

MACHINE LEARNING APPROACHES FOR BIOLOGICAL AND
PHYSIOLOGICAL DATA

RAISSA TILLADA RELATOR



A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Division of Electronics and Informatics
Graduate School of Science and Technology
Gunma University

February 2015

Raissa Tillada Relator: *Machine Learning Approaches for Biological and Physiological Data*, © February 2015

*I don't know anything,
but I do know that everything is interesting if you go into it deeply enough.*

— Richard Feynman

Para kay nanay at tatay.

ABSTRACT

Machine learning has proven itself to be a powerful tool in different tasks such as classification, pattern recognition, and data mining and analysis. However, with the increasing amount of available information, different structures and nature of data require different approaches for data utilization to warrant optimal results. With the aim to exploit biological and physiological data used in recently trending fields such as bioinformatics, chemoinformatics, neuroinformatics and brain-computer interfaces, learning algorithms that can outperform conventional methods in these areas are developed in this thesis.

The first part of this work, which concerns the use of biological data, focuses on predicting interactions between drugs and proteins, and finding active sites in enzymes. For the first task, a variant of canonical correlation analysis is introduced and utilized to improve the performance of learning machines in predicting drug-protein interactions, where the approach is derived from the generalized eigenvalue problem. As for the latter, a learning algorithm exploiting Bregman divergences is developed to determine a weighting scheme to be used in the computation of the deviation in finding active sites in enzymes.

The second part, which addresses physiological data, focuses on task classification using EEG signals. The data signals can usually be modeled as a vector sequence or set of vectors. For this type of input in classification tasks, a kernel function is proposed. This thesis also presents a link between the said kernel function and an existing Grassmann kernel.

The performances of all methods proposed in this work are investigated using real-world data. Empirical results show the efficiency and potential advantages of the proposed algorithms over conventional methods.

ACKNOWLEDGMENTS

First and foremost, I would like to give my sincerest gratitude to my supervisor, Professor Tsuyoshi Kato, for all his support and patience as my mentor, as well as my guardian in a foreign land away from home. His efforts and advices have made me a better student, researcher, and individual. This thesis would not have been possible without his supervision.

I am also grateful to Professor Yoichi Seki, Professor Hidetoshi Yokoo, Professor Naohya Ohta, and Professor Yoshikuni Onozato for agreeing to serve on my committee, and setting aside some time off their very busy schedule to review this thesis.

I am thankful to past and present members of Kato Laboratory, especially to Eisuke Ito and Yoshihiro Hirohashi for their inputs and assistance in my research. I also appreciate the researchers I had the opportunity to collaborate with. I would like to give due thanks to all my professors in Gunma University for their guidance and for imparting their knowledge. Likewise, I would like to acknowledge the people from the Student Support Section and department office who have been so kind and accommodating all throughout my stay in the university.

I would always be grateful for friends I have made here in Japan, some of whom have been like family to me. Special mention goes to Meerin and Ou, who have been dear friends even in tough times. To my teachers and classmates in the language classes, thank you very much for making my stay more interesting and enjoyable. I would also like to extend my thanks to friends back home and abroad, for keeping in touch and for the small messages that bring smiles.

I would like to thank my family for their unending love, support and encouragement. And most of all, to W. There are not enough words to describe how truly grateful I am. *Maraming salamat.*

Finally, I would like to acknowledge the generous help of the Japanese Ministry of Education, Culture, Sports, Science and Technology for funding my PhD studies.

CONTENTS

1	INTRODUCTION	1
2	PRELIMINARIES	5
2.1	Notational Conventions	5
2.2	The Binary Classification Problem	5
2.3	Support Vector Machines	6
2.4	Kernel Functions	9
3	PREDICTING PROTEIN-LIGAND INTERACTIONS	11
3.1	Related works	13
3.2	Methodology	14
3.2.1	Overview of the Algorithm	14
3.2.2	Weighted CCA	17
3.2.3	Weighted SVM	18
3.2.4	Weighting schemes	18
3.3	Experiments and Results	19
3.3.1	Data and Experimental Settings	19
3.3.2	Performance Evaluation Criteria	20
3.3.3	Effects of Using CCA and Weighting	21
3.3.4	Weighted SVM vs Classical SVM	24
3.3.5	Using Interaction Profiles	24
3.4	Summary	25
3.5	Supporting Theories and Proofs	26
3.5.1	Generalized Eigendecomposition	26
3.5.2	Linear Weighted CCA	29
3.5.3	Kernel Weighted CCA	32
3.5.4	Weighted SVM	34
4	ENZYME ACTIVE SITE PREDICTION	35
4.1	Active Site Search Method	37
4.2	Learning Algorithm	39
4.2.1	Optimization Algorithm	41
4.3	Experiments and Results	42
4.4	Summary	43
5	CLASSIFYING TASKS FROM EEG SIGNALS	45
5.1	Preliminaries	47
5.2	Grassmann Kernels and Related Methods	48
5.3	Mean Polynomial Kernel	50
5.4	Mean Polynomial Kernel and Projection Kernel Relationship	51
5.5	Experiments and Results	52

5.5.1	EEG Signal Task Classification	52
5.5.2	Efficiency Comparison	55
5.5.3	Discussion	56
5.6	Summary	57
5.7	Proofs and Discussion	57
5.7.1	Derivation of Equation (5.3)	57
5.7.2	Derivation of Equation (5.4)	57
5.7.3	Explicit Representation of Features	58
6	CONCLUSION AND FUTURE DIRECTIONS	59
	BIBLIOGRAPHY	61

