Pairwise Rational Kernels Applied to Metabolic Network Predictions

by

Abiel Roche Lima

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Doctor of Philosophy

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
February 2015

© Copyright 2015 by Abiel Roche Lima

Thesis advisor

Michael Domaratzki and Brian Fristensky

Author

Abiel Roche Lima

Pairwise Rational Kernels Applied to Metabolic Network Predictions

Abstract

Metabolic networks are represented by the set of metabolic pathways. Metabolic pathways are a series of chemical reactions, in which the product from one reaction serves as the input to another reaction. Many pathways remain incompletely characterized, and in some of them not all enzyme components have been identified. One of the major challenges of computational biology is to obtain better models of metabolic pathways. Existing models are dependent on the annotation of the genes. This propagates error accumulation when the pathways are predicted by incorrectly annotated genes.

Pairwise kernel frameworks have been used in supervised learning approaches, e.g., Pairwise Support Vector Machines (SVMs), to predict relationships among two pairs of entities. Rational kernels are based on transducers to manipulate sequence data, computing similarity measures between sequences or automata. Rational kernels take advantage of the smaller and faster representation and algorithms of weighted finite-state transducers. They have been effectively used in problems that handle large amount of sequence information such as protein essentiality, natural language

processing and machine translations.

We propose a new framework, Pairwise Rational Kernels (PRKs), to manipulate pairs of sequence data, as pairwise combinations of rational kernels. We develop experiments using SVM with PRKs applied to metabolic pathway predictions in order to validate our methods. As a result, we obtain faster execution times with PRKs than other kernels, while maintaining accurate predictions. Because raw sequence data can be used, the predictor model avoids the errors introduced by incorrect gene annotations.

We also obtain a new type of Pairwise Rational Kernels based on automaton and transducer operations. In this case, we define new operations over two pairs of automata to obtain new rational kernels. We also develop experiments to validate these new PRKs to predict metabolic networks. As a result, we obtain the best execution times when we compare them with other kernels and the previous PRKs.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my advisors and committee members for their help and guidance. I will also THANK my son, my partner, my relatives, my friends, some of my enemies and all the people who have supported me along the way.

This work has been funded by Natural Sciences and Engineering Research Council of Canada (NSERC) and Microbial Genomics for Biofuels and Co-Products from Biorefining Processes (MGCB2 project).

Chapter 1

Introduction

Machine learning is an area of artificial intelligence that develops algorithms to construct and study systems learned from data [Bishop, 2006]. One machine learning area is called supervised learning algorithms, where a model is inferred from labelled training data and used to predict new data [Brouard et al., 2012]. See Fig. 1.1 for a graphical representation. Supervised learning algorithms consist of two processes:

1. the training process, which uses input data and known responses or outcomes to the data to build a predictor model, as shown in Fig. 1.1 (a), and

2. the predictor process, which predicts responses to new data using the model generated in the training process, as shown in Fig. 1.1 (b).

Supervised learning can be used for Classification (i.e., for responses from a fixed number of categories, such as '+1' or '−1') and Regression (i.e., for responses that are real numbers).

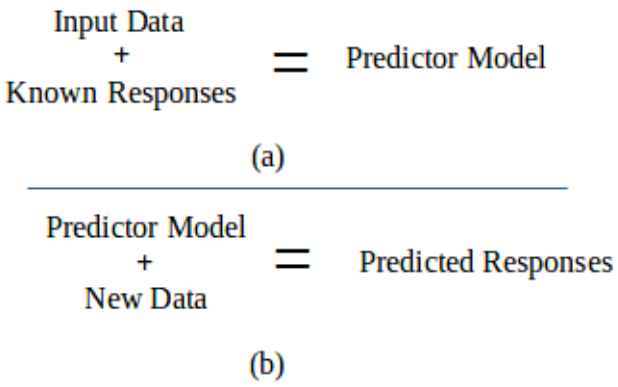

Figure 1.1: Learning Methods. (a) Training Process. (b) Predictor Process.

Pairwise classification methods are supervised learning algorithms used for classification of new "pairs" of entities [Brunner et al., 2012]. Pairwise classification approaches have been used for predicting biological networks, including metabolic networks [Ben-Hur and Noble, 2005; Kashima et al., 2010; Tsuda and Noble, 2004]. They have been also applied to other contexts such as social networks to classify co-authorship relations [O'Madadhain et al., 2005] and predict semantic relationships such as "friend-of" relations in web page links [Taskar et al., 2003].

Some of these supervised learning methods use kernels. Kernels are used to find and study general types of relations on different types of data [Cortes and Mohri, 2009]. Learning tasks need to handle and compute large amounts of input data and known responses. Kernel estimation and evaluation methods, used along with learning tasks, allow the use of linear time algorithms, avoiding computationally intensive solutions [Moreau, 2012]. Moreover, kernel methods permit the integration of heterogeneous types of data [Yamanishi, 2010]. As a consequence, predicting models can

be constructed using disparate types of data. For example biological models can be obtained by integrating proteomic, genomic and any other type of data as result of high-throughput analysis methods [De Keersmaecker et al., 2006].

Pairwise kernels are a type of kernels that describe similarity measures between two pairs of entities [Ben-Hur and Noble, 2005]. Methods involving pairwise kernels are more expensive in terms of processing than methods using regular kernels [Yamanishi et al., 2005]. When pairwise kernels are used to handle sequence data, even longer processing times and more storage are needed in comparison to kernels to manipulate non-sequence data [Kashima et al., 2010].

In this thesis, to combat these expenses, we employ rational kernels. Rational kernels are kernels based on weighted finite-state transducers that represent similarity measures between sequences or automata [Cortes et al., 2004]. Rational kernels take advantage of transducer algorithms and representations to manipulate sequence data [Cortes and Mohri, 2009]. Automata and transducers have been used in applications including natural language [Allauzen et al., 2004; Lothaire, 2005], image processing [Albert and Kari, 2009; Mohri et al., 2005b] and bioinformatics [Bradley and Holmes, 2007; Holmes, 2003; Westesson et al., 2011].

Biological networks are formal models that store data and encode molecular interactions in a cell. They help to characterize biological entities and their interaction patterns [Yoon et al., 2012]. Relevant types of biological networks include: transcriptional regulatory networks [Lee et al., 2002], signal transduction networks [Janes, 2006], protein-protein interaction networks [Ben-Hur and Noble, 2005], and metabolic

networks [Helden et al., 2012]. This thesis is focused on metabolic networks.

Metabolic networks allow the modelling of molecular systems to understand the underlying biological mechanisms in a cell [Faust and Helden, 2012]. Metabolic networks are represented by the set of metabolic pathways, as is shown in Fig. 1.2 (a). Metabolic pathways are chemical reactions in the cell where enzymes are involved to produce other compounds (see Fig. 1.2 (b) as a graphical representation of a metabolic pathway). For example, in a glycolysis pathway, glucose is broken down into smaller products, such as carbon dioxide and water [Demain et al., 2005]. The experimental determination of metabolic networks is still very challenging [Beurton-Aimar et al., 2014]. Thus, there have been several efforts to develop supervised learning methods to determine genes coding for missing enzymes and predict unknown parts of metabolic networks [Karp et al., 2011; Osterman and Overbeek, 2003].

Some methods for predicting metabolic networks have recently been developed in the framework of kernel methods [Kashima et al., 2010; Kotera et al., 2013; Yamanishi, 2010]. Development of pathway databases such as KEGG [Kanehisa et al., 2008] and BioCyc [Caspi et al., 2012] have made available data to allow analysis of the current knowledge of metabolic networks, using new kernels and supervised learning methods.

As the general motivations of this research, we consider that the genome sequencing projects have provided biologists with a large amount of data regarding the set of genes found in several organisms. Efforts are now focused on how this information fits together and how biomolecules interact [Khatri et al., 2012; Tatusova et al., 2014]. Thus, there are demands for computer applications to analyze, model and predict the

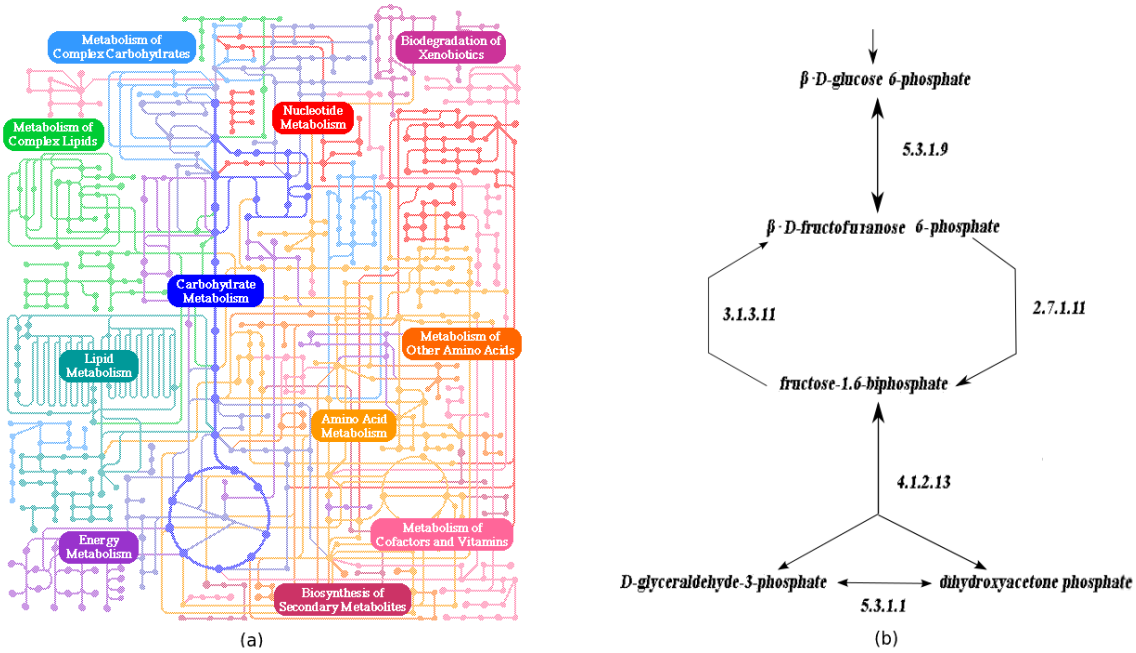

Figure 1.2: Examples of Metabolic Network and Metabolic Pathway. (a) Metabolic Network. (b) Partial Metabolic Pathway.

network relations inside the cells [Beurton-Aimar et al., 2014]. Many of the available methods to predict metabolic networks are based on the assumption that the genome annotation is correct. Then, the metabolic pathway predictions will be wrong if there are errors in the annotation. Normally, expensive computational costs, such as long processing times and storage, are needed in these types of applications [Pruitt, 2014; Yamanishi, 2010]. These costs are even higher when raw sequence data are used. Thus, one of the main challenges for computer scientists is to develop algorithms that can make better predictions in an acceptable period of time [Kotera et al., 2013; Kyrpides et al., 2014].

As the computer science contribution of this thesis, we define for first time a new framework, called Pairwise Rational Kernels (PRKs), which are the combinations of

pairwise and rational kernels. PRKs are used with supervised learning methods to manipulate sequence data, speeding up the computation and manipulation of the kernels. PRKs can be applied to any problem where two pairs of sequence data need to be manipulated, for example on web-based, semantic or biological networks. We define seven different PRKs, based on the state-of-the-art of pairwise kernel combinations, rational kernel representations and automaton operations.

In order to validate the new framework (i.e., the seven PRKs), we use Support Vector Machine (SVM) methods with these kernels to predict metabolic networks (see a general idea of the method in Fig. 1.3). PRKs are used to efficiently compute the similarities between the pairs of nucleotide sequences corresponding to the genes. We design several experiments applying SVMs and PRKs to predict metabolic pathways of the species *Saccharomyces cerevisiae*. This species is selected because other data

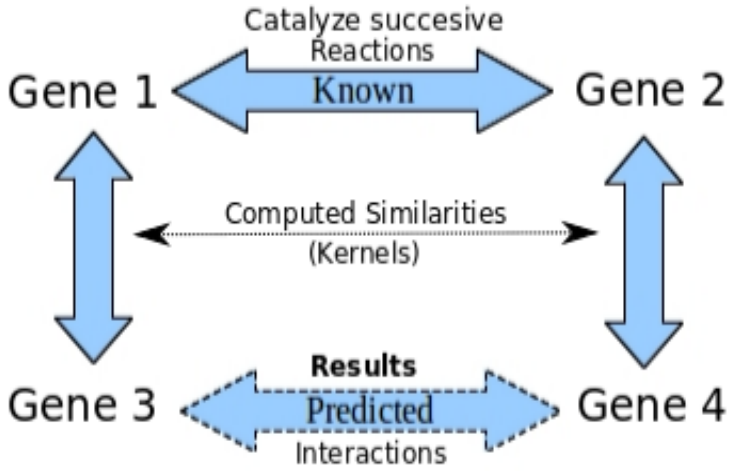

Figure 1.3: Given known interacting genes (gene 1 and gene 2), similarity measures are computed in order to predict other gene interactions (gene 3 and gene 4).

and kernel models (e.g., Ben-Hur and Noble [2005]; Kashima et al. [2010]; Yamanishi [2010]) are available to validate and compare our proposal. As results, when only sequence data are used, better execution times are obtained, while the accuracy values are maintained and sometimes improved. As raw sequence data can be used during the prediction, the error accumulation due to wrong genome annotation may be avoided.

Chapter 2

Background

This thesis relates to several areas in computer and biological sciences. We define in this chapter the main terms and concepts that describe the results obtained during the research.

2.1 Automata and Transducers

Automata define a mathematical formalism to analyze and model real problems through useful machines [Rabin and Scott, 1959]. An automaton has a set of states (generally represented by circles), and transitions (generally represented by arrows). The automaton moves from one state to another state (makes a transition) when activated by an event or function. One variant of an automaton is called finite state machine. A finite-state machine can be used to model a simple system, such as turn-

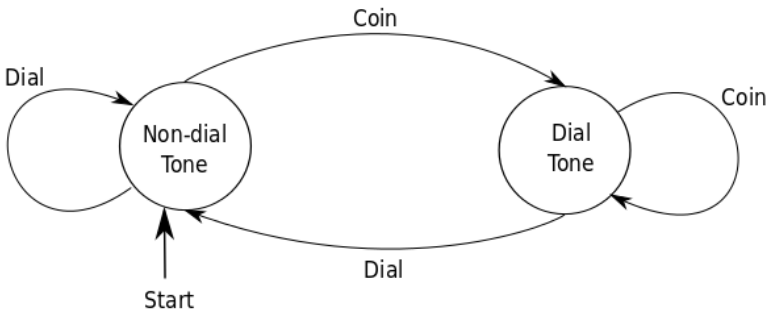

Figure 2.1: Finite-state machine diagram for a pay phone.

stiles or transit lights, or complex systems such as sophisticated spaceship controls. Fig. 2.1 shows an example of a simple finite-state machine which models a pay phone. The pay phone has two states (i.e., "Non-dial Tone" or "Dial Tone") and two inputs ("Coin" and "Dial") that affect what state the machine is in. The automaton starts in the "Non-dial Tone" state. When "Coin" is inserted, the machine goes to "Dial Tone", otherwise if the entry is "Dial" the machine stays in the same state. When the automaton is in the "Dial Tone" state and "Coin" is inserted, it stays in the same state, else if "Dial" is read, it moves to the "Non-Dial Tone". Detailed versions of this type of models can be used to develop in-built software applications in digital pay phones to control calls, payments and other services.

Automata work over sequence of symbols, where Σ^* denotes all the finite sequences using the symbols over the alphabet Σ, including ϵ that represents the empty symbol. In order to formally define automata and transducers, we will follow the notations used by Cortes and Mohri [2009]. An automaton A is a 5-tuple $(\Sigma, Q, I, F, \delta)$ [Rabin and Scott, 1959] where Σ is the input alphabet set, Q is the state set, $I \subset Q$ is the subset of initial states, $F \subset Q$ is the subset of final states, and $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q$

is the transition set. A transition $\iota \in \delta$ describes the actions of moving from one state to another when a condition (input symbol) is encountered.

Similarly, a Finite-State Transducer (FST) is an automaton where an output label is included in each transition in addition to the input label. Based on the above definition, a FST T is a 6-tuple $(\Sigma, \Delta, Q, I, F, \delta)$ [Cortes et al., 2004], where the new term Δ is the output alphabet and the transition set δ is now $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times Q$. Similar to the previous definition, a transition $\iota \in \delta$ is the action of moving from one state to another when input symbol is encountered and the output symbol is produced.

In addition, Automata and Finite-State Transducers can be weighted, where each transition is labelled with a weight. Thus, a Weighted Automaton (WA) is a 7-tuple $(\Sigma, Q, I, F, \delta, \lambda, \rho)$ and a Weighted Finite-State Transducer (WFST) is a 8-tuple $(\Sigma, \Delta, Q, I, F, \delta, \lambda, \rho)$ Cortes et al. [2004], where the new terms λ and ρ are: $\lambda : I \to \mathbb{R}$, the initial weight function, and $\rho : F \to \mathbb{R}$, the final weight function. The new transitions for the WAs and WFSTs are $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times \mathbb{R} \times Q$ and $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{R} \times Q$, respectively, where $\mathbb{R}$ represents the weights as real numbers.

We denote as $p[\iota]$ and $n[\iota]$ the origin and destination of the transition ι, where $\iota \in \delta$ and $w[\iota]$ is the associated weight. A path π is denoted as $\pi = \iota_1 ... \iota_k \in \delta^*$, where the initial state is $p[\pi] = p[\iota_1]$, the final state is $n[\pi] = n[\iota_k]$ and the weight $w[\pi] = w[\iota_1] * ... * w[\iota_k]$. The set of all paths P from the initial state I to the final state F with input and output sequences x and y, respectively, is denoted as $P(I, x, y, F)$.

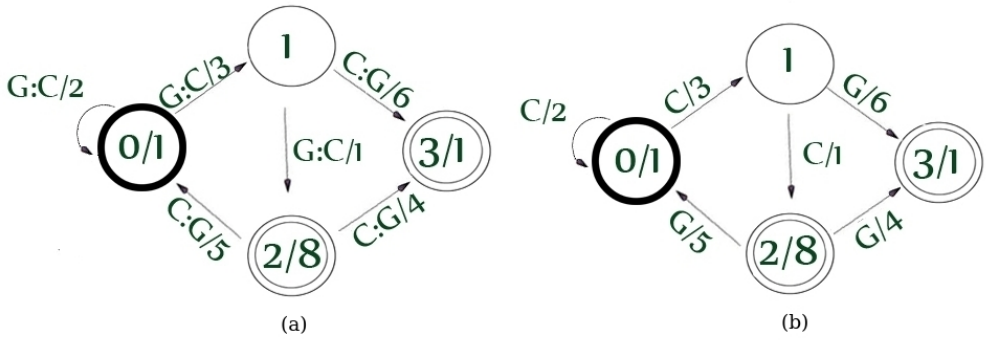

Figure 2.2: Weighted transducer and Weighted Automaton representing sequences in the alphabet $\Sigma = \Delta = \{G, C\}$. (a) Weighted Transducer T. (b) Weighted Automaton A (A is obtained by projecting the output of T).

The weight associated to the transducer T for a pair of sequences (x, y) is $T(x, y) = \sum_{\pi \in P(I,x,y,F)} \lambda(p[\pi])w[\pi]\rho(n[\pi])$, where $(x, y) \in \Sigma^* \times \Delta^*$. The weighted transducer T is *regulated* if the final $T(x, y)$ value is in $\mathbb{R} \cup \{\infty\}$. This definition can also be extended for weighted automata, considering only the input or output labels of a weighted transducers [Cortes and Mohri, 2009]. All the transducers and automata in this research are regulated.

As an example, a weighted transducer is shown in Fig. 2.2 (a). We use as delimiters the colon to separate the input and output labels of the transitions and the slash to separate the weight values (i.e., the notation is *input:output/weight*). States are represented by circles, where the set of initial states are bold circles and the set of final states are double circles. Only the initial and final states have associated weights (the notation is *state/weight*). Example 2.1 shows how to compute the weight to the transducer T (i.e., $T(x, y)$) for two given sequences x and y. In this case, we define the alphabets $\Sigma = \{G, C\}$ and $\Delta = \{G, C\}$.

Example 2.1. The weight (or value) associated to the transducer T in Fig. 2.2 (a)

for the pair $(x, y) = (GGC, CCG) \in \Sigma^* \times \Delta^*$ is computed as:

$$T(GGC, CCG) = 1 * 2 * 3 * 6 * 1 + 1 * 3 * 1 * 4 * 1 = 48,$$

considering that there are two accepting paths labelled with input GCC and output CCG. These paths are:

$$Path\ 1 : State\ 0 \mapsto State\ 0 \mapsto State\ 1 \mapsto State\ 3,$$

$$Path\ 2 : State\ 0 \mapsto State\ 1 \mapsto State\ 2 \mapsto State\ 3.$$

The initial and final values in the terms of $T(GGC, CCG)$ correspond to the weights of the initial and final states.

Fig. 2.2 (b) shows a graph representation of a weighted automaton. It can be obtained as the output projection of the transducer T where the input labels are omitted. Thus, the alphabet Δ is $\Delta = \{G, C\}$ and the weight computation of the automaton A for two given sequences is shown in Example 2.2.

