fewer iterations than the iterative algorithms. The batch algorithms are also faster than the iterative algorithms when parallelism computation is used in each iteration.

The faster convergence is due to update rule of the batch algorithms. In the batch algorithms, we assume that the SOM will converge to a stable state. At this stable state, the weight vector of a given unit u at iteration t is equal to the one at iteration $t+1$. So in a stable state we have:

$$\forall u, \quad E_t\{h_t\left(BMU\left(x(t)\right), u\right)\left(x(t) - w_u(t)\right)\} = 0 \tag{4.5}$$

where t is the iteration step, $x(t)$ is the selected input data at step t, $w_u(t)$ is the unit weight vector at iteration t and E_t is the mathematical expectation value over the iteration.

With this equation we can define the update rule of the batch SOM algorithm. In this work, we use two batch SOM algorithms, one for the vectorial data which we call BSOM and one for the kernel data which we call BKSOM. For the vectorial data we follow the algorithm defined by Kohonen in [67]. The update rule of the BSOM is defined such that:

$$w_u\left(t+1\right) = \frac{\sum_i h_t\left(BMU\left(x_i, u\right)\right) x_i}{\sum_i h_t\left(BMU\left(x_i, u\right)\right)} \tag{4.6}$$

For the kernel data, we follow the algorithm defined in [15]. The kernel SOMs have a particularity, their unit neurons are a linear combination of the mapped input data such that:

$$w_u = \sum_i \rho_{u,i} \Phi(x_i) \tag{4.7}$$

where Φ is the kernel function and $\rho_{u,i}$ represents the linear combination coefficient of the neuron unit u for the input data i. The update rule of the BKSOM is defined such that:

$$\rho_{u,i}\left(t+1\right) = \frac{h_t\left(BMU\left(x_i, u\right)\right)}{\sum_j h_t\left(BMU\left(x_j, u\right)\right)} \tag{4.8}$$

Moreover, to reduce the time and memory consumption due to the kernels, we combine the BKSOM with the Bagged Kernel SOM presented in [86]. The idea behind the Bagged Kernel SOM is to reduce the dimensionality of the computed kernel. To do so, they generate multiple smaller kernels which they called bag and compute kernel SOMs on these bags. From each SOM, they extract the most representative input data such that they select the ones associated with the highest ρ value for each neuron units. Finally, they compute a kernel SOM with the most representative input data. In our combination that we call Bagged Batch Kernel SOM (BBKSOM), we replace the iterative kernel SOM of the Bagged Kernel SOM by it's batch version.

4.2.3 Third layer

The third layer combines the outputs of the SOMs presented in the second layer to train the single final map. The output of each SOM represents a similarity between the input data and the prototypes in the maps. We define a Gaussian similarity measure $s^r(x_i, p_u^r)$ between x_i, the i$^{\text{th}}$ element of X, and p_u^r, the u$^{\text{th}}$ prototype of the r$^{\text{th}}$ map such that:

$$s^r(x_i, p_u^r) = \exp\left(-\gamma \parallel x_i - p_u^r \parallel^2\right) \tag{4.9}$$

For the vectorial SOMs (BSOM), $\parallel x_i - p_u^r \parallel^2$ is computed using the euclidean. For kernel SOMs (BBKSOM), the distance between an input data x_i and a neuron unit u is computed by $\parallel \Phi^r(x_i) - p_u^r \parallel^2$ where $\Phi^r(x_i)$ is the kernel function of the source r for the input data x_i. This distance is computed using the kernel trick such that:

$$\begin{aligned}\parallel \Phi^r(x_i) - p_u^r \parallel^2 &= K^r(x_i, x_i) + \parallel \sum_j \rho_{u,j}^r \Phi^r(x_j) \parallel^2 - 2\Phi^r(x_i) \sum_j \rho_{u,j}^r \Phi^r(x_j) \\ &= K^r(x_i, x_i) + \sum_{j,j'} \rho_{u,j}^r \rho_{u,j'}^r K^r(x_i, x_j) - 2\sum_j \rho_{u,j}^r K^r(x_i, x_j)\end{aligned}$$

where $K^r(x_i, x_j)$ is the value of the kernel function Φ^r for the input data x_i and x_j.

We define a function $\theta_r : X \to \mathbb{R}^{U^r}$, returning the similarity between an element x_i in X and all the prototypes of the SOM for the source r such that:

$$\begin{aligned}\theta_r(x_i) &= [s^r(x_i, p_1^r), s^r(x_i, p_2^r), ..., s^r(x_i, p_{U^r}^r)] \\ &\quad for\ r = 1, ..., d\end{aligned}$$

Let be $\theta : X \to \mathbb{R}^{U^{total}}$ (with $U^{total} = \sum_{r=1}^{d} U^r$) a function returning a vector composed of the similarity between an element x_i in X and all the prototypes of all the maps (see Figure 4.1) defined by:

$$\theta(x_i) = [\theta_1(x_i), \theta_2(x_i), ..., \theta_d(x_i)] \tag{4.10}$$

With these definitions we can create a map S^{final} defined by U^{final} prototypes using the function $\theta(x)$ as an input.

Let be W the three dimensional $(U^{final} \times d \times U^j)_j$ structure containing the weights vectors of the prototypes such that $W = [w_1, w_2, ..., w_{Ufinal}]$ with $w_u = [w_{u,1}, w_{u,2}, ..., w_{u,d}]$ and $w_{u,i} = [w_{u,i}^1, w_{u,i}^2, ..., w_{u,i}^{U^i}]$. We define a two dimensional $(U^{final} \times d)$ matrix α containing the prototypes local source weights such that $\alpha = [\alpha_1, \alpha_2, ..., \alpha_{Ufinal}]$ with $\alpha_u = [\alpha_{u,1}, \alpha_{u,2}, ..., \alpha_{u,d}]$.

The BMU denoted $BMU(x)$ of the final map S^{final} is found by using the dot product of the input and the weight vectors:

$$BMU(x) = arg \max_{u \in U^{final}} \sum_{r=1}^{d} \alpha_{u,r} \left(\theta_r(x) \cdot w_{u,r}\right) \tag{4.11}$$

We define a new objective function f that we maximize, in order to create the final map S^{final} by learning the local source weights for each cluster (represented by the map prototypes), as follows:

$$f(x_i) = \sum_{u=1}^{U^{final}} \sum_{r=1}^{d} h_t(x_i, u) \alpha_{u,r} \left(w_{u,r} \cdot \theta_r(x_i)\right) \tag{4.12}$$

We update alternatively the final prototypes and the source weights using the gradient descent method as follows:

$$w_{u,r}(t+1) = w_{u,r}(t) + \mu(t)\frac{\partial f}{\partial w_{u,r}}$$
$$\alpha_{u,r}(t+1) = \alpha_{u,r}(t) + \mu'(t)\frac{\partial f}{\partial \alpha_{u,r}}$$

where $\mu(t) = 1 - \frac{t}{T}$ and T is the maximal number of iterations. The gradients of W and α are computed as follows:

$$\frac{\partial f}{\partial w_{u,r}} = h_t(x_i, u)\alpha_{u,r}\theta_r(x_i)$$
$$\frac{\partial f}{\partial \alpha_{u,r}} = h_t(x_i, u)\left(w_{u,r}.\theta_r(x_i)\right)$$

The vectors $w_{u,r}$ and $\alpha_{u,r}$ are then normalized by fixing the Euclidean norm of the prototype weights and source weights vectors to 1.
The originality of our approach lies in including the local source weights optimization in the SOM learning algorithm. The convergence of our approach depends of the convergence of the final map (the third layer). As in the multiple kernel SOM [91], the cost induced by the learning of the α parameter is moderate. In practice, we multiplied by five the number of iterations necessary to train the classical SOM in order to insure the convergence of our learning algorithm.

4.2.4 Discussion

As presented before, we combine the information of the different sources using their respective SOM inside the algorithm and create a final SOM with their outputs. We associate to each cluster obtained in the final SOM a prototype representing the local weights of the sources associated to the different SOMs. Compared to MKSOM[91], the only one SOM algorithm in the literature which learns a global source weights for all the clusters, our algorithm learns local weights of the data sources for each cluster. Localized multiple kernel learning approach was recently applied to k-means clustering [46], [78], [128]. Lei et al. [75] proposed an approach that can be formulated as a convex optimization problem over a given cluster structure. Instead of giving the same kernel weights for all input instances, the kernel combination is sample specific. Different kernel weights are calculated at the sample level rather than at the group level. In spite of the performance improvements proposed by these methods, the learning of a very large number of parameters leads to very expensive computation. Moreover, it is difficult to interpret the obtained results. In our approach, we associate a different sources combination to each cluster in order to distinguish it from the other clusters. This modelling makes sense in several application domains. Different categories of objects do not necessarily share the same importance of data sources. To our knowledge, two approaches have been proposed in this sense. In [132], Yang et al. incorporate the notion of group in the MKL (Multiple Kernel Learning) framework using support vector machines for objects categorization. In [88], Mu and Zou use the graph embedding framework to tune the kernel weights that vary at the cluster level. They introduce in their work a non-uniform MKL. Finally, our method allows to learn mixed source weights at the cluster level in a self organizing map, which performs clustering as well as data visualization. This is very useful since it allows to interpret and explain the clustering results.

4.3 MSSOM validation

4.3.1 Datasets

We evaluate our approach called MSSOM on six datasets (see Table 4.1). We generate two artificial datasets in order to show the interest of our approach and we use four datasets obtained from the UCI database [81] for the evaluation.

Dataset	#Instances	#Attributes	#Classes
Artificial 1	800	4	4 balanced classes
Artificial 2	800	4	4 balanced classes
Dermatology	366	34	6 classes (1:112, 2:61, 3:72, 4:49, 5:52, 6:20)
E.coli	336	7	4 classes (1:143, 2:116, 3:52, 4:25)
Iris	150	4	3 balanced classes
WDBC	685	10	2 classes (1:458, 2:241)

Table 4.1: Overview of the considered datasets.

The first artificial dataset is composed of four clusters of 4-dimensional points. For each dimension there are two clusters that can be separated from the other ones and so this dimension is informative for these clusters. The separated clusters are represented by Gaussian distributions with a standard deviation of 0.1 with two different centers (-5 and 5). The other non-separated clusters (noise) are represented by uniform distributions between -4 and 4. For the second artificial dataset, we use also four clusters containing 4-dimensional points. In this dataset, each cluster has one informative dimension. As for the first generated dataset, the separated clusters are represented by Gaussian distributions (with a standard deviation of 0.1 and a center of 1). The non-separated ones are obtained using a uniform distribution between -5 and 0. In Table 4.2 we show the informative dimensions for each cluster for both the datasets.

	Informative dimension							
	Artificial 1 dataset				Artificial 2 dataset			
Cluster id	0	1	2	3	0	1	2	3
0	I	I	N	N	I	N	N	N
1	I	N	I	N	N	I	N	N
2	N	I	N	I	N	N	I	N
3	N	N	I	I	N	N	N	I

Table 4.2: Clusters informative dimension for the first and second artificial datasets where N stands for noise and I stands for informative.

4.3.2 Protocol

To show the ability of our approach MSSOM to handle the mixed datasets, we used three representations of attributes (sources) in the considered datasets:

1. Datasets containing only numerical data: Each source is represented by a numerical attribute.

2. Datasets containing only kernels: We represent each attribute of the dataset by a Gaussian kernel defined by: $K(x, y) = \exp(-\gamma \times \|x - y\|^2))$. In order to select the parameter σ, we follow a simple rule defined in [50]. The value of parameter γ is computed as follows: $\gamma = \frac{\sqrt{2*U}}{dist_{max}}$ where U is the number of neurons and $dist_{max}$ represents the maximal distance between the instances.

3. Mixed datasets containing kernels and numerical data: In this case, we represent, for each dataset, some attributes using numerical values and others using kernels.

In the following, we will note by MSSOM_num the variant of our approach where only numerical attributes are used, MSSOM_kernel the one where only kernels are used and MSSOM_mix the one that uses mixed data sources. We compare the results obtained by the different variants of our approach to the SOM [67] which uses the numerical attributes and to MKSOM [91] which uses the kernels. The three methods use two common parameters which are the grid dimension and the maximal number of iterations. For all the datasets we use a 3x3 grid and the maximal number of iterations is equal to five times the number of instances. We used small grids in order to highlight the clustering capability of our approach. However, our method works well with higher map sizes and produces the same kind of results as classical SOM. The other parameters of MKSOM are selected by doing a grid search. MSSOM and SOM methods use the same neighbourhood function. For MSSOM_kernel and MSSOM_mix we use we used bags of 100 input data for each kernel sources. The bags were generated such that an input data is part of approximatively 30 bags in order to have a stable estimation of the representatives input data.

To measure the clustering performance, we use the Normalized Mutual Information (NMI) and the purity measures. The NMI measures how the classes are distributed in the clusters. A high NMI means that the clusters tend to contain data of one class and the class tends to be in one cluster. The purity measures the distribution of classes in the clusters. A high purity means that the clusters tend to contain only one class of data. These measures are defined by:

$$NMI = \frac{2 \times \sum_{i=1}^{U} \sum_{j=1}^{C} N_{ij} \times \log\left(\frac{N}{N_i \times N_j}\right)}{\sum_{i=1}^{U} N_i \times \log\left(\frac{N_i}{N}\right) + \sum_{j=1}^{C} N_j \times \log\left(\frac{N_j}{N}\right)}$$

$$Purity = \frac{1}{N} \sum_{i=1}^{U} \max_{j} (N_{ij})$$

where N is the number of instances, N_i is the number of instances associated to the cluster i, N_j is the number of the instances in class j and $N_{i,j}$ is the number of instances of class j associated to the cluster i.