LIST OF FIGURES

Figure 2.1	An illustration of the margin.	7
Figure 2.2	An illustration of the kernel trick.	10
Figure 3.1	Protein-ligand matrix and descriptors.	12
Figure 3.2	Illustration of Classical CCA vs Weighted CCA.	16
Figure 3.3	Average performance of all methods for the task of protein-ligand prediction.	22
Figure 3.4	Histograms illustrating the comparisons between the proposed method WW and other methods.	23
Figure 3.5	ROC curves.	25
Figure 4.1	The active site search problem.	37
Figure 4.2	Comparison between a local site and a template.	38
Figure 4.3	Dataset for training.	39
Figure 4.4	Average performance of all methods for the task of enzyme active site prediction.	43
Figure 5.1	Sample position map of EEG sensors.	46
Figure 5.2	EEG signals as a sequence of vectors.	46
Figure 5.3	Flow of methodology for computing the kernel value for the Grassmann kernels and the mean polynomial kernel.	49
Figure 5.4	Average performance of all methods for the EEG signal task classification.	54
Figure 5.5	Cumulative ratio distribution of the eigenvalues as dimension m is varied.	54

LIST OF TABLES

Table 3.1	Abbreviation of methods.	20
Table 5.1	Time complexity comparison of the kernels.	55

INTRODUCTION

We are currently in the era of “big data.” This data deluge came with the progression of modern technology. With the evolution of the Internet, high tech machineries, advanced medical technologies, and personal computers, a plethora of data of various types is continuously being generated. And there has been a demand to harness these to extract some valuable information that can be used to our advantage.

Machine learning has proven itself to be a powerful tool in different tasks such as classification, pattern recognition, and data mining and analysis [16, 29, 52, 72, 74]. These tasks can further be extended to varying applications such as in biometrics and security, computer vision, data intrusion, drug design, medical imaging, and social data analysis. However, with the increasing amount of available information, different structures and nature of data require different approaches for data utilization to warrant optimal results. Hence, there is a need to continuously challenge the conventional methods.

Machine learning algorithms can be divided into 3 types: supervised, unsupervised, and reinforcement learning. Supervised learning makes use of a part of the whole data for training to allow the algorithm to “learn” some mapping that can perform predictions. Here, the presence of the training data set allows us to measure the performance of the algorithm by defining error metrics, such as the difference between the predicted values and the true values.

On the other hand, in the unsupervised approach, the goal is usually to find some noteworthy patterns that can be utilized for data analysis. Thus, this is also sometimes referred to as *knowledge discovery* [52]. Unlike in supervised learning, the type of patterns that are expected to be uncovered are not explicitly known, neither can any error metric be defined. Examples of this type include clustering, hidden Markov models, and computational methods such as principal component analysis and singular value decomposition, among others.

Reinforcement learning may be the less commonly used type compared to the previous two. This form of learning is inspired by behavioral psychology, where reward and punishment are usually given to teach subjects how to act. By doing this, the subject is trained on which tasks he/she is encouraged to perform, and which are not. Thus, this also arises in areas such as control theory and robotics. In this work, however, we will only focus on supervised learning. In particular, we are interested in binary classification tasks where training data are utilized to approximate some relationship between

input and output variables, which can further be used to categorize new inputs into one of two classes.

THESIS ORGANIZATION AND CONTRIBUTIONS

The studies involved in this thesis contribute to the improvement of machine learning methods for binary classification problems. In particular, we focus our works on bioinformatics- and brain-computer interface-related themes. Following a short introduction of the binary classification problem, support vector machines (SVMs) and kernel methods, and some notations introduced in Chapter 2, the contributions of this thesis are divided and summarized accordingly:

- In Chapter 3, we present a variant of canonical correlation analysis and use this to improve the performance of learning machines in predicting interactions between ligands/drugs and proteins. Protein-ligand interaction prediction plays an important role in drug design and discovery. However, wet lab procedures are inherently time consuming and expensive due to the vast number of candidate compounds and target genes. Hence, computational approaches became imperative and have become popular due to their promising results and practicality. Such methods require high accuracy and precision outputs for them to be useful, thus, the problem of devising such an algorithm remain very challenging. In this chapter, we propose an algorithm employing both support vector machines and an extension of canonical correlation analysis (CCA). Following assumptions of recent chemogenomic approaches, we explore the effects of incorporating bias on similarity of compounds. We introduce kernel weighted CCA as a means of uncovering any underlying relationship between similarity of ligands and known ligands of target proteins.
- In the succeeding Chapter 4, we exploit some Bregman distances to develop a learning algorithm that automatically determines the weights for atoms in finding active sites in enzymes. Prediction of active sites in enzyme is very important for the study of proteins and for practical applications such as drug design. Underlying mechanisms for enzyme reaction are based on the local structures of their active sites. Because of this, the mean square deviation has often been used for protein local structure comparison. To improve the ability of such simple measure, various types of template-based methods that compare local sites have been developed to date. In this work, we introduce parameters for the deviation, as well as regularization functions using Bregman divergences to model a new machine

learning algorithm that determines the parameters of the square deviation.

- In Chapter 5, we propose a kernel function for data modeled as vector sequences or sets of vectors, such as physiological data like electroencephalography (EEG) signals. Classification tasks in brain-computer interface research have presented several applications, biometrics and cognitive training, for instance. However, like in any other discipline, determining suitable representation of data has been challenging, and recent approaches have deviated from the familiar form of one vector for each data sample. In this chapter, we consider a kernel between vector sets, motivated by recent studies where data are approximated by linear subspaces, in particular, methods that were formulated on Grassmann manifolds. The kernel takes a more general approach given that it can also support input data that can be modeled as a vector sequence, and not necessarily requiring it to be a linear subspace. We also discuss how the kernel can be associated with the Projection kernel, a known Grassmann kernel.

Finally, Chapter 6 concludes our work with a summary and discussion of future works related to the proposed methods.