Example 2.2. The weight (or value) associated to the Automaton A in Fig. 2.2 (b) for $y = CCG \in \Delta^*$ is computed as:

$$A(CCG) = 1 * 2 * 3 * 6 * 1 + 1 * 3 * 1 * 4 * 1 = 48$$

considering that there are two accepting paths labelled with CCG. These paths are:

$$Path\ 1 : State\ 0 \mapsto State\ 0 \mapsto State\ 1 \mapsto State\ 3,$$

$$Path\ 2 : State\ 0 \mapsto State\ 1 \mapsto State\ 2 \mapsto State\ 3.$$

The initial and final values in the terms of $A(CCG)$ correspond to the weights of the initial and final states.

There are several operations defined on automata and transducers, such as *sum* $+$ (union) and *product* $\cdot$ (concatenation) [Mohri, 2009]. Given two transducers T_1 and T_2, the sum is defined as $(T_1 + T_2)(x, y) = T_1(x, y) + T_2(x, y)$ and the product is

$(T_1 \cdot T_2)(x, y) = \sum_{x_1 x_2 = x, y_1 y_2 = y} T_1(x_1, y_1) \cdot T_2(x_2, y_2)$, $\forall (x, y) \in \Sigma^* \times \Delta^*$.

Other operations are *inverse* and *composition*. Given any transducer T, the *inverse* T^{-1} is the transducer obtained when the input and output labels are swapped for each transition. The *composition* operation of the transducers T_1 and T_2 with matching input and output alphabets boh equal to Σ is a weighted transducer, denoted by $T_1 \circ T_2$, provide that the sum given by $(T_1 \circ T_2)(x, y) = \sum_{z \in \Sigma^*} T_1(x, z) T_2(z, y)$ is well defined in $\mathbb{R}$ for all $(x, y) \in \Sigma^*$ [Cortes and Mohri, 2009].

2.2 Kernels Methods

Kernel Methods are used in supervised learning approaches for classification, ranking, regression and clustering tasks [Lee et al., 2005]. The recent popularity of kernel methods have been described by Hofmann et al. [2008]. This popularity is based on the fact that linear methods have been extensively studied and applied to the theory and algorithms of statistics and machine learning for many years. Linear methods make decisions using values from linear combination of the variables. However, nonlinear methods are required to make successful prediction in real world data problems. To convert the data into a new space where linear methods can be applied, a nonlinear mapping function, Φ, can be used to map each point of the input space (where data are nonlinearly separable) X to a high-dimensional feature space (where data are linearly separable) F. Fig. 2.3 shows a graphical representation. The efficiency and practicality of these solutions can be compromised for high dimensionality of F, since dot products are computationally expensive. Kernel methods solve this problem

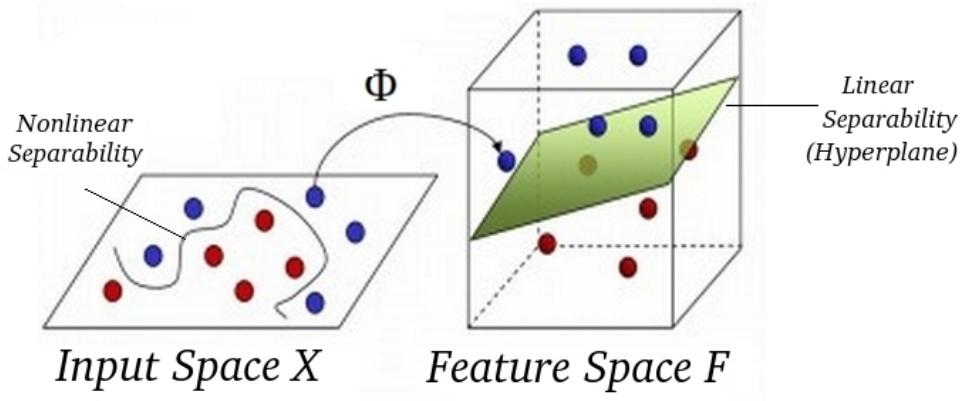

Figure 2.3: Mapping points from input space X to high-dimensional feature space F using function Φ.

using a kernel function, k, which coincides with the product of their images in the new feature space. Kernel methods also allow working on the same mathematical framework across different types of data [Moreau, 2012].

Given a kernel $k : X \times X \rightarrow \mathbb{R}$, the value $k(x, y)$ coincides with the dot product of their images $\Phi(x)$ and $\Phi(y)$ in the feature space F, i.e., $k(x, y) = \Phi(x) \cdot \Phi(y)$ for $\forall x, y \in X$. k also defines a similarity measure between x and y. For a set of points $x_i, y_i \in X$, $k(x_i, y_i)$ represents a matrix with their values. This matrix needs to be positive definite and symmetric - PDS (i.e., fulfills the Mercer's condition [Mercer, 1909]). This condition is necessary to use the matrix for training convergence in algorithms such as Support Vector Machines [Lee et al., 2005] (see Section 2.3 for more details about SVM).

Hofmann et al. [2008] described some examples of kernels, such as Polynomial kernels, spline kernels, convolutions kernels, ANOVA kernels, n-grams kernels, trees kernels and graph kernels. A suitable kernel that captures the implicit patterns of the data is

the most helpful tool in a supervised learning approach based on kernel methods [Ben-Hur et al., 2008]. There are no rules for choosing a kernel, and finding an appropriate kernel can be a trial-and-error process.

In this research, we have conveniently divided the type of kernels in two groups: non-sequence and sequence kernels. Roughly speaking, the sequence kernels (also called string kernels) are functions that manipulate data as finite sequence of symbols of different lengths (e.g., the path kernel [Takimoto and Warmuth, 2004], PFAM kernel [Allauzen et al., 2008; Ben-Hur and Noble, 2005; Gomez et al., 2003] and motif kernel [Huang and Brutlag, 2001; Kuang et al., 2005]). Non-sequence kernels handle any other type of data, in this case, binary or numerical (e.g., gene expression, gene localization, and phylogenetic data [Kashima et al., 2010; Vert et al., 2007; Yamanishi, 2010]). We have defined specific non-sequence and sequence kernels in the context of supervised learning methods in Section 4.2.

2.2.1 Rational Kernels

In order to manipulate sequence data, FSTs provide a simple representation as well as efficient algorithms such as composition and shortest-distance [Cortes et al., 2004]. Rational Kernels, based on Finite-State Transducers, are effective for analyzing sequences with variable lengths [Cortes and Mohri, 2009].

As a formal definition, a function $k : \Sigma^* \times \Delta^* \to \mathbb{R}$ is a *rational kernel* if there exists a WFST U such that k coincides with the function defined by U, i.e., $k(x, y) = U(x, y)$ for all sequences $x, y \in \Sigma^* \times \Delta^*$ [Cortes and Mohri, 2009]. From now on, we consider

input and output alphabets with the same symbols (i.e., $\Sigma = \Delta$), and only the terms Σ and Σ^* will be used.

In order to compute the value of $U(x, y)$ for a particular pair of sequences $x, y \in \Sigma^* \times \Sigma^*$, the composition algorithm of weighted transducers is used [Cortes and Mohri, 2009]:

- First, M_x, M_y are considered as trivial weighted transducers representing x, y respectively, where $M_x(x, x) = 1$ and $M_x(v, w) = 0$ for $v \neq x$ or $w \neq x$. M_x is obtained using the linear finite automata representing x by augmenting each transition with an output label identical to the input label and by setting all transition, initial and final weights to one. M_y is obtained in a similar way by using y.

- Then, by definition of weighted transducer composition:
  $(M_x \circ U \circ M_y)(x, y) = M_x(x, x)U(x, y)M_y(y, y)$.
  Considering $M_x(x, x) = 1$ and $M_y(y, y) = 1$, we obtain $(M_x \circ U \circ M_y)(x, y) = k(x, y)$, i.e., the sum of the weights of all paths of $M_x \circ U \circ M_y$ is exactly $U(x, y) = k(x, y)$.

Based on this representation, there is a two-step algorithm to obtain $k(x, y) = U(x, y)$ [Cortes and Mohri, 2009].

Algorithm 1 *Rational Kernel Computation*

INPUT: pair of sequences (x, y) and a WFST U
(i) compute N using composition as $N = M_x \circ U \circ M_y$
(ii) compute the sum of all paths of N using shortest-distance algorithm, which is equal to $U(x, y)$.
RESULTS: value of $k(x, y) = U(x, y)$

Using Algorithm 1, the overall complexity to compute one value for the rational kernel is $\mathcal{O}(|U||M_x||M_y|)$, where $|U|$ remains constant. In the practice, this complexity is reduced to $\mathcal{O}(|U|+|M_x|+|M_y|)$ in many kernels which have been used in areas such as natural language processing and computational biology. For example, the Algorithm 1 for the n-gram kernel, described in Section 2.2.2, has a linear complexity.

As mentioned above, kernels used in training methods classification algorithms (e.g., SVM) need to be PDS. Cortes et al. [2004] have proven a result that gives a general method to construct a PDS rational kernel using any WFSTs.

Theorem 1 ([Cortes et al., 2004]). If T is an arbitrary weighted transducer, then $U = T \circ T^{-1}$ defines a PDS rational kernel.

2.2.2 n-gram Kernel as a Rational Kernel

Hofmann et al. [2008] has defined a class of similarity measures between two biological sequences as a function of the number of equal subsequences that they have. As an example of such measures is the spectrum kernel defined by Leslie et al. [2004]. Similarity values are the results of summing all the products of the counts for the same subsequences. It is also referred to in computational biology as the k-mer or n-gram kernel. In the rest of this thesis, we use the term n-gram to follow the notation of Hofmann et al. [2008] and Cortes and Mohri [2009].

The n-gram kernel is defined as $k_n(x, y) = \sum_{|z|=n} c_x(z)c_y(z)$ for a fixed integer n, which represents subsequences of length n. Here, $c_a(b)$ is the number of times that

Table 2.1: Counting how many time z's occurs in x (i.e., $c_x(z)$) and how many times z occurs in y (i.e., $c_y(z)$). Computing the final kernel value (i.e., $k_2(x, y)$).

z	**x**=CCGCC	**y**=GGGCC	$c_x(z)c_y(z)$
CC	$c_x(z) = 2$	$c_x(z) = 1$	2
GC	$c_x(z) = 1$	$c_x(z) = 1$	1
CG	$c_x(z) = 1$	$c_x(z) = 0$	0
GG	$c_x(z) = 0$	$c_x(z) = 2$	0
$k_2(x, y)$	-	-	**3**

the subsequence b appears in a. k_n can be represented as a rational kernel using the weighted transducer $U_n = T_n \circ T_n^{-1}$, where the transducer T_n is defined as $T_n(x, z) = c_x(z)$, for all $x, z \in \Sigma^*$ with $|z|= n$ Cortes et al. [2004]. Example 2.3 shows how the 2-gram ($n = 2$) kernel is computed for a given pair of sequences.

Example 2.3. Given the alphabet $\Sigma = \{G, C\}$ and the sequences $x = CCGCC$, $y = GGGCC$, then the 2-gram kernel considers all the subsequences $z \in \Sigma^*$, with $|z|= 2$, i.e., $z = CC$, $z = GC$, $z = CG$, $z = GG$, where:

- $c_x(z)$ counts how many times z occurs in x, e.g., for $z = CC$, z occurs in $x = CCGCC$ twice (i.e., $\underline{CC}G\underline{CC}$), then $c_x(z) = 2$,
- $c_y(z)$ counts how many times z occurs in y, e.g., for $z = CC$, z occurs in $y = GGGCC$ once (i.e., $GGG\underline{CC}$), then $c_y(z) = 1$,

Finally, $k_2(x, y) = k_2(CCGCC, GGGCC) = \sum_{|z|=2} c_x(z)c_y(z)$. Table 2.1 shows the corresponding values of $c_x(z)$ and $c_y(z)$ for all different z's, as well as the final value of $k_2(x, y)$.

k_n can be represented as a rational kernel using the weighted transducer $U_n = T_n \circ T_n^{-1}$, where the transducer T_n is defined as $T_n(x, z) = c_x(z)$, for all $x, z \in \Sigma^*$ with $|z|= n$

[Cortes et al., 2004]. Fig. 2.4 represents the transducer T_2 for the alphabet $\Sigma = \{G, C\}$ that counts how many time CC appears in any given sequence.

The n-gram rational kernel can be constructed in time $\mathcal{O}(|U_n|+|M_x|+|M_y|)$, as described by Allauzen et al. [2008]; Mohri [2009].

Allauzen et al. [2008] extended the construction of this kernel, k_n, to measure the similarity between sequences represented by automata. Firstly, they define the count of a sequence z in a weighted automaton A as $c_A(z) = \sum_{u \in \Sigma^*} c_u(z)A(u)$, where u ranges over the set of sequences in Σ^* which can be represented by the automaton A. This equation represents the sums obtained for each u, of how many times z occurs in u multiplied by the weight (or value) associated to the sequence u in the automaton A.

Then, the similarity measure between the weighted automata A_1 and A_2, according

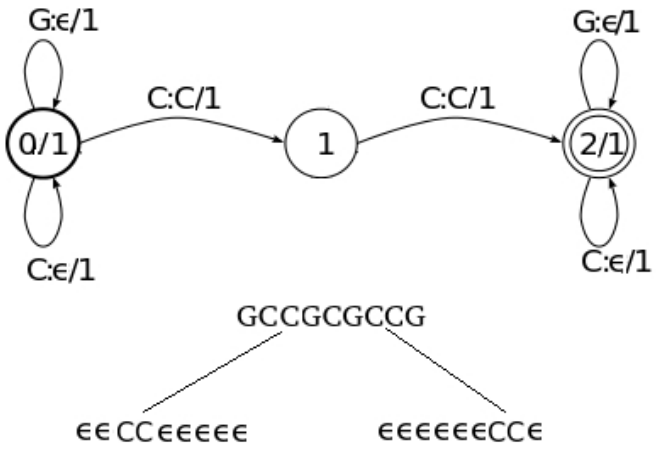

Figure 2.4: Counting transducer T_2 for $\Sigma = \{G, C\}$. T_2 counts how many times the subsequence CC occurs in the input sequence $GCCGCGCCG$, i.e., $T_2(GCCGCGCCG, CC) = 2$, because the $\underline{CC}$ occurs twice in $G\underline{CC}GCG\underline{CC}G$ (ϵ represents the empty symbol).

to the n-gram kernel k_n, is defined as:

$$k_n(A_1, A_2) = \sum_{x,y \in X} (A_1 \circ T_n \circ T_n^{-1} \circ A_2)(x, y) = \sum_{|z|=n} c_{A_1}(z) c_{A_2}(z) \tag{2.1}$$

Yu et al. [2010] have verified that n-gram sequence kernels alone are not strong enough to predict protein interactions. We address their concerns by combining n-gram with other kernels that include evolutionary information (see Section 2.4.1 for more details).

2.2.3 Pairwise Kernels

We apply kernel methods to the problem of predicting relationships between two given entities, i.e., pairwise prediction. Typically, the construction of pairwise kernels K are based on combinations of simple kernels, k, which define similarities to predict or classify one single entity. For example, a simple kernel can be defined with the PFAM database [Gomez et al., 2003] to learn a SVM model and predict protein essentiality of a single protein (entity) [Allauzen et al., 2008].

In this research, four different pairwise kernels, K, are defined based on the simple kernels $k : X \times X \to \mathbb{R}$ and the entities $x_1, y_1, x_2, y_2 \in X$:

1. Direct Sum Learning Pairwise Kernel (defined by Hertz et al. [2004]):

$$K_{DS}((x_1, y_1), (x_2, y_2)) = k(x_1, x_2) + k(x_1, y_2) + k(y_1, x_2) + k(y_1, y_2) \tag{2.2}$$

2. Pairwise Tensor Product Kernel or Kronecker Kernel (defined separately by Ben-Hur and Noble [2005], Basilico and Hofmann [2004] and Oyama and Manning [2004]):

$$K_T((x_1, y_1), (x_2, y_2)) = k(x_1, x_2) \cdot k(y_1, y_2) + k(x_1, y_2) \cdot k(y_1, x_2) \tag{2.3}$$

3. Metric Learning Pairwise Kernel (defined by Vert et al. [2007]):

$$K_M((x_1, y_1), (x_2, y_2)) = (k(x_1, x_2) - k(x_1, y_2) - k(y_1, x_2) + k(y_1, y_2))^2 \tag{2.4}$$

4. Cartesian Pairwise Kernel (defined by Kashima et al. [2010]):

$$\begin{aligned} K_C((x_1, y_1), (x_2, y_2)) = k(x_1, x_2) \cdot \delta(y_1 = y_2) + \delta(x_1 = x_2) \cdot k(y_1, y_2) \\ + k(x_1, y_2) \cdot \delta(y_1 = x_2) + \delta(x_1 = y_2) \cdot k(y_1, x_2) \end{aligned} \tag{2.5}$$

   where $\delta(x = y) = 1$ if $x = y$ or 0 otherwise, for $\forall x, y \in X$.

Using these functions the pairwise kernel, K, relates the images of $\Phi(x_1)$, $\Phi(y_1)$, $\Phi(x_2)$ and $\Phi(y_2)$. All these pairwise functions guarantee the symmetry of the pairwise kernels K, $K((x_1, y_1), (x_2, y_2)) = K((x_2, y_2), (x_1, y_1))$, where $x_1, x_2, y_1, y_2 \in X$. Also, if the simple kernel k is PDS (satisfies the Mercer condition), the resulting pairwise kernel K also is PDS, for each of the pairwise kernels defined above [Brunner et al., 2012; Kashima et al., 2010].

The following example shows how a pairwise kernel can be represented:

Example 2.4. Given the simple PFAM kernel k and the proteins x and y, where $k_{PFAM}(x, y)$ [Ben-Hur and Noble, 2005] describes similarity measure based on PFAM database between the protein x and y (see Section 4.2.2 for more details), then, the Tensor Product Pairwise Kernel [Basilico and Hofmann, 2004; Ben-Hur and Noble, 2005; Oyama and Manning, 2004], $K((x_1, y_1), (x_2, y_2))$ is computed using the simple PFAM Kernel k_{PFAM} as:
$K((x_1, y_1), (x_2, y_2)) = k_{PFAM}(x_1, x_2) \cdot k_{PFAM}(y_1, y_2) + k_{PFAM}(x_1, y_2) \cdot k_{PFAM}(y_1, x_2)$,
where x_1, y_1, x_2, y_2 represent genes or proteins (see .

2.3 Support Vector Machine

The rationale for the preceding discussion on representing disparate types of data as kernels is to enable us to use them in machine learning formalism such as Support Vector Machines (SVMs). SVMs are used for classification and regression analysis, defined as supervised models with associated learning algorithms [Cortes and Vapnik, 1995]. In this thesis, we use SVMs for classification.

SVMs represents the data as objects (e.g., each point in Fig. 2.3 represent an object) in a feature space. As a training set, several examples (vectors) classified in two categories are given. A SVM is trained to find a hyperplane that separates the vector space in two parts (see in Fig. 2.3 the Hyperplane in the Feature Space F). Each part of the feature space groups the examples into the same category (in Fig. 2.3 all the red points belong to one category and blue points belong to a different category). The optimal hyperplane is defined by the largest distance (margin) to the nearest

data point of both categories (this nearest point is called a support vector). Then, new examples and composition can be classified depending their localization in the feature space related to the hyperplane [Cortes and Vapnik, 1995].

In the real world problems, many data can not be linearly separable. To solve it, SVM methods can be applied, providing two solutions:

- assume a soft margin, where few given examples may fall on the wrong part of the hyperplane during the training process, or
- use a kernel function, that map the given examples to a high-dimensional feature space (also just called the "feature space") where a linearly separable hyperplane is possible (as is shown in Fig. 2.3).

In this research, we use both, by computing some of the kernels described in Section 2.2 and implementing the soft margin SVM [Cortes and Vapnik, 1995].

The SVM formulation using a kernel function for binary classification can be described as follows:
Given training data (x_i, d_i) where $i = 1, .., n$, d_i has binary values (e.g., $+1$ or -1, corresponding to two categories) and the mapping function Φ, then the SVM methods find the optimal hyperplane, $w^T\Phi(x) + b = 0$, which separates the points into two categories, where w is the normal vector and b the offset of the hyperplane. An example can be seen in Fig. 2.3 for a 3-dimensional feature space, where red and blue points are separated by a hyperplane. As a result, a hyperplane is obtained with the maximum margin and minimum loss on the training data.

To implement the SVM method, there exists two different formulations: the primal and dual formulations [Platt, 1998]. Regularly, the primal formulation cannot be solved directly, because the mapping function is unspecified [Cortes and Vapnik, 1995]. Thus, the solution is obtained by moving to the dual formulation of the optimization problem, where the kernel function is included. The dual decision function predicts which category the unseen example x is classified. This function is:

$$f(x) = \sum_i^n \alpha_i k(x_i, x) + b,$$

where k is the kernel, x_i is the set of training examples ($i = 1, .., n$) and α and b are the learned parameters during the training process, α is obtained from the Lagrange Multipliers as a function of w [Cortes and Vapnik, 1995]. Using f, unseen examples x are classified (e.g., if $f(x) >= 0$, x is classified into the $+1$ category, else if $f(x) < 0$, x is classified into the -1 category).