4.3.3 Results

Figure 4.2 shows the performance of our methods against two state-of-art methods. On the generated datasets as well as on the real world datasets, the different variants of our method give the best results. Our method gives an NMI and a purity close to 1 for both artificial datasets. The performance of MSSOM is not sensitive to the different attributes (sources) representations. This means that all the variants of our MSSOM are able to well separate the clusters. These good results can be explained by the ability of our method to associate an adequate source combination for each cluster instead of one combination for the whole space.
We can see that all the variants of MSSOM give good results on the real datasets compared to the SOM state-of-art methods. For example, on the Iris dataset, our method gives an NMI greater than 0.7 and a purity

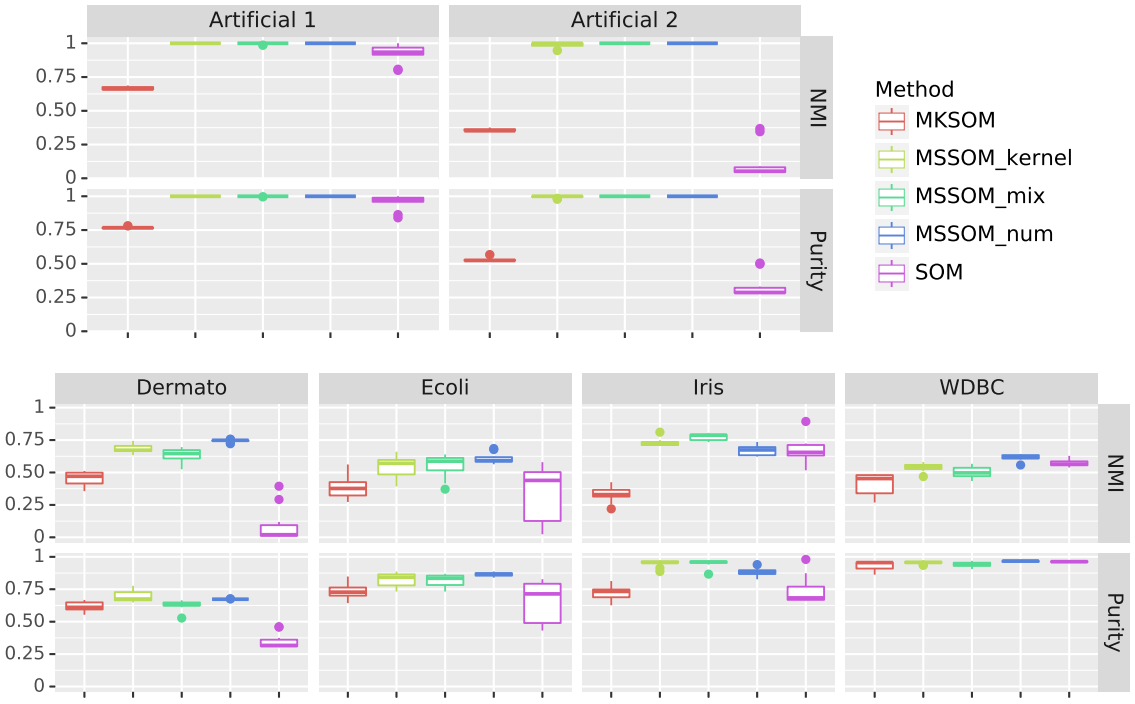


Figure 4.2: Performance of the different methods on the artificial and real datasets.

greater than 0.9 (for the three variants) when the classical SOM shows an NMI of 0.7 and a purity lower than 0.8 and the MKSOM shows an NMI lower than 0.4 and a purity lower than 0.8. This confirms that learning the local source combinations for each cluster allows to improve the clustering results.

Figure 4.3 shows the source weights for each unit (neuron) in the final map for Artificial 1 and Artificial 2 datasets, as well as the distribution of the examples among the map units (neurons). We can see that the units 0: $(0, 0)$, 2: $(0, 2)$, 6: $(2, 0)$ and 8: $(2, 2)$ represent well the four classes. We can also see clearly that for the four units representing the classes, the informative sources have the greatest weights. For example, on the Artificial 1 dataset, the unit 0 which contains the inputs of class 0 has sources weights close to 0.4 for the first and the second sources and close to 0.1 for the other sources. The sources with the biggest weight values correspond to the informative ones. On the Artificial 2 dataset, the unit 2 which contains the inputs of class 1 has approximatively a source weight of 0.56 for the second source which is the informative one. The weight values of the other sources are around 0.1. The MKSOM method gives the same kernel weights for all the clusters. It gives the greatest weight for the fourth sources in Artificial 1 dataset, and for the second source in Artificial 2 dataset. The other weight values are close to 0. This does not make sense on our artificial datasets.

In Figure 4.4 we propose a new informative visualization of our approach MSSOM, where we represent the source weights in the final map. Each neuron in the final map is represented by a set of four subplots (one for each source). For a given neuron in the final map, each subplot allows to identify for each cluster its representative neuron in the sub maps.

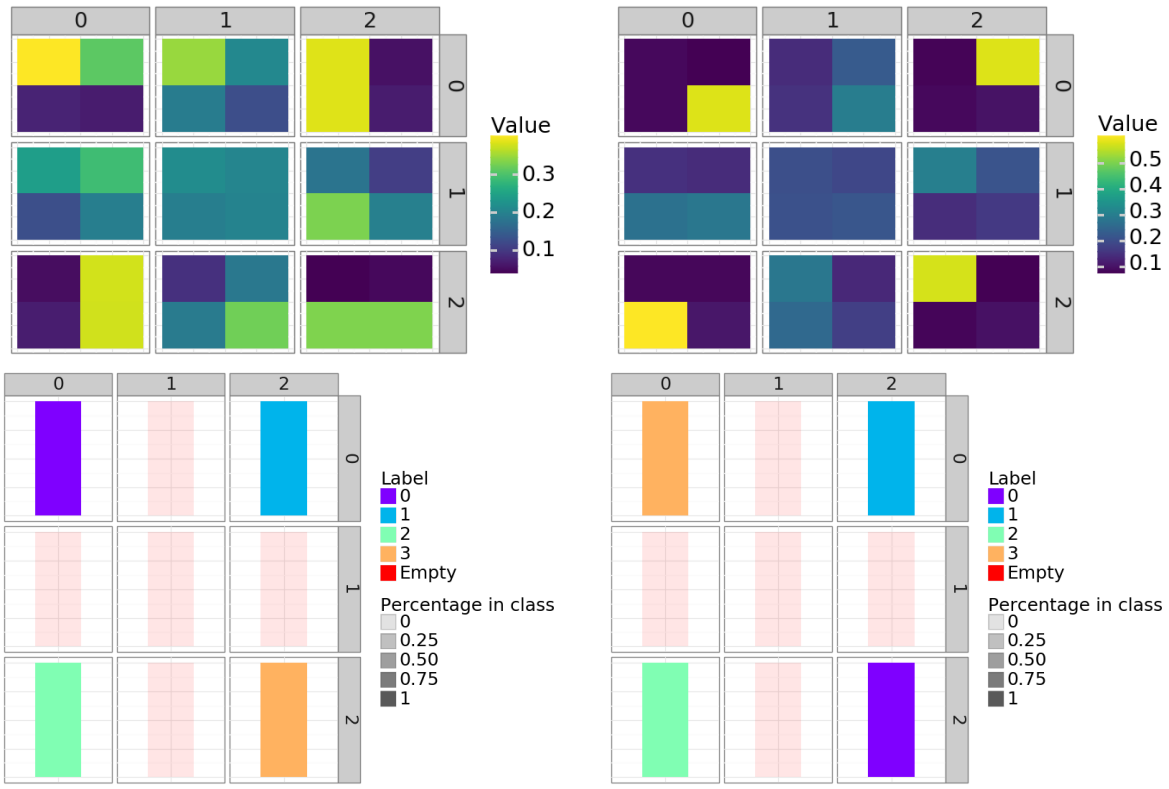


Figure 4.3: Sources weights (top) and label repartition (bottom) of the final map for Artificial 1 dataset (left) and Artificial 2 dataset (right) using the MSSOM_kernel.

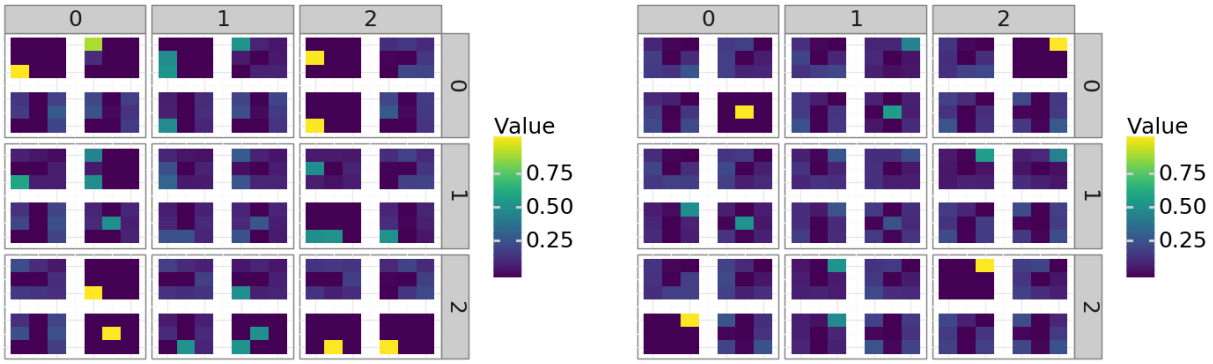


Figure 4.4: Visualization of the final map weights of MSSOM on the kernel version of Artificial 1 (left) and Artificial 2 (right) datasets.

4.4 Conclusion

In this work we present a new approach based on SOM which is able to combine heterogeneous (numerical and complex) data sources and to learn the source weights locally for each cluster. This property is very important for ncRNAs classification because with this property, we can compute for each ncRNA class a specific source weight. We show, using artificial datasets, that our method can select informative sources for each cluster. We show that our method gives good results and is competitive to other SOM approaches. Moreover, our approach can handle hybrid data sources by representing in an efficient way the numerical and complex datasets. Experimental results show that our approach is not sensitive to the kind of inputs we use (kernels, numerical or mixed data). In all the cases, we obtain approximatively the same performances.

Chapter 5

CRSOM, a classifier of ncRNAs based on heterogeneous data sources

In this chapter, we propose a supervised classifier using heterogeneous sources. These sources can represent complex data sources using kernels and numerical data sources without any transformation. As in MSSOM, our approach learns the source weights localy at the level of a cluster. This new supervised classifier is composed of rejection options in order to improve the reliability of the predictions and to identify potential new classes. For this purpose, we use self-organizing maps (SOM), multi layer perceptrons (MLP) and a rejection option. This chapter is organized as follows: we first present the related works to the supervised classification using multiple heterogeneous sources. Then we present a new supervised classifier called CRSOM, that is a combination of MSSOM [100] (see Chapter 4) and SLSOM [102] (see Chapter 2). Finally, we present the performance of CRSOM for the classification of ncRNAs.

5.1 Multiple sources supervised classification related work

In this section, we present how supervised learning can be performed using multiple sources of data. We can distinguish three levels of data fusion: early data integration where the sources are combined (or concatenated) into a single source before building the model [140][70], intermediate data integration where the sources are combined through inference of a global model [42], and late data integration where one model is generated for each source or for each sources combination and then these models are combined into one global model [106] [121] [134] [130] [61]. We focus on the kernel-based methods for data fusion called multiple kernel learning (MKL) [45]. MKL methods use a (linear or nonlinear) kernel combination of kernels in order to improve the results of the methods that use a single kernel. They can be classified as intermediate data integration approaches since each source is represented by a kernel. Every kernel method can be potentially extended to the MKL framework. MKL methods can be applied to perform: classification, regression, clustering, multi-sources learning, etc. Existing MKL methods can be cetegorized using six properties [45]: the learning method, the functional form, the target function, the training method, the base learner, and the computational complexity. The kernel methods can use different approaches to compute the kernel combination. In [45], the authors have divided the existing approaches into five categories:

- Fixed rules approaches that dot not need any parameter learning. These approaches are limited to define a fixed combination (e.g., summation or multiplication of the kernels) before training.

- Heuristic approaches that find the combination parameters for each kernel by looking the performance results obtained by training the model using each kernel separately.
- Optimization approaches that lean the kernel combination parameters in the learning step by solving the related optimization problem.
- Bayesian approaches that define priors on the combination parameters that are considered as random variables before performing inference.
- Boosting approaches that add a new kernel iteratively to the combination until the optimal performance is reached.

The majority of the Multiple Kernels Learning (MKL) approaches [32, 9, 133, 14] associate one global kernel combination for all the predictions. If we use kernel learning for classification, each class will be represented by the same kernel combination. To overcome this drawback, the authors of [44] present an approach based on local source weights learning. The authors propose a new SVM MKL where different kernel weights can be set for different regions of the input space. The input space regions are delimited by gating functions.
In our work we were interested by using neural networks as base learner. Recent works about deep learning data fusion define what they call multi-modal structure deep neural networks [111] [80]. They propose a new way to integrate heterogeneous sources (views) and capture their high level associations. The principle of these approaches is to learn a sub-network for each source and then integrate the outputs of the learned sub-networks in a common higher layer of the hole network.

5.2 Multiclass classification algorithm based on a combination of MSSOM and SLSOM

We propose here a new approach called CRSOM, to classify using multiple heterogeneous data. CRSOM combines MSSOM [100] and SLSOM [102] to produce a powerful supervised classifier for data analysis. This new classifier combines the different sources using SOMs. As in the multi-model approaches, we compute a SOM for each data source and combine the obtained SOMs into them with a final SOM. The supervised part is represented by a multi layer perceptron (MLP) with a rejection option.
The combination of MSSOM and SLSOM gives to CRSOM two interesting properties. The first property is the exploitation of multiple heterogeneous sources of data. We showed previously that MSSOM is able to use multiple sources of data to compute a SOM by combining each source at the neuron level. We also showed that MSSOM is able to identify for each class its relevant data sources. The second property is the classification of known classes and the identification of potential new classes. We show previously that SLSOM is able to accurately identify known classes. We also show in SLSOM how we can identify potential new classes by combining rejection options and the SOM.
CRSOM is a neural network composed of two major parts. The first one is the unsupervised part which is composed of the MSSOM. The second one is the supervised part of SLSOM. The resulting network is composed of at least five layers (in Figure 5.1 we illustrate an architecture example of CRSOM with two hidden layers in the supervised part).

1. The first layer, called input layer, represents the input data. As in MSSOM, the first layer of CRSOM can represent vectorial and non vectorial data.