PUBLISHED RESULTS

The works contained in this thesis have resulted in 2 journal publications in the Information Processing Society of Japan (IPSJ) Transactions on Bioinformatics [61], and in the Institute of Electronics, Information and Communication Engineers (IEICE) Transactions on Information and Systems [62]. The contents have also been disseminated at relevant conferences and SIG meetings, where a total of 5 articles have been accepted for presentation [63, 64, 65, 66, 67].

PRELIMINARIES

In this chapter we introduce the basic notations and briefly review some mathematical concepts that have been incorporated in this thesis. References [11, 16, 29, 52, 72, 74] are recommended to readers who wish to have more detailed information about support vector machines, kernel methods, machine learning, and other related subjects.

2.1 NOTATIONAL CONVENTIONS

We will use \mathbb{R} and \mathbb{N} to denote the set of real and natural numbers, \mathbb{R}^n and \mathbb{N}^n for the set of n -dimensional real and natural vectors, and $\mathbb{R}^{m \times n}$ for the set of $m \times n$ real matrices. The set of nonnegative real numbers is represented by \mathbb{R}_+ , and the set of positive real numbers by \mathbb{R}_{++} . For any $n \in \mathbb{N}$, we will use \mathbb{N}_n to represent the set of natural numbers less than or equal to n . We will denote vectors by bold-faced lower-case letters and matrices by bold-faced upper-case letters. Entries of vectors and matrices are not bold-faced. The transpose of a matrix \mathbf{M} is given by \mathbf{M}^\top , the inverse is \mathbf{M}^{-1} , while the $n \times n$ identity matrix is \mathbf{I}_n . The n -dimensional vector whose entries are all one is denoted by $\mathbf{1}_n$, while $\mathbf{0}_n$ is the $n \times 1$ vector of all zeros. The operation $\text{diag}(\mathbf{M})$ outputs a vector whose entries are the diagonal entries of matrix \mathbf{M} , while $\text{diag}(\mathbf{v})$ returns the diagonal matrix with the entries of vector \mathbf{v} along the diagonal. We will use \mathbb{S}^n to denote the set of $n \times n$ symmetric matrices, and $\mathbb{O}^{m \times n}$ the set of $m \times n$ orthonormal matrices, i.e. $\mathbb{O}^{m \times n} \equiv \{\mathbf{M} \in \mathbb{R}^{m \times n} \mid \mathbf{M}^\top \mathbf{M} = \mathbf{I}_n\}$. This definition implies that $\mathbb{O}^{m \times n} = \emptyset$ if $m < n$. For simplicity, we shall also assume that the input space \mathcal{X} is a Euclidean space, i.e., $\mathcal{X} \subseteq \mathbb{R}^n$.

2.2 THE BINARY CLASSIFICATION PROBLEM

Before our introduction on support vector machines and kernel methods, we start by formally defining the classification problem in supervised learning.

As previously stated, classification, as supervised learning, aims to learn or approximate a mapping from inputs denoted by $\mathbf{x} \in \mathbb{R}^n$ to outputs represented by y , given some data for training. The input vectors \mathbf{x} are usually called *features* or *attributes*. And the value of the output variable y is either nominal or categorical, depending on the classification task, which tells us the true class of the input.

In the simplest case where y is nominal and the number of classes is equal to two, we have $y \in \{\pm 1\}$. This is referred to as the *2-class* or *binary classification problem*. This may further be extended into N number of classes, or what we call the *multi-classification* problem. In the case where $y \in \mathbb{R}$, we call it *regression*. In this work, we will mainly focus on binary classification problems.

To formally define classification problem, suppose we have a set of ℓ training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell}$ consisting of input-output pairs. The task is to find an approximate *decision function* $f : \mathcal{X} \mapsto \mathcal{Y}$ using the \mathcal{D} .

2.3 SUPPORT VECTOR MACHINES

First introduced in 1992 by V. Vapnik and colleagues from the AT&T Labs, support vector machines (SVMs) are known to have the ability of being universal approximators of any multivariate function much similar to neural networks. Despite their strong theoretical background, SVMs did not receive much attention until publications showing their excellent performance in practical applications such as text categorization, digit recognition and computer vision arise (cf. [16, 72] for some examples).

To address the task of classifying data, Vapnik et al. first considered a class of hyperplanes with equation

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0,$$

in some dot product space \mathcal{X} , where $\mathbf{w} \in \mathcal{X}$ and $b \in \mathbb{R}$, corresponding to the decision function

$$f(\mathbf{x}) = \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b). \quad (2.1)$$

The value $\langle \mathbf{w}, \mathbf{x} \rangle + b$ is also sometimes referred to as the *SVM score*, as it also provides a level or degree of confidence in classifying \mathbf{x} . Following this, they then proposed a learning algorithm for linearly separable problems.

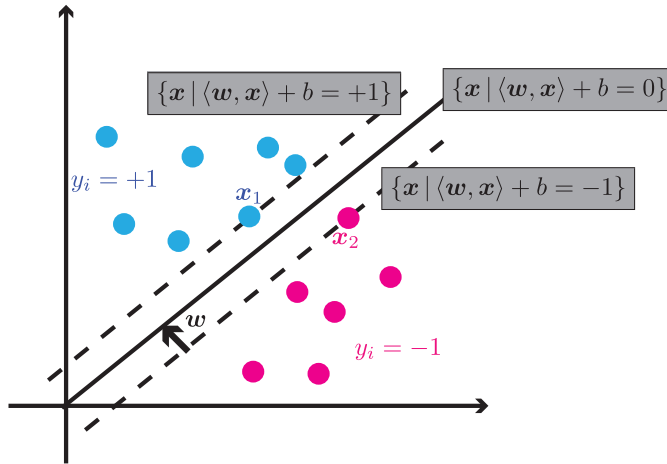


Figure 2.1: **An illustration of the margin.** Here, the data points on the left belong to the class where $y = +1$, while the points on the right belong to $y = -1$. The two classes are separated by the hyperplane with equation $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$, where \mathbf{w} is the vector normal to the hyperplane. Between the dashed lines with equations $\langle \mathbf{w}, \mathbf{x} \rangle + b = +1$ and $\langle \mathbf{w}, \mathbf{x} \rangle + b = -1$, the size of the *margin* is equal to $\frac{2}{\|\mathbf{w}\|}$.