The implementation of the SVM dual formulation used in this thesis is the sequential minimal optimization (SMO) technique [Platt, 1998] from LIBSVM package [Chang and Lin, 2011]. In the SMO algorithm, the problem is divided into 2 sub-problems using coordinate descent and solve analytically [Cristianini and Shawe-Taylor, 2000]. The parameters include the kernel k, the Lagrange Multipliers parameters α and the hyperparameter for regularization C. The parameter C controls the trade-off between achieving a low error on the training data and minimising the norm of the weights [Cristianini and Shawe-Taylor, 2000]. In this thesis, we set the parameter $C = 1$, then the penalization for trained vectors being missclassified is high. Since our goal is to compare our results with other previous works, such as Kashima et al. [2010],

we set the parameters with similar values.

2.3.1 Pairwise Support Vector Machine

Pairwise SVMs classify whether if a pair (x_1,y_1) belong to the same category or to a different category. Then, while SVM methods classify simple entities, pairwise SVM methods classify pair of entities [Brunner et al., 2012].

Let us formally define the binary Pairwise Support Vector Machine formulation, following Brunner et al. [2012]:
Given a training dataset $((x_i, y_j), d_i))$, where d_i has binary values (i.e., the pair (x_i, y_j) is classified as $+1$ or -1), $i = 1, .., n$ and the mapping function Φ, then the Pairwise SVM methods find the optimal hyperplane, $w^T\Phi(x_i, y_i) + b = 0$, which separates the points in two categories.

Similarly, the dual formulation can be defined using the pairwise kernels described in Section 2.2.3. The dual decision function for this case is:

$$f(x, y) = \sum_i^n \alpha_i K((x_i, y_i), (x, y)) + b,$$

where K is the pairwise kernel, (x_i, y_j) is the set of training examples α_{ij} and b are the learned parameters during the training process and α_{ij} is obtained from the Lagrange Multipliers as a function of w. f classifies the new pairs (x, y). For example $f(x, y) >= 0$, (x, y) is classified as $+1$, and $f(x, y) < 0$, (x, y) is classified as -1. As an implementation of SVM, we use sequential minimal optimization (SMO), using the pairwise kernel K and the parameter C set as 1 (see Section 2.3 for more details).

2.4 Metabolic Networks

In this work, a metabolic network is represented as a graph, in which the vertices are the enzymes, and the edges are the enzyme-enzyme relations (i.e., two proteins are enzymes that catalyze successive reactions in known pathways). Fig. 2.5 represents a graphical transition from a metabolic pathway to a graph.

In a traditional representation of a metabolic pathway, enzymes are vertices (nodes), and metabolites are edges (branches). Following Yamanishi [2010], we represent it differently, where the interactions between pairs of enzymes are considered discrete data points. For example, in Fig. 2.5 (a), the enzyme numbered 5.3.1.9 can create D-fructose-6-phosphate as a product, which is in turn used as a substrate by the enzyme numbered 2.7.1.11. This means there is an enzyme-enzyme relation between 5.3.1.9 and 2.7.1.11. Then, we create a graph in which enzyme-enzyme relations become edges and enzymes are nodes as is shown in Fig. 2.5 (b). If there is a relation between two enzymes, such a relation is classified as $+1$ (i.e., interacting pair). Enzyme-enzyme pairs for which no relation exists are classified as -1 (non-interacting pairs). Fig. 2.5 (c) describes these classifications, which are used as training set in the SVM method.

2.4.1 Dataset

In the experiments of this thesis, we used information of the yeast *Saccharomyces cerevisiae* [Sikorski and Hieter, 1989]. This species was selected because it is a well-

studied organism. Moreover, several models, using kernels, have been described and tested using data from this species such as Ben-Hur and Noble [2005]; Kashima et al. [2010]; Yamanishi [2010]. Therefore, this is a good candidate to compare our methods, implementations and results with other methods that also predict biological networks for *Saccharomyces cerevisiae*.

The data for this species were taken from the KEGG pathway [Kanehisa et al., 2008] and converted to a graph as described in the previous section. There were 755 nodes and 2575 interacting pairs in the graph for this species. As we used SVM method for

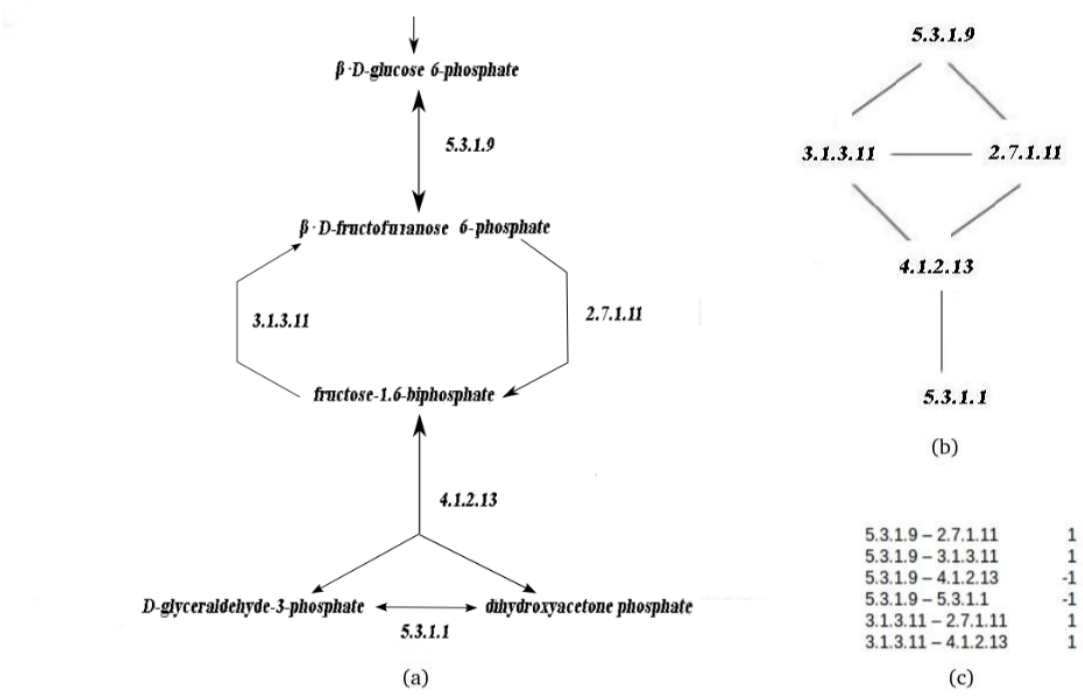

Figure 2.5: Conversion from a metabolic network to a graph representation. (a) Part of the Glycolysis Pathways, from BioCyc Database Caspi et al. [2012]; Latendresse et al. [2012]. (b) The resulting graph with the nodes (enzymes) and edges (enzyme-enzyme relations). (c) Table that represents known enzymes relations (EC numbers related are classified as +1 and non-related as -1).

the metabolic network inference, we found an unbalanced proportions of interacting (+1) and non-interacting (−1) classified pairs (e.g., for this dataset there were 282060 non-interacting pairs). In order to balance our dataset, we followed the procedure recommended by Yu et al. [2010], using the program BRS-noint to select non-interacting pairs. Yu et al. [2010] describes the bias towards non-interacting pair selection during the training process and the accuracy estimation. To solve it, the BRS-noint program is used to create a "balanced" negative set to maintain the right distribution of non-interacting and interacting pairs. As a result, we obtained 2574 non-interacting pairs for a total of 5149 pairs in the training process that we used in all the experiments.

2.4.2 Bioinformatics Data

Biological data are being generated from several genome sequencing projects [Ellegren, 2014; Kyrpides et al., 2014; Pruitt, 2014; Tatusova et al., 2014]. In this thesis, we use sequence data from the known genes of the species *Saccharomyces cerevisiae*. We take the information in FASTA format. FASTA represents as single characters each nucleotide or amino acid of a biological sequence, and includes other data such as the sequence name and comments [Pearson, 1990]. Two examples of a nucelotide sequence in FASTA format can be found in Appendix A.

2.5 Using Kernel Methods and Pairwise SVM to Predict Metabolic Networks

Based on Section 2.3.1, the input data, considered as the training example dataset $((x_i, y_i), d_i)$, is a set of known pairs of enzymes (or genes) classified in two categories (interacting or non-interacting pairs). Fig. 2.6 (a) shows an example of the input data. In this example, enzymes are represented by either EC number (e.g., 5.3.1.9) or gene nomenclature (e.g., YAR071W).

A pairwise kernel is computed based on Section 2.2. Fig. 2.6 (b) represents the kernel $K((x_1, y_1), (x_2, y_2))$. Different types of data associated to the enzymes can be used. Examples of sequence and non-sequence data to compute the pairwise kernel are described in Chapter 4 Section 4.2. In this thesis, we focus on improving the pairwise kernel computation and representation using sequence data (See Chapter 5 and 6 for examples using nucleotide data). The advantage to using sequence data is the gene does not need to be annotated. We can directly use nucleotide or peptide sequences to predict the metabolic networks. Then, the errors from misannotation can be eliminated or diminished. Based on Example 2.4, if we use the $(x_1, y_1), (x_2, y_2)$ variables as genes represented in Fig 2.6(a), e.g., $x_1 = \text{YAR071W}, y_1 = \text{YAL002W}, x_2 = \text{YDR127W}, y_2 = \text{YAL038W}$, then the Tensor Product Pairwise Kernel, using the simple PFAM kernel is

$K((x_1, y_1), (x_2, y_2)) = k_{PFAM}(\text{YAR071W}, \text{YDR127W}) \cdot k_{PFAM}(\text{YAL002W}, \text{YAL038W})$
$+ k_{PFAM}(\text{YAR071W}, \text{YAL038W}) \cdot k_{PFAM}(\text{YAL002W}, \text{YDR127W}),$

A Pairwise SVM based on the dual formalism of the optimization problem is represented in Fig. 2.6 (c). The parameters α_{ij} and b are learned, using the pairwise kernel, K, and the training dataset, (x_i, y_i). Finally, new pairs of enzymes (x, y), can be classified as interacting or not-interacting, depending the evaluation of the f function (see an example representation in Fig. 2.6 (d)). By predicting the gene interactions of the other unseen examples, the metabolic pathways may be predicted.

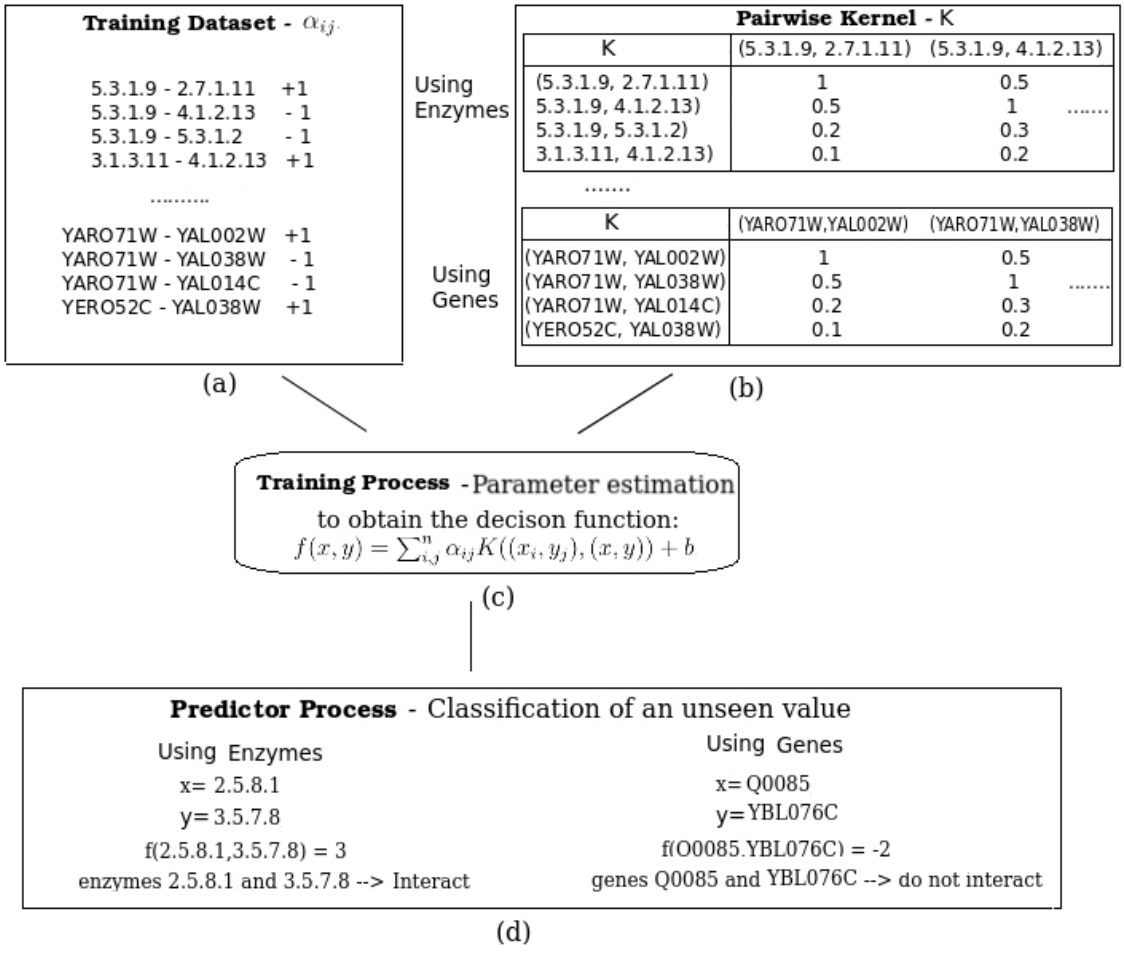

Figure 2.6: Diagram of pairwise SVM applied to metabolic network prediction. (a) An example of the pairs in the training set using the EC numbers (top) or gene names (bottom). (b) The pairwise kernel as a matrix, where the numerical values in each cell correspond to a measure of similarities, given two pairs of EC numbers (top) or two pairs of gene names (bottom). (c) A model is trained to estimate the parameters $\alpha_{i,j}$ and b of the decision function f. (d) Given a new pair of EC numbers (left) or gene names (right) the decision function is evaluated and the pair is classified as interacting or non-interacting.

Chapter 3

Related Works

To increase the understanding of cells, biological network analysis tools have been developed. Supervised learning and kernel methods have been used in some of these tools to reduce complexity. In this chapter, we discuss the most relevant papers related to these topics.

3.1 Metabolic Networks

Bioinformatics applied to metabolic network predictions have become an increasingly active area of research in the past few years [Khatri et al., 2012]. Khatri et al. [2012] have grouped the applications to predict metabolic pathways by the type of analysis these applications perform: Over-Representation Analysis (ORA), Functional Class Scoring (FCS) and Pathway Topology (PT). ORA methods use statistical analysis,

such as hypergeometric, chi-square, and binomial distribution to analyze the group of genes showing large changes in expressions versus the fraction of genes in a particular pathway [Draghici et al., 2003; Khatri et al., 2002]. FCS methods use the same statistical methods as ORA methods for the gene-level analysis, but also consider that small but well coordinated changes have significant effects in the pathways, then incorporate the gene-set (or pathway) statistical analysis [Backes et al., 2007; Dinu et al., 2007]. PT methods further improve ORA and FCS by also take into consideration the number and type of interactions, topology, reactions and additional information available in databases such as KEGG [Okuda et al., 2008], MetaCyc [Karp et al., 2011; Latendresse et al., 2012] and Reactome [Matthews et al., 2009]. Some of these applications and links are shown in Table. 3.1. The most effective of these Functional Pathways Analysis tools are PT methods because they integrate previous knowledge and information into the statistical analysis.

One of the most popular and precise software applications to predict metabolic networks is Pathway Tools, which use information from BioCyc and MetaCyc databases [Karp et al., 2011; Latendresse et al., 2012]. Pathway Tools implements the following two part algorithm:

1. the Inference phase infers the reactions catalyzed by the organism from the set of enzymes present in the annotated genome,

2. the Pathway Inference phase infers the metabolic pathways present in the organism from the reactions found in the phase 1 and using information of catalyzed reactions of similar organisms presented in the database.

Table 3.1: Pathway Analysis tools.

Types	**Tools**	**Links**
Over-representation Analysis (ORA)	ONTO-Express [Draghici et al., 2003]	`http://vortex.cs.wayne.edu`
	GenMAPP [Doniger et al., 2003]	`http://www.genmapp.org`
	GenMerge [Castillo-Davis and Hartl, 2003]	`http://genmerge.cbcb.umd.edu`)
	WebGestalt [Zhang et al., 2005]	`http://bioinfo.vanderbilt.edu/webgestalt`
Functional Class Scoring (FCS)	sigPathway [Tian et al., 2005]	Standalone(BioConductor)
	SAM-GS [Dinu et al., 2007]	`http://www.ualberta.ca/~yyasui/software`
	T-profiler [Boorsma et al., 2005]	`http://www.t-profiler.org`
	GeneTrail [Backes et al., 2007]	`http://www.genetrail.bioinf.uni-sb.de`
Pathway Topology (PT)	ScorePAGE [Rahnenfuhrer et al., 2004]	No implementation available
	Pathway-Express [Draghici et al., 2007]	`http://vortex.cs.wayne.edu`
	SPIA [Tarca et al., 2009]	Standalone (BioConductor)
	NetGSA [Shojaie and Michailidis, 2009]	No implementation available

Considering BioCyc and MetaCyc have a large amount of available data, this application can potentially make precise metabolic pathway predictions [Caspi et al., 2012]. However, the Inference Phase is based on the annotated gene, and if there are errors on the annotation, the inferred pathways will not be correct. Therefore, these methods intrinsically carry error accumulations due to incorrect genome annotations.

Despite the efforts developing pathway analysis approaches, there are outstanding challenges. Khatri et al. [2012] state the next high-throughput technologies. This knowledge will allow better understanding of the biological systems and increase the utility and relevance of pathway analysis. In this thesis, we address the metabolic pathway analysis using new computational approaches based on kernel methods that integrate different types of data. Our proposal belongs to PT methods and focuses on sequence data, to avoid annotation problems, combined with supervised learning

methods, to obtain precise predictions.

3.2 Rational Kernels in Bioinformatics

Finite-state transducers have been used in different research areas, such as natural language processing [Allauzen et al., 2004; Lothaire, 2005; Mohri, 1997], image and optical pattern recognition [Albert and Kari, 2009; Mohri et al., 2005b; Oncina and Sebban, 2006] and bioinformatics [Bradley and Holmes, 2007; Westesson et al., 2011]. In bioinformatics, FSTs provide a bridge between graph representation of biological data and finite-state machine solutions [Bradley and Holmes, 2007]. FSTs have been used as a framework for studying phylogenetic and multiple alignments [Westesson et al., 2011] as well as protein classification and essentiality problems [Allauzen et al., 2008; Roche-Lima, 2007].

Rational kernels have been used in bioinformatics problems to manipulate sequence kernels (see Section 2.2.1 for definitions of sequence and rational kernels). Many of the sequence kernels used in computational biology can be represented as special instances of rational kernels, e.g., locality-improved kernels [Zien et al., 2000], remote homology detection [Kuang et al., 2005] and mismatch kernels [Leslie et al., 2004]. Cortes et al. [2004] showed that rational kernels are a good representation and an algorithmic framework to use with sequence kernels.

Allauzen et al. [2008] proposed a solution to predict protein essentiality by manipulating sequence data using rational kernels. The authors designed two sequence

kernels (called general domain-based kernels), which are instances of rational kernels. To handle the large amount of data (6190 domains each with around 3000 protein sequences), automata were used to represent the rational kernels. Their results showed that the final rational kernels favourably predicted protein essentiality. Based on the fact that the rational kernels described by Allauzen et al. [2008] can be extended to other problems in bioinformatics, we define new kernels to applied to metabolic network predictions. In this case, we combine sequence data in a pairwise way, which intrinsically need to deal with large amount of sequences, to create the pairwise rational kernel framework, described in Chapters 5 and 6.

3.3 Kernel and Pairwise Kernels to Predict Biological Networks

Pairwise kernels have also applied to several problems in bioinformatics. Protein-protein interaction (PPI) networks have been predicted using pairwise kernel methods [Ben-Hur and Noble, 2005]. Ben-Hur and Noble [2005] described the tensor product pairwise kernel (also called the Kronecker kernel), which measures similarities between two pairs of proteins. They used sequence, non-sequence and combined kernels with SVM to predict PPI networks. They obtained the best accuracy (Area Under the Curve AUC=0.78) with the sequence kernels and combining them with other data sources, similar to other authors such as Yu et al. [2010] (see more information about AUC in Section 4.3.1). Yu et al. [2010] use sequence and other kernels also to predict

protein-protein interaction networks. They stated that the AUC values obtained by random selection of data for training machine learning tools results in biased towards proteins with large numbers of interactions. The author proposed to use the balanced sampling techniques to combat bias in the training set. They obtained results with balanced sampling, which range from 0.5-0.75 across several different kernels.

Vert et al. [2007] proposed a new pairwise kernel, called the metric learning pairwise kernel, based on the metric distance for graph inference. They implemented this pairwise kernels combined with SVMs to predict two biological networks, using non-sequence data. As a result, the accuracy was improved (AUC values around 0.8) upon previous state-of-the-art implementations with another pairwise kernel. They have also arrived to the conclusion that kernel combinations improve upon individual kernels.