2. The second layer contains the sub-maps computed by MSSOM.

3. The third layer is the final SOM produced by MSSOM.

4. The last layers represent an MLP as in SLSOM.

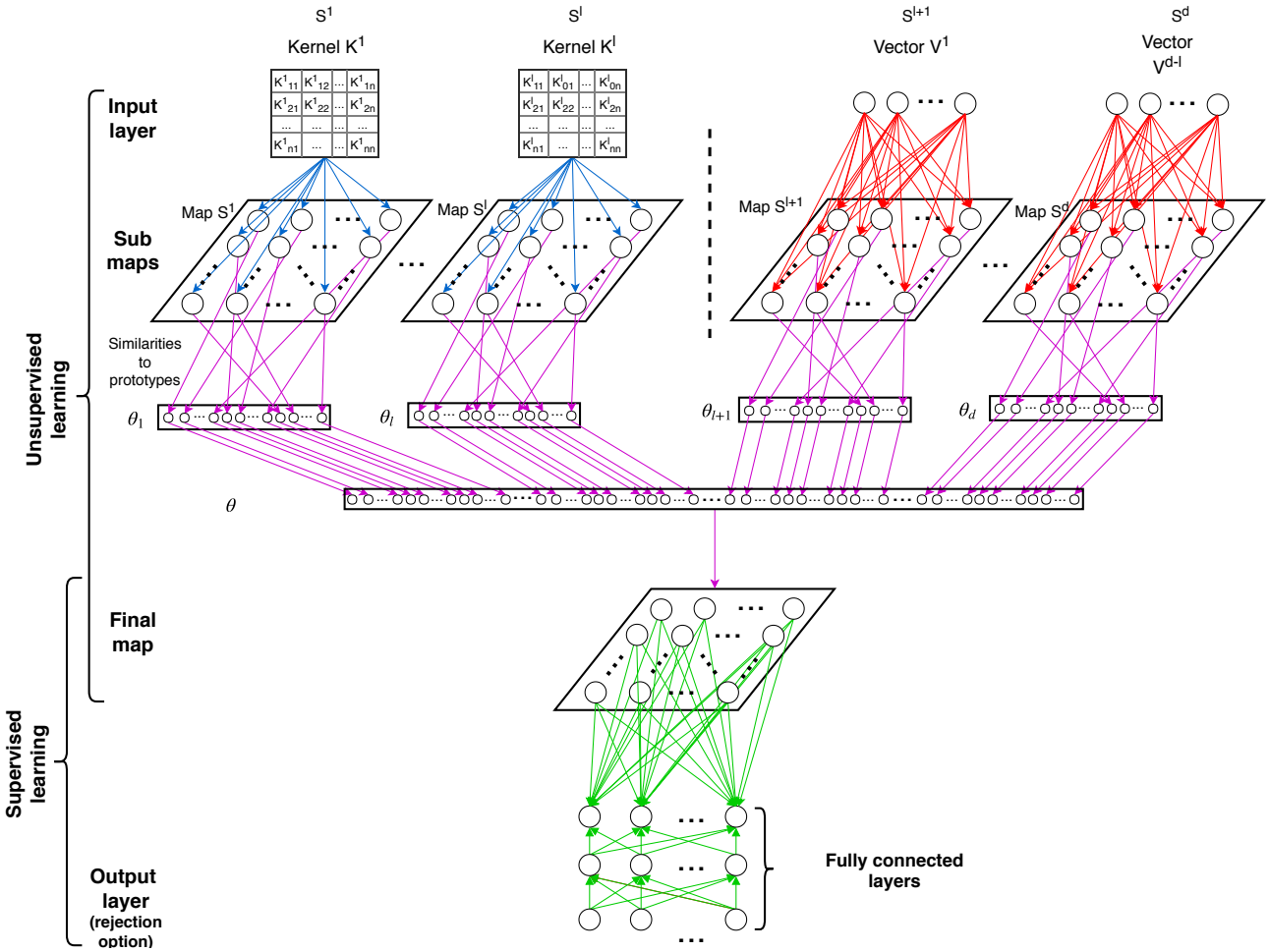


Figure 5.1: Example of CRSOM architecture with a two hidden layers MLP.

Let be a dataset $X = \{x_1, x_2, ..., x_n\}$ where x_i is described using d sources $S = \{S^1, S^2, ..., S^d\}$. These sources can be represented by numerical vectorial data or by complex data. Let be $K = \{K^1, K^2, ..., K^l\}$ the kernels representing the complex sources and $V = \{V^1, V^2, ..., V^{d-l}\}$ the vectors representing the numerical sources where K^i represents the complex features of the source i and V^i the numerical features of the source $i + l$. Let be $Y = \{y_1, y_2, ..., y_n\}$ where y_i is a vector representing the label of the input data x_i. The dimension of y_i is equal to the number of classes. The goal of CRSOM is to learn a function f : X $\rightarrow$ Y, where X is the input space represented by the d sources and Y the output labels. During the training process, we first train the unsupervised part of CRSOM (layers 1 to 3). Then, we train the supervised part using the output of the final map.

Training of the unsupervised part

In MSSOM, the weights are learned online. In the online learning, only one input data is used at each iteration. With the objective function defined in MSSOM (Equation 4.12), we optimize the final prototypes and the source weights using the gradient descent method. As in MSSOM, the vectors $w_{u,r}$ and $\alpha_{u,r}$ must be

normalized after each iteration. In our case, we normalized $w_{u,r}$ and $\alpha_{u,r}$ such that their respective sum are equal to 1. Moreover, we constrained $w_{u,r}$ and $\alpha_{u,r}$ to be positive. Another alternative would be to use mini batch learning. In the mini batch learning, we consider a subset of input data at each iteration when the weights are updated. Theoretically, mini batch learning is more stable and can converge faster. But in our case, the mini batch learning did not improve the results of CRSOM. There was no noticeable label repartition differences for the same number of iterations. Therefore, we keep the online learning.

Training of the supervised part

SLSOM uses an activation function between the SOM and the MLP that is based on Gaussian similarity. Here in CRSOM we use a different SOM activation pattern function between the final SOM and the MLP which uses dot product. The dot product activation is faster and more suitable for our final map which uses dot product to determine the BMUs. The SOM activation pattern $a_{u,i}^{out}$ of the neuron unit u of the final map for the input data i is defined such that:

$$a_{u,i}^{out} = \sum_{r=1}^{d} \alpha_{u,r} \left(\theta_r(x) \cdot w_{u,r}\right) \tag{5.1}$$

The weights of the hidden layers and the output layer of the MLP are learned as in SLSOM. We use the cross-entropy cost function and use the Adam [64] optimizer to optimize the weights of the MLP. We also use the l2-norm regularization in order to avoid overfitting. As in SLSOM, CRSOM uses both the distance and ambiguity local rejection options. The rejection options allow to improve the performances by reducing the ambiguity and to identify potential new classes of ncRNAs.

5.3 Features for ncRNA classification

In order to classify ncRNAs, we define two sources of data: the k-mer frequencies (of size 3, 4 and 5) and the secondary structure generated by RNAfold [83].

In machine learning, kernels are useful for complex data representation or for non linear separation of classes. Since the SOM performs a non linear separation of the input data, we use kernels with SOM only to represent complex data.

We use vectorial representation of the k-mer frequencies and use the batch SOM algorithm in order to produce its sub-map. The k-mer frequencies vectors have a length of 1344 where each component represents a sequence k-mer frequencies.

For the secondary structure, we test two different representations. In the first one, we represent the secondary structure by a kernel and in the second one, we represent the secondary structure by a vector. The secondary structure kernel is based on the edit distance function of RNAlib (the c++ library of RNAfold) on the string representation of the secondary structure. The string representation of an RNA secondary structure is composed of parenthesis and points. A couple of parenthesis (an opening and closing parenthesis) represents a bind between two nucleotides and a point represents an absence of bind. The edit distance function quantifies the number of modifications needed to transform one string into an other. For the RNAs, it represents the number of modifications needed to transform a secondary structure into another. With the computed distances, we generate a Gaussian kernel. We test different values of γ on our dataset and select the one such that $\gamma = \frac{7.0}{dist_{max}}$ where $dist_{max}$ represents the maximal distance between the instances. Then, we use the BBKSOM (Bagged Batch Kernel SOM) (see Equation 4.2.2) to produce its sub-map.

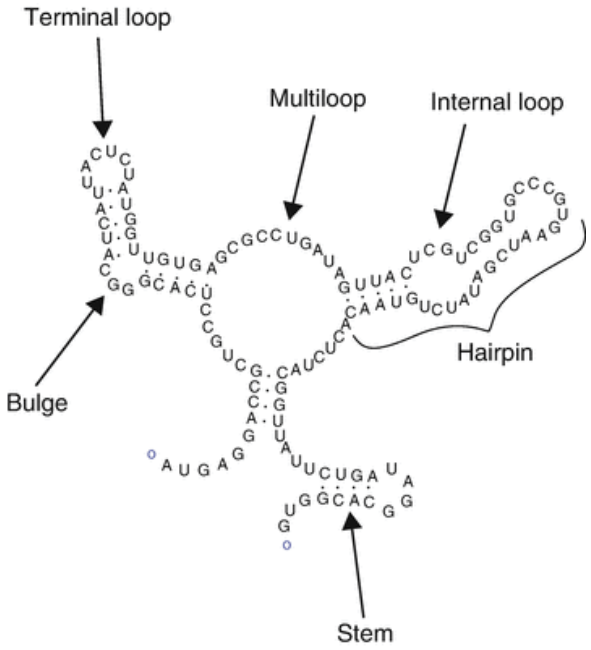


Figure 5.2: The different types of secondary structure motifs (Source: [116]).

In the second representation, we represent the secondary structures by vectors where each component represents the occurrence number of a secondary structure motif or group of motifs. In our current implementation, we work with five motifs which are the bulge (B), the hairpin loop (H), the internal loop (I), the steam (S) and the multiloop (M) (see Figure 5.2). With the found secondary structure motifs, we construct a graph of motifs where the nodes represent secondary structure motifs and a vertex exists between two nodes if their corresponding secondary structure motifs are consecutive. In this graph, each non stem node is connected to at least one stem node because two non stem motifs can not be consecutive. For example, we can not have a multiloop followed by a bulge or an internal loop. From the graph, we can define what we call secondary structure k-mer. A secondary structure k-mer represents a succession of nodes. For example, a valid 2-mer would be SH (a stem followed by a hairpin loop) or SB (a stem followed by a bulge). Then as for the sequence k-mer, we combine all secondary structure k-mer occurrences to form a vector. In our case, we search the secondary structure k-mers (of size 1 to 5) and thus obtain vectors of size 145.

5.4 CRSOM validation

In this Section, we evaluate the performance of CRSOM for the classification of ncRNAs. We compare our approach to nRC [35], which is to our knowledge the latest and the most performant tool existing in the literature for ncRNA classification. For our comparison, we use the dataset of nRC available in their github repository. This dataset is composed of 13 ncRNA classes coming from the Rfam database 12 [89]. The authors of nRC select the following classes: micro RNA (miRNA), two ribosomal RNA (rRNA) classes (5s rRNA and 5.8S rRNA), ribozyme, three small nucleolar RNA (snoRNA) classes (CD-box, HACA-box and scaRNA), transfer RNA (tRNA), two other ribozymes related classes (intron gpI and intron gpII), internal ribosome entry site (IRES), leader (the upstream sequence of a mRNA), and riboswitch. The training and test sets are composed respectively of 6320 and 2600 ncRNAs. We keep from the training and test sets the ncRNAs which contain only A, C, T or G in their sequence in order to keep only the well defined sequences. The filtered sets are composed of 6161 ncRNAs for the training set and 2530 for the test set.

Organisation id	Layer 1	Layer 2	Layer 3
1	119	None	None
2	154	83	None
3	83	154	None
4	172	119	66
5	119	172	66

Table 5.1: Number of perceptrons in the hidden layer of the tested MLP architectures.

5.4.1 Protocol

To evaluate CRSOM on ncRNAs, we generate a confusion matrix on the test set as in nRC. We also measure the performance of CRSOM by using the accuracy, F1-score, MCC, precision, recall and specificity measures:

$$\text{Accuracy} = \frac{1}{ncl} \times \sum_c \frac{TP_c + TN_c}{TP_c + TN_c + FP_c + FN_c}$$

$$\text{F1-score} = \frac{1}{ncl} \times \sum_c \frac{2TP_c}{2TP_c + FP_c + FN_c}$$

$$\text{MCC} = \frac{1}{ncl} \times \sum_c \frac{TP_cTN_c - FP_cFN_c}{\sqrt{(TP_c + FP_c)(TP_c + FN_c)(TN_c + FP_c)(TN_c + FN_c)}}$$

$$\text{Precision} = \frac{1}{ncl} \times \sum_c \frac{TP_c}{TP_c + FP_c}$$

$$\text{Recall} = \frac{1}{ncl} \times \sum_c \frac{TP_c}{TP_c + FN_c}$$

$$\text{Specificity} = \frac{1}{ncl} \times \sum_c \frac{TN_c}{TN_c + FP_c}$$

where TP_c, TN_c, FP_c and FN_c represent respectively the true positive, true negative, false positive and false negative with c the class considered as the positive class and the other classes as one negative class, and ncl is the number of classes.

For the unsupervised part, we vary the number of map sizes and select the one that gives the best organization and generalization results. For each sub map, we use a rectangular grid of size 15x15. Both sources (k-mer and secondary structure) produced similar organisation. For the final map, we also use a 15x15 rectangular grid with a total of 225 neuron units. The unsupervised part has been trained with 61610 iterations.

For the supervised part, we vary three parameters: the epochs number of the MLP, the regularization parameter and different architectures of the MLP. We tested five values of regularization parameter: 0.00001, 0.0001, 0.001, 0.01 and 0.1 and six different MLP architectures that are described in Table 5.1. The number of hidden layers in the MLP varies from one to three. We tested different values of epochs during the test of CRSOM on ncRNAs. We trained our MLP with 3000 epochs and save the results for each 100 epochs.

5.4.2 Results

We evaluated CRSOM by highlighting three aspects. In the first one, we compare the performance of our method using five different sets of features in order to select the best model. The second one focuses on the impact of the number of epochs on the accuracy. And finally, we analyse the SOMs computed by CRSOM and

its results.

We use our approach in two ways: without rejection option and with local rejection option. The different classes of the dataset have been well studied and defined and thus we do not expect to find potential new classes. So we use only the ambiguity rejection rule with a rejection threshold for each class equal to the quantile x of the difference between the class output and the second largest output. Here, we test 4 values for x which are 0.05, 0.1, 0.15 and 0.2.

Performances with the different feature sets

In the previous method section, we defined the features used to classify ncRNAs which, are sequence k-mers and secondary structure. We proposed to encode the sequence k-mers as vectorial data. We also proposed two representations for the secondary structure, one using vectors and one using a kernel matrix.

Here we show the performances of our method when using each data source separately and when we combine sequence k-mers with a secondary structure representation. We thus test five sets of features: sequence k-mer vectors, secondary structure k-mer vectorial data, secondary structure kernel data, k-mer with secondary structure vectorial data, k-mer vectorial data with secondary structure kernel data. In the following, we note each of these feature sets by: "V kmer", "V ss", "K ss", "V kmer ss" and "V kmer K ss" respectively.

For each feature set, we have retained the best model with the best parameters based on their precision. The best regularization parameter was set to 0.0001. For the set "V kmer", the best model was computed with 1600 epochs and with one hidden layer of 119 perceptrons. For the set "V ss", "K ss" and "V kmer K ss", the best model was computed with three hidden layers of respectively 119, 172 and 66 perceptrons and respectively 3000 epochs, 2600 epochs and 2300 epochs. For the set "V kmer ss", the best model was computed with 2800 epochs, with three hidden layers of respectively 172, 119 and 66 perceptrons.

Figure 5.3 shows the performances of our approach using the five features sets. The features set "V ss" shows the worst performances with a recall of 0.14. The results on the features set "K ss" shows that our kernel matrix representation of secondary structure is better than our vectorial representation with a recall of 0.68. The feature set "V kmer" shows good results with an recall of 0.82 and a specificity of 0.98. The features set "V kmer ss" combines the vectorial representation of sequence and secondary structure k-mers. This features set shows better results than the "V ss" but worse than if we used the sequence k-mer alone. When we combine the sequence k-mers with the secondary structure kernel matrix (V kmer K ss), we have comparable results with the "V kmer" set. But, with the combination we are able to better identify the ambiguous prediction and thus increase our performances with the ambiguity rejection option.