In the context of the simple binary classification problem, SVMs use hyperplanes to create classifiers that maximize the margin, such as that in Figure 2.1. By maximizing the margin, classification error is minimized when new data points are introduced. With the classifier being more robust to perturbations in the data, the problem of underfitting or loose classification, and overfitting or tight classification of the data can be avoided. The binary classification problem is then equivalent to the task of finding a hyperplane that separates the data such that the margin of separation between any training point and the hyperplane is a maximum. This can be formulated as

$$\begin{aligned} \text{maximize} \quad & \min \{ \|\mathbf{x} - \mathbf{x}_i\| : \mathbf{x} \in \mathcal{X}, \langle \mathbf{w}, \mathbf{x} \rangle + b = 0, i = 1, \dots, \ell \} \\ \text{wrt} \quad & \mathbf{w} \in \mathcal{X}, b \in \mathbb{R}. \end{aligned} \quad (2.2)$$

A unique solution for this problem exists and is called the *optimal hyperplane*, also referred to as the *maximal margin classifier*.

In order to construct the optimal hyperplane, the learning problem for SVMs presented in (2.2) can be rewritten as the constrained quadratic optimization problem

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} \|\mathbf{w}\|^2 \\ \text{wrt} \quad & \mathbf{w} \in \mathcal{X}, b \in \mathbb{R}, \\ \text{subject to} \quad & y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1, \forall i = 1, 2, \dots, \ell, \end{aligned} \quad (2.3)$$

where \mathbf{w} is the normal vector to the hyperplane. To solve this quadratic optimization problem, Lagrange multipliers α_i 's are introduced, and the Lagrangian function is given by

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^{\ell} \alpha_i (y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1).$$

Hence, problem (2.3) has the primal form

$$\begin{aligned} \text{minimize} \quad & \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^{\ell} \alpha_i (y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1), \\ \text{wrt} \quad & \mathbf{w} \in \mathcal{X}, b \in \mathbb{R}, \\ \text{subject to} \quad & y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1, \forall i = 1, 2, \dots, \ell, \end{aligned}$$

and dual form given by

$$\begin{aligned} \text{maximize} \quad & W(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ \text{wrt} \quad & \boldsymbol{\alpha} \in \mathbb{R}^{\ell}, \\ \text{subject to} \quad & \alpha_i \geq 0, \forall i = 1, \dots, \ell, \text{ and } \sum_{i=1}^{\ell} \alpha_i y_i = 0. \end{aligned}$$

By minimizing the Lagrangian \mathcal{L} with respect to the primal variables \mathbf{w} and b , and maximizing it with respect to the dual variables α_i 's, we can obtain the saddle point which will lead us to the solution. To do this, the Karush-Kuhn-Tucker (KKT) conditions are imposed. The KKT conditions state that at the saddle point, the partial derivatives with respect to the primal variables must be zero, i.e.,

$$\begin{aligned} \frac{\partial}{\partial b} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) &= - \sum_{i=1}^{\ell} \alpha_i = 0 \quad \text{and} \\ \frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) &= \mathbf{w} - \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i = 0. \end{aligned}$$

And thus we must have

$$\sum_{i=1}^{\ell} \alpha_i = 0 \quad \text{and} \quad \mathbf{w} = \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i. \quad (2.4)$$

The vectors \mathbf{x}_j in the solution vector \mathbf{w} in Equation (2.4) whose coefficients α_j are nonzero are called *support vectors*. In Figure 2.1, these are the data points \mathbf{x}_1 and \mathbf{x}_2 , which are points lying on the margin represented by the dashed lines. Finally, using the solution in Equation (2.4) expressed in terms of support vectors, the decision

function from Equation (2.1) can now be rewritten as the *support vector classifier*

$$f(\mathbf{x}) = \text{sgn} \left(\sum_{i=1}^{\ell} \alpha_i y_i \langle \mathbf{x}, \mathbf{x}_i \rangle + b \right). \quad (2.5)$$

It can be observed from the form that these classifiers only operate with the inner product. Hence, they can still be very efficient even in high dimensional input spaces.

There are also other variants of SVM such as that using soft-margin, one-class SVM, and support vector regression. However, we will no longer include the vast details of these types of SVM.

2.4 KERNEL FUNCTIONS

Kernel methods are a class of pattern recognition algorithms, with their best known element as the support vector machine. Since linear classifiers such as SVM may not work well when data has a more complex pattern, a nonlinear mapping ϕ from the original input space \mathcal{X} to some higher dimensional space \mathcal{F} is often introduced to address this issue. A *kernel* is then defined as a function K such that for any two data points $\mathbf{x}_1, \mathbf{x}_2$ in an input space \mathcal{X} ,

$$K(\mathbf{x}_1, \mathbf{x}_2) = \langle \phi(\mathbf{x}_1), \phi(\mathbf{x}_2) \rangle,$$

where ϕ is a mapping from \mathcal{X} to an inner product space \mathcal{F} called a *feature space*. Hence, kernel methods are a class of algorithms that perform by mapping the input data into a high dimensional feature space. This is done using the so-called *kernel trick* which is primarily based on Mercer's theorem:

Theorem 2.1 (Mercer's Theorem). *Any continuous, symmetric, positive semidefinite kernel function $K(\mathbf{x}, \mathbf{y})$ can be expressed as a dot product in a high dimensional space.*

The theorem implies that any algorithm employing the inner product operation can be applied with the kernel trick. For instance, the binary decision function in (2.5) for the support vector classifier can easily be rewritten as

$$f(\mathbf{x}) = \text{sgn} \left(\sum_{i=1}^{\ell} \alpha_i y_i \langle \phi(\mathbf{x}), \phi(\mathbf{x}_i) \rangle + b \right) = \text{sgn} \left(\sum_{i=1}^{\ell} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b \right).$$

The kernel trick is utilized especially in cases when data is not linearly separable. By applying the kernel trick, data is projected in a higher dimensional space \mathcal{F} before performing classification. This gives rise to linear classifiers or hyperplanes in the feature space \mathcal{F} , and hence creating nonlinear classifiers or hypersurfaces when projected back to the original input space, like in the example in Figure 2.2.