Similarly, Kashima et al. [2010] described a new pairwise kernel (the Cartesian Kernel) and tested it to predict biological networks. The Cartesian kernel is computed as a Kronecker sum of two matrices. It allows faster processing time and less matrix storage needs. They applied this kernel to predict Protein-Protein Interaction and metabolic networks. In this case, the accuracy was around the same values compared to other methods, however the performance was sped up by 16 times.

Other applications based on kernel methods have already been developed for metabolic network prediction [Kashima et al., 2010; Kotera et al., 2012, 2013; Vert et al., 2007; Yamanishi, 2010]. These applications are focused on determining proteins coding for missing enzymes and predicting the whole or unknown parts of the metabolic

pathways using knowledge from pathways databases (e.g., KEGG database). Thus, we can group these application into the Pathway Topology methods, as described in Section 3.1).

Yamanishi [2010] and Kotera et al. [2012] described the theory and implementation of the web application "GEne Network Inference Engine based on Supervised analysis" (GENIES). GENIES allowed prediction of the unknown parts of metabolic networks using supervised graph inference and kernel methods. Several learning algorithms were implemented in GENIES to learn the parameters of the decision functions based on graph kernel inference methods [Kato et al., 2005; Yamanishi, 2010; Yamanishi et al., 2004, 2005]. Yamanishi [2010] developed experiments to evaluate and compare these methods, using data from annotated genomes. As a result, they obtained accuracy values which range from 0.5 to 0.8, but they did not mention about the execution times. Besides, the experiments were deployed using only non-sequence data. The authors stated that pairwise SVM methods have high processing time and memory consumption. Then, they did not implement SVM-based methods. This was one of the motivation to extend our experiments described in Chapter 4 (see the results in Table 4.1, Experiments II and IV).

Chapter 4

Inferring Metabolic Networks by Sequence Kernels[1]

To infer metabolic networks, machine learning approaches have been developed in the framework of kernel methods [Kashima et al., 2010; Kotera et al., 2012, 2013; Yamanishi, 2010]. Yamanishi [2010] describe several supervised graph inference algorithms to predict biological networks. However, the authors used neither sequence data, nor SVM-based methods. They stated that pairwise SVMs suffer from scalability problems, because the time complexity of the quadratic programming problem is $\mathcal{O}(n^6)$ and the space complexity for storing the kernel matrix is $\mathcal{O}(n^4)$ (where n is the number of known genes of the metabolic network). The authors also mentioned the advantages of using metric learning algorithms for supervised graph inference,

[1]This Chapter is based on the accepted work "Supervised Learning Methods to infer Metabolic Network using Sequence and Non-sequence Kernels", at the International Workshop of Machine Learning in System Biology (MLSB'2013), as part of the ISMB/ECCB'2013 International Conference, Berlin, Germany, July 2013. .

because these methods are more efficient than SVM-based methods.

In this chapter, we develop several experiments using metric learning and SVM-based methods to predict metabolic networks. We have also used available non-sequence and sequence data with these methods. Indeed, we aim to develop and compare metabolic network prediction methods and data using:

- metric learning algorithms and non-sequence data (similar to Yamanishi [2010]),
- metric learning algorithms with sequence data,
- pairwise SVM algorithms with non-sequence data, and
- pairwise SVM algorithms with sequence data.

4.1 Methods

4.1.1 Metric learning Algorithms

As can be seen in Fig. 2.5, metabolic networks are conveniently converted to a graph. Then, a supervised graph inference method with metric learning can be used for the metabolic network prediction problem. A formalism of the problem can be defined as follow:

- given an undirected graph $\Gamma = (V, E)$ (i.e., metabolic network),
  with a set of vertices $V = \{v_1, v_2, ..., v_n\}$ (i.e., enzymes),
  a set of edges $E \subseteq (V \times V)$ (i.e., enzyme-enzyme interactions), and

an additional set of vertices $V' = (v_{n+1}, ..., v_N)$ (i.e., new enzymes),

- infer the set of new edges $E' \subset V' \times (V + V') \cup (V + V') \times V'$ that involve the additional vertices in V' (i.e., predict interactions of the new enzymes with other enzymes).

Several algorithms have been proposed to implement the supervised graph inference with metric learning methods to solve the problem above [Yamanishi, 2010; Yamanishi et al., 2005]. Some of these algorithms were Kernel Canonical Correlation Analysis, Distance Metric Learning, Kernel Matrix Regression, Penalized Kernel Matrix Regression, and expectation-maximization. Yamanishi [2010] described in details their implementations and applications to metabolic network prediction. Penalized Kernel Matrix Regression (PKMR) [Yamanishi and Vert, 2007] was the method with the best accuracy results in their experiments.

In this research, we implement the PKMR method, using the *R* GENIES library [Kotera et al., 2012]. Our goal was to develop similar experiments but using sequence data. PKMR methods received as an input the known part of the the metabolic network previously converted to a kernel [Yamanishi, 2010]. Data were also converted to other types of kernels (see Section 4.2 for a detailed explanation). Similar to SVM, a decision function was learned using the training set. Then, the unseen examples can be classified to predict interactions between pair of enzymes.

4.1.2 Pairwise SVM Algorithms

We followed the description of Pairwise kernels and Pairwise SVM in Sections 2.3.1 and 2.2.3, respectively. Fig. 2.6 gives a general idea how to predict metabolic networks using pairwise kernels and pairwise SVMs. In the experiments of this Chapter, we computed the Pairwise Tensor Product Kernel described in Formula 2.3. We also used the sequential minimal optimization (SMO) technique [Platt, 1998] from LIBSVM package [Chang and Lin, 2011] to implement the pairwise SVM.

4.2 Data

We used available data from the metabolic network of the species *Saccharomyces cerevisiae* in order to validate and compare those methods (see Section 2.4.1 for more details). Different representations of the metabolic network were used, depending of the method. Metric learning technique used the diffusion kernel [Kondor and Lafferty, 2002] to represent the graph. Pairwise SVM method, instead, converted the graph into a training dataset using the enzymes relations as is described in Section 2.4. In this experiments, we considered all 2575 interacting pairs (classified as $+1$), and 50% of the total of non-interacting pairs (classified as -1), for a total of 143605 pairs.

4.2.1 Non-sequence Data

In our context, non-sequence kernels manipulate data that is binary or numerical. We used three different type of non-sequence data: gene expression, gene localization and phylogenetic. All these data have been used in other research as kernels [Kashima et al., 2010; Vert et al., 2007; Yamanishi, 2010].

Gene expression data were obtained by Yamanishi [2010]. They used the results from 157 microarray experiments [Eisen et al., 1998; Spellman et al., 1998]. Each gene was associated with a 157-element numerical vector that represented the results from the experiments. Gaussian RBF Kernel was used to manipulate this data and we defined the same parameters that Yamanishi [2010] used in their experiments. We denoted the final final kernel as k_{exp}.

The gene localization data were represented as a 23-element binary vector for each gene, following Yamanishi [2010]. A total of 23 intracellular localizations were defined (e.g., mitochondrion, Golgi, nucleus and others). The value was 1, if the gene was present in the intracellular localization, or otherwise 0. Similar to Yamanishi [2010], we used the linear kernel applied to these data with the same parameters. We denoted this kernel as k_{loc}.

The phylogenetic profile data were obtained from 145 organisms which describe the set of orthologous genes. These organisms were selected based on the criteria defined in Yamanishi [2010]. Each gene was associated with a 145-element binary vector. The values was 1 if the gene was presented in this organism or otherwise 0. A Gaussian RBF kernel was used to compute this data with the same parameters used

by Yamanishi [2010]. This final kernel was denoted as k_{phy}.

4.2.2 Sequence Data

Sequence kernels define similarities over finite sequences of symbols with different lengths . Then, the sequence data are converted to sequence kernels. In our research, we used three sequence kernels, Pfam, Motif and Spectrum, defined by Ben-Hur and Brutlag [2003]. We chose these sequence kernels based on their results in other papers such as Allauzen et al. [2008]; Ben-Hur and Brutlag [2003]; Yu et al. [2010].

We used the Pfam kernel [Gomez et al., 2003] obtained by Ben-Hur and Brutlag [2003]. The Pfam kernel was computed based on a set of Hidden Markov Models (HMMs) where each gene that codes for an enzyme was compared with every HMM in the Pfam database. The E-value statistics were obtained as features for the 13672 domain HMMs in the Pfam version 26.0 [Punta et al., 2012]. Thus, each protein was represented by a vector of 13672 log E-values and the kernel was computed based on these vectors (see also Allauzen et al. [2008] for more details). We denoted this kernel as k_{pfam}.

The Motif kernel [Ben-Hur and Brutlag, 2003] was also used. It was obtained by calculating how many times a discrete sequence motif matches each of the protein sequences. The eMotif database [Huang and Brutlag, 2001] was used to extract the discrete sequence motifs. A vector of E-values was associated for each of the proteins (genes coding for the proteins). The kernel was finally computed as a dot products of those vectors (Ben-Hur and Noble [2005] describe all the details). The kernel was

called k_{motif}.

Finally, the Spectrum kernel defined by Leslie et al. [2004] was considered in this research. This is a model whose features count how many times a n-gram (k-mer) appeared in a sequence. Each gene had an associated vector of n-gram counts (we have consider the parameter $n = 3$). Similar to the data above, the kernel was calculated as an explicit dot product using the associated feature vectors. We denoted this kernel as k_{ngram}.

4.3 Experiment Description

Our main goal is to compare supervised learning methods (i.e., Metric Learning and pairwise Support Vector Machine) using different types of data (i.e., non-sequence and sequence). We developed four groups of experiments and organized them by methods and type of data (See Table 4.1). In the first two experiments, the PKMR method was used with non-sequence data (Group I) and sequence data (Group II). In the second set of experiments, the pairwise SVM was also used with non-sequence (Group III) and sequence (Group IV) data. They all were executed on a PC intel i7CORE with 8MB RAM. The PKMR parameters were used similar to Yamanishi [2010]. SVM implementation is the sequential minimal optimization (SMO), using as an input the Pairwise Tensor Product Kernel obtained from the kernels mentioned above along with the parameter C set as 1 ($C = 1$) following other papers [Kashima et al., 2010] (see Section 2.3 for more details).

4.3.1 Model Validation

The performance of predictive models, such as PKMR and SVM, can be estimated using the cross-validation method. In k-fold cross-validation, the original dataset is partitioned into k equal-sized subsets, randomly. Then, the model is trained k times. Each time, one of the k subsets is reserved for testing and all the remaining $k-1$ subsets are used for training. The final value is obtained as the average of the k results (see Kohavi et al. [1995] for more details).

A Receiver Operating Characteristic (ROC) curve is a plot of the True Positive Rate (TPR) versus the False Positive Rate (FPR) for different possible cut-offs of a binary classifier system. A cut-off defines a level for discriminating positive and negative categories. For example, if the predictor produces outputs as real numbers, a cut-off can be used on the outputs to generate binary values. ROC curve analysis is used to assess the overall discriminatory ability of the binary classifiers (PKMR and SVM). In this thesis, the area under the curve (AUC score) has been used as a metric to evaluate the strength of the classification, i.e., AUC scores close to 1 indicate the strongest values.

Table 4.1: AUC scores and processing times (seconds)

Exper	Methods	Data type and Predictor Kernel	AUC score	Runtime (sec)
I	PKMR	***Non-Sequence***		
		k_{exp}	0.660	300
		k_{loc}	0.503	240
		k_{phy}	0.775	240
		$k_{exp} + k_{loc} + k_{phy}$	0.755	350
II	PKMR	***Sequence***		
		k_{pfam}	0.797	450
		k_{motif}	0.782	430
		k_{ngram}	0.725	420
		$k_{pfam} + k_{motif} + k_{ngram}$	0.817	480
III	SVM	***Non-Sequence***		
		k_{exp}	0.791	9620
		k_{loc}	0.696	7800
		k_{phy}	0.802	7980
		$k_{exp} + k_{loc} + k_{phy}$	0.818	9980
IV	SVM	***Sequence***		
		k_{pfam}	0.887	12060
		k_{motif}	0.868	12000
		k_{ngram}	0.840	11760
		$k_{pfam} + k_{motif} + k_{ngram}$	0.898	12120

4.4 Results and Discussion

4.4.1 Comparing Data

As can be seen in Table 4.1, we obtained better accuracy when the sequence kernels were used (experiments II and IV). When we compared the sequence kernels within the same learning method (e.g., in PKRM method), the accuracy was improved from AUC=0.503 (the lowest value in Experiment I) to AUC=0.817 (the highest value with kernel in Experiment II). However their execution times were doubled. The best results were obtained using the sequence data, but they were also the most time

consuming.

4.4.2 Comparing Methods

We compared the accuracy and performance of PKRM and SVM methods. The SVM method outperformed the precision values of PKMR. For example, using the PKMR method, the AUC score for k_{exp} (Experiment I) is 0.660 compared to 0.791 (Experiment III) with the same kernel but with the SVM method. However, the execution times increased considerably for SVMs (see the execution times for Experiments I and II in comparison with Experiments III and IV in Table 4.1).

In Fig. 4.1, we have represented the results of both methods only using sequence kernels. The strongest accuracy were obtained with pairwise SVM method, however the execution times are very high. The best results were the combination of all the sequence kernels ($k_{pfam} + k_{motif} + k_{ngram}$) and pairwise SVM method.

4.5 Partial Conclusions and Further Motivations

We have developed for the first time experiments using sequence data with PKRM and SVM methods to predict metabolic networks. Vert et al. [2007] and Kashima et al. [2010] have used SVM methods to predict metabolic networks, but only using non-sequence data (similar to the data we described in Section 4.2.1). Furthermore, Yamanishi [2010] used PKMR with non-sequence data and mentioned the scalability problems of SVM-based methods to predict metabolic networks.

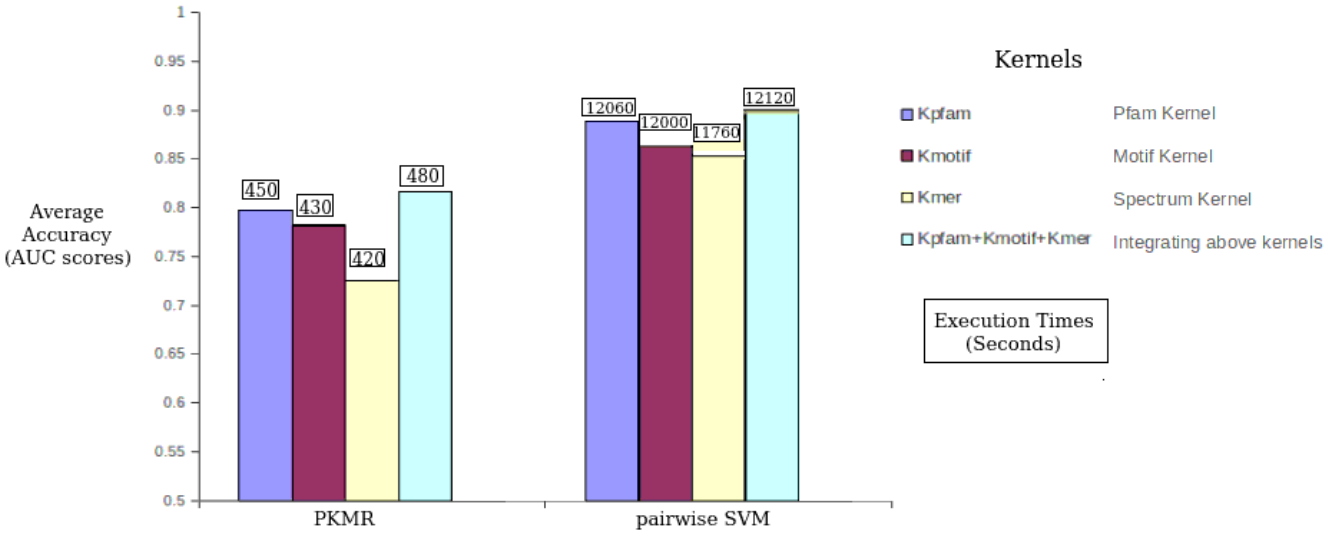

Figure 4.1: Comparison of the methods (SVM and kernel matrix regression) using sequence data kernels, related to accuracy and execution times.

As we have mentioned, other tools to predict metabolic networks and missing enzymes are based on the gene annotation [Karp et al., 2011; Latendresse et al., 2012]. However, current genome annotation pipelines may fail to correctly assign identities to score genes, while failing to detect other genes. One advantage of using the sequence data is that it bypasses the annotation steps, and the errors associated with these steps. Thus, we have proposed to use methods based on sequence data (e.g., using nucleotide or protein sequences). But, the performance is a limitation, as we have observed in these experiments.

Based on our results, the optimal solution should keep the strong accuracy of pairwise SVM method with sequence data, but should decrease the execution times. Brunner et al. [2012] proposed a method to improve pairwise SVM algorithm performance, by enforcing the symmetry of the pairwise kernel, i.e., optimizing the data representation.

We propose a framework that use finite-state transducers to represent and manipulate the sequence data as rational kernels. Then pairwise kernel and pairwise SVM can be benefited from the compact representation and algorithms of FSTs. The next Chapters describe this new framework, called Pairwise Rational Kernels (PRKs).

Chapter 5

Pairwise Rational Kernels[1]

In this chapter, we define four different pairwise rational kernels, based on the state-of-the-art of pairwise kernel combinations and rational kernels. We also describe the general algorithm to obtain the kernels. Finally, we design several experiments by applying PRKs to predict the metabolic network of the species *Saccharomyces cerevisiae*, using the SVM-based algorithm in the training process. As a result, when PRKs are used, the methods execute faster in comparison to other pairwise kernels. Also, when we use the pairwise rational kernel combined with other simple kernels that include evolutionary information, the accuracy values have been improved.

[1]This Chapter is based on the paper "Metabolic Network Prediction Through Pairwise Rational Kernels". Published in BMC Bioinformatics, September 2014. 15:318. doi:10.1186/1471-2105-15-318. http://www.biomedcentral.com/1471-2105/15/318.

5.1 Pairwise Rational Kernels

Based on the notations of rational kernels in Section 2.2.1 and pairwise kernel in Section 2.2.3, we define PRKs as:

Definition 5.1. Given $X \subseteq \Sigma^*$ and a transducer U, then a function $K : (X \times X) \times (X \times X) \to \mathbb{R}$ is:

- ***Direct Sum Pairwise Rational Kernel*** (K_{PRKDS}) if

  $$K((x_1, y_1), (x_2, y_2)) = U(x_1, x_2) + U(y_1, y_2) + U(y_1, x_2) + U(x_1, y_2)$$

- ***Tensor Product Pairwise Rational Kernel*** (K_{PRKT}) if

  $$K((x_1, y_1), (x_2, y_2)) = U(x_1, x_2) \cdot U(y_1, y_2) + U(x_1, y_2) \cdot U(y_1, x_2)$$

- ***Metric Learning Pairwise Rational Kernel*** (K_{PRKM}) if

  $$K((x_1, y_1), (x_2, y_2)) = (U(x_1, x_2) - U(x_1, y_2) - U(y_1, x_2) + U(y_1, y_2))^2$$

- ***Cartesian Pairwise Rational Kernel*** (K_{PRKC}) if

  $$K((x_1, y_1), (x_2, y_2)) = U(x_1, x_2) \cdot \delta(y_1 = y_2) + \delta(x_1 = x_2) \cdot U(y_1, y_2)$$
  $$+U(x_1, y_2) \cdot \delta(y_1 = x_2) + \delta(x_1 = y_2) \cdot U(y_1, x_2)$$

  where $\delta(x = y) = 1$ if $x = y$ and 0 otherwise, $\forall x, y \in X$.

Following Theorem 1 in Section 2.2.1, if we construct U using a weighted transducer T, such as $U = T \circ T^{-1}$, then we guarantee U is a Positive Definite and Symmetric (PDS) kernel. PDS is a needed condition to use kernels in training classification algorithms. Since all the kernels defined above are results of PDS kernel operations, the PRK kernels are also PDS [Horn and Johnson, 2012].

5.1.1 Algorithms

We have designed a general algorithm, Algorithm 2, to compute the kernels, using the composition of weighted transducers. This is a an extension of Algorithm 1. It uses as an input the transducers M_{x_1}, M_{y_1}, M_{x_2}, M_{y_2}, that represent the sequences $x_1, y_1, x_2, y_2 \in X$ and the Weighted Finite-State Transducer U, and outputs the value of $K((x_1, y_1), (x_2, y_2))$.