We can see that the "V kmer" set and the "V kmer K ss" give comparable results with nRC. These results suggest that the sequence k-mers are powerful to classify ncRNAs. It has to be noticed that the performances with the "V ss" and "K ss" sets show worse results than nRC. So our secondary structure representation is not has good as the one used in nRC. This can be explained by the difference of secondary structure representation between our approach and the one of nRC. In nRC, they search the sub-graphs that are the most represented in the secondary structure. Moreover, another secondary structure prediction tool is used (ipknot [107]) which allow to predict particular secondary structure such as the pseudoknots. With a better secondary structure representation, CRSOM can give better results. However, with the rejection option we improve the performances of CRSOM that become better than nRC with the "V kmer K ss" feature set. The rejection option computed with x equal to 0.2 reject 46% inaccurate predictions but also 13% of accurate predictions for a total of 20% of rejected predictions.

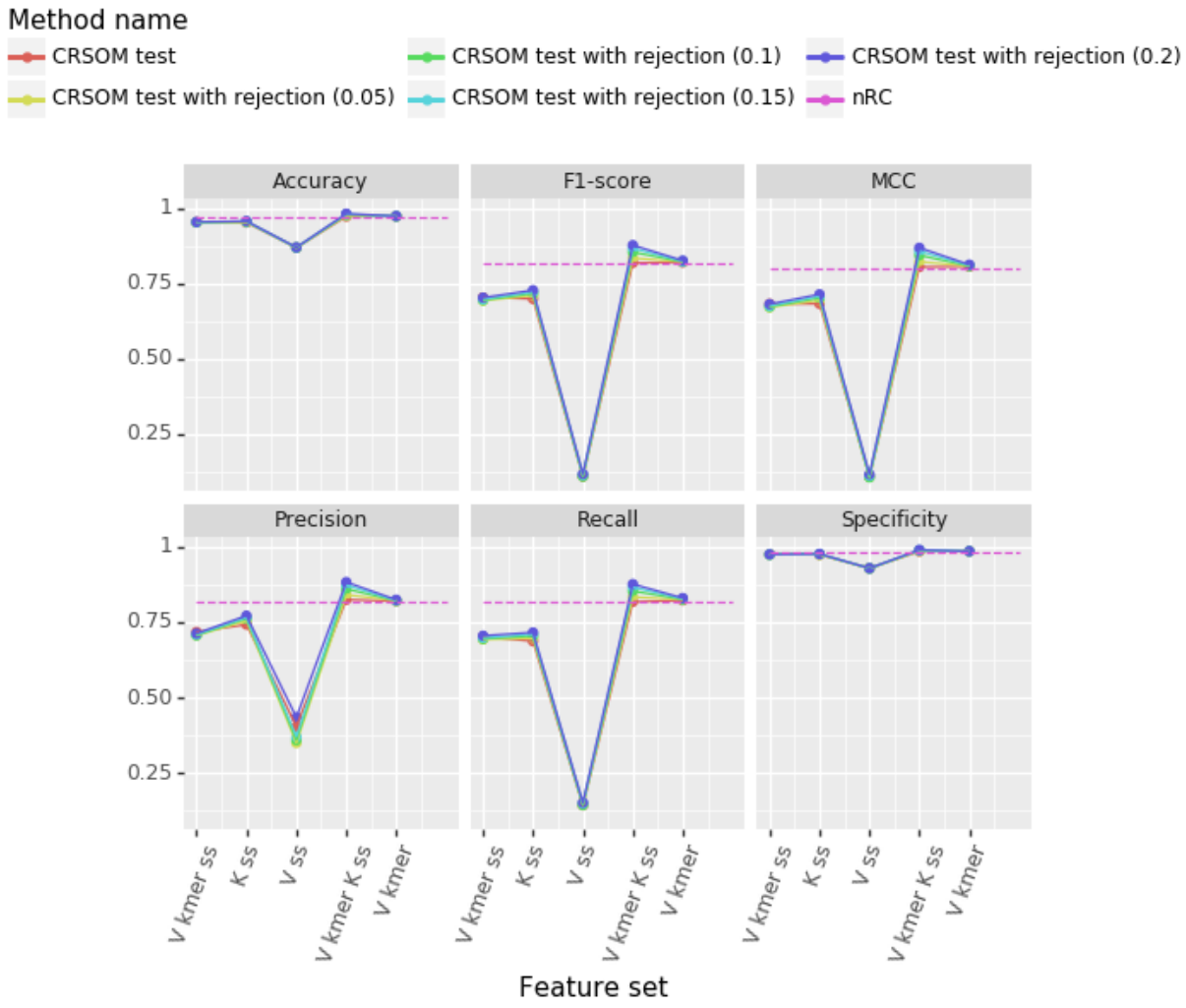


Figure 5.3: Performance of CRSOM on 5 features sets: sequence k-mer vectors (V kmer), secondary structure k-mer vectors (V ss), secondary structure kernel (K ss), sequence and secondary structure k-mer vectors (V kmer ss) and sequence k-mer vector and secondary structure k-mer (V kmer K ss). The dashed line represent the performances of nRC.

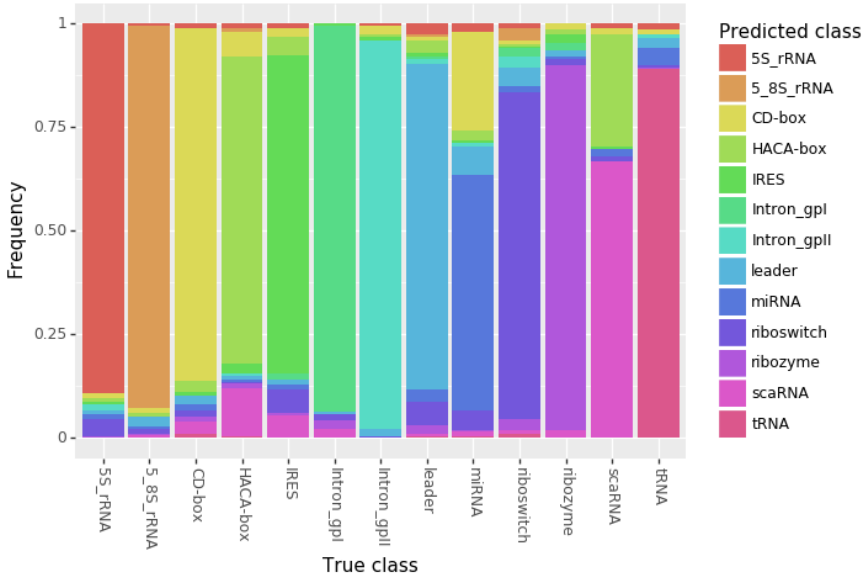


Figure 5.4: CRSOM prediction frequencies for each known class.

Figures 5.4 and 5.5 shows respectively for CRSOM and nRC, how the prediction of each class is distributed. We can see that CRSOM predict accurately most of the classes with frequencies over 75% except for the HACA-box, IRES, miRNA and scaRNA. Moreover, CRSOM gives better results than nRC on the 5.8 rRNA, CD-box, HACA-box, miRNA, riboswitch and ribozyme but worse results for the IRES, Intron group I and the scaRNA. The performances on the other ncRNAs are comparable.

Impact of epochs number

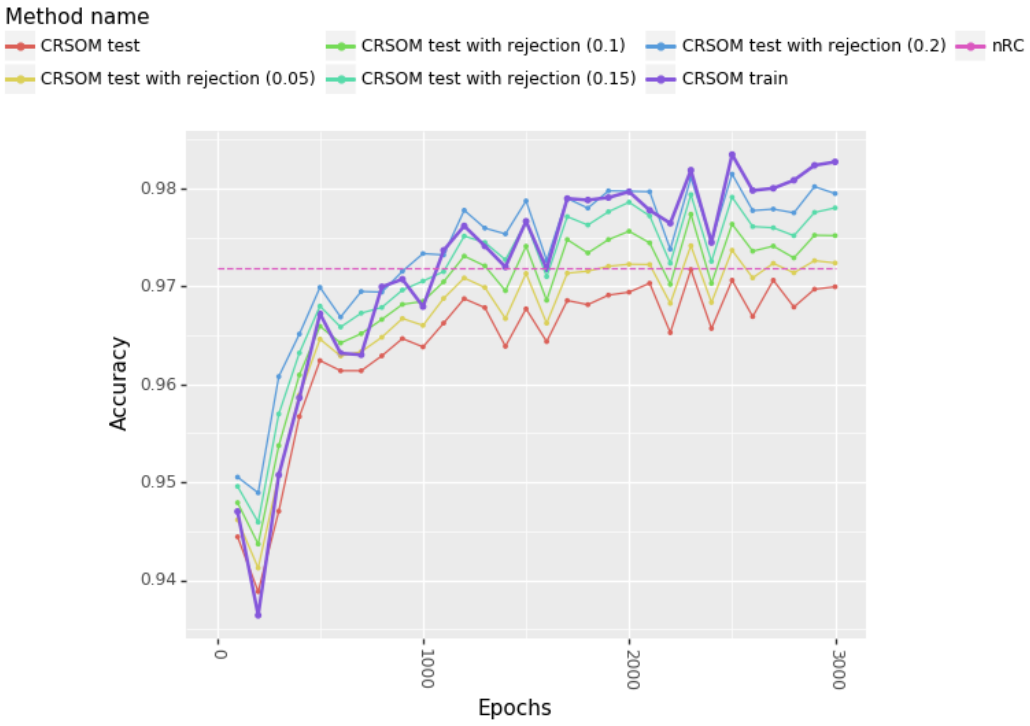


Figure 5.6: Accuracy of CRSOM with the "V kmer K ss" feature set in function of the epochs number.

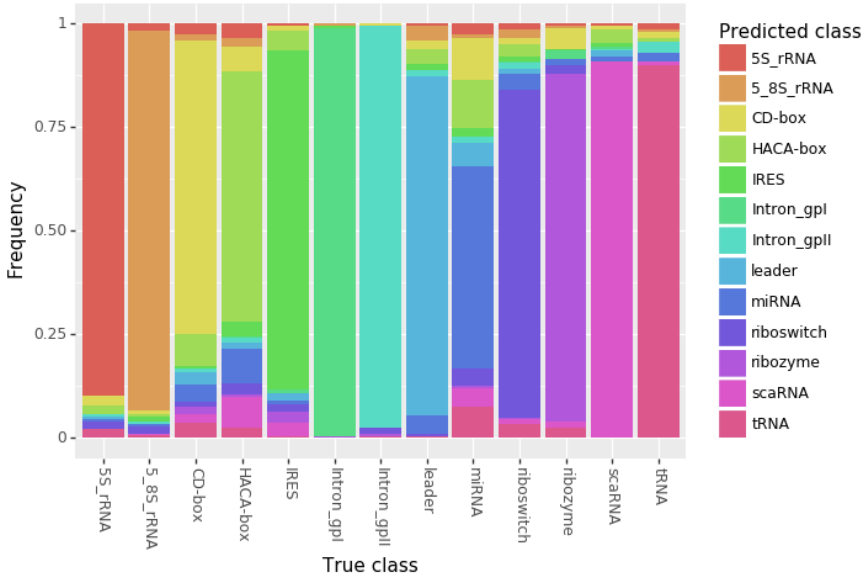


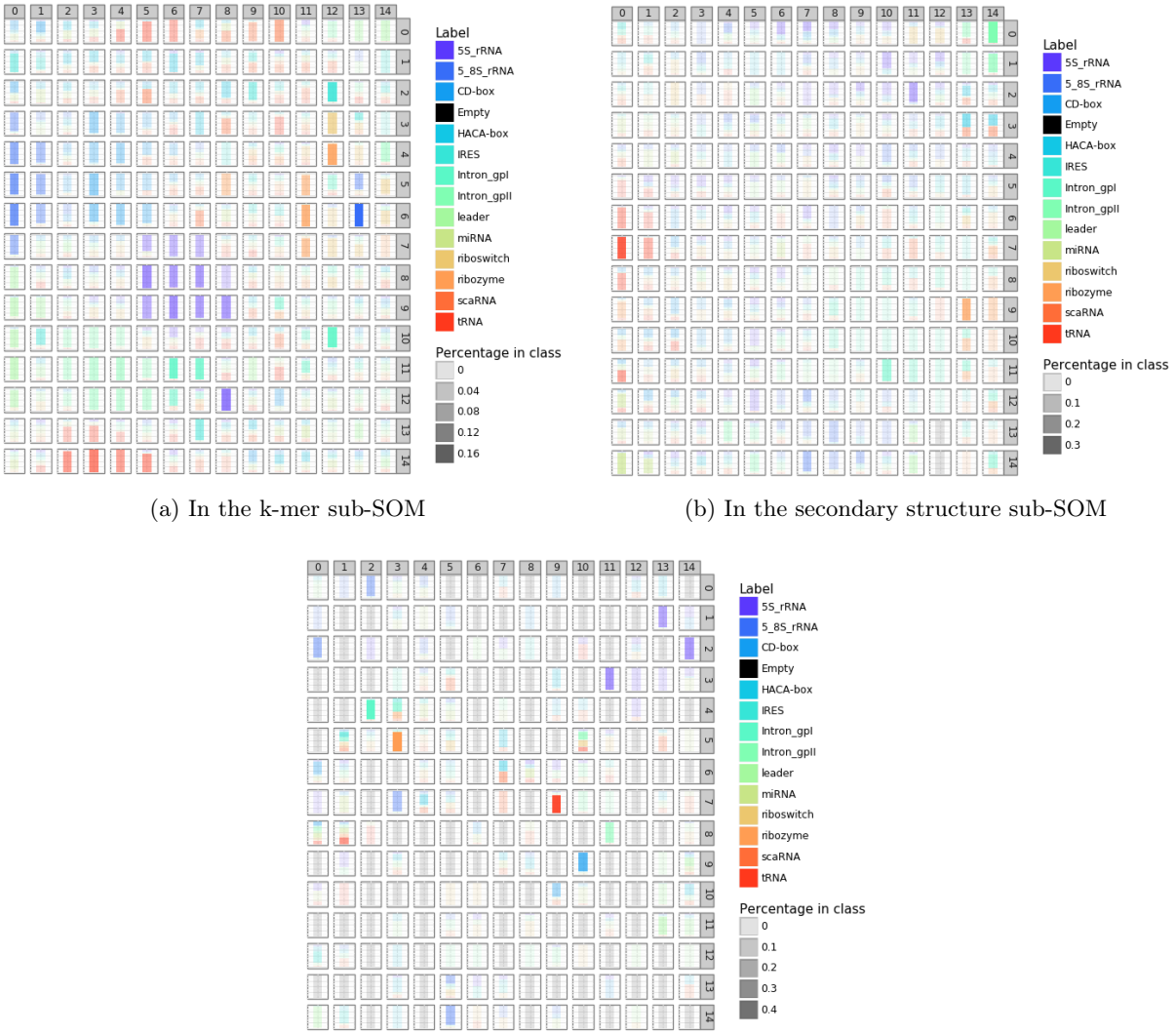
Figure 5.5: nRC prediction frequencies for each known class.