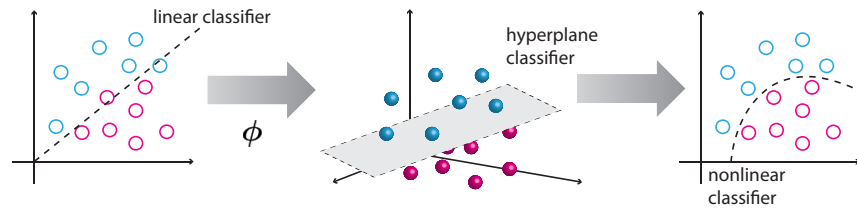


Figure 2.2: **An illustration of the kernel trick.**

Indeed, kernel-based algorithms have come a long way since their introduction due to their many advantages and potential. At present, there is an extensive list of kernels used in varying applications and types of data. Among them, the most recognized and widely-used are the linear kernel ($k(\mathbf{x}_1, \mathbf{x}_2) = \langle \mathbf{x}_1, \mathbf{x}_2 \rangle$), the polynomial kernel ($k(\mathbf{x}_1, \mathbf{x}_2) = \langle \mathbf{x}_1, \mathbf{x}_2 \rangle^d + c$, $c \in \mathbb{R}$), and the Gaussian RBF kernel ($k(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\sigma^2}\right)$, $\sigma > 0$) functions. Aside from the fact that kernel functions have provided algorithms a bridge between linearity and nonlinearity, their performance have been proven comparable to, if not better than, existing algorithms in various areas where they have been exploited. Also, applying the kernel trick is very straightforward and new kernels are easy to construct. And lastly, there is less concern given to the dimension of the feature space due to the simple dot product operation that renders the algorithms computationally inexpensive than usual. For these reasons, recent machine learning trends have involved “kernelizing” some already established algorithms.

Drug discovery is a multi-staged process which involves the determination of existing interactions between a compound and a protein. Many drugs are developed depending on the reaction they produce when coupled with the respective proteins acting during a biological process in the body. However, only a few existing interactions have actually been validated through experiments. Moreover, wet lab procedures are inherently time consuming and expensive due to the vast number of candidate compounds and target genes. Hence, computational approaches became imperative and have become popular due to their promising results and practicality.

The protein-ligand interaction prediction problem can be viewed as a task of filling up a protein-ligand matrix whose rows represent the candidate compounds and the columns represent the target proteins as shown in the example in Figure 3.1a. A matrix entry is +1 if there is interaction between the corresponding drug and target. Otherwise, -1. Only a few interactions have actually been verified and recorded which makes the protein-ligand matrix sparse. Termed as the 'chemogenomic approach' by Rognan [68], the ultimate goal of this task is to identify all the ligands of each target, thus, fully matching the ligand and target spaces [3].

Many *in silico* methods have already been developed to address this problem. We can classify these methods into two: the structure or docking approach and the ligand-based approach. Docking approaches make use of 3D structures of the chemical compounds or the proteins to find protein-ligand pairs which are more likely to bind [2, 4, 7]. On the other hand, ligand-based techniques usually employ machine learning algorithms in comparing known ligands and candidate ligands of a certain target even without any prior information regarding their structure [14, 33, 80]. In this study, we shall make use of the ligand-based approach.

There are two ways of approaching the task of interaction prediction: one is by using the global model [2, 56], and another one is via the local model [3, 33, 80]. The global model utilizes a large interaction matrix and imputation of missing values is done simultaneously. Each cell in the interaction matrix is considered as a sample to which statistical methods are applied. Descriptors of ligands in the form of a feature matrix and some information for target proteins are combined to generate a fused profile for each cell in the interaction matrix. An advantage is that interaction prediction for target proteins with few known interactions can still be formed. However, since the model aims to exploit information from similar

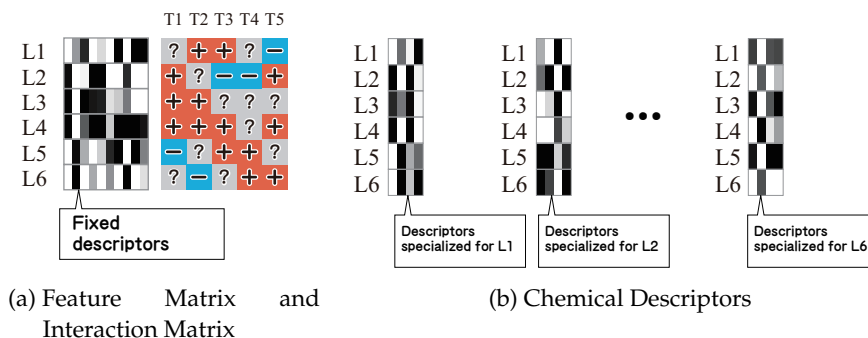


Figure 3.1: **Protein-ligand matrix and descriptors.** In the example depicted in (a), the prediction task is to impute 11 missing entries in the 6×5 protein-ligand matrix using 10-dimensional raw descriptors of ligands. The problem can be divided into six sub-problems, each of which is to complete a row in the protein-ligand matrix. Our algorithm extracts compact descriptors specialized in each sub-problem.

columns, some useful information for learning the rule for prediction may be corrupted by information from irrelevant columns.