Algorithm 2 *Pairwise Rational Kernel Computation*

INPUT: pairs of sequences $(x_1, y_1), (x_2, y_2)$ and WFST U

(i) obtain M_{x_1}, M_{y_1}, M_{x_2}, M_{y_2} and use transducer composition to compute:

$$N_1 = M_{x_1} \circ U \circ M_{x_2}$$
$$N_2 = M_{x_1} \circ U \circ M_{y_2}$$
$$N_3 = M_{y_1} \circ U \circ M_{x_2}$$
$$N_4 = M_{y_1} \circ U \circ M_{y_2}$$

(ii) compute the sum of all paths of N_1, N_2, N_3, N_4 using shortest-distance algorithm

(iii) compute the formulas in Definition 5.1:

$$K_{PRKDS}((x_1, y_1), (x_2, y_2)) = N_1 + N_2 + N_3 + N_4$$
$$K_{PRKT}((x_1, y_1), (x_2, y_2)) = N_1 \cdot N_4 + N_2 \cdot N_3$$
$$K_{PRKM}((x_1, y_1), (x_2, y_2)) = (N_1 - N_2 - N_3 + N_4)^2$$
$$K_{PRKC}((x_1, y_1), (x_2, y_2)) = N_1 \cdot \delta(y_1 = y_2) + N_2 \cdot \delta(y_1 = x_2) + N_3 \cdot \delta(x_1 = y_2) + N_4 \delta(x_1 = x_2)$$

RESULTS: value of $K((x_1, y_1), (x_2, y_2))$

In our implementation described below, we use the n-gram rational kernel as the rational kernel U (see Section 2.2.2 for more details). Then, the complexity of the steps (i) and (ii) are $\mathcal{O}(|M_{x_1}|+|M_{y_1}|+|M_{x_2}|+|M_{y_2}|)$. Step (iii) adds a constant time complexity. We conclude that PRKs based on n-gram kernels can also be computed in time $\mathcal{O}(|M_{x_1}|+|M_{y_1}|+|M_{x_2}|+|M_{y_2}|)$.

5.2 Methods

In this section, we describe the data and algorithms to predict metabolic networks using pairwise rational kernels and pairwise SVMs (see Fig. 2.6 for a general example).

5.2.1 Data, Kernel and SVMs

We used data from the yeast *Saccharomyces cerevisiae* as is described in Section 2.4.1. The known part of the metabolic network was converted in a graph and we then obtained the pairs of training set, corresponding to Fig. 2.6 (a). The PRK representation and computation coincide with Fig. 2.6 (b). Here, we describe the computation of PRKs, given the data from the yeast *Saccharomyces cerevisiae*:

- each of the 755 known genes were represented as a trivial weighted automaton (i.e., $A_{x_1}, A_{x_2}, ...A_{x_{755}}$) using the nucleotide sequence,
- the n-gram kernel, with $n = 3$, was used as a rational kernel, then $U(A_{x_1}, A_{x_2}) = \sum_{|z|=3} c_{A_{x_1}}(z) c_{A_{x_2}}(z)$ (see Section 2.2.2 for more details),
- Algorithm 2 was implemented to obtain the K values,
- as an example, the Tensor Product Pairwise Rational Kernel in Definition 5.1 is obtained by:

$$\begin{aligned} K_{PRKTP}((x_1, y_1), (x_2, y_2)) &= U(A_{x_1}, A_{x_2}) * U(A_{y_1}, A_{y_2}) + U(A_{x_1}, A_{y_2}) * U(A_{y_1}, A_{x_2}) \\ &= \sum_{|z|=3} c_{A_{x_1}}(z) c_{A_{x_2}}(z) * \sum_{|z|=3} c_{A_{y_1}}(z) c_{A_{y_2}}(z) \\ &\quad + \sum_{|z|=3} c_{A_{x_1}}(z) c_{A_{y_2}}(z) * \sum_{|z|=3} c_{A_{y_1}}(z) c_{A_{x_2}}(z), \end{aligned}$$

- finally, all the PRK kernels K with positive eigenvalues were normalized to avoid the fact that longer sequences may contain more n-grams, resulting in more similarities [Allauzen et al., 2008].

We implemented this method to compute the PRKs using Open Finite-State Transducer (OpenFST) library [Allauzen et al., 2007] and OpenKernel library [Allauzen and Mohri, 2012]. The input data were nucleotide sequences of known genes, and the outputs were the pairwise rational kernel values as a similarity measure between pairs. Example 5.1 shows the input and output values for the method described above, equivalent to Fig. 2.6 (b), but using sequence data.

Example 5.1. Given nucleotide sequences x_1, y_1, x_2, y_2, which represent abbreviated examples of known genes in the dataset,

x_1 = GCTAAATTGGACAAATCTCAATGAAATTGTCTTGG

y_1 = ATGTCCTCGTCTTCGTCTACCGGGTACAGAAAA

x_2 = CATGACTAAAGAAACGATTCTGGTAGTTATTTGGCGG

y_2 = ATCTACAAGCGAACCAGAGTCTTTCTGCAGGCTTAGAT

and the Rational Kernel U corresponding to the 3-gram rational kernel (see Fig. 2.4 as an example of an n-gram rational kernel for $n = 2$), then, $U(x, y)$ values are computed. For example, if we consider the size-3 subsequence $z = TCT$, the following factors are obtained (see Section 2.2.2 for more details):

- $c_{A_{x_1}}(z) = 2$ because, TCT appears twice in

  GCTAAATTGGACAAA<u>TCT</u>CAATGAAATTG<u>TCT</u>TGG,

- $c_{A_{y_1}}(z) = 2$ because, TCT appears twice in

  ATGTCCTCGTCTTCGTCTACCGGGTACAGAAAA,

- $c_{A_{x_2}}(z) = 1$ because, TCT appears once in

  CATGACTAAAGAAACGATTCTGGTAGTTATTTGGCGG, and

- $c_{A_{y_1}}(z) = 3$ because, TCT appears three times in

  ATCTACAAGCGAACCAGAGTCTTTCTGCAGGCTTAGAT.

and

$U(x_1, x_2) = \sum_{|z|=3} c_{A_{x_1}}(z) * c_{A_{x_2}}(z) = 2 * 1 + ...$

$U(y_1, y_2) = \sum_{|z|=3} c_{A_{y_1}}(z) * c_{A_{y_2}}(z) = 2 * 3 + ...$

$U(x_1, y_2) = \sum_{|z|=3} c_{A_{x_1}}(z) * c_{A_{y_2}}(z) = 2 * 3 + ...$

$U(y_1, x_2) = \sum_{|z|=3} c_{A_{y_1}}(z) * c_{A_{x_2}}(z) = 2 * 1 + ...$

with these values, the K_{PRKT} is computed for $(x_1, y_1), (x_2, y_2)$ as:

$$K_{PRKT}((x_1, y_1), (x_2, y_2)) = U(x_1, x_2) * U(y_1, y_2) + U(x_1, y_2) * U(y_1, x_2),$$

For example, if the result of $K_{PRKT}((x_1, y_1), (x_2, y_2))$ is equal to 0.3, it means that the measure of similarities of these two pairs is 0.3. Appendix B describes the implementation of PRKs in more detail.

As SVM implementation, we used the sequential minimal optimization (SMO) algorithm from the package LIBSVM [Chang and Lin, 2011] in combination with the OpenKernel library. As an input, the four Pairwise Rational Kernels described in Definition 5.1 were used, and the C parameter was set to 1, following other papers

such as Kashima et al. [2010] (see Section 2.3 for more details).

During the training process, the decision function was obtained. The prediction process allows classification of new pairs of nucleotide sequences as interacting or not interacting, by evaluating the decision function. Example 5.2 shows a description of the prediction process, similar to the process described in Fig. 2.6 (d), but using nucleotide sequences.

Example 5.2. This example describe the predictor process. Suppose we want to know if the sequences:

x = CTCAAAGTCTTAATGCTTGGACAAATTGAAATTGG, and

y = TCTACAGAGTCGTCCTTCGTCTACCGGGAAAAT

interact or do not interact. The decision function, $f(x, y)$, was previously obtained during the training process (see Section 2.3 for more details). If the resulting value of evaluating the decision function $f(x, y)$ is greater than 0, the pair (x, y) interact, otherwise the pair (x, y) do not interact. Suppose that the evaluation is:

$$f(x, y) = f(CTCAAAGTCTTAATGCTTGGACAAATTGAAATTGG,$$
$$TCTACAGAGTCGTCCTTCGTCTACCGGGAAAAT) = +3.$$

then, we conclude that these nucleotide sequences, (x, y), interact in the context of the metabolic network of the yeast *Saccharomyces cerevisiae*.

In this case, we used 755 genes during the training process, but the species *Saccharomyces cerevisiae* has more than 6000 genes. Then, the rest of the metabolic network can be predicted by classifying all other pairs of genes (or pairs of raw nucelotide se-

quences), as interacting or non-interacting, using the decision function f. Note that the decision function is obtained once during the training process, but can be used as often as needed during the prediction process.

The advantage of using sequence data is that nucleotide sequences can be used, even if it is not annotated. Also, any other type of sequence data, e.g., from high-throughput analysis, can be considered and combined, using a similar implementation.

5.2.2 Experiment Description

We used pairwise SVM with PRKs for metabolic network prediction, using the data and algorithms described above. A detailed explanation about the experiments can be found in Appendix B. We ran experiments for twelve different kernels. Firstly, we used four PRKs described in Definition 5.1 using the 3-gram rational kernel (i.e., $K_{PRKDS-3gram}, K_{PRKT-3gram}, K_{PRKM-3gram}$ and $K_{PRKC-3gram}$). In addition, a combination of PRKs with other kernels were considered. We included the phylogenetic kernel (K_{phy}) [Yamanishi, 2010] and PFAM kernel (K_{pfam}) [Ben-Hur and Noble, 2005] (these kernel were described in Section 4.2.2). Then, a second set of experiments were developed combining PRKs with the phylogenetic kernel (i.e., $K_{PRKDS-3gram} + K_{phy}, K_{PRKT-3gram} + K_{phy}, K_{PRKM-3gram} + K_{phy}$ and $K_{PRKC-3gram} + K_{phy}$). Finally, we combined PRKs with the PFAM kernel, obtaining $K_{PRKDS-3gram} + K_{pfam}, K_{PRKT-3gram} + K_{pfam}, K_{PRKM-3gram} + K_{pfam}$ and $K_{PRKC-3gram} + K_{pfam}$ kernels. Considering the phylogenetic and PFAM kernels were PDS, the resulting combinations were also PDS [Horn and Johnson, 2012].

All the experiments were executed on a PC intel i7CORE, 8MB RAM. To validate the model, we used the 10-fold cross validation method and measured the Area Under the Curve of Receiver Operating Characteristic (AUC ROC) score described in Section 4.3.1. We ran each experiment ten times and calculated the average of the AUC scores and processing time values. The graphics of the ROC curves were obtained based on one of the ten experiments.

In addition, the 95% Confidence Intervals (CIs) have been computed, following the method described by Cortes and Mohri [2005]. The authors provide a distribution-independent technique to compute confidence intervals for average AUC values. The variance depends on the number of positive a negative examples (2575 and 2574 in our cases) and the number of classification errors, ranging between 889 and 1912 in our cases.

5.3 Results and Discussion

Table 5.1 and Fig. 5.1 show the results. As can be seen, the experiments using only the PRK have the best execution times (Exp. I) because of using finite-state transducers that speed up the processing. However, the accuracy is not comparable to Experiments II and III. Similar results were obtained by Yu et al. [2010] with PPI networks. They stated that simple sequence based kernels, such as n-gram, do not properly predict protein-protein interactions. However, when Yu et al. [2010] combined sequence kernels with other kernels that incorporated evolutionary information, the accuracy of the model predictors were improved. We obtained similar results applied to metabolic

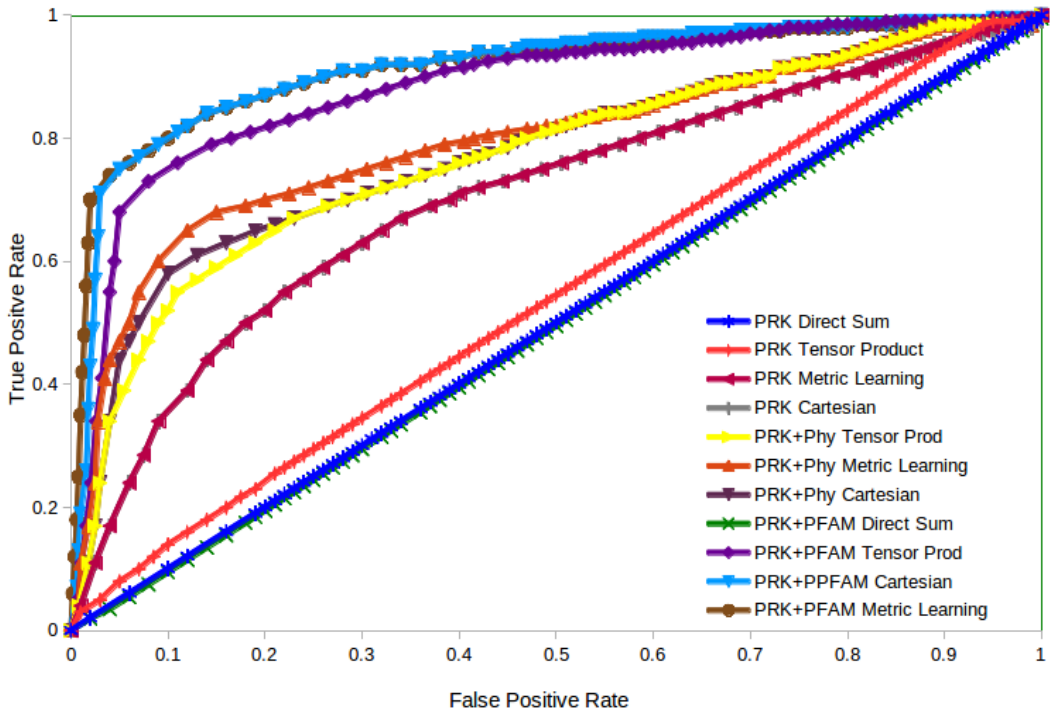

Figure 5.1: ROC Curves for the Pairwise Rational Kernels.

networks predictions. When we incorporated kernels with evolutionary information, i.e., K_{phy} and K_{pfam} in Experiments II and III, respectively, accuracies were improved while maintaining adequate processing times. The best accuracy value was obtained by combining the PRK-Metric-3gram and PFAM kernels (AUC ROC=0.844). Other authors have used similar kernel combinations to improve the prediction of metabolic networks, such as Vert et al. [2007] and Yamanishi [2010]. However, rational kernels have not been used for any of them. Although Yamanishi [2010] did not recommend using SVM-based methods to predict metabolic networks because of the high computational costs, we have obtained good accuracy values within reasonable execution times, using pairwise SVMs and PRKs.

Table 5.1: Average AUC ROC scores, confidence intervals and processing times for various PRKs.

	Type	Kernel	Average AUC score	Runtime (sec)	Confidence Intervals
I	Pairwise Rational Kernels (PRK) (3-gram)	PRK-Direct-Sum ($K_{PRKDS-3gram}$)	0.499	15.0	[0.486, 0.512]
		PRK-Tensor-Product ($K_{PRKT-3gram}$)	0.597	16.2	[0.589, 0.605]
		PRK-Metric-Learning ($K_{PRKM-3gram}$)	0.641	17.4	[0.633, 0.648]
		PRK-Cartesian ($K_{PRKC-3gram}$)	0.640	15.0	[0.632, 0.647]
II	PRKs combined with phylogenetic data (K_{phy} Non-sequence kernel)	PRK-Direct-Sum+Phy ($K_{PRKDS-3gram} + K_{phy}$)	0.425	136.2	[0.411, 0.438]
		PRK-Tensor+Phy ($K_{PRKT-3gram}$+K_{phy})	0.733	135.6	[0.725, 0.741]
		PRK-Metric+Phy ($K_{PRKM-3gram}$+K_{phy})	0.761	139.2	[0.753, 0.768]
		PRK-Cartesian+Phy ($K_{PRKC-3gram}$+K_{phy})	0.742	132.6	[0.734, 0.749]
III	PRKs combined with PFAM data (K_{pfam} sequence kernel)	PRK-D-Sum+PFAM ($K_{PRKDS-3gram}$+K_{pfam})	0.493	136.2	[0.480, 0.506]
		PRK-Tensor+PFAM ($K_{PRKT-3gram}$+K_{pfam})	0.827	136.8	[0.819, 0.834]
		PRK-Metric+PFAM ($K_{PRKM-3gram}$+K_{pfam})	0.844	140.4	[0.837, 0.850]
		PRK-Cartesian+PFAM ($K_{PRKC-3gram}$+K_{pfam})	0.842	132.0	[0.835, 0.849]

In the current experiments, the best result was AUC=0.844, while the execution time is only 140.4 seconds. In Chapter 4, we obtained the best accuracy values equal to 0.898, when the pairwise PFAM kernel was computed, however the execution time was over 12100 seconds. Furthermore, our experiments improved the results reported by Yamanishi [2010], where the best AUC value was 0.831 using the integrated kernels without the chemical compatibility.

Ben-Hur and Noble [2005] report an average AUC value of 0.78 for PFAM kernels,

while Yamanishi [2010] reports an AUC of 0.77 for the PHY kernel for predicting *Saccharomyces cerevisiae* metabolic pathways. In Chapter 4, we developed similar experiments but using SVM methods. As a result, we obtain AUC values of 0.887 for PFAM kernel and 0.802 for PHY kernel, with execution times of 12060 and 7980 seconds, respectively. However, in all cases a random selection of negative and positive training data was used. As noted by Yu et al. [2010], the AUC values obtained by random selection of data for training machine learning tools are due to in a bias towards genes (or proteins) with large numbers of interactions. As such, the high AUC results in these previous works cannot be directly compared to the results in this Chapter. We have employed the balanced sampling techniques suggested by Yu et al. [2010] to combat bias in the training set. Our results, with AUC values in the range 0.5-0.844, are comparable to and exceed in cases the results obtained by Yu et al. [2010] with balanced sampling, which range from 0.5-0.75 across several different kernels for protein interaction problems. We have also obtained these results in execution times of 15-140 seconds. With the exception of the direct sum kernel, all of the confidence intervals are above the behavior of a random classifier.

We developed one more experiment with the PFAM kernel as a simple kernel of the Pairwise Tensor Product (K_{pfam}) using a balanced sampling as suggested by Yu et al. [2010]. Note that it is not a PRK; it is a regular pairwise kernel using PFAM as a simple kernel, similar to the example in the Section 2.5. As a result, the average AUC was 0.61 and the execution time was 122 seconds. When we compare these values with the results in Table 5.1 Exp. I, we can see that the kernels $K_{PRKM-3gram}$ and $K_{PRKC-3gram}$ have better average accuracy (i.e., 0.641 and 0.640, respectively)

with lesser average execution times (17.4 and 15.0 seconds, respectively). In addition, when the Pairwise Rational Kernel 3-gram was combined with the PFAM kernel in the Exp. III, (i.e., Tensor Product Pairwise Rational Kernel - $K_{PRKT-3gram}$+K_{pfam}), the average accuracy value (AUC=0.827) was better than the Pairwise Tensor Product (K_{pfam}), while the execution time just was increased 14.8 seconds (i.e., from 122 seconds, using K_{pfam}, to 134.8 seconds, using $K_{PRKT-3gram}$+K_{pfam}).

The Cartesian Kernel has not been widely used since it was defined by Kashima et al. [2010]. Kashima et al. [2010] used expression (k_{exp}), gene localization (k_{loc}) and phylogenetic (k_{phy}) kernels to predict metabolic networks (see Section 4.2.1 for more details about these kernels). Each of these are non-sequence kernels. In the current experiments we computed, for the first time, the pairwise Cartesian kernel with a rational kernel (sequence kernel) to represent sequence data for metabolic network prediction. Cartesian kernels [Kashima et al., 2010] have been defined as an alternative to improve the computational performance of the Tensor Product Pairwise Kernel [Ben-Hur and Noble, 2005]. In the three experiments shown in Table 5.1, we confirmed this definition, as we have obtained better accuracy and execution times when we used the Cartesian Pairwise Rational Kernel ($K_{PRKC-3gram}$) rather than the Tensor Product Rational Kernel ($K_{PRKT-3gram}$). Comparing our results with Kashima et al. [2010], we obtained better AUC values (i.e., 0.844 vs 0.79), and approximately the same average of the execution times (i.e., 93 seconds). Kashima et al. [2010] used non-sequence data and random selection of positive and negative data for training.

Chapter 6

Pairwise Rational Kernels by Automata Operations [1]

Pairwise rational kernels (PRKs) have already been defined as the combination of pairwise kernels and rational kernels to manipulate sequence data in Chapter 5. In this definition, we have first obtained the sequence kernel, i.e., rational kernel, and later compute their pairwise combinations.

As the sequences can be represented by automata, we propose a new method to obtain PRKs, where, first, we compute the pairwise combinations and, later, we use rational kernels to represent the final automata. Fig. 6.1 shows a comparison between the PRKs obtained in Chapter 5 (Fig. 6.1 (a)) and the PRKs we propose

[1]This Chapter is based on the paper "Pairwise Rational Kernels Based on Automaton Operations", accepted by International Conference of Implementation and Application of Automata (CIAA'2014) and published in Springer's Lecture Notes in Computer Science, Volume 8587, pp. 332-345, 2014.

here (Fig. 6.1 (b)).

In both cases, we have sequence data as input and we convert them to automata. In part (a), we use the automata to obtain the rational kernels, and later we compute the pairwise operations. In part (b), we make the pairwise combinations using the automata operations and later we obtain the rational kernels, using the final automata.

We define three new PRKs using automata operations such as sum and product (described in Section 2.1), equivalent to Direct Sum and tensor Product pairwise kernels (described in Section 2.2.3). We propose a general algorithm to compute the PRKs. Finally, we run experiments to predict metabolic networks of *Saccharomyces cerevisiae* using these methods. As a result, we obtain better performance than PRKs defined in Chapter 5.