Figure 5.6 we show how the number of epochs is involved in the performances of CRSOM with the optimal set of parameters described before. As expected, at the beginning of the training phase, the accuracy increase with the number of epochs. we see a stabilization around 1000 epochs. From 1100 epochs the training performances of CRSOM start to be higher than the nRC performances. As expected, the performances of CRSOM on the tests sets increase with increase of the rejection option. It has to be noticed that when the rejection option of CRSOM is high, the performances on the test set are close to those of training set.

Data analysis with CRSOM

An interesting property of SOMs is their ability to give a new representation of data. Here, we show how we can make data analysis and visualization using CRSOM. Figure 5.7 illustrates the predicted classes repartition in the neurons units of the final map (see Figure 5.7c), the sequence k-mer sub-map (see Figure 5.7a) and the secondary structure kernel representation sub-map (see Figure 5.7b). In Figure 5.7a, we can see that some classes are well defined. For example, we have the rRNAs (5S and 5.8S) in the middle left, the ribozymes in the middle right and the tRNAs in the bottom left. On the contrary, in Figure 5.7b, the classes are mixed except for some Intron group II on the top right corner, some ribozymes in the right and tRNAs on the left. As a result, the classes repartition in the final map can not represent correctly all the classes. In Figure 5.7c, we can see that the classes that are well defined in both sub-maps are well defined in the final map. For example, some ribozymes on the left and tRNAs in the middle are represented by only on neuron unit. The good representation of these classes explains why CRSOM gives good results on these classes and lower on the others.

Figure 5.8 shows the source weights of CRSOM. We can see that most of the neurons have a small biases towards the secondary structure source. For some neuron units, we have higher source weights towards the source containing secondary structure (middle left of the map). Some of these neuron units are in an ambiguous area in the middle left and contain IRES and scaRNA (see Figure 5.7c). But, the neuron units representing ribozyme and the tRNA in the Figure 5.7c show also a high bias towards secondary structure. This bias can be explain for the tRNA by its high secondary structure conservation. We can not make the



(a) In the k-mer sub-SOM

(b) In the secondary structure sub-SOM

(c) In the final SOM

Figure 5.7: Predicted label repartition obtained on the training set (without rejection) with the "V kmer K ss" feature set.

same theory about the ribozyme because there are no proof of high conservation in this class.

5.5 Conclusion

In this chapter, we presented a new supervised classifier with rejection options based on SOM, which combines multiple heterogeneous sources of data. This new classifier is able to learn the source weights for each neuron unit of the final map. We compared CRSOM to nRC [35], on a dataset composed of 13 classes of ncRNAs that cover the major classes of ncRNAs.

In our experiments, we used three types of data: the sequence k-mer, and two representations of the secondary structure, one using vectors and one using a kernel matrix. CRSOM gives better results when we combine sequence k-mers and the secondary structure kernel representation. With this combination, we show that CRSOM gives comparable results to nRC and even better results when we use the ambiguity rejection option. However, we can expect better performances of CRSOM. Our experiments show that our secondary structure representation needs some improvements because our results using only the secondary structure feature is

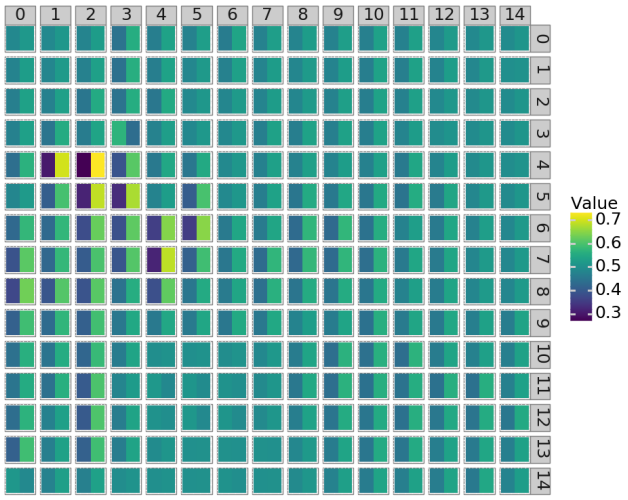


Figure 5.8: Source weights of CRSOM with the "V kmer K ss" feature set. For each neuron unit we show the weight for the sequence k-mer (left) and the weights for the secondary structure kernel matrix (right).

worst those of nRC. Moreover, we show with our data analysis that the representation given by the secondary structure does not separate enough the RNAs which explains our results. Therefor, the oerforamnces of CRSOM can be clearly improved with a better secondary structure representation and additional sources that we present in the perspectives.

Conclusions and perspectives

6.1 Conclusion

The biological context of this thesis was the identification and classification of ncRNAs. The existing approaches do not take into account the biological context of the ncRNAs. The approaches used for the discrimination between coding and non-coding RNAs take the assumption that the coding and non-coding RNAs can be separable. But the separation between coding and non-coding ncRNAs is not clear. Moreover, most of the approaches used for the classification of the ncRNAs into sub classes use only one source of data. By doing so, they show good performances only for the ncRNAs that are well defined using the data source. Another limitations is their lack of biological insight to analyse the ncRNAs. None of the existing methods propose a way to visualize or analyse the predicted ncRNAs. To overcome the limitations, we developped in this thesis, two tools based on self-organizing map: a tool called IRSOM [103] that classify RNA sequences into coding or non-coding RNAs, and a tool called CRSOM that classify predicted non-coding RNAs into different classes. IRSOM is an adaptation of SLSOM [101], a supervised classifier with rejection option we developed. CRSOM is based on a combination of SLSOM and MSSOM [100], a multi sources classifier method we also developed. Thanks to the use of SOM, our tools allow data analysis and visualization.

6.1.1 Identification of ncRNAs using a single SOM, MLP and rejection options

We first developed SLSOM, a supervised classifier with rejection options using SOM. SLSOM is a neural network composed of at least 3 layers (input layer, SOM layer and an MLP with at least one layer). The training of SLSOM is composed of two steps. In the first one, the SOM layer is trained using the unsupervised learning algorithm of SOM. The MLP is trained using backpropagation. We avoid backpropagation in the SOM layer in order to keep the unsupervised learned organization. The rejection option is composed of two rules, the distance and the ambiguity rules. The distance rejection rule rejects the prediction of an input data if the input data is far from the known classes and thus rejects outliers or elements belonging to potential new classes. The ambiguity rejection rule rejects the predictions of input data that are in ambiguous areas (areas where two or more classes are overlapping).

We tested and validated our approach on multiple artificial and real world datasets such as MNIST [72]. We show that SLSOM gives comparable or better results than the classical supervised classifier and the existing supervised SOM. On all the datasets, SLSOM is in the top 3 approaches and gives better results than all the supervised SOM. Moreover, the particularity of SLSOM is its combination of rejection options. We show how the performances of SLSOM can be improved by rejecting ambiguous predictions and input data that are far from the known classes.

We also show how the rejection option (and more particularly the distance rejection rule) can be used to

identify potential new classes. To highlight this property, we trained the SOM layer with the complete training set and the MLP with a reduced training set where one class has been removed. Afterwards, we used the learned model on the test set containing elements of all the classes and show the repartition of the predictions in the SOM. The rejected elements have formed a cluster in the SOM. After verification, all rejected predictions corresponded to the examples that belong to the missing class during the training of the MLP.
The first biological application of SLSOM leads to the development of IRSOM. IRSOM uses a variant of SLSOM designed for the discrimination of coding and non-coding RNAs. The difference between IRSOM and SLSOM is in its learning algorithm. In IRSOM, the SOM layer is optimized two times, with an unsupervised learning as in SLSOM, and during the training of the MLP with the backpropogation. We have used backpropagation because we did not expect to discover potential new classes in this case (an RNA is coding, non-coding or ambiguous) and the backpropagation improves the discrimination power of the SOM. As a consequence, the rejection option of IRSOM is composed only of the ambiguity rejection rule.
We evaluated IRSOM on different species coming from different reigns. We generated two types of models, a species specific model and a cross-species model. For both models we identify the best rejection thresholds by using cross-validation. We show that for complex species, such as the human or the mouse, we have high rejection thresholds (around 0.8). And for simpler species such as E. coli or S. cerevisiae, we have low rejection thresholds (around 0.1). With this rejection thresholds, we show that our method gives comparable results to CNCI [115], CPAT [127], CPC2 [62] and PLEK [77]. We also show that IRSOM is able to identify the ambiguous RNAs (the one that are in the overlapping areas between coding and non-coding RNAs). These ambiguous RNAs show interesting properties such as an ORF coverage around 0.5 (the length of the maximal ORF represents half the length of the transcript) and a high ORF frequency (which means a high number of ORFs in regard to the number of start codon).

6.1.2 Classification of ncRNAs using multiple sources

With IRSOM, we show that the combination of SOM, MLP and rejection options can be successfully used for the prediction of ncRNAs. Even if IRSOM combines different sources of data (k-mer frequencies, ORF and codon bias) it can not be used to classify ncRNAs using vectorial and complex data and we can not use the same source combination for all the classes. For example, the piRNAs do not rely on the secondary structure to be classified when the miRNAs or the tRNA rely on it. So we need to integrate multiple heterogeneous sources of data. For this purpose, we developed the Multiple Source SOM (MSSOM). MSSOM is a three layers neural network trained using unsupervised learning. In MSSOM, each source of data is represented by a SOM (called sub-SOM) and then combined into a final SOM. By doing so, we can obtain a different SOM for each data source and thus exploit differently the available information. Moreover, each neuron unit in the final map has its own source weights.
As for SLSOM, we validated MSSOM on artificial and real world datasets. MSSOM has been tested in different scenarios where the data sources are represented by vectors, kernels or a mix of vectors and kernels. MSSOM showed comparable or better performances than the other SOM algorithms combining multiple sources of data. Moreover, the use of kernels to represent vectorial data does not improve the performances of MSSOM. This can be explained by the non linear discrimination given by the sub-SOMs. Finally, we showed that MSSOM is able to identify the informative sources for each class of data.
MSSOM allows to obtain a SOM that can integrate multiple heterogeneous data sources and with SLSOM, we can classify and identify potential new classes using a SOM. We combine MSSOM and SLSOM into one

classifier called CRSOM. This classifier overcomes the limitations of the state-of-art with its data visualization for each source and a global high level visualisation obtained from the final SOM that combines them and its rejection option that identify ambiguous predictions and help to identify potential new classes.
CRSOM contains at least five layers. The first three layers correspond to those of MSSOM and the last ones to an MLP as in SLSOM. We used two types of data to classify ncRNAs, the sequence with sequence k-mer as vectors and the secondary structure with a Gaussian kernel using the edit distance.
We evaluated and compared CRSOM to nRC [35] on a dataset which contains 13 classes of ncRNAs. We show that when combining sequence k-mers and secondary structure kernel, we have comparable performances with nRC, and with the ambiguity rejection option, we have better results than nRC. We also show how CRSOM can be used to make data analysis by analysing the predicted label repartition in the final map and in the sub-maps.

6.2 Perspectives

The works achieved during this thesis are a good starting point towards the identification and classification of ncRNAs. These works overcome some limitations of the state-of-art. In this section, we will discuss about some possible improvements .

6.2.1 Integration of new data sources

In these works, we test different data sources such as k-mer frequencies, ORF, codon bias and the secondary structure. But for some ncRNA classes ,these data sources will not be sufficient. For example, if we want to predict piRNAs, we need epigenetics data and also RNA interaction data to highlight the interactions between piRNAs transposons.

New data sources

Secondary structure: Currently, we predict the secondary structure of the input RNAs with RNAfold. RNAfold [83] is a good secondary structure predictor that shows good results but it does not predict secondary structures with pseudoknots. So one of our perspectives for the secondary structure data sources would be to test other secondary prediction tools such as BiokoP [74] or IPknot [107]. Another solution would be to consider multiple possible secondary structures instead of just one. But, the use of multiple secondary structures will increase the computational cost of the secondary structure feature representation.
Moreover, our current representation of the secondary structure is not optimal due to two limitations. The first limitation is related to the secondary structure motifs we detect for our vectorial representation. This representation does not take into account the length and the nucleotides compositions of the secondary structure motifs. The second limitation concerns our secondary structure kernel which rely on the edit distance between two secondary structures represented by strings containing dots and parenthesis. The edit distance implemented in the RNAlib package does not take into account the constraints of the parenthesis.
To overcome these limitations, we can inspire ourself from the secondary structure representation defined in nRC. The results of CRSOM, shows how this representation is better than our representation of secondary structure. In nRC, authors look for the most representatives sub-graphs in each secondary structure and represent the secondary structure based on these sub-graphs. But the sub-graphs does not give enough insight on the secondary structure and thus are hard to use in data analysis.

Epigenetic: Epigenetic data have been used previously to classify ncRNAs. In COME [54], the authors use data related to histone modifications (H3K36me3 and H3K4me3) for discriminating coding and non-coding RNAs. They show that histone modifications give a good sensitivity but a poor specificity and the feature importance score computed by Random Forest was around 0.02 when the DNA conservation or the protein conservation was around 0.07. In IpiRId [14], which identify piRNAs, the authors used epigenetic data such as the G-quadruplex, CpG islangs and the histone modifications. They show that epigenetic data give an accuracy greater than 60 % when used alone, with even an accuracy of 90 % for the histone on the fly dataset. In the article [30], the authors show the correlation between the ncRNA classes based on epigenetic data. They consider 420 epigenetic elements such as chromatin state, histone modifications and transcription factors. By using the Pearson's correlation, they highlight two super groups of ncRNAs. Moreover, some ncRNA classes show high correlation. For example, the CD-box and snoRNA have a correlation of 0.94, the HACA-box and snoRNA have a correlation of 0.65 and the piRNA and the rRNA have a correlation of 0.61. Some ncRNA classes also show a high anticorrelation coefficient. For example, the piRNA and the CD-box have a correlation of -0.62. From these papers, we can deduce that epigenetic data can be useful to classify ncRNAs.

Conservation: Another potential data sources would be the RNAs conservation. Here, we discuss two types of conservation: the sequence conservation and the secondary structure conservation.
In the article [92], the authors show interesting results about the sequence conservation of ncRNAs. Their study was based on several ncRNAs coming from human and mouse. Among these ncRNAs, the miRNA and the snoRNA show high conservation (greater than 90% for the miRNA and between 80% and 90 % for the snoRNA). On the contrary, the long non-coding RNAs (lncRNAs) show a conservation lower than 70%. But the conservation distribution of these RNAs with 50 nucleotides windows is defined such that a part on the lncRNAs have a conservation around 0 and the rest is greater than 60%. The authors suggested that this conservation can be related to short domains of the lncRNAs that need sequence specific interactions (as the miRNAs for example). So, there is a possibility to find a common motif among each ncRNA classes and thus classify them base on their sequence conservation.
The secondary structure conservation of ncRNAs can also be used as a new data source. Some classes of ncRNAs show a high secondary structure conservation. All the approaches classifying ncRNAs using secondary structure rely on these properties. For example, All miRNA precursors have the same secondary structure composed of an hairpin, tRNAs have a "t" shape, and for a given reign, all the rRNAs shapes are known.
To integrate conservation in our model, we can define a set of representative ncRNAs for each class. Then we can compute a conservation score between the input RNAs and the representative ncRNAs. With the computed conservation scores, we can define a conservation vector for each input RNA such that each component of this vector represents the conservation score of the input RNA to representative ncRNAs.