Meanwhile, in the local model approach, prediction is made for each column of the protein-ligand table independently — the approach finds unknown chemical compounds which are similar to known ligands interacting with the target protein of interest. The local model often suffers from a small-sample problem. Many columns in the protein-ligand interaction matrix include few positive interactions, causing machine learning algorithms to be trained with few positive samples despite very high dimensionality of ligand descriptors.

The goal of determining interactions between targets and compounds is established under twofold assumptions [3, 68]: First is that compounds with similar properties tend to share targets. And, targets with similar ligands share similarities in structures such as binding sites. These have been verified by recent studies by considering drug side effects [12] and similarities among ligands [51]. Moreover, integrated approaches exploring both protein and compound similarities have also been investigated [10, 33, 85]. Thus, recent methodologies have allowed us to make predictions on interactions based on similarity measures for ligands and targets.

Motivated by the assumption that similar ligands tend to have similar target proteins [42, 73], our goal is to uncover any underlying relationship between a set of ligands and exploit this relationship, together with some known ligand-target interactions, to predict new interactions. We search for ligands with strong associations by finding correlations between them using their features.

In this chapter, we present a weighted extension of canonical correlation analysis (WCCA) in the reproducing kernel Hilbert space

(RKHS) in an attempt to introduce advantageous properties of local models to the global model approach. To estimate the missing entries in each row of the interaction matrix, we use kernel WCCA (KWCCA) to extract essential features which are specialized in imputation of the corresponding row. The extracted features are compact enough for local models to be trained with a small training set composed from the column. Through the experiments with data of G protein-coupled receptors (GPCRs) and odorants, the prediction performance is shown to be improved when our algorithm is applied compared to several existing methods.

3.1 RELATED WORKS

A popular and useful technique in investigating relationships between sets of data is the so-called canonical correlation analysis (CCA) [28]. First introduced by Hotelling [30], CCA generally aims to find linear transformations which maximize the correlation between a pair of data. However, the common information extracted from the data sources may not be as useful if nonlinear correlations exist. For this reason, kernel CCA (KCCA) was introduced to offer an alternative solution via the kernel trick, where CCA is performed in a reproducing kernel Hilbert space (RKHS), typically a higher dimensional feature space [1].

Several variants of CCA have been developed and applied to different problem settings. For instance, Yu et al. [93] introduced weights to CCA. Although we also introduce weighting in our proposed method, the authors' purpose and formulation are totally different from ours: they assumed more than two data sources and weight each source, whereas, in our formulation, we assume two data sources and each sample is weighted. On the other hand, in a biologically-related setting, Yamanishi et al. [90] employed multiple KCCA and integrated KCCA for gene cluster extraction. One is done by maximizing the sum of pairwise correlations and the other by maximizing correlation of combination of attributes.

For the problem of functional site prediction, Gonzalez et al. [24] incorporated KCCA to find amino acid pairs and protein functional classifications which are maximally correlated. This technique was motivated by the *Xdet* method [57] and CCA was employed as an alternative to computing Pearson correlation.

The indefinite kernel CCA (IKCCA) was developed by Samarov et al. [71] with a motivation similar to ours. They removed the similarity of samples outside the neighborhood to refine the analysis. The operation often yields an indefinite matrix. IKCCA finds a definite matrix close to the indefinite matrix to perform CCA on the definite matrix. However, their usage of employing CCA is different from ours: the inputs of their approach are positive pairs of ligands and proteins,

whereas our approach applies CCA to two different types of ligand profiles. IKCCA is formulated with a saddle-point problem that is solved by minimizing a maximum, but the numerical algorithm to solve the problem has not been shown.

Another important variant is sparse CCA [87, 91] which uses *lasso* or *elastic net* techniques to encourage loading matrices to be sparse. This approach was also applied to a set of protein-ligand pairs with positive interactions in order to elucidate meaningful chemical descriptors in [91]. Another is the Supervised Regularized CCA [23] which allows integration of multimodal data. Such method can be very useful when involving non-image and image data samples.

3.2 METHODOLOGY

3.2.1 Overview of the Algorithm

Our approach consists of two stages: First, we consider sub-problems, each of which involves imputation on a single row in the interaction matrix, and use weighted CCA to extract a compact vector representation for each sub-problem. Then, we apply SVM for prediction of each cell using the corresponding descriptor extracted in the previous stage. This technique is overviewed as follows.

Chemical profiles obtained from chemical structures contain numerous features that are not important for prediction. Extracting significant features from such profiles is crucial for accurate prediction of protein-ligand interaction. To accomplish this, we have to find effective low-dimensional representations of the original chemical profiles lying in the extremely high-dimensional *chemical space*.

Interaction profiles describe the existence and the absence of interactions with several target proteins. More often than not, target proteins share similar properties. For this reason, interaction profiles approximately span a low-dimensional space, say \mathbb{R}^m , which we shall also extract from a high-dimensional *interaction space*, in a similar fashion as the chemical profiles.

Canonical correlation analysis uses a set of chemical profiles and interaction profiles to find two projection functions, Φ_{ch} and Φ_{in} , simultaneously: The projection Φ_{ch} is from the chemical space to the low-dimensional canonical space \mathbb{R}^m , and Φ_{in} is from the interaction space to \mathbb{R}^m . The images of Φ_{ch} are used to approximate the images of Φ_{in} . The projections obtained by CCA are shown mathematically to be the minimizer of the expected deviation of the image of Φ_{ch} from the image of Φ_{in} .