6.1 General Definitions and Operation

We begin by defining some notations, algorithms and implementations in order to develop the new PRKs. Firstly, we define V_Σ alphabet as:

Definition 6.1. Given the alphabet Σ, the new alphabet V_Σ is defined as

$V_\Sigma = \{\binom{a}{b} : a, b \in \Sigma\} \cup \{\binom{a}{\epsilon}, \binom{\epsilon}{b} : a, b \in \Sigma\}$, where ϵ is the empty character.

We can use the alphabet V_Σ to combine words over Σ into a word which can be handle as a single unit.

Example 6.1. Given the alphabet $\Sigma = \{A, G, C, T\}$ and the nucleotide sequences

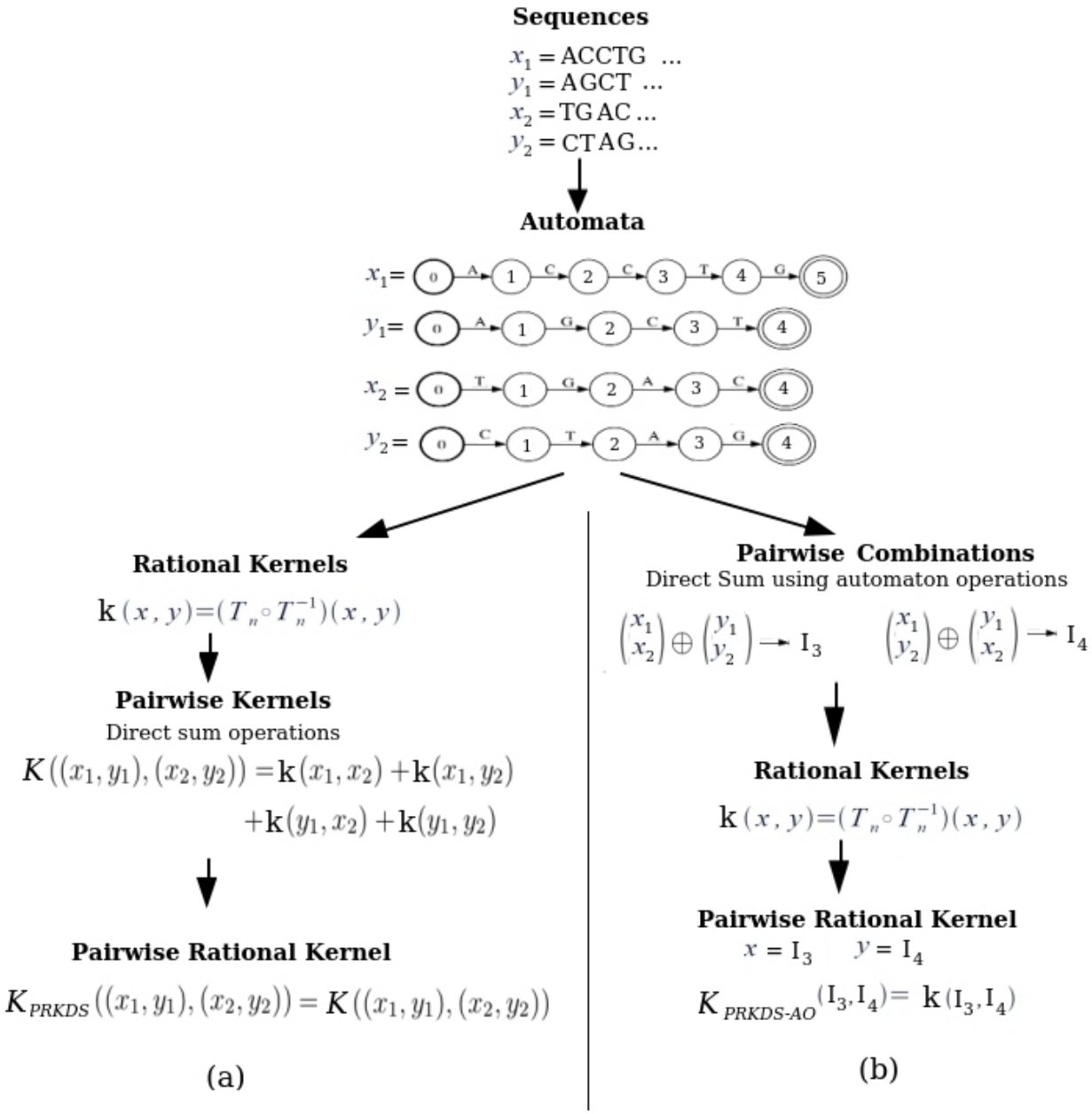

Figure 6.1: Comparison between the families of PRKs we propose in this research. (a) PRKs computed by obtaining the rational kernels first and the pairwise operations later as in Chapter 5. (b) PRKs computed by obtaining the pairwise operation first using automaton operations, and the rational kernels later.

$x = AGGCCCGTA$, $y = CCCGTA$, then $\binom{x}{y} = \binom{A}{C}\binom{G}{C}\binom{G}{C}\binom{C}{G}\binom{C}{T}\binom{C}{A}\binom{G}{\epsilon}\binom{T}{\epsilon}\binom{A}{\epsilon}$.

In this example, a new sequence over the alphabet V_Σ is created. Each symbol from the nucleotide sequence x goes to the top and each symbol from y goes to the bottom. When the nucleotide sequences have different lengths, the ϵ symbol is used. For example, the first symbol from x is the base A and the first symbol from y is the base C, the new symbol from the alphabet V_Σ is $\binom{A}{C}$. The lengths of x and y are different, as x is longer than y. Therefore, the last few symbols in V_Σ have the corresponding symbols from x and the empty symbol (ϵ) in the bottom (i.e., $\binom{G}{\epsilon}\binom{T}{\epsilon}\binom{A}{\epsilon}$).

The following operations over the alphabet V_Σ are defined.

Definition 6.2. Given $\binom{x_1}{y_1}, \binom{x_2}{y_2} \in V_\Sigma^*$, then

- $[\uparrow\uparrow]$ is the Top-Top Operator where $\binom{x_1}{y_1}[\uparrow\uparrow]\binom{x_2}{y_2} = \binom{x_1}{x_2}$
- $[\downarrow\downarrow]$ is the Bottom-Bottom Operator where $\binom{x_1}{y_1}[\downarrow\downarrow]\binom{x_2}{y_2} = \binom{y_1}{y_2}$
- $[\uparrow\downarrow]$ is the Top-Bottom Operator where $\binom{x_1}{y_1}[\uparrow\downarrow]\binom{x_2}{y_2} = \binom{x_1}{y_2}$
- $[\downarrow\uparrow]$ is the Bottom-Top Operator where $\binom{x_1}{y_1}[\downarrow\uparrow]\binom{x_2}{y_2} = \binom{y_1}{x_2}$

for all operations $\diamond \in [\uparrow\uparrow], [\downarrow\downarrow], [\uparrow\downarrow], [\downarrow\uparrow]$ and all languages $L_1, L_2 \subseteq V_\Sigma^*$, we have $L_1 \diamond L_2 = \bigcup_{x \in L_1, y \in L_2} x \diamond y$, based on Kari [1994], where the languages L_1, L_2 are subsets of sequences of V_Σ^*.

Example 6.2. Given the nucleotide sequences $x_1 = ACCTG$, $y_1 = AGCT$, $x_2 = TGAC$, $y_2 = CTAG$ and their respective sequences in V_Σ^*

$\binom{x_1}{y_1} = \binom{A}{A}\binom{C}{G}\binom{C}{C}\binom{T}{T}\binom{G}{\epsilon}$ and $\binom{x_2}{y_2} = \binom{T}{C}\binom{G}{T}\binom{A}{A}\binom{C}{G}$,

the *Top-Top operation* is

$\binom{x_1}{y_1}[\uparrow\uparrow]\binom{x_2}{y_2} = \binom{x_1}{x_2}$, where $\binom{x_1}{x_2} = \binom{A}{T}\binom{C}{G}\binom{C}{A}\binom{T}{C}\binom{G}{\epsilon}$.

The *Top-Top operation* takes the two sequences in the top and creates a new sequence in the alphabet V_Σ^*. In this example, the two top sequences are x_1 and x_2, then the new sequence $\binom{x_1}{x_2} \in V_\Sigma^*$ is created. Equivalently, the results for the other operations are:

the *Bottom-Bottom operation*:

$\binom{x_1}{y_1}[\downarrow\downarrow]\binom{x_2}{y_2} = \binom{y_1}{y_2} = \binom{A}{C}\binom{G}{T}\binom{C}{A}\binom{T}{G}$,

the *Top-Bottom operation*:

$\binom{x_1}{y_1}[\uparrow\downarrow]\binom{x_2}{y_2} = \binom{x_1}{y_2} = \binom{A}{C}\binom{C}{T}\binom{C}{A}\binom{T}{G}\binom{G}{\epsilon}$, and

the *Bottom-Top operation*:

$\binom{x_1}{y_1}[\downarrow\uparrow]\binom{x_2}{y_2} = \binom{y_1}{x_2} = \binom{A}{T}\binom{G}{G}\binom{C}{A}\binom{T}{C}$.

Fig. 6.2 shows automata as a result of some of the operations described above. At the beginning, the nucleotide sequences are represented as automata. Part (a) and part (b) show the automata for the *Top-Top* and *Bottom-Bottom* operations, respectively.

6.1.1 Automata Operations

Now, we define a set of automata operators (based on sum and product operations) to create the new family of Pairwise Rational Kernels.

Definition 6.3. Let $M_{\binom{x}{y}}$ represent the trivial automata of $\binom{x}{y}$, $\forall\ \binom{x}{y} \in V_\Sigma^*$. The following pairwise automata operators are defined:

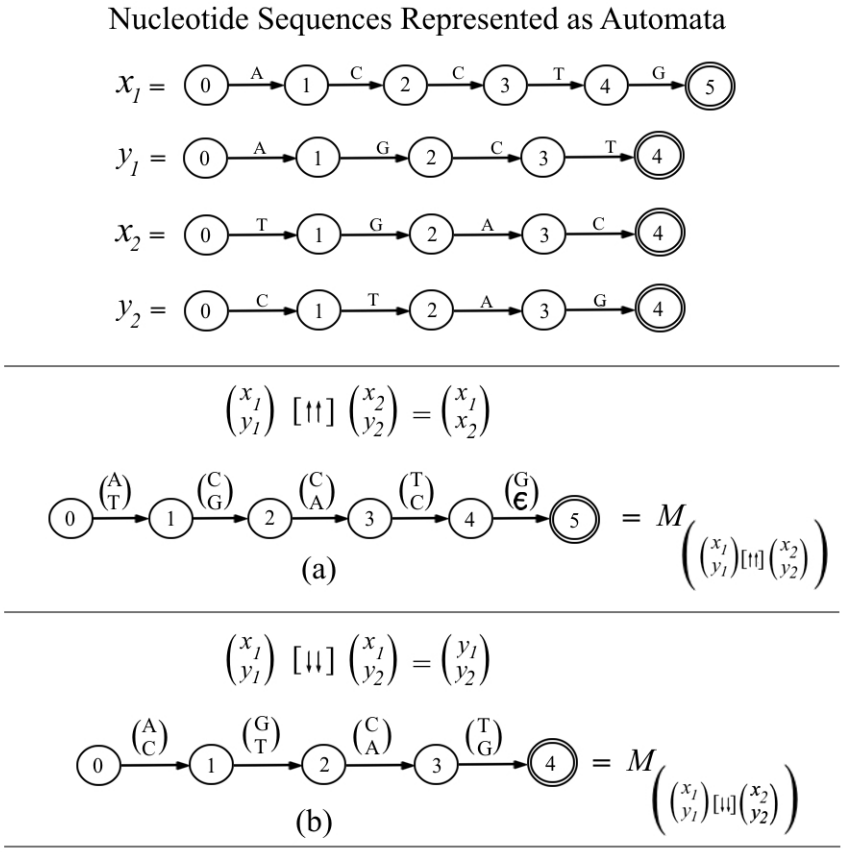

Figure 6.2: Example of automata as a result of Top-Top and Bottom-Bottom operations. First, the given nucleotide sequences from Example 6.2 are represented as automata. (a) Automaton as a result of *Top-Top operation*. (b) Automaton as a result of *Bottom-Bottom operation*.

- Direct-Sum-Left Pairwise Automata Operator ($[\leftleftarrows]$) where

$$M_{\binom{x_1}{y_1}}[\leftleftarrows]M_{\binom{x_2}{y_2}} = M_{\left(\binom{x_1}{y_1}[\uparrow\uparrow]\binom{x_2}{y_2}\right)} \oplus M_{\left(\binom{x_1}{y_1}[\downarrow\downarrow]\binom{x_2}{y_2}\right)}$$

- Direct-Sum-Right Pairwise Automata Operator ($[\rightrightarrows]$) where

$$M_{\binom{x_1}{y_1}}[\rightrightarrows]M_{\binom{x_2}{y_2}} = M_{\left(\binom{x_1}{y_1}[\downarrow\uparrow]\binom{x_2}{y_2}\right)} \oplus M_{\left(\binom{x_1}{y_1}[\downarrow\uparrow]\binom{x_2}{y_2}\right)}$$

- Tensor-Product-Left Pairwise Automata Operator ($[\Leftarrow]$) where

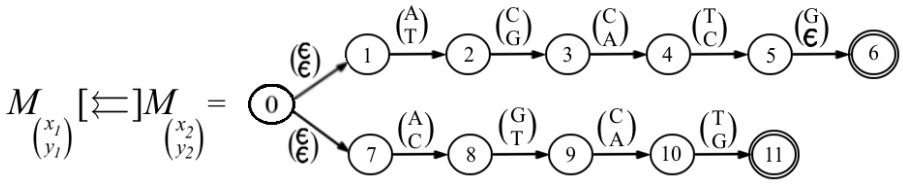

Figure 6.3: Example of an Automaton as result of the Direct-Sum-Left Pairwise Automata Operation (i.e., $M_{\binom{x_1}{y_1}}[\leftleftarrows]M_{\binom{x_2}{y_2}} = M_{\left(\binom{x_1}{y_1}[\uparrow\uparrow]\binom{x_2}{y_2}\right)} \oplus M_{\left(\binom{x_1}{y_1}[\downarrow\downarrow]\binom{x_2}{y_2}\right)}$).

$$M_{\binom{x_1}{y_1}}[\Leftarrow]M_{\binom{x_2}{y_2}} = M_{\left(\binom{x_1}{y_1}[\uparrow\uparrow]\binom{x_2}{y_2}\right)} \otimes M_{\left(\binom{x_1}{y_1}[\downarrow\downarrow]\binom{x_2}{y_2}\right)}$$

- Tensor-Product-Right Pairwise Automata Operator ($[\Rightarrow]$) where

$$M_{\binom{x_1}{y_1}}[\Rightarrow]M_{\binom{x_2}{y_2}} = M_{\left(\binom{x_1}{y_1}[\uparrow\downarrow]\binom{x_2}{y_2}\right)} \otimes M_{\left(\binom{x_1}{y_1}[\downarrow\uparrow]\binom{x_2}{y_2}\right)}$$

In this context, we have used operations over automata, $\oplus$ and $\otimes$, that we defined in Section 2.1 as *sum* (union) and *product* (concatenation). We have used these new symbols just to make a difference over automata as graphical representation, but basically there are the same operations.

Example 6.3. Given the sequences in V_Σ^*

$\binom{x_1}{y_1} = \binom{A}{A}\binom{C}{G}\binom{C}{C}\binom{T}{T}\binom{G}{\epsilon}$ and $\binom{x_2}{y_2} = \binom{T}{C}\binom{G}{T}\binom{A}{A}\binom{C}{G}$,

from Example 6.2, the *Direct-Sum-Left Pairwise Automata Operator* produces the automaton shows in Fig. 6.3, as a result of the *sum* operation over the automata obtained in Fig. 6.2 (a) and (b).

Similarly, Fig. 6.4 shows the automaton as a result of the *product* operation over the same automata in Fig. 6.2 (a) and (b), which represents the *Tensor-Product-Left Pairwise Automata Operator*.

In this section, we define operations over automata that equivalently make pairwise

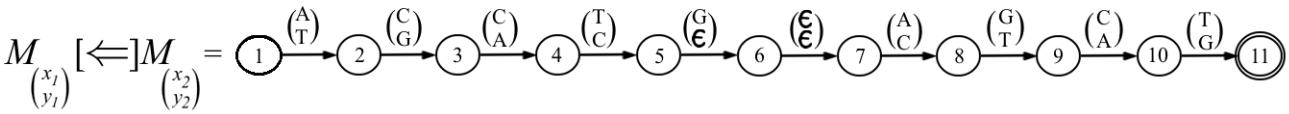

Figure 6.4: Example of an Automaton as result of the Tensor-Product-Left Pairwise Automata Operation (i.e., $M_{\binom{x_1}{y_1}}[\Leftarrow]M_{\binom{x_2}{y_2}} = M_{\left(\binom{x_1}{y_1}[\Uparrow]\binom{x_2}{y_2}\right)} \otimes M_{\left(\binom{x_1}{y_1}[\Downarrow]\binom{x_2}{y_2}\right)}$).

relations over data (in this case, sequence data). For example, the Direct Sum Learning Pairwise Kernel, described in Formula 2.2, is obtained by the sum operation of simple kernel k, evaluated in pairs of data. Similarly, the Tensor Learning Pairwise Kernel, Formula 2.3, is the combination of product and sum operations over the simple kernels, using also pairs of data.

In this research, we are using sequence data. We have represented them in a new alphabet, V_Σ, as pairs of sequences. We converted these pairs of data in automata, over the new alphabet. Then, pairwise operations over automata can be made first, producing final automata as a result. Rational kernels can be obtained from these final automata. Based on this concept, we define a new family of kernels in the next section.

6.2 Pairwise Rational Kernels based on Automata Operations

We introduce then the new Pairwise Rational Kernels using automata operations.

Definition 6.4. Given $X \subseteq \Sigma^*$ and a transducer U, then a function

$K : (X \times X) \times (X \times X) \to \mathbb{R}$ is defined as

- a ***Direct-Way Pairwise Rational Kernel on Automaton Operations*** ($K_{PRKDW-AO}$) if
  $K((x_1, y_1), (x_2, y_2)) = U(I_1, I_2)$, where $I_1 = M_{\binom{x_1}{y_1}}$ and $I_2 = M_{\binom{x_2}{y_2}}$.

- a ***Direct-Sum Pairwise Rational Kernel on Automaton Operations*** ($K_{PRKDS-AO}$) if
  $K((x_1, y_1), (x_2, y_2)) = U(I_3, I_4)$, where $I_3 = M_{\binom{x_1}{y_1}}[\leftleftarrows]M_{\binom{x_2}{y_2}}$ and
  $I_4 = M_{\binom{x_1}{y_1}}[\rightrightarrows]M_{\binom{x_2}{y_2}}$.

- a ***Tensor Pairwise Rational Kernel on Automaton Operations*** ($K_{PRKTP-AO)}$) if
  $K((x_1, y_1), (x_2, y_2)) = U(I_5, I_6)$, where $I_5 = M_{\binom{x_1}{y_1}}[\Leftarrow]M_{\binom{x_2}{y_2}}$ and
  $I_6 = M_{\binom{x_1}{y_1}}[\Rightarrow]M_{\binom{x_2}{y_2}}$.

The first PRK, i.e., $K_{PRKDW-AO}$, is obtained directly from the automata that represent the pairs (x_1, y_1) and (x_2, y_2) in the new alphabet V_Σ^* (i.e., $\binom{x_1}{y_1}$ and $\binom{x_2}{y_2}$). The other two PRKs represent the Direct-Sum ($K_{PRKDS-AO}$) and Tensor Product ($K_{PRKTP-AO)}$) pairwise kernels. These new PRKs differ from previous PRK definitions in Chapter 5, Section 5.1, in the way the data are represented and kernels are computed. Our motivations to develop these new kernels are based on automata operations to optimize the performance of kernel computation.

6.2.1 Algorithms

We use the general Algorithm 3 to compute the Pairwise Rational Kernels defined above. This algorithm is based on the basic idea of the Algorithm 1 in Section 2.2.1. The input of the algorithm is the pairs $(x_1, y_1), (x_2, y_2)$. In the first step, the I_i automata are computed using Definition 6.3, then the transducer composition and shortest-distance algorithm is used to finally obtain the Pairwise Rational Kernel values.