Integration in a database

By increasing the number of data sources, we can improve our model but we need a solution to manage all the data used and thus need big data solutions. In the case of ncRNAs, we are not dealing with a lot of inputs but with complex data that contain multiple relationships. This problem can be represented by a graph containing three types of nodes, one for the RNAs, one for the RNA classes and one for the data sources (see Figure 6.9). In this graph the data source values are encoded by edges between the nodes representing RNAs and the nodes representing data sources. This graph can include distance or kernel data with edges between nodes

representing RNAs. Moreover, this graph can be extended to encode additional information for data analysis. For example, we can include Gene ontology data or functional annotation.

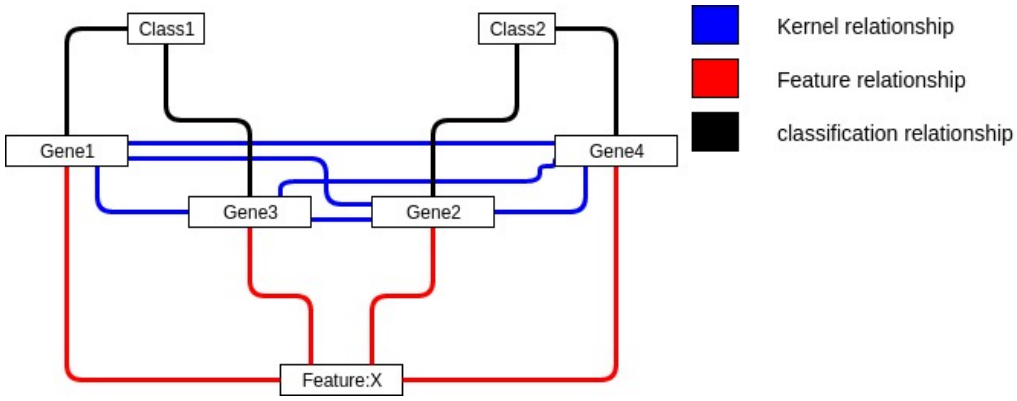


Figure 6.9: Example of a graph encoding RNA data.

Currently there exist multiple graph databases such as ArangoDB, Cassandra, Neo4j or OrientDB. These databases handle graph natively. Graph databases show better performances than SQL databases for data containing a lot of relationships. This performance difference is explained by the cost induced by the join in SQL. In graph databases, this cost is greatly reduced by storing the relationships between the database instances.

6.2.2 Algorithms improvement

Several improvements can be made to our algorithms:

Growing SOM

Here we discuss how to use self growing networks methods instead of SOM with fixed structure to overcome the fixed structure of SOM. These methods can be trained without specifying the number of neuron units and adapt their structure to the input data. We can distinguish two types of growing neural network: those relying on a specific structure such as the growing SOM [12, 39, 4] and those not relying on a structure such as the neural gas [38, 104]. Here we focus on the method relying on a specific structure because they are more suited for data analysis and visualization.
The articles [12], [39] and [4] present methods relying on a structure. In the articles [12] and [4], the authors use the same approach where they added a new neuron unit if the BMU of an input data is at the border of the SOM. But in [4], authors introduce a threshold that controls the growth such that new neurons are added to the SOM if and only if the error of the BMU is greater than the threshold. In [39], the author added a column or a row of neuron units after a given number of iterations. After these iterations, the author identified the neuron that has been the BMU most of the time and added a row or column between that neuron and its farther neighbor.

Handle missing data

Missing data is a common problem with biological data and can have different origins. For example, with RNA-seq data, the expression profile of some RNAs can be missing if these RNAs are not enough expressed. Currently, we handle missing data with the complete case analysis approach where input data with missing

values are removed from the datasets. But this way to do is not optimal because for some classes we have a low amount of available training input data and it is not possible remove them from the training set.
In the article [47], the author present a review about missing data analysis. He first presents the three types of missing data: the missing completely at random (MCAR), the missing at random (MAR) and the missing not at random (MNAR). MCAR means that there is no correlation between the missingness of a given variable and the other variables. MAR means that there is a correlation between the missingness and the observable variables but not necessarily between the missing data and the observable data. MNAR means that the missing data is correlated with a non observable variable.
In the articles [99, 108, 47], the authors present several approaches to handle missing data. The most common approach is called listwise deletion or complete case analysis (the one we currently use). Another common approach is called pairwise deletion or available case analysis. The available case analysis allows to use more data than the complete case analysis. With the available case analysis approach, we use the input data with missing values only if they have data for a given computation. For example, let be three data sources A, B and C. Let be an input data x with missing data for the source B. If we use available case analysis with MSSOM or CRSOM, we can use x to compute the sub-maps of A and C and thus don't lose all its information.
Another approach would be to impute values (i.e replace missing value by a given value). There exist multiple methods to impute values. For example, we can replace missing values for a given variable by the mean value of this variable or use Expected Maximization (EM) to estimate the missing values. In the case of SOM, we propose to compute the BMU of an input data with missing values only with its available data. Then we can impute the missing values using the weights of the BMU or by randomly sampling from the closest input data (i.e the ones that are assigned to the BMU).

Loss function integrating rejection option

With our rejection options, we need to set the rejection thresholds. Currently, we need to manually find the best rejection threshold by doing grid search. This grid search can be time consuming depending on the number of input data and the number of classes (as a quick reminder, we have to set two rejection thresholds for each class in the local rejection option).
A better strategy would be to optimize the rejection thresholds (local and global thresholds). This can be done after or during the training process of our supervised SOM. In the first approach, we can compute the Pareto front for the global rejection and the extended pareto front for the local one [37]. In the second one, we use the chow's method [23] to compute the rejection thresholds.
For both methods we need to set the cost of a rejection for each class. The Chow's method is easier to integrate in our algorithms because it can be directly associated to the output layer. But the first approach is more interesting, because with the Pareto we will optimize the weights of our neural networks depending on the rejection options.

Semi-supervised backpropagation

Currently, CRSOM is trained following two steps. Firstly, we train the sub-maps and the final map. Secondly, we train the MLP using backpropagation. By doing so, we keep the unsupervised organization of the SOM and thus make data analysis and visualization without the influence of classes. But the representation of the input data given by the final map does not take into account the known classes.

An alternative training would use backpropagation on the whole network but in a "semi-supervised" way. This will allow us to improve the classification results and to keep the organization of the discovered classes unchanged in the SOM.

6.2.3 Application to the sex determinism of the Cucumis melo

These works has been developed in collaboration with the FLOwer and CArpel Development (FLOCAD) team at the Institute of Plant Science Paris Saclay (IPS2). The FLOCAD team has two research axis which are the sex determinism and the translation in plants. The translation research focuses on the creation of plants with new phenotype using TILLING and ECOTILLING.
Our project is the identification of the ncRNAs involved on the sex determinism of the Cucumis melo. This project is motivated by previous works that show the presence of ncRNAs in sex determinism mechanism ([105], [66] and [3]). In the article [3], the authors highlight the presence of a small RNA that is responsible of the sex determinism of the Diospyros (persimmon). Moreover in the article [66], the authors show how a piRNA can induce the sexual differentiation of the Bombyx mori (silkworm).
We performed some experimentations on the Cucumis melo. We ran IRSOM (with the Arabidopsis thaliana model) and CPC2 [62] on the Cucumis melo data. We used the Arabidopsis thaliana model for IRSOM because we couldn't create a specific model due to lack of high confident data. We compared the results of IRSOM to those of CPC2. We found out that 95% of the predictions were common between both tools. When we increased the rejection threshold of IRSOM, we reduced the prediction difference between IRSOM and CPC2. Currently we couldn't pursue these experiments due to the creation of a new genome version of the Cucumis melo that improves the gene annotation and thus changes the sequences of some RNAs. And thus, we need to launch again our identification of the ncRNAs with IRSOM and then classify them using CRSOM.

Bibliography

[1] Martín Abadi et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems". In: (). URL: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45166.pdf.

[2] Qaisar Abbas et al. "A Review of Computational Methods for Finding Non-Coding RNA Genes". In: *Genes (Basel).* 7.12 (Dec. 2016), p. 113. ISSN: 2073-4425. DOI: 10.3390/genes7120113. URL: http://www.mdpi.com/2073-4425/7/12/113.

[3] Takashi Akagi et al. "A Y-chromosome–encoded small RNA acts as a sex determinant in persimmons". In: *Science* 346.6209 (2014), pp. 646–650.

[4] Damminda Alahakoon, Saman K Halgamuge, and Bala Srinivasan. "Dynamic self-organizing maps with controlled growth for knowledge discovery". In: *IEEE Trans. neural networks* 11.3 (2000), pp. 601–614.

[5] Shea J Andrews and Joseph A Rothnagel. "Emerging evidence for functional peptides encoded by short open reading frames". In: *Nature Reviews Genetics* 15.3 (2014), p. 193.

[6] Federico Ariel et al. "Battles and hijacks: noncoding transcription in plants". In: *Trends in plant science* 20.6 (2015), pp. 362–371.

[7] Roberto T Arrial, Roberto C Togawa, and Marcelo de M Brigido. "Screening non-coding RNAs in transcriptomes from neglected species using PORTRAIT: case study of the pathogenic fungus Paracoccidioides brasiliensis". In: *BMC bioinformatics* 10.1 (2009), p. 239.

[8] Nenad Bartonicek, Jesper L V Maag, and Marcel E Dinger. "Long noncoding RNAs in cancer: mechanisms of action and technological advancements". In: (). DOI: 10.1186/s12943-016-0530-6. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4884374/pdf/12943_2016_Article_530.pdf.

[9] Asa Ben-Hur et al. "Support vector machines and kernels for computational biology". In: *PLoS computational biology* 4.10 (2008), e1000173.

[10] Tanya Z Berardini, Leonore Reiser, et al. "The arabidopsis information resource: Making and mining the "gold standard" annotated reference plant genome". In: *genesis* 53.8 (2015), pp. 474–485. ISSN: 1526-968X. DOI: 10.1002/dvg.22877. URL: http://dx.doi.org/10.1002/dvg.22877.

[11] Steffen Bickel and Tobias Scheffer. “Multi-view clustering.” In: *ICDM*. Vol. 4. 2004, pp. 19–26.

[12] Justine Blackmore and Risto Miikkulainen. “Incremental grid growing: Encoding high-dimensional structure into a two-dimensional feature map”. In: *Proceedings of the IEEE International Conference on Neural Networks (ICNN'93)*. Vol. 1. Citeseer. 1993, pp. 450–455.

[13] Avrim Blum and Tom Mitchell. “Combining labeled and unlabeled data with co-training”. In: *Proc. Elev. Annu. Conf. Comput. Learn. theory*. ACM. 1998, pp. 92–100.

[14] Anouar Boucheham et al. “IpiRId: Integrative approach for piRNA prediction using genomic and epigenomic data”. In: *PLoS One* 12.6 (2017), e0179787.

[15] Romain Boulet et al. “Batch kernel SOM and related Laplacian methods for social network analysis”. In: *Neurocomputing* 71.7 (2008), pp. 1257–1273.

[16] Jocelyn Brayet et al. “Towards a piRNA prediction using multiple kernel fusion and support vector machine”. In: *Bioinformatics* 30.17 (2014), pp. i364–i370.

[17] Thomas R. Cech and Joan A. Steitz. “The noncoding RNA revolution - Trashing old rules to forge new ones”. In: *Cell* 157.1 (2014), pp. 77–94. ISSN: 10974172. DOI: 10.1016/j.cell.2014.03.008. URL: http://dx.doi.org/10.1016/j.cell.2014.03.008.

[18] Kamalika Chaudhuri et al. “Multi-view clustering via canonical correlation analysis”. In: *Proc. 26th Annu. Int. Conf. Mach. Learn.* ACM. 2009, pp. 129–136.

[19] Jia Cheng et al. “piRNA, the new non-coding RNA, is aberrantly expressed in human cancer cells”. In: *Clinica chimica acta* 412.17-18 (2011), pp. 1621–1625.

[20] Liam Childs et al. “Identification and classification of ncRNA molecules using graph properties”. In: *Nucleic Acids Res.* 37.9 (2009), pp. 1–12. ISSN: 03051048. DOI: 10.1093/nar/gkp206.

[21] Yoonsuck Choe, Joseph Sirosh, and Risto Miikkulainen. “Laterally Interconnected Self-Organizing Maps in Hand-Written Digit Recognition.” In: *Adv. Neural Inf. Process. Syst.* (1996), pp. 736–742.

[22] Seo-Won Choi, Hyun-Woo Kim, and Jin-Wu Nam. “The small peptide world in long noncoding RNAs”. In: *Briefings in Bioinformatics* (2018), bby055. DOI: 10.1093/bib/bby055. eprint: /oup/backfile/content_public/journal/bib/pap/10.1093_bib_bby055/1/bby055.pdf. URL: http://dx.doi.org/10.1093/bib/bby055.

[23] C Chow. “On optimum recognition error and reject tradeoff”. In: *IEEE Trans. Inf. theory* 16.1 (1970), pp. 41–46.

[24] The RNAcentral Consortium, The RNAcentral Consortium, and The RNAcentral Consortium. “RNAcentral: a comprehensive database of non-coding RNA sequences”. In: *Nucleic Acids Res.* 45.D1 (Jan. 2017), pp. D128–D134. ISSN: 0305-1048. DOI: 10.1093/nar/gkw1008. URL: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkw1008.

[25] The UniProt Consortium. "UniProt: the universal protein knowledgebase". In: *Nucleic Acids Res.* 45.D1 (Jan. 2017), pp. D158–D169. ISSN: 0305-1048. DOI: 10.1093/nar/gkw1099. URL: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkw1099.

[26] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Mach. Learn.* 20.3 (1995), pp. 273–297. ISSN: 1573-0565. DOI: 10.1007/BF00994018. URL: http://dx.doi.org/10.1007/BF00994018.

[27] Thomas Cover and Peter Hart. "Nearest neighbor pattern classification". In: *IEEE Trans. Inf. theory* 13.1 (1967), pp. 21–27.

[28] Claudio De Stefano, Carlo Sansone, and Mario Vento. "To reject or not to reject: that is the question-an answer in case of neural classifiers". In: *IEEE Trans. Syst. Man, Cybern. Part C (Applications Rev.* 30.1 (2000), pp. 84–94.

[29] Thomas Derrien, Rory Johnson, et al. "The GENCODE v7 catalog of human long noncoding RNAs: analysis of their gene structure, evolution, and expression". In: *Genome Res.* 22.9 (2012), pp. 1775–1789.

[30] Mikhail G Dozmorov et al. "Systematic classification of non-coding RNAs by epigenomic similarity". In: *BMC bioinformatics*. Vol. 14. 14. BioMed Central. 2013, S2.