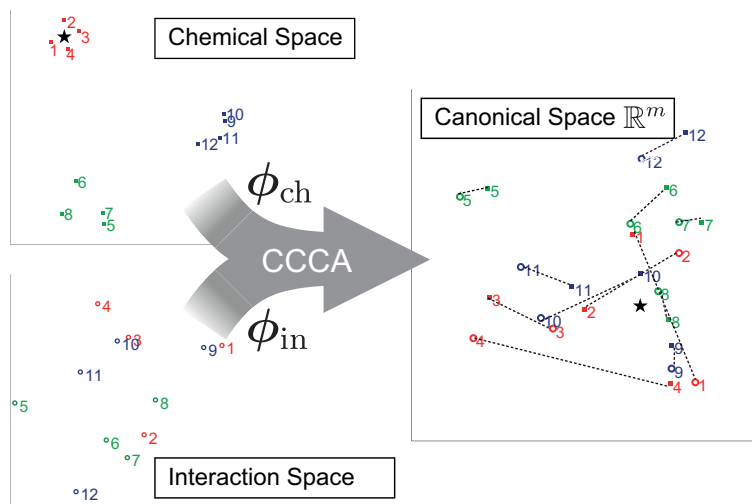
Figure 3.2a is an illustration of how CCA works with chemical profiles and interaction profiles. In this figure, the shaded squares are data representations of the feature vector of each ligand in the chemical space. While the open circles are the data representation

of the interaction vector of each ligand in the interaction space. The images under Φ_{ch} and Φ_{in} of these data points are plotted in the canonical space, and their corresponding images are linked with a dashed line. CCA finds the projections Φ_{ch} and Φ_{in} so that the average squared length of the dashed lines is minimized.

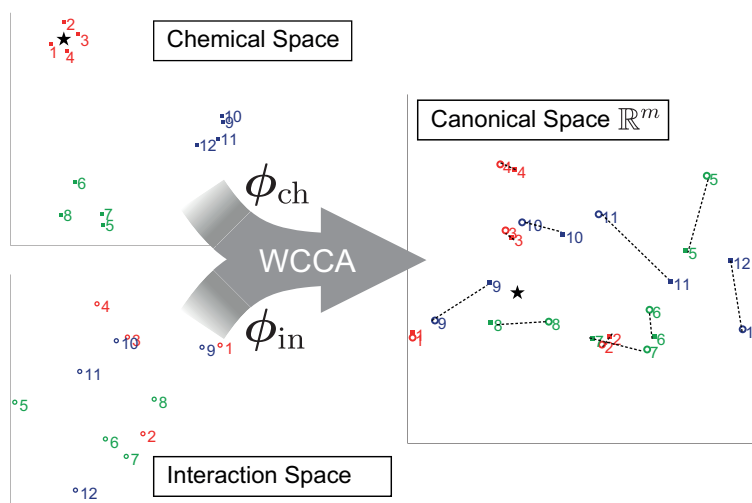
In application to protein-ligand interaction prediction, estimating the images for all ligands is not necessary; it is only for the ligand whose interactions we wish to predict that the image of the chemical compound is desired to be well approximated. To obtain a good approximation for a ligand of interest, it is sufficient to estimate projections so that only the images of similar ligands are approximated well. The precisions of the approximations for ligands dissimilar to the ligand of interest barely affect the accuracy of the solution. This consideration motivated us to assign weights to ligands according to their similarity to the ligand of interest, and to extend the classical CCA so that the weighted average deviation is minimized. The weighted CCA almost disregards ligands with small weights to find projections, achieving more accurate approximations for the ligand of interest. We refer to the extension of CCA as weighted CCA.

Figure 3.2b illustrates the effects of weighted CCA when weights are added to similar ligands. In this context, we define similarity as the measure of affinity between features of compounds. This can be represented by the distance between the data representation of the ligands in the chemical space. In the given figure, the chemical profile for a ligand of interest is marked with a star, and profiles of similar ligands are colored red. In a similar manner, we interpret points of the same color as ligands sharing similarities in their chemical properties, hence their grouping in the chemical space. The two figures, (a) and (b), allow us to compare classical CCA with weighted CCA: the deviations for red points in (b) are smaller than those in (a). The deviations for other ligands are larger, which hardly worsen the performance of predicting the interaction of the protein of interest.

The final prediction result is obtained in the post-processing stage using SVM. The images of the projections are used for SVM learning. SVM is trained well if a good training set is given. Hence, ligands with poor approximations by CCA, which are noisy for SVM learning, are preferably excluded. The images are already in a low-dimensional space in which SVM learning works well even with a small training set, encouraging us to assign smaller weights to ligands with poor approximations for SVM learning.



(a) Classical CCA



(b) Weighted CCA

Figure 3.2: **Illustration of Classical CCA vs Weighted CCA.** Our approach projects chemical and interaction profiles into a low-dimensional canonical space so that the images are close to each other. The star point represents the ligand of interest, and red points are ligands sharing similarities with the ligand of interest. Although the classical CCA minimizes the average deviation over all the ligands, to achieve accurate prediction, it is sufficient that the deviations between the images of the target ligand and the ligands similar to it are small. The weighted CCA works with arbitrarily specified weights, which ensures small deviations for red points by giving them larger weights.

3.2.2 Weighted CCA

In this subsection we present the details of weighted CCA. We denote the chemical profile and the interaction profile, respectively, by a p_{ch} -dimensional vector \mathbf{x}^{ch} and a p_{in} -dimensional vector \mathbf{x}^{in} . Assuming that the functions $\Phi_{\text{ch}} : \mathbb{R}^{p_{\text{ch}}} \rightarrow \mathbb{R}^m$ and $\Phi_{\text{in}} : \mathbb{R}^{p_{\text{in}}} \rightarrow \mathbb{R}^m$ are affine transformations allows us to express them as

$$\Phi_{\text{ch}}(\mathbf{x}^{\text{ch}}) = \mathbf{W}_{\text{ch}}^{\text{T}}(\mathbf{x}^{\text{ch}} - \boldsymbol{\mu}_{\text{ch}}), \quad \Phi_{\text{in}}(\mathbf{x}^{\text{in}}) = \mathbf{W}_{\text{in}}^{\text{T}}(\mathbf{x}^{\text{in}} - \boldsymbol{\mu}_{\text{in}}),$$

where $\mathbf{W}_{\text{ch}} \in \mathbb{R}^{p_{\text{ch}} \times m}$, $\boldsymbol{\mu}_{\text{ch}} \in \mathbb{R}^{p_{\text{ch}}}$, $\mathbf{W}_{\text{in}} \in \mathbb{R}^{p_{\text{in}} \times m}$, and $\boldsymbol{\mu}_{\text{in}} \in \mathbb{R}^{p_{\text{in}}}$ are their respective parameters. We wish to find the pair of projection functions minimizing the expected deviation between the images given by

$$J(\Phi_{\text{ch}}, \Phi_{\text{in}}) \equiv \mathbb{E}[\|\Phi_{\text{ch}}(\mathbf{x}^{\text{ch}}) - \Phi_{\text{in}}(\mathbf{x}^{\text{in}})\|^2],$$

where \mathbb{E} is the expectation operator.

The expected deviation can be reduced arbitrarily by setting the projections so that the images are scaled down. A trivial solution is $\mathbf{W}_{\text{ch}} = \mathbf{0}$ and $\mathbf{W}_{\text{in}} = \mathbf{0}$ at which the expected deviation vanishes for any dataset. To avoid trivial solutions, the size of the images is adjusted by fixing the second moment matrices, $\mathbb{E}[\Phi_{\text{ch}}(\mathbf{x}^{\text{ch}})\Phi_{\text{ch}}(\mathbf{x}^{\text{ch}})^{\text{T}}]$ and $\mathbb{E}[\Phi_{\text{in}}(\mathbf{x}^{\text{in}})\Phi_{\text{in}}(\mathbf{x}^{\text{in}})^{\text{T}}]$, to identity matrices.

The expectation appearing in the derivation and the second moment matrices operates according to an empirical probabilistic distribution. Supposing n ligands are given, the chemical profiles are denoted by $\mathbf{x}_1^{\text{ch}}, \mathbf{x}_2^{\text{ch}}, \dots, \mathbf{x}_n^{\text{ch}}$, and the interaction profiles by $\mathbf{x}_1^{\text{in}}, \mathbf{x}_2^{\text{in}}, \dots, \mathbf{x}_n^{\text{in}}$. If we define an empirical distribution as

$$q(\mathbf{x}^{\text{ch}}, \mathbf{x}^{\text{in}}) = \sum_{j=1}^n v_j \delta(\mathbf{x}^{\text{ch}} - \mathbf{x}_j^{\text{ch}}) \delta(\mathbf{x}^{\text{in}} - \mathbf{x}_j^{\text{in}}),$$

with weights v_1, v_2, \dots, v_n whose sum is one and $\delta(\cdot)$ is the Dirac delta function, then the expected deviation is reduced to the weighted average of deviation and can be expressed as

$$J(\Phi_{\text{ch}}, \Phi_{\text{in}}) = \sum_{j=1}^n v_j \|\Phi_{\text{ch}}(\mathbf{x}_j^{\text{ch}}) - \Phi_{\text{in}}(\mathbf{x}_j^{\text{in}})\|^2. \quad (3.1)$$

This implies that approximations are refined locally by setting the weights so that ligands dissimilar from the target ligand are given smaller weights.

The optimal projections can be computed via the generalized eigendecomposition, as given in Algorithm 3.1 in Subsection 3.5.2. When setting $v_j = 1/n$, the algorithm is shown to be equivalent to the classical CCA. Hence, we can say that weighted CCA is an extension of the classical CCA.

Kernelization of weighted CCA is formulated with a similarity function of chemical profiles $K_{\text{ch}}(\mathbf{x}_i^{\text{ch}}, \mathbf{x}_j^{\text{ch}})$ and a similarity function of

interaction profiles $K_{\text{in}}(\mathbf{x}_i^{\text{in}}, \mathbf{x}_j^{\text{in}})$ without using the vectors themselves explicitly. These similarity functions are said to be valid kernels guaranteeing the theory of the algorithms, which map the profiles nonlinearly into other (typically high-dimensional) spaces \mathcal{H}_{ch} and \mathcal{H}_{in} , respectively, called an RKHS. Kernelized weighted CCA finds affine-transforms from RKHS to a canonical space \mathbb{R}^m , so that the expected deviation between images in \mathbb{R}^m is minimized. If we denote the composite mapping functions by $\boldsymbol{\psi}_{\text{ch}}$ and $\boldsymbol{\psi}_{\text{in}}$, respectively, the optimal solution is given by

$$\boldsymbol{\psi}_{\text{ch}}(\mathbf{x}^{\text{ch}}) = \mathbf{A}_{\text{ch}}^{\text{T}} \mathbf{D}_{\text{v}}^{1/2} \bar{\mathbf{k}}_{\text{ch}}(\mathbf{x}^{\text{ch}}), \quad \boldsymbol{\psi}_{\text{in}}(\mathbf{x}^{\text{in}}) = \mathbf{A}_{\text{in}}^{\text{T}} \mathbf{D}_{\text{v}}^{1/2} \bar{\mathbf{k}}_{\text{in}}(\mathbf{x}^{\text{in}}).$$

The algorithm for computing the two matrices, $\mathbf{A}_{\text{ch}} \in \mathbb{R}^{m \times n}$ and $\mathbf{A}_{\text{in}} \in \mathbb{R}^{m \times n}$, is presented in Algorithm 3.2 in Subsection 3.5.3. The functions $\bar{\mathbf{k}}_{\text{ch}}(\cdot)$ and $\bar{\mathbf{k}}_{\text{in}}(\cdot)$ are called *empirical kernel mappings*