Algorithm 3 *Pairwise Rational Kernel Computation based on Automaton Operation*

INPUT: pairs $(x_1, y_1), (x_2, y_2)$ and WFST U

(i) Compute the operations in Definition 6.4:

$$I_1 = M_{\binom{x_1}{y_1}} \qquad I_2 = M_{\binom{x_2}{y_2}},$$
$$I_3 = M_{\binom{x_1}{y_1}}[\leftleftarrows]M_{\binom{x_2}{y_2}} \qquad I_4 = M_{\binom{x_1}{y_1}}[\rightrightarrows]M_{\binom{x_2}{y_2}},$$
$$I_5 = M_{\binom{x_1}{y_1}}[\Leftarrow]M_{\binom{x_2}{y_2}} \qquad I_6 = M_{\binom{x_1}{y_1}}[\Rightarrow]M_{\binom{x_2}{y_2}}.$$

(ii) use transducer composition to compute:

$$N_1 = I_1 \circ U \circ I_2$$
$$N_2 = I_3 \circ U \circ I_4$$
$$N_3 = I_5 \circ U \circ I_6$$

(iii) use a shortest-distance algorithm algorithm to compute the sum of the weights of all paths of N_1, N_2, N_3, and finally

$$K_{PRKDW-AO}((x_1, y_1), (x_2, y_2)) = N_1$$
$$K_{PRKDS-AO}((x_1, y_1), (x_2, y_2)) = N_2$$
$$K_{PRKT-AO}((x_1, y_1), (x_2, y_2)) = N_3$$

RESULTS: values of $K((x_1, y_1), (x_2, y_2))$

This algorithm is also an extension of Algorithm 1, Section 2.2.1. The complexity of the step (i) is $\mathcal{O}(|M_{x_1}|+|M_{y_1}|+|M_{x_2}|+|M_{y_2}|)$, based on the linearity of the *sum* and *product* automaton operations [Cortes and Mohri, 2009]. Steps (ii) and (iii) depend of the final sizes of the automata I_1, I_2, I_3, I_4. The complexities of these steps are

$\max(\mathcal{O}(|I_1|+|I_2|), \mathcal{O}(|I_3|+|I_4|), \mathcal{O}(|I_5|+|I_6|)$. So, we conclude that PRKs based on n-gram kernels can also be computed in linear time.

6.3 Methods

We developed a set of experiments to apply PRKs using automaton operations and pairwise SVMs to predict metabolic networks (see Fig. 2.6 for a general example).

6.3.1 Data, Kernel and SVMs

To make a comparison with previous results, we used the same data set of metabolic pathways of the yeast *Saccharomyces cerevisiae*, described in see Section 2.4.1 and used in Section 5.2.

Algorithm 3 was adapted and implemented to compute the PRKs that coincides with the pairwise kernel computation, exemplified in Fig. 2.6 (b). The steps to compute the PRKs using automaton operations are:

- the nucleotide sequences of the 755 known genes (i.e., $x_1, x_2, ...x_{755}$) were combined and converted to the new alphabet V_{Σ}^{*}, representing pair of sequences, (i.e., $\binom{x_1}{x_2}, \binom{x_1}{x_3}, ..., \binom{x_{754}}{x_{755}}$)
- operations over the new alphabet in Definition 6.2 were computed,
- the resulting sequences were converted to automata (e.g., automata in Fig. 6.2),

- the pairwise operations in Definition 6.3 were executed to obtain the final automata. Fig. 6.3 and 6.4 are examples that represent the pairwise direct sum (i.e., I_3) and tensor product (i.e., I_5), respectively.

- n-gram kernel, with $n = 3$, was used as a rational kernel, e.g., $K_{PRKDW-AO}(I_3, I_4) = U(I_3, I_4) = \sum_{|z|=3} c_{I_3}(z)c_{I_4}(z)$ (see Section 2.2.2 for more details)

- finally, all the PRK kernels K with positive eigenvalues were normalized to avoid the fact that longer sequences may contain more n-grams, resulting in more similarities [Allauzen et al., 2008].

Similar to the implementation in Chapter 5, we developed software applications to compute the kernels using the Open Finite-State Transducer (OpenFST) library [Allauzen et al., 2007] and the OpenKernel library [Allauzen and Mohri, 2012]. Appendix B shows more details about the algorithms and implementations. PRKs defined above can be combined with SVM methods to predict metabolic pathways, as described in Section 2.5. As we have used in our previous experiments, the pairwise SVM implementation used the sequential minimal optimization (SMO) technique from the package LIBSVM [Chang and Lin, 2011], in combination with the OpenKernel library. As an input, the new PRKs described in Definition 6.4 were used along with the hyperparameter C set to 1, following other papers such as Kashima et al. [2010] (see Section 2.3 for more details).

6.3.2 Experiment Description

The experiments were separated in three different groups:

- Exp I included the new Pairwise Rational Kernels ($K_{PRKDW-AO-3gram}$, $K_{PRKDS-AO-3gram}$ and $K_{PRKTP-AO-3gram}$
- Exp II considered the combination of the new PRKs with the phylogenetic (K_{phy}) kernel [Yamanishi, 2010] ($K_{PRKDW-AO-3gram+phy}$, $K_{PRKDS-AO-3gram+phy}$ and $K_{PRKTP-AO-3gram+phy}$),
- Exp III included the new PRKs with the PFAM kernel [Ben-Hur and Noble, 2005] ($K_{PRKDW-AO-3gram+pfam}$, $K_{PRKDS-AO-3gram+pfam}$ and $K_{PRKTP-AO-3gram+pfam}$).

More information about Phylogenetic and PFAM kernels can be found in Section 4.2.2. A PC intel i7CORE with 8MB RAM was used to execute the experiments. We used 10-fold cross-validation methods and the AUC ROC indicator. We collect the AUC scores and execution times, running each experiment ten times. The graphics of the ROC curves were obtained based on one of the ten experiments, similar to Section 5.2.2.

6.4 Results and Discussion

Table 6.1 and Fig. 6.5 show the results. SVM performance and execution times were grouped by the set of experiments defined above. When only n-gram kernels were

Table 6.1: Average AUC values and processing times for Pairwise Rational Kernels using automaton operations.

	Kernel	AUC	Time (sec)
I	PRK-Direct-Way ($K_{PRKDW-AO-3gram}$)	0.532	9.60
	PRK-Direct-Sum ($K_{PRKDS-AO-3gram}$)	0.526	11.4
	PRK-Tensor-Product ($K_{PRKTP-AO-3gram}$)	0.648	12.0
II	PRK-Direct-Way + phy kernel ($K_{PRKDW-AO-3gram+phy}$)	0.674	129.8
	PRK-Direct-Sum + phy kernel ($K_{PRKDS-AO-3gram+phy}$)	0.533	133.6
	PRK-Tensor-Product + phy kernel ($K_{PRKTP-AO-3gram+phy}$)	0.789	131.8
III	PRK-Direct-Way + PFAM kernel ($K_{PRKDW-AO-3gram+pfam}$)	0.771	129.9
	PRK-Direct-Sum + PFAM kernel ($K_{PRKDS-AO-3gram+pfam}$)	0.538	133.4
	PRK-Tensor-Product + PFAM kernel ($K_{PRKTP-AO-3gram+pfam}$)	0.877	132.0

used (Exp I), the best accuracy value was obtained with the Tensor-Product Pairwise Rational Kernel ($K_{PRKTP-AO-3gram}$), as has also occurred in other experiments, such as Chapter 5. However, the fastest execution time was obtained with the PRK-Direct-Way ($K_{PRKDW-AO-3gram}$), which has been defined and used in this research for the first time. Unfortunately, using only n-gram kernels yields low accuracy values (AUC). These results coincided with Yu et al. [2010], who recommended using other kernels with evolutionary information (i.e., Phylogenetics and PFAM kernels) to improve the predictor accuracy. Thus, we obtained the results in Exp II and Exp III with Phylogenetic and PFAM kernels, respectively. The accuracy values were improved in all cases, while maintaining adequate processing times. The best accuracy value was AUC=0.877, corresponding to *PRK-Tensor-Product + PFAM* ($K_{PRKTP-AO-3gram+pfam}$) kernel (Exp III). Furthermore, this is the highest accuracy

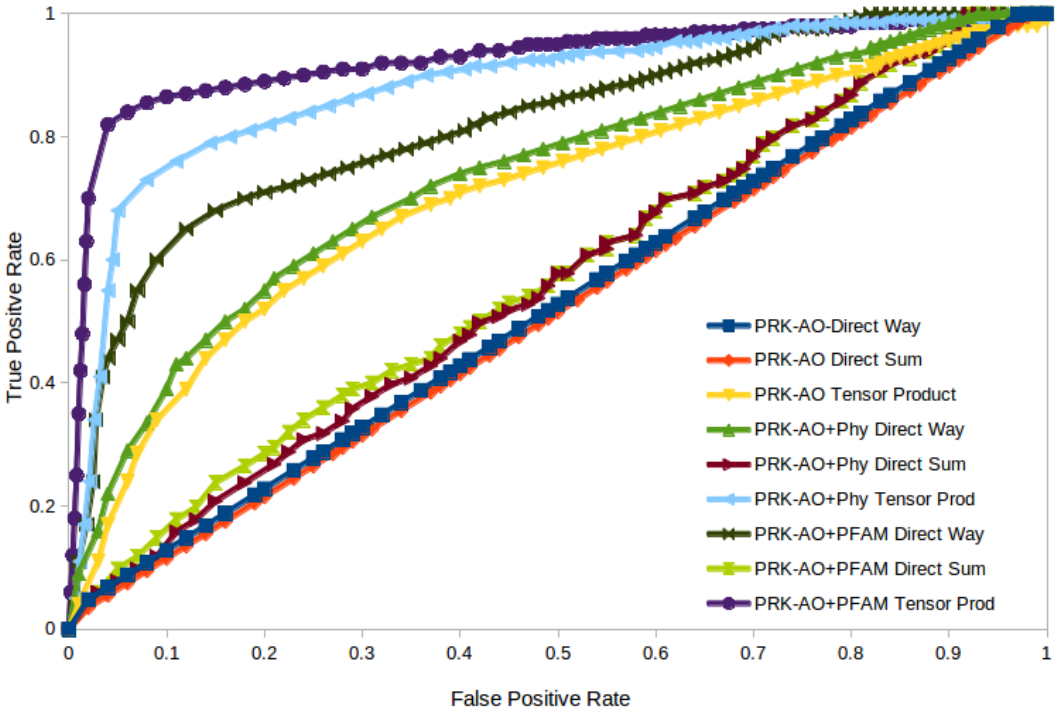

Figure 6.5: ROC Curves for the Pairwise Rational Kernels obtained by automata operations.

obtained with all the PRKs included in this research, which was also computed with an adequate time of 132.0 seconds.

Chapter 7

Conclusion

In this research, we compared for the first time sequence and non-sequence kernels to predict metabolic networks. We obtained in our experiments that sequence kernels have better accuracy than non-sequence kernels. We also showed that pairwise Support Vector Machines and pairwise kernels yield better precision values than Penalized Kernel Matrix Regression method. However, the pairwise Support Vector Machine and kernel methods were computationally expensive in terms of space and execution times.

We introduced for the first the time a new framework called Pairwise Rational Kernels. Pairwise Rational Kernels are the combination of rational kernels (based on finite-state transducers) and pairwise kernels. We defined the framework, developed a general algorithm and tested Pairwise Rational Kernels to predict metabolic networks. We obtained better execution times using PRKs than other kernels. We have also

defined for the first time the pairwise rational Cartesian kernel and used it to predict metabolic networks. This kernel improved the execution time of the Pairwise Rational Tensor Product Kernel, while yielding similar accuracy values.

Furthermore, we proposed a new type of Pairwise Rational Kernels based on automata operations. The new Pairwise Rational Kernels first make pairwise combinations using automata operations and later obtain the rational kernel. We also defined the new notations, operations, kernels and algorithms. Using these new kernels, we obtained the best performance for predicting metabolic network based on Pairwise Rational Kernels.

In these methods, the learning process are executed once to obtain the decision function. The decision can be used as many times as necessary to predict interaction between the other sequences in the species and predict the metabolic network. The methods in this research used sequence data (e.g., nucleotide sequences) to predict these interactions. Genes do not need to be correctly annotated as the raw sequences can be used. In this way, our methods were able to avoid error accumulation due to incorrect gene annotations.

The power of using kernels is that almost any sort of data can be represented using kernels. Therefore, completely disparate types of data can be combined to add power to kernel-based machine learning methods [Fu, 2014]. For example, coefficients describing relative amounts of metabolites involved in a biochemical reaction (i.e., stochiometric data) can also be represented as kernels and added to strengthen the predicting model. For example, the reaction catalyzed by fructose-bisphosphate al-

dolase [EC:4.1.2.13] splits 1 molecule of fructose 1,6-bisphosphate into 2 molecules of glyceraldehyde 3-phosphate, where the relative amounts of substrate and product are represented by the coefficients 1 and 2, respectively. A stoichiometric kernel therefore would encode coefficients for all substrates and products, where enzymes that do not interact would have stoichiometric coefficients of 0. Other authors [Aceves-Lara et al., 2008; Bernard and Bastin, 2005; Mailier et al., 2013] have defined and used similar types of stochiometric data, which can be converted into kernels to be consider with PRKs.

As future work, the model should be tested in other species to predict their metabolic networks. As well, models for one species can be tested to determine if they are able to predict metabolic networks of other species. From the computer science area, new pairwise rational kernels based on automata operations may be developed using other operations such as difference and intersection. Likewise, other pairwise kernels, such as Catersian and metric learning, may be represented using automata operations. Moreover, we have only concentrated on the pairwise kernel representation, however the SVM implementation may be improved using gradient descent algorithm [Allauzen et al., 2011].

PRKs can be used in kernel methods where data are sequences. PRKs improve the performance by taking advantage of the representation and algorithms of weighted finite-state transducers. Some of the areas where PRKs can be applied are pattern recognition, language processing, social networks and information retrieval [Bishop, 2006; Mohri et al., 2005a; O'Madadhain et al., 2005; Taskar et al., 2003].

Appendix A

FASTA data for *S. cerevisiae*

As an example of the FASTA format, we have two sequences from *S. cerevisiae S288C* accession number NC001142.

```
>YAL036C RBG1 SGDID:S000000034, Chr I from 76152-75043, Genome Release
64-1, reverse complement, Verified ORF, "Member of the DRG family of
GTP-binding proteins; associates with translating ribosomes; interacts with
Tma46p, Ygr250cp, Gir2p and Yap1p via two-hybrid"
ATGTCTACTACAGTTGAAAAAATCAAAGCTATCGAAGATGAAATGGCCCGTA
AACAAGGCCACATCTTTCCATTTGGGTCAACTGAAGGCCAAGCTGGCCAAAC
GAATTGTTGACCAGTGCTTCATCCGGCAGCGGTGGTGGTGCTGGTATTGGTT
GCTAGAACTGGTGTGGCCAGTGTGGGGTTTGTCGGGTTCCCGTCGGTGGGGA
TTACTGTCCAAGTTGACTGGTACTGAGTCTGAAGCAGCTGAGTACGAGTTTA
GTTACCGTCCCCGGTGTCATTCGTTATAAAGGTGCCAAGATCCAAATGTTGG
GGTATTATCGATGGTGCTAAGGATGGTAGAGGTAGAGGTAAGCAAGTTATTG
AGAACCTGTAACCTGTTATTTATCATCCTAGATGTGAACAAACCCTTGCATC
ATCATTGAGAAGGAACTGGAAGGTGTGGGGATTCGTCTGAATAAAACTCCGC
GAAGATGTTGTTACCATCTTGAAAAAGTGA

>YAL037W YAL037W SGDID:S000000035, Chr I from 74020-74823, Genome Release
64-1, Uncharacterized ORF, "Putative protein of unknown function"
ATGGATATGGAAATCGAAGATTCAAGCCCCATAGATGACCTGAAGTTACAAA
```

ACCAATGTTTATTTTGGACCCTGTGAGATATTGACACAACCTATTCTTTTGC
AATATTAAGTTCATCATTGGTGTCAATCTAAGTACTGAAAAGATAGCGTCGT
CAGTATTTCAGGAACTCTAATTCGGTAGTCGTGAATCTTTGCTCACCAACTA
GTAGCAACAAAGAAGGCCGCAATTGATTTGTATATACGAAACAATACAATAC
AAATTCGTTGGACAGTACTTGCAGATGGGCAAAAAGATAAAAACATCTTTAA
CAAACCGATACAATCCAATCACTGCCCCAGTTTTGTAATTCGAATGTCCTCA
ACCACAAAGCGAGGTCGCTTTTGA

Appendix B

Details about Algorithms

B.1 Experiments with PRKs

We describe in details the experiments developed in Chapter 5.

B.1.1 Pre-processing the data

The process we followed was:

- obtain the nucleotide sequences for each of the 755 genes from the FASTA file, convert each of them to an automaton, and save in separate files (i.e., 001.fst, ..., 755.fst files),
- obtain a file with the interacting genes from the training dataset with the format:

gene001 gene005 interact

gene012 gene113 interact

gene244 gene389 interact

.....

- use the web program "BRS-nonint" Yu et al. [2010] to select the right number of non-interacting samples, using as an input the set of interacting pairs,
- create the final file with the training dataset in LIBSVM format.

B.1.2 Obtaining the Pairwise Rational Kernel

The process we followed to compute the pairwise kernel was:

- Create a text file with the names of the files of each of the automata (i.e., "001.fst", "002.fst",..., "755.fst"),
- Create the simple 3-gram rational kernel (k) with the transducers in the list producing the kernel file, using the OpenKernel library,
- Create the final four PRKs, using the pairwise combinations defined in Section 5.1, where $U(x, y) = k(x, y)$.

B.1.3 Validating the model

Finally, the SVM was executed and the results obtained as follow:

- Execute the SVM in LIBSVM library, using as an input the training dataset and the PRKs. The SVM implementation was SMO in the Dual Form. The hyperparameter for regularization was set to $C = 1$ to be compatible with other results such as Kashima et al. [2010].
- Using cross-validation with $fold = 10$, the Area Under the Curve (AUC) values were obtained as a measurement accuracy, as well as the execution times in the training process.

B.2 Experiments with PRKs obtained as Automaton Operations

We describe in details the experiments developed in Chapter 6. See Fig. 6.1 for a comparison of the methods to compute the PRKs described in Chapter 5 and Chapter 6.

B.2.1 Pre-processing the data

The process we followed was:

- obtain the nucleotide sequences for each of the 755 genes from the FASTA file,

convert each of them to an automaton, and save in separate files (i.e., 001.fst, ..., 755.fst files),

- obtain a file with the interacting genes from the training dataset with the format:

  gene001 gene005 interact

  gene012 gene113 interact

  gene244 gene389 interact

  .....

- use the web program "BRS-nonint" Yu et al. [2010] to select the right number of non-interacting samples, using as an input the set of interacting pairs,
- create the final file with the training dataset in LIBSVM format.

B.2.2 Obtaining the Pairwise Rational Kernel

The process we followed to compute the pairwise kernel was:

- Create the 3 types of final automata as results of the pairwise automata operations described in Section 6.1.1, and combine to produce the final automata used in the Definition 6.3 (the programs and scripts use the OpenFST library to implement automata operations).
- Using the final automata, create the three different rational kernels (3-gram rational kernels), using the OpenKernel library.

B.2.3 Validating the model

Finally, the SVM was executed and the results obtained as follow:

- Execute the SVM in LIBSVM library, using as an input the training dataset and the PRKs. The SVM implementation was SMO in the Dual Form. The hyperparameter for regularization was set to $C = 1$ to be compatible with other results such as Kashima et al. [2010].
- Using cross-validation with $fold = 10$, the Area Under the Curve (AUC) values were obtained as a measurement accuracy, as well as the execution times in the training process.

B.3 Dataset and Programs

Programs, scripts and data files are available at:

`http://www.cs.umanitoba.ca/~aroche/auxiliary.htm`

Appendix C

Terms and Concepts

C.1 Biological Science terms

- DNA (Deoxyribonucleic acid): molecules that encodes genetic instructions as hereditary material. DNA is in the nucleos of the cell. DNA is made up of the following bases: adenine (A), guanine (G), cytosine (C), and thymine (T).
- Gene: basic unit of heredity that defines the physical and functional regulations. It is made up of DNA and acts as instructions to make proteins.
- RNA (Ribonucleic acid): molecules responsible for multiple vital roles in the coding, decoding, regulation, and expression of genes. RNA is made up of the following bases: adenine (A), guanine (G), cytosine (C), and uracil (U).
- Nucleotide: the chemical component that DNA and RNA are made up (i.e.,

DNA - A, G, C, T, RNA - A, G, C, U). Nucleotides are composed of a nitrogenous base, a five-carbon sugar (ribose or deoxyribose), and at least one phosphate group that defines the type.

- Genome: genetic components of an organism. It includes all the genes in a DNA, as well as other sequences that do not code for a DNA. The human genome, for example, was decode few years ago and it has over 3 billion base pairs of sequences.

- Genomic: related to study the function and structure of genomes, applying DNA sequencing and bioinformatics methods (e.g., to determine the entire DNA sequence of organisms).

- Proteomic: study the functions and structures of the set of proteins of an organism or system. This study is obtained by large-scale experimental analysis of proteins, for example, protein purification and mass spectrometry.

- Enzymes: Biological molecules (mostly proteins) that catalyze metabolic reactions. Enzymes react with other compounds to obtain other compounds or final products.

- High-throughput data: data (i.e., proteomic, genomic and other type of 'omic data) obtained as a result of automated processing of cellular studies, e.g., imaging processing, gene expression microarrays or genome wide screening.

- Transcriptional regulatory networks: collection of sequence segments that interact with each other, governing the expression levels of proteins. The set of

nodes represent proteins or mRNAs (mature RNA). The set of edges represent individual molecular reactions, where the products of one node affect those of another node.

- Signal transduction networks: the connection of transduced signals between cells. Cells interact or communicate with each other to coordinate cell activities and actions. These networks try to simulate these interactions, where the nodes are the cells and the edges their interactions. They have been extensively used to study the development of organisms, tissue repairs, and diseases such as immunity problems, cancer and diabetes.

- Protein-protein interaction networks: are networks which proteins in a cell are represented as nodes and edges represent some physical contacts between these proteins as a result of biochemical events. These biochemical events have been studied from different perspective, such as biochemistry, quantum chemistry, molecular dynamics, and others. It allows to define a protein-protein interaction networks to study, for example, the muscle contraction physiology, where all the proteins and their interactions can be represented.