[31] Florian Erhard and Ralf Zimmer. "Classification of ncRNAs using position and size information in deep sequencing data". In: *Bioinformatics* 26.18 (2010), pp. i426–i432. ISSN: 14602059. DOI: 10.1093/bioinformatics/btq363.

[32] Eleazar Eskin et al. "Mismatch string kernels for SVM protein classification". In: *Advances in neural information processing systems*. 2003, pp. 1441–1448.

[33] Xiao-Nan N Fan and Shao-Wu W Zhang. "lncRNA-MFDL: identification of human long non-coding RNAs by fusing multiple features and using deep learning". In: *Mol. BioSyst.* 11.3 (2015), pp. 892–897. ISSN: 1742-206X. DOI: 10.1039/c4mb00650j. URL: http://xlink.rsc.org/?DOI=C4MB00650J%20http://www.ncbi.nlm.nih.gov/pubmed/25588719.

[34] Mario Fasold et al. "DARIO: A ncRNA detection and analysis tool for next-generation sequencing experiments". In: *Nucleic Acids Res.* 39.SUPPL. 2 (2011), pp. 112–117. ISSN: 03051048. DOI: 10.1093/nar/gkr357.

[35] Antonino Fiannaca et al. "nRC: non-coding RNA Classifier based on structural features". In: *BioData Min.* 10.1 (Dec. 2017), p. 27. ISSN: 1756-0381. DOI: 10.1186/s13040-017-0148-2. URL: http://biodatamining.biomedcentral.com/articles/10.1186/s13040-017-0148-2.

[36] James W Fickett. "Recognition of protein coding regions in DNA sequences". In: *Nucleic acids research* 10.17 (1982), pp. 5303–5318.

[37] Lydia Fischer, Barbara Hammer, and Heiko Wersing. "Optimal local rejection for classifiers". In: *Neurocomputing* 214 (2016), pp. 445–457.

[38] Bernd Fritzke. “A growing neural gas network learns topologies”. In: *Advances in neural information processing systems*. 1995, pp. 625–632.

[39] Bernd Fritzke. “Growing grid—a self-organizing network with constant neighborhood range and adaptation strength”. In: *Neural processing letters* 2.5 (1995), pp. 9–13.

[40] Giorgio Fumera, Fabio Roli, and Giorgio Giacinto. “Reject option with multiple thresholds”. In: *Pattern Recognit.* 33.12 (2000), pp. 2099–2101.

[41] Francis Galton. “Regression towards mediocrity in hereditary stature.” In: *J. Anthropol. Inst. Gt. Britain Irel.* 15 (1886), pp. 246–263.

[42] Olivier Gevaert et al. “Predicting the prognosis of breast cancer by integrating clinical and microarray data with Bayesian networks”. In: *Bioinformatics* 22.14 (2006), e184–e190. DOI: 10.1093/bioinformatics/btl230. eprint: /oup/backfile/content_public/journal/bioinformatics/22/14/10.1093/bioinformatics/btl230/2/btl230.pdf. URL: http://dx.doi.org/10.1093/bioinformatics/btl230.

[43] Federico Girosi, Michael Jones, and Tomaso Poggio. “Regularization Theory and Neural Networks Architectures”. In: *Neural Comput.* 7 (1995), pp. 219–269.

[44] Mehmet GöNen and Ethem AlpaydıN. “Localized algorithms for multiple kernel learning”. In: *Pattern Recognition* 46.3 (2013), pp. 795–807.

[45] Mehmet Gönen and Ethem Alpaydın. “Multiple kernel learning algorithms”. In: *Journal of machine learning research* 12.Jul (2011), pp. 2211–2268.

[46] Mehmet Gönen and Adam A Margolin. “Localized Data Fusion for Kernel k-Means Clustering with Application to Cancer Biology”. In: *Adv. Neural Inf. Process. Syst. 27*. Ed. by Z Ghahramani et al. Curran Associates, Inc., 2014, pp. 1305–1313. URL: http://papers.nips.cc/paper/5236-localized-data-fusion-for-kernel-k-means-clustering-with-application-to-cancer-biology.pdf.

[47] John W Graham. “Missing data analysis: Making it work in the real world”. In: *Annual review of psychology* 60 (2009), pp. 549–576.

[48] Xingli Guo et al. “Advances in long noncoding RNAs: Identification, structure prediction and function annotation”. In: *Brief. Funct. Genomics* 15.1 (2016), pp. 38–46. ISSN: 20412657. DOI: 10.1093/bfgp/elv022.

[49] David J Hand and Keming Yu. “Idiot’s Bayes—not so stupid after all?” In: *Int. Stat. Rev.* 69.3 (2001), pp. 385–398.

[50] S Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing Company, 1994, pp. 236–284. ISBN: 9780132265560. URL: https://books.google.fr/books?id=VIJmPwAACAAJ.

[51] Thomas Hecht, Mathieu Lefort, and Alexander Gepperth. "Using self-organizing maps for regression: the importance of the output function". In: *European Symposium on Artificial Neural Networks (ESANN)*. Bruges, Belgium, Apr. 2015. URL: https://hal.archives-ouvertes.fr/hal-01251011.

[52] Tin Kam Ho. "Random decision forests". In: *Doc. Anal. Recognition, 1995., Proc. Third Int. Conf.* Vol. 1. IEEE. 1995, pp. 278–282.

[53] Gali Housman and Igor Ulitsky. "Methods for distinguishing between protein-coding and long noncoding RNAs and the elusive biological purpose of translation of long noncoding RNAs". In: *Biochim. Biophys. Acta - Gene Regul. Mech.* 1859.1 (Jan. 2016), pp. 31–40. ISSN: 18749399. DOI: 10.1016/j.bbagrm.2015.07.017. URL: http://linkinghub.elsevier.com/retrieve/pii/S1874939915001728.

[54] Long Hu et al. "COME: a robust coding potential calculation tool for lncRNA identification and characterization based on multiple features." In: *Nucleic Acids Res.* 45.1 (Jan. 2017), e2. ISSN: 1362-4962. DOI: 10.1093/nar/gkw798.

[55] Nicholas T Ingolia. "Ribosome profiling: new views of translation, from single codons to genome scale". In: *Nature Reviews Genetics* 15.3 (2014), p. 205.

[56] Hisao Ishibuchi and Manabu Nii. "Neural networks for soft decision making". In: *Fuzzy Sets Syst.* 115.1 (2000), pp. 121–140.

[57] Eric Augusto Ito et al. "BASiNET—BiologicAl Sequences NETwork: a case study on coding and non-coding RNAs identification". In: *Nucleic Acids Research* (2018), gky462. DOI: 10.1093/nar/gky462. eprint: /oup/backfile/content_public/journal/nar/pap/10.1093_nar_gky462/1/gky462.pdf. URL: http://dx.doi.org/10.1093/nar/gky462.

[58] Saakshi Jalali et al. "Computational approaches towards understanding human long non-coding RNA biology". In: *Bioinformatics* 31.14 (2015), pp. 2241–2251. ISSN: 14602059. DOI: 10.1093/bioinformatics/btv148.

[59] D Jevsinek Skok et al. "Genome-wide in silico screening for micro RNA genetic variability in livestock species". In: *Animal genetics* 44.6 (2013), pp. 669–677.

[60] Ioanna Kalvari et al. "Rfam 13.0: shifting to a genome-centric resource for non-coding RNA families". In: *Nucleic Acids Res.* (Nov. 2017). ISSN: 0305-1048. DOI: 10.1093/nar/gkx1038. URL: http://academic.oup.com/nar/article/doi/10.1093/nar/gkx1038/4588106.

[61] Meina Kan et al. "Multi-view discriminant analysis". In: *IEEE transactions on pattern analysis and machine intelligence* 38.1 (2016), pp. 188–194.

[62] Yu-Jian Kang, De-Chang Yang, et al. "CPC2: a fast and accurate coding potential calculator based on sequence intrinsic features". In: *Nucleic Acids Res.* 45.W1 (2017), W12–W16.

[63] Kaori Kashi et al. “Discovery and functional analysis of lncRNAs: methodologies to investigate an uncharacterized transcriptome”. In: *Biochimica et Biophysica Acta (BBA)-Gene Regulatory Mechanisms* 1859.1 (2016), pp. 3–15.

[64] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR* abs/1412.6980 (2014). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

[65] Sila Kittiwachana and Kate Grudpan. “Supervised Self Organizing Maps for explonatory data analysis of running waters on physiochemical parameters: a case study in Chiang Mai, Thailand”. In: *KKU Res.j* 20.1 (2015), pp. 1–11.

[66] Takashi Kiuchi et al. “A single female-specific piRNA is the primary determiner of sex in the silkworm”. In: *Nature* 509.7502 (2014), pp. 633–636.

[67] Teuvo Kohonen. *Self-Organizing Maps - third edition*. Springer-Verlag Berlin Heidelberg, 2001. DOI: 10.1007/978-3-642-56927-2. URL: http://www.springer.com/us/book/9783540679219.

[68] Teuvo Kohonen. “The’neural’phonetic typewriter”. In: *Computer (Long. Beach. Calif).* 21.3 (1988), pp. 11–22.

[69] Lei Kong et al. “CPC: Assess the protein-coding potential of transcripts using sequence features and support vector machine”. In: *Nucleic Acids Res.* 35 (July 2007), W345–W349. ISSN: 03051048. DOI: 10.1093/nar/gkm391. URL: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkm391.

[70] Gert R. G. Lanckriet et al. “A statistical framework for genomic data fusion”. In: *Bioinformatics* 20.16 (2004), pp. 2626–2635. DOI: 10.1093/bioinformatics/bth294. eprint: /oup/backfile/content_public/journal/bioinformatics/20/16/10.1093/bioinformatics/bth294/2/bth294.pdf. URL: http://dx.doi.org/10.1093/bioinformatics/bth294.

[71] King Wai Lau, Hujun Yin, and Simon Hubbard. “Kernel self-organising maps for classification”. In: *Neurocomputing* 69.16 (2006), pp. 2033–2040.

[72] Yan LeCun, Corinna Cortes, and Christopher J.C Burges. *THE MNIST DATABASE of handwritten digits*.

[73] Byunghan Lee et al. “deepTarget: end-to-end learning framework for microRNA target prediction using deep recurrent neural networks”. In: *Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*. ACM. 2016, pp. 434–442.

[74] Audrey Legendre, Eric Angel, and Fariza Tahi. “Bi-objective integer programming for RNA secondary structure prediction with pseudoknots”. In: *BMC bioinformatics* 19.1 (2018), p. 13.

[75] Yunwen Lei et al. “Localized Multiple Kernel Learning - A Convex Approach”. In: *CoRR* abs/1506.0 (2015). arXiv: 1506.04364. URL: http://arxiv.org/abs/1506.04364.

[76] Yuk Yee Leung et al. "CoRAL: Predicting non-coding RNAs from small RNA-sequencing data". In: *Nucleic Acids Res.* 41.14 (2013), pp. 1–10. ISSN: 03051048. DOI: 10.1093/nar/gkt426.

[77] Aimin Li, Junying Zhang, and Zhongyin Zhou. "PLEK: a tool for predicting long non-coding RNAs and messenger RNAs based on an improved k-mer scheme". In: *BMC Bioinformatics* 15.1 (Sept. 2014), p. 311. ISSN: 1471-2105. DOI: 10.1186/1471-2105-15-311. URL: http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-15-311.

[78] Miaomiao Li et al. "Multiple Kernel Clustering with Local Kernel Alignment Maximization". In: *Proc. Twenty-Fifth Int. Jt. Conf. Artif. Intell. IJCAI 2016, New York, NY, USA, 9-15 July 2016*. 2016, pp. 1704–1710. URL: http://www.ijcai.org/Abstract/16/244.

[79] Yong Li and Hongkai Xiong. "Union of data-driven subspaces via subspace clustering for compressive video sampling". In: *Data Compression Conf. Proc.* (2014), pp. 63–72. ISSN: 10680314. DOI: 10.1109/DCC.2014.21.

[80] Muxuan Liang et al. "Integrative data analysis of multi-platform cancer data with a multimodal deep learning approach". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 12.4 (2015), pp. 928–937.

[81] M Lichman. *UCI Machine Learning Repository*. 2013. URL: http://archive.ics.uci.edu/ml.

[82] Jinfeng Liu, Julian Gough, and Burkhard Rost. "Distinguishing Protein-Coding from Non-Coding RNAs through Support Vector Machines". In: *PLoS Genet.* 2.4 (2006), e29. ISSN: 1553-7390. DOI: 10.1371/journal.pgen.0020029. URL: http://dx.plos.org/10.1371/journal.pgen.0020029.

[83] Ronny Lorenz et al. "ViennaRNA Package 2.0". In: *Algorithms for Molecular Biology* 6.1 (2011), p. 26.

[84] Jun Lu et al. "MicroRNA expression profiles classify human cancers". In: *nature* 435.7043 (2005), p. 834.

[85] Lina Ma, Vladimir B. Bajic, and Zhang Zhang. "On the classification of long non-coding RNAs". In: *RNA Biology* 10.6 (2013). PMID: 23696037, pp. 924–933. DOI: 10.4161/rna.24604. eprint: https://doi.org/10.4161/rna.24604. URL: https://doi.org/10.4161/rna.24604.

[86] Jérôme Mariette et al. "Bagged kernel SOM". In: *Adv. Self-Organizing Maps Learn. Vector Quantization*. Springer, 2014, pp. 45–54.

[87] César L C Mattos and Guilherme A Barreto. "ARTIE and MUSCLE models: building ensemble classifiers from fuzzy ART and SOM networks". In: *Neural Comput. Appl.* 22.1 (2013), pp. 49–61.

[88] Yadong Mu and Bingfeng Zhou. "Non-uniform Multiple Kernel Learning with Cluster-based Gating Functions". In: *Neurocomput.* 74.7 (2011), pp. 1095–1101. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2010.11.001. URL: http://dx.doi.org/10.1016/j.neucom.2010.11.001.

[89] Eric P. Nawrocki et al. “Rfam 12.0: updates to the RNA families database”. In: *Nucleic Acids Research* 43.D1 (2015), pp. D130–D137. DOI: 10.1093/nar/gku1063. eprint: /oup/backfile/content_public/journal/nar/43/d1/10.1093_nar_gku1063/2/gku1063.pdf. URL: http://dx.doi.org/10.1093/nar/gku1063.

[90] Nuala A O’Leary et al. “Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation.” In: *Nucleic Acids Res.* 44.D1 (Jan. 2016), pp. D733–45. ISSN: 1362-4962. DOI: 10.1093/nar/gkv1189. URL: http://www.ncbi.nlm.nih.gov/pubmed/26553804%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4702849.

[91] Madalina Olteanu, Nathalie Villa-Vialaneix, and Christine Cierco-Ayrolles. “Multiple Kernel Self-Organizing Maps”. In: *Eur. Symp. Artif. Neural Networks, Comput. Intell. Mach. Learn.* Esann (2013), p. 83.