- Glycolysis pathway: a metabolic pathway that converts glucose into pyruvate. It is a really well studied in many organisms where all the enzymes and compounds involved in the process have been identified.

- Phylogenetics: area of study of evolution factors among group of organisms, determined by the relations through molecular sequencing data and morphological data.

- Sequence alignments: representation of two or more sequences that allow the identification regions of similarities in biological sequences as result of functional, structural, or evolutionary relationships between the sequences.

C.2 Computer Science terms

- Feature space: an abstract space defined by a feature extraction procedure that transforms raw data into sample vectors of some fixed length (see Fig. 2.3 as an example).

- Dot product: operation that takes as input two vectors with the same length and returns a single number, representing their computations (e.g., the dot product of $X = (x_1, x_2, x_3, x_4)$ and $Y = (y_1, y_2, y_3, y_4)$ is $X \cdot Y = x_1 * y_1 + x_2 * y_2 + x_3 * y_3 + x_4 * y_4$).

- Hyperplane: a "flat" $n - 1$-dimensional subspace in an n-dimension space (see Fig. 2.3 as an example).

- Lagrange Multipliers: Method for finding the local maxima and minima of a function subject to equality constraints [Bellman, 1956].

- Hidden Markov Model HMM: a machine learning method that involves Markov process with hidden states. HMMs allow the recovery of a data sequence that is not immediately observable, using other dependant data on the sequence that is observable.

Bibliography

C. A. Aceves-Lara, E. Latrille, N. Bernet, P. Buffiere, and J. P. Steyer. A pseudo-stoichiometric dynamic model of anaerobic hydrogen production from molasses. *Water research*, 42(10):2539–2550, 2008.

J. Albert and J. Kari. *Handbook of weighted automata, EATCS Monographs on Theoretical Computer Science*, chapter Digital image compression. Springer, 2009.

C. Allauzen and M. Mohri. Openkernel library, 2012. URL `http://www.openfst.org/twiki/bin/view/Kernel`.

C. Allauzen, M. Mohri, and M. Riley. Statistical modeling for unit selection in speech synthesis. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. doi: 10.3115/1218955.1218963.

C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri. OpenFST: A general and efficient weighted finite-state transducer library. In *Implementation and Application of Automata*, pages 11–23. Springer, 2007.

C. Allauzen, M. Mohri, and A. Talwalkar. Sequence kernels for predicting protein es-

sentiality. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 9–16, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4.

C. Allauzen, C. Cortes, and M. Mohri. A dual coordinate descent algorithm for SVMs combined with rational kernels. *International Journal of Foundations of Computer Science*, 22(08):1761–1779, 2011.

C. Backes, A. Keller, J. Kuentzer, B. Kneissl, N. Comtesse, Y. A. Elnakady, R. Mller, E. Meese, and H.-P. Lenhof. Genetrail–advanced gene set enrichment analysis. *Nucleic acids research*, 35(Web Server issue):W186–W192, 2007.

J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *Proceedings of the twenty-first international conference on Machine learning*, page 9. ACM, 2004.

R. Bellman. Dynamic programming and lagrange multipliers. *Proceedings of the National Academy of Sciences of the United States of America*, 42(10):767, 1956.

A. Ben-Hur and D. Brutlag. Remote homology detection: a motif based approach. *Bioinformatics*, 19(suppl 1):i26–i33, 2003.

A. Ben-Hur and W. S. Noble. Kernel methods for predicting protein–protein interactions. *Bioinformatics*, 21(suppl 1):i38–i46, 2005.

A. Ben-Hur, C. S. Ong, S. Sonnenburg, B. Schölkopf, and G. Rätsch. Support vector machines and kernels for computational biology. *PLoS computational biology*, 4 (10):e1000173, 2008.

O. Bernard and G. Bastin. On the estimation of the pseudo-stoichiometric matrix for macroscopic mass balance modelling of biotechnological processes. *Mathematical biosciences*, 193(1):51–77, 2005.

M. Beurton-Aimar, T. V.-N. Nguyen, and S. Colombié. Metabolic network reconstruction and their topological analysis. In *Plant Metabolic Flux Analysis*, pages 19–38. Springer, 2014.

C. M. Bishop. *Pattern recognition and machine learning*. Springer, New York, 2006. ISBN 0387310738, 9780387310732.

A. Boorsma, B. C. Foat, D. Vis, F. Klis, and H. J. Bussemaker. T-profiler: scoring the activity of predefined groups of genes using gene expression data. *Nucleic acids research*, 33(Web Server issue):W592–W595, 2005.

B. Bostanci and E. Bostanci. An evaluation of classification algorithms using McNemar's test. In *Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012)*, pages 15–26. Springer, 2013.

R. K. Bradley and I. Holmes. Transducers: an emerging probabilistic framework for modeling indels on trees. *Bioinformatics*, 23(23):3258–3262, 2007.

C. Brouard, J. Dubois, C. Vrain, D. Castel, M. Debily, and F. D'Alche-Buc. Learning a markov logic network for supervised gene regulation inference: application to the id2 regulatory network in human keratinocytes. In *Sixth International Workshop on*

Machine Learning in Systems Biology (MLSB 2012), Basel, Switzerland, September 2012.

C. Brunner, A. Fischer, K. Luig, and T. Thies. Pairwise support vector machines and their application to large scale problems. *Journal of Machine Learning Research*, 13:2279–2292, 2012.

R. Caspi, T. Altman, K. Dreher, C. A. Fulcher, P. Subhraveti, I. M. Keseler, A. Kothari, M. Krummenacker, M. Latendresse, L. A. Mueller, et al. The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of pathway/genome databases. *Nucleic acids research*, 40(D1):D742–D753, 2012.

C. I. Castillo-Davis and D. L. Hartl. Genemerge–post-genomic analysis, data mining, and hypothesis testing. *Bioinformatics (Oxford, England)*, 19(7):891–892, 2003.

C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

C. Cortes and M. Mohri. Learning with weighted transducers. In *Proceedings of the 2009 conference on Finite-State Methods and Natural Language Processing: Post-proceedings of the 7th International Workshop FSMNLP 2008*, pages 14–22, Amsterdam, The Netherlands, The Netherlands, 2009. IOS Press. ISBN 978-1-58603-975-2.

C. Cortes and M. Mohri. Confidence intervals for the area under the roc curve. *Advances in neural information processing systems*, 17:305, 2005.

C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

C. Cortes, P. Haffner, and M. Mohri. Rational kernels: theory and algorithms. *J. Mach. Learn. Res.*, 5:1035–1062, 2004. ISSN 1532-4435.

N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

S. C. J. De Keersmaecker, I. M. V. Thijs, J. Vanderleyden, and K. Marchal. Integration of omics data: how well does it work for bacteria? *Molecular microbiology*, 62 (5):1239–1250, 2006.

A. L. Demain, M. Newcomb, and J. H. D. Wu. Cellulase, clostridia, and ethanol. *Microbiology and Molecular Biology Reviews*, 69(1):124–+, 2005.

I. Dinu, Y. Yasui, J. D. Potter, T. Mueller, Q. Liu, A. J. Adewale, G. S. Jhangri, G. Einecke, K. S. Famulski, and P. Halloran. Improving gene set analysis of microarray data by sam-gs. *BMC bioinformatics*, 8(1):242–242, 2007.

S. W. Doniger, N. Salomonis, K. D. Dahlquist, K. Vranizan, S. C. Lawlor, and B. R. Conklin. Mappfinder: using gene ontology and genmapp to create a global gene-expression profile from microarray data. *Genome Biology*, 4(1):7, 2003.

S. Draghici, P. Khatri, R. P. Martins, G. C. Ostermeier, and S. A. Krawetz. Global functional profiling of gene expression. *Genomics*, 81(2):98–104, 2003.

S. Draghici, P. Khatri, A. L. Tarca, K. Amin, A. Done, C. Voichita, C. Georgescu,

and R. Romero. A systems biology approach for pathway level analysis. *Genome research*, 17(10):1537–1545, 2007.

M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

H. Ellegren. Genome sequencing and population genomics in non-model organisms. *Trends in ecology & evolution*, 29(1):51–63, 2014.

K. Faust and J. Helden. *Bacterial Molecular Networks*, chapter Predicting Metabolic Pathways by Sub-network Extraction. Methods in Molecular Biology. Springer, 2012.

Y. Fu. Kernel methods and applications in bioinformatics. In *Springer Handbook of Bio-Neuroinformatics*, pages 275–285. 2014.

S. M. Gomez, W. S. Noble, and A. Rzhetsky. Learning to predict protein–protein interactions from protein sequences. *Bioinformatics*, 19(15):1875–1881, 2003.

J. Helden, A. Toussaint, and D. Thieffry. *Bacterial Molecular Networks*, chapter Bacterial Molecular Networks: Bridging the Gap Between Functional Genomics and Dynamical Modelling. Methods in Molecular Biology. Springer, 2012.

T. Hertz, A. Bar-Hillel, and D. Weinshall. Boosting margin based distance functions for clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 50. ACM, 2004.

T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

I. Holmes. Using guide trees to construct multiple-sequence evolutionary hmms. *Bioinformatics*, 19:i147–i157, 2003.

R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge University Press, 2012.

J. Y. Huang and D. L. Brutlag. The emotif database. *Nucleic Acids Research*, 29(1): 202–204, 2001.

J. Ingraham. Pathway: glycolysis i (from glucose-6p), May 2014. URL `http://biocyc.org/ECOLI/NEW-IMAGE?type=PATHWAY&object=GLYCOLYSIS`.

K. A. Janes. Data-driven modelling of signal-transduction networks. *Nature Reviews Molecular Cell Biology*, 7(11):820–828, 2006.

M. Kanehisa, M. Araki, S. Goto, M. Hattori, M. Hirakawa, M. Itoh, T. Katayama, S. Kawashima, S. Okuda, T. Tokimatsu, et al. KEGG for linking genomes to life and the environment. *Nucleic acids research*, 36(suppl 1):D480–D484, 2008.

L. Kari. On language equations with invertible operations. *Theoretical Computer Science*, 132(1):129–150, 1994.

P. D. Karp, M. Latendresse, and R. Caspi. The pathway tools pathway prediction algorithm. *Standards in Genomic Sciences*, 5(3):424–429, 2011.

H. Kashima, S. Oyama, Y. Yamanishi, and K. Tsuda. Cartesian kernel: An efficient

alternative to the pairwise kernel. *IEICE TRANSACTIONS on Information and Systems*, 93(10):2672–2679, 2010.

T. Kato, K. Tsuda, and K. Asai. Selective integration of multiple biological data for supervised network inference. *Bioinformatics*, 21(10):2488–2495, 2005.

M. Kellis, N. Patterson, M. Endrizzi, B. Birren, and E. S. Lander. Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature*, 423 (6937):241–254, 2003.

P. Khatri, S. Draghici, G. C. Ostermeier, and S. A. Krawetz. Profiling gene expression using onto-express. *Genomics*, 79(2):266–270, 2002.

P. Khatri, M. Sirota, and A. J. Butte. Ten years of pathway analysis: Current approaches and outstanding challenges. *PLoS Comput Biol*, 8(2):e1002375, 02 2012. doi: 10.1371/journal.pcbi.1002375.

R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, volume 14, pages 1137–1145, 1995.

R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, volume 2, pages 315–322, 2002.

C. Kosiol, I. Holmes, and N. Goldman. An empirical codon model for protein sequence evolution. *Molecular Biology and Evolution*, 24(7):1464–1479, 2007.

M. Kotera, Y. Yamanishi, Y. Moriya, M. Kanehisa, and S. Goto. GENIES: gene

network inference engine based on supervised analysis. *Nucleic acids research*, 40 (W1):W162–W167, 2012.

M. Kotera, Y. Tabei, Y. Yamanishi, T. Tokimatsu, and S. Goto. Supervised reconstruction of metabolic pathways from metabolome-scale compound sets. *Bioinformatics*, 29(13):i135–i144, 2013.

R. Kuang, E. Ie, K. Wang, K. Wang, M. Siddiqi, Y. Freund, and C. Leslie. Profile-based string kernels for remote homology detection and motif extraction. *Journal of bioinformatics and computational biology*, 3(3):527, 2005.

N. C. Kyrpides, T. Woyke, J. A. Eisen, G. Garrity, T. G. Lilburn, B. J. Beck, W. B. Whitman, P. Hugenholtz, and H.-P. Klenk. Genomic encyclopedia of type strains, phase i: the one thousand microbial genomes (kmg-i) project. *Standards in Genomic Sciences*, 9(3), 2014.

M. Latendresse, S. Paley, and P. Karp. Browsing metabolic and regulatory networks with biocyc. In *Bacterial Molecular Networks*. Springer, 2012.

K. H. Lee, D. Lee, K. Lee, and D.-W. Kim. Possibilistic support vector machines. *Pattern Recognition*, 38(8):1325–1327, 2005.

T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, Z. Bar-Joseph, G. K. Gerber, N. M. Hannett, C. T. Harbison, C. M. Thompson, I. Simon, J. Zeitlinger, E. G. Jennings, H. L. Murray, D. B. Gordon, B. Ren, J. J. Wyrick, J.-B. Tagne, T. L. Volkert, E. Fraenkel, D. K. Gifford, and R. A. Young. Transcriptional regulatory networks in saccharomyces cerevisiae. *Science (New York, N.Y.)*, 298(5594):799–804, 2002.

T. Lengauer. Introduction to biological significance of bioinformatics topics, May 2014. URL `http://www.cs.cmu.edu/~blmt/Seminar/SeminarMaterials/IntroMolBasDisease.html`.

C. S. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for svm protein classification. In *Pacific symposium on biocomputing*, volume 7, pages 566–575, 2002.

C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004.

M. Lima. Visual complexity, May 2014. URL `http://www.visualcomplexity.com/vc/index.cfm?trend=Protein%20Interactions`.

G. Longo, May 2014. URL `http://dame.dsf.unina.it/machine_learning.html`. [Online; accessed 09-May-2014].

M. Lothaire. *Applied Combinatorics on Words*. Cambridge University Press, 2005.

J. Mailier, M. Remy, and A. V. Wouwer. Stoichiometric identification with maximum likelihood principal component analysis. *Journal of mathematical biology*, 67(4): 739–765, 2013.

L. Matthews, G. Gopinath, M. Gillespie, M. Caudy, D. Croft, B. de Bono, P. Garapati, J. Hemish, H. Hermjakob, B. Jassal, A. Kanapin, S. Lewis, S. Mahajan, B. May, E. Schmidt, I. Vastrik, G. Wu, E. Birney, L. Stein, and P. D'Eustachio. Reactome knowledgebase of human biological pathways and processes. *Nucleic acids research*, 37(Database issue):D619–D622, 2009.

Q. McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, pages 415–446, 1909.

M. Mohri. Finite-state transducers in language and speech processing. *Comput. Linguist.*, 23(2):269–311, June 1997. ISSN 0891-2017.

M. Mohri. Weighted automata algorithms. In *Handbook of weighted automata*, pages 213–254. Springer, 2009.

M. Mohri, F. Pereira, and M. Riley. *Statistical natural language processing.* Applied Combinatorics on Words, 2005a.

M. Mohri, F. Pereira, and M. Riley. Weighted automata in text and speech processing. *CoRR*, abs/cs/0503077, 2005b.

Y. Moreau. Kernel methods for genomic data fusion. In *Sixth International Workshop on Machine Learning in Systems Biology (MLSB 2012)*, Basel, Switzerland, September 2012.

S. Okuda, T. Yamada, M. Hamajima, M. Itoh, T. Katayama, P. Bork, S. Goto, and M. Kanehisa. KEGG atlas mapping for global analysis of metabolic pathways. *Nucleic acids research*, 36(Web Server issue):W423–W426, 2008.

J. O'Madadhain, J. Hutchins, and P. Smyth. Prediction and ranking algorithms for event-based network data. *ACM SIGKDD Explorations Newsletter*, 7(2):23–30, 2005.

J. Oncina and M. Sebban. Learning stochastic edit distance: Application in handwritten character recognition. *Pattern Recognition*, 39(9):1575–1587, 2006.

A. Osterman and R. Overbeek. Missing genes in metabolic pathways: a comparative genomics approach. *Current opinion in chemical biology*, 7(2):238–251, 2003.

S. Oyama and C. D. Manning. Using feature conjunctions across examples for learning pairwise classifiers. In *Machine Learning: ECML 2004*, pages 322–333. Springer, 2004.

W. R. Pearson. Rapid and sensitive sequence comparison with fastp and fasta. *Methods in enzymology*, 183:63–98, 1990.

J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

K. D. Pruitt. NCBI eukaryotic genomes: Current status, new reports, and annotation summary. In *Plant and Animal Genome XXII Conference*. Plant and Animal Genome, 2014.

M. Punta, P. C. Coggill, R. Y. Eberhardt, J. Mistry, J. Tate, C. Boursnell, N. Pang, K. Forslund, G. Ceric, J. Clements, et al. The Pfam protein families database. *Nucleic acids research*, 40(D1):D290–D301, 2012.

M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM journal of research and development*, 3(2):114–125, 1959.

J. Rahnenfuhrer, F. S. Domingues, J. Maydt, and T. Lengauer. Calculating the statistical significance of changes in pathway activity from gene expression data. *Statistical applications in genetics and molecular biology*, 3:Article16, 2004.

A. Roche-Lima. Bioinformatics applied to genetic study of rumen microorganism. In *2nd Conference of IT in Agriculture Scienc*, Havana, Cuba, November 2007.

A. Shojaie and G. Michailidis. Analysis of gene sets based on the underlying regulatory network. *Journal of computational biology : a journal of computational molecular cell biology*, 16(3):407–426, 2009.

R. S. Sikorski and P. Hieter. A system of shuttle vectors and yeast host strains designed for efficient manipulation of dna in saccharomyces cerevisiae. *Genetics*, 122(1):19–27, 1989.

P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular biology of the cell*, 9(12):3273–97, 1998.

E. Takimoto and M. Warmuth. Path kernels and multiplicative updates. *Journal of Machine Learning Research*, 4(5):773–818, 2004.

A. L. Tarca, S. Draghici, P. Khatri, S. S. Hassan, P. Mittal, J.-s. Kim, C. J. Kim,

J. P. Kusanovic, and R. Romero. A novel signaling pathway impact analysis. *Bioinformatics*, 25(1):75–82, 2009.

B. Taskar, M.-F. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *Advances in neural information processing systems*, page None, 2003.

T. Tatusova, S. Ciufo, B. Fedorov, K. ONeill, and I. Tolstoy. Refseq microbial genomes database: new representation and annotation strategy. *Nucleic acids research*, 42 (D1):D553–D559, 2014.

L. Tian, S. A. Greenberg, S. W. Kong, J. Altschuler, I. S. Kohane, and P. J. Park. Discovering statistically significant pathways in expression profiling studies. *Proceedings of the National Academy of Sciences of the United States of America*, 102 (38):13544–13549, 2005.

K. Tsuda and W. S. Noble. Learning kernels from biological networks by maximizing entropy. *Bioinformatics*, 20(suppl 1):i326–i333, 2004.

V. N. Vapnik and S. Kotz. *Estimation of dependences based on empirical data*, volume 41. Springer-Verlag New York, 1982.

J.-P. Vert, J. Qiu, and W. Noble. A new pairwise kernel for biological network inference with support vector machines. *BMC bioinformatics*, 8(Suppl 10):S8, 2007.

T. Watanabe, D. Kessler, C. Scott, M. Angstadt, and C. Sripada. Disease prediction based on functional connectomes using a scalable and spatially-informed support vector machine. *NeuroImage*, 2014.

O. Westesson, G. Lunter, B. Paten, and I. Holmes. Phylogenetic automata, pruning, and multiple alignment. *arXiv preprint arXiv:1103.4347*, 2011.

Y. Yamanishi. Supervised inference of metabolic networks from the integration of genomic data and chemical information. *Elements of Computational Systems Biology. Wiley*, pages 189–212, 2010.

Y. Yamanishi and J. Vert. Kernel matrix regression. In *12th International Conference on Applied Stochastic Models and Data Analysis.*, 2007.

Y. Yamanishi, J. Vert, and M. Kanehisa. Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics*, 20(Suppl 1):i363–i370, 2004.

Y. Yamanishi, J. Vert, and M. Kanehisa. Supervised enzyme network inference from the integration of genomic data and chemical information. *Bioinformatics*, 21(suppl 1):i468–i477, 2005.

B.-J. Yoon, X. Qian, and S. M. E. Sahraeian. Comparative analysis of biological networks hidden markov model and markov chain-based approach. *Ieee Signal Processing Magazine*, 29(1):22–34, 2012.

J. Yu, M. Guo, C. J. Needham, Y. Huang, L. Cai, and D. R. Westhead. Simple sequence-based kernels do not predict protein–protein interactions. *Bioinformatics*, 26(20):2610–2614, 2010.

B. Zhang, S. Kirov, and J. Snoddy. Webgestalt: an integrated system for exploring gene sets in various biological contexts. *Nucleic acids research*, 33(Web Server issue):W741–W748, 2005.

A. Zien, G. Ratsch, S. Mika, B. Schalkopf, T. Lengauer, and K. R. Macller. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics (Oxford, England)*, 16(9):799–807, 2000.