[92] Ken C Pang, Martin C Frith, and John S Mattick. “Rapid evolution of noncoding RNAs: lack of conservation does not mean lack of function”. In: *Trends in genetics* 22.1 (2006), pp. 1–5.

[93] Bharat Panwar, Amit Arora, and Gajendra Ps Raghava. “Prediction and classification of ncRNAs using structural information”. In: *BMC Genomics* 15.1 (2014), p. 127. ISSN: 1471-2164. DOI: 10.1186/1471-2164-15-127.

[94] François Payre and Claude Desplan. “RNA. Small peptides control heart activity.” In: *Science* 351.6270 (Jan. 2016), pp. 226–227. ISSN: 1095-9203. DOI: 10.1126/science.aad9873. URL: http://www.ncbi.nlm.nih.gov/pubmed/26816363%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5150219.

[95] F Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *J. Mach. Learn. Res.* 12 (2011), pp. 2825–2830.

[96] K Pelckmans and B De Moor. “Mutual Spectral clustering: Microarray Experiments Versus Text Corpus”. In: *Work. Probabilistic Model. Mach. Learn. Struct. Syst. Biol.* Helsinki, Finland, 2006, pp. 55–58. URL: http://www.cs.helsinki.fi/group/bioinfo/events/pmsb06/final/pelckmans_et_al.pdf.

[97] Diane M Pereira et al. “Delivering the promise of miRNA cancer therapeutics”. In: *Drug discovery today* 18.5-6 (2013), pp. 282–289.

[98] Anton S Petrov et al. “Secondary structures of rRNAs from all three domains of life”. In: *PLoS One* 9.2 (2014), e88222.

[99] Therese D Pigott. “A review of methods for missing data”. In: *Educational research and evaluation* 7.4 (2001), pp. 353–383.

[100] Ludovic Platon, Farida Zehraoui, and Fariza Tahi. “Localized multiple sources Self-Organizing Map”. In: *25th International Conference on Neural Information Processing: ICONIP 2018 (To appear)*. 2018.

[101] Ludovic Platon, Farida Zehraoui, and Fariza Tahi. "Self-organizing maps with supervised layer". In: *2017 12th Int. Work. Self-Organizing Maps Learn. Vector Quantization, Clust. Data Vis.* (June 2017), pp. 1–8. DOI: 10.1109/WSOM.2017.8020022. URL: http://ieeexplore.ieee.org/document/8020022/.

[102] Ludovic Platon, Farida Zehraoui, and Fariza Tahi. "Self-organizing maps with supervised layer". In: *Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM), 2017 12th International Workshop on*. IEEE. 2017, pp. 1–8.

[103] Ludovic Platon et al. "IRSOM, a reliable identifier of ncRNAs based on supervised self-organizing maps with rejection". In: *Bioinformatics* 34.17 (2018), pp. i620–i628.

[104] Yann Prudent and Abdellatif Ennaji. "An incremental growing neural gas learns topologies". In: *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*. Vol. 2. IEEE. 2005, pp. 1211–1216.

[105] Raphael Heiko Rastetter, Craig Allen Smith, and Dagmar Wilhelm. "The role of non-coding RNA in male sex determination and differentiation". In: *Reproduction* (2015), REP–15.

[106] Matteo Re and Giorgio Valentini. "Integration of heterogeneous data sources for gene function prediction using decision templates and ensembles of learning machines". In: *Neurocomputing* 73.7-9 (2010), pp. 1533–1537.

[107] Kengo Sato et al. "IPknot: fast and accurate prediction of RNA secondary structures with pseudoknots using integer programming". In: *Bioinformatics* 27.13 (2011), pp. i85–i93.

[108] Joseph L Schafer and John W Graham. "Missing data: our view of the state of the art." In: *Psychological methods* 7.2 (2002), p. 147.

[109] Urminder Singh et al. "PLncPRO for prediction of long non-coding RNAs (lncRNAs) in plants and its application for discovery of abiotic stress-responsive lncRNAs in rice and chickpea". In: *Nucleic Acids Research* 45.22 (2017), e183. DOI: 10.1093/nar/gkx866. eprint: /oup/backfile/content_public/journal/nar/45/22/10.1093_nar_gkx866/2/gkx866.pdf. URL: http://dx.doi.org/10.1093/nar/gkx866.

[110] Ricardo Gamelas Sousa et al. "Robust classification with reject option using the self-organizing map". In: *Neural Comput. Appl.* 26.7 (2015), pp. 1603–1619.

[111] Nitish Srivastava and Ruslan R Salakhutdinov. "Multimodal learning with deep boltzmann machines". In: *Advances in neural information processing systems*. 2012, pp. 2222–2230.

[112] Georges St.Laurent, Claes Wahlestedt, and Philipp Kapranov. "The Landscape of long noncoding RNA classification". In: *Trends Genet.* 31.5 (2015), pp. 249–251. ISSN: 13624555. DOI: 10.1016/j.tig.2015.03.007. arXiv: 15334406. URL: http://dx.doi.org/10.1016/j.tig.2015.03.007.

[113] Kun Sun et al. "iSeeRNA: identification of long intergenic non-coding RNA transcripts from transcriptome sequencing data." In: *BMC Genomics* 14 Suppl 2.Suppl 2 (2013), S7. ISSN: 1471-2164. DOI: 10.1186/1471-2164-14-S2-S7. URL: http://www.ncbi.nlm.nih.gov/pubmed/23445546%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3582448.

[114] Lei Sun et al. "lncRScan-SVM: a tool for predicting long non-coding RNAs using support vector machine". In: *PloS one* 10.10 (2015), e0139654.

[115] Liang Sun et al. "Utilizing sequence intrinsic composition to classify protein-coding and long non-coding transcripts". In: *Nucleic Acids Res.* 41.17 (Sept. 2013), e166–e166. ISSN: 03051048. DOI: 10.1093/nar/gkt646. URL: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkt646.

[116] Fariza Tahi, Anouar Boucheham, et al. "In Silico Prediction of RNA Secondary Structure". In: *Promoter Associated RNA*. Springer, 2017, pp. 145–168.

[117] Christophe Tav et al. "miRNAFold: a web server for fast miRNA precursor prediction in genomes". In: *Nucleic Acids Research* 44.W1 (2016), W181–W184. DOI: 10.1093/nar/gkw459. eprint: /oup/backfile/content_public/journal/nar/44/w1/10.1093_nar_gkw459/5/gkw459.pdf. URL: http://dx.doi.org/10.1093/nar/gkw459.

[118] Sebastien Tempel et al. "miRBoost: boosting support vector machines for microRNA precursor classification". In: *RNA* (2015).

[119] Sébastien Tempel and Fariza Tahi. "A fast ab-initio method for predicting miRNA precursors in genomes". In: *Nucleic Acids Res.* 40.11 (2012), e80–e80.

[120] Rashmi Tripathi et al. "Deeplnc, a long non-coding rna prediction tool using deep neural network". In: *Network Modeling Analysis in Health Informatics and Bioinformatics* 5.1 (2016), p. 21.

[121] JWC Van Lint and Serge P Hoogendoorn. "A robust and efficient method for fusing heterogeneous data from traffic sensors on freeways". In: *Computer-Aided Civil and Infrastructure Engineering* 25.8 (2010), pp. 596–612.

[122] Pavankumar Videm et al. "BlockClust: Efficient clustering and classification of non-coding RNAs from short read RNA-seq profiles". In: *Bioinformatics* 30.12 (2014), pp. 274–282. ISSN: 14602059. DOI: 10.1093/bioinformatics/btu270.

[123] Thomas Voegtlin. "Recursive self-organizing maps". In: *Neural Networks* 15.8 (2002), pp. 979–991.

[124] Weijian Wan and Donald Fraser. "A multiple self-organizing map scheme for remote sensing classification". In: *Int. Work. Mult. Classif. Syst.* Springer. 2000, pp. 300–309.

[125] Chunyu Wang et al. "Computational Approaches in Detecting Non-Coding RNA". In: *Curr. Genomics* 14 (2013), pp. 371–377. ISSN: 1389-2029. DOI: 10.2174/1389202913149990005.

[126] Jialei Wang, Jinfeng Zhuang, and Steven C H Hoi. "Unsupervised Multiple Kernel Learning". In: *J. Mach. Learn. Res. - Proc. Track* 20 (2011), pp. 129–144.

[127] Liguo Wang et al. "CPAT: Coding-potential assessment tool using an alignment-free logistic regression model". In: *Nucleic Acids Res.* 41.6 (2013), pp. 1–7. ISSN: 03051048. DOI: 10.1093/nar/gkt006. URL: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkt006.

[128] Qiang Wang et al. "Local kernel alignment based multi-view clustering using extreme learning machine". In: *Neurocomputing* 275 (2018), pp. 1099–1111. DOI: 10.1016/j.neucom.2017.09.060. URL: https://doi.org/10.1016/j.neucom.2017.09.060.

[129] Stefan Washietl et al. "RNAcode: robust discrimination of coding and noncoding regions in comparative sequence data". In: *Rna* (2011).

[130] Shuo Xiang et al. "Bi-level multi-source learning for heterogeneous block-wise missing data". In: *NeuroImage* 102 (2014), pp. 192–206.

[131] Chang Xu, Dacheng Tao, and Chao Xu. "A Survey on Multi-view Learning". In: *CoRR* abs/1304.5 (2013). arXiv: 1304.5634. URL: http://arxiv.org/abs/1304.5634.

[132] Jingjing Yang et al. "Group-Sensitive Multiple Kernel Learning for Object Categorization". In: *ICCV*. 2009.

[133] Jieping Ye et al. "Heterogeneous data fusion for alzheimer's disease study". In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2008, pp. 1025–1033.

[134] Lei Yuan et al. "Multi-source feature learning for joint analysis of incomplete multiple heterogeneous neuroimaging data". In: *NeuroImage* 61.3 (2012), pp. 622–632.

[135] Daniel R Zerbino, Premanand Achuthan, et al. "Ensembl 2018". In: *Nucleic Acids Res.* 46.D1 (2018), pp. D754–D761. DOI: 10.1093/nar/gkx1098. URL: http://dx.doi.org/10.1093/nar/gkx1098.

[136] Yi Zhang et al. "A review on recent computational methods for predicting noncoding RNAs". In: *BioMed research international* 2017 (2017).

[137] Bin Zhao, James T Kwok, and Changshui Zhang. "Multiple Kernel Clustering". In: *SDM*. 2009, pp. 638–649.

[138] Xuran Zhao, Nicholas Evans, and Jean-Luc Dugelay. "A subspace co-training framework for multi-view clustering". In: *Pattern Recognition Letters, Elsevier, 8 December 2013* (2013). URL: http://www.eurecom.fr/publication/4190.

[139] Dengyong Zhou and Christopher J C Burges. "Spectral clustering and transductive learning with multiple views". In: *ICML '07 Proc. 24th Int. Conf. Mach. Learn.* New York, NY, USA: ACM, 2007, pp. 1159–1166. ISBN: 978-1-59593-793-3. DOI: http://doi.acm.org/10.1145/1273496.1273642.

[140] M. Žitnik and B. Zupan. "Data Fusion by Matrix Factorization". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.1 (Jan. 2015), pp. 41–53. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2014.2343973.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Algorithmes ab initio pour l'identification et la classification des ARNs non-codants.

Mots clés : ARNs non-codants, Apprentissage automatique, Réseau de neuronnes, Cartes auto-organisatrices

Résumé : L'identification des ARN non codants (ARNncs) permet d'améliorer notre compréhension de la biologie. Actuellement, les fonctions biologiques d'une grande partie des ARNncs sont connues. Mais il reste d'autres classes à découvrir. L'identification et la classification des ARNncs dépend de plusieurs types de données hétérogènes (séquence, structure secondaire, interaction avec d'autres composants biologiques, etc.) et nécessite l'utilisation de méthodes appropriées.
Durant cette thèse, nous avons développé des méthodes basées sur les cartes auto-organisatrices (SOM). Les SOMs nous permettent d'analyser et de représenter les ARNncs par une carte où la topologie des données est conservée. Nous avons proposé un nouvel algorithme de SOM qui permet d'intégrer plusieurs sources de données sous forme numérique ou sous forme complexe (représentée par des noyaux). Ce nouvel algorithme que nous appelons MSSOM calcule une SOM pour chaque source de données et les combine à l'aide d'une SOM finale. MSSOM calcule pour chaque cluster la meilleure combinaison de sources. Nous avons par ailleurs développé une variante supervisée de SOM qui s'appelle SLSOM. SLSOM classifie les classes connues à l'aide d'un perceptron multicouches et de la sortie d'une SOM. SLSOM intègre également une option de rejet qui lui permet de rejeter les prédictions incertaines et d'identifier de nouvelles classes.
Ces méthodes nous ont permis de développer deux nouveaux outils bioinformatiques. Le premier est l'application d'une variante de SLSOM appelé IRSOM, pour la discrimination entre les ARN codants et non-codants. Nous avons montré que IRSOM permet de séparer les ARN codants des non-codants et d'identifier les ARN ambigüs avec l'option de rejet et la visualisation des SOMs. Le second s'appelle CRSOM et permet d'identifier les ARNncs connus et de découvrir de nouvelles classes en utilisant des sources de données hétérogènes. Nous avons montré que CRSOM, avec seulement deux sources de données, obtient des performances comparables à l'outil de référence (nRC) sans rejet et des performances supérieures avec le rejet.

Title: Algorithms for ab initio identification and classification of non-coding RNAs.

Keywords: Non-coding RNAs, Machine learning, Neural network, Self Organizing Map

Abstract: The non-coding RNA (ncRNA) identification helps to improve our comprehension of biology. We know the biological functions for a majority of ncRNA classes. But we don't know all the classes of ncRNAs. Besides, the identification and classification of ncRNAs rely on multiple heterogeneous sources of data (sequence, secondary structure, interaction with other biological components, etc.).
During this thesis, we developed methods relying on Self-Organizing Maps (SOM). The SOMs are used to analyze and represent the ncRNAs by a map of clusters where the topology of the data is preserved. We proposed a new SOM version called MSSOM which can handle multiple sources of data composed of numerical data or complex data (represented by kernels). MSSOM combines data sources by using a SOM for each source and learns the weights of each source at the cluster level. We also proposed a supervised variant of SOM with rejection called SLSOM. SLSOM is able to identify and classify the known classes using multi layer perceptron and the output of a SOM. The rejection options associated to the output layer allow to reject the unreliable prediction and to identify the potential new classes.

These methods lead to the development of two new bioinformatic tools. The first one called, IRSOM, is a variant of SLSOM for the discrimination of coding and non-coding RNAs. We showed that IRSOM is able to separate the coding and non-coding RNAs efficiently and to identify ambiguous RNAs using our rejection option and the SOM visualization. The second one is called CRSOM and is able to identify known ncRNAs and discover potential new classes by using multiple heterogeneous data sources. We show that CRSOM gives comparable performances with the reference tool (nRC) with only two data sources and without reject, and better results than nRC with the rejection option.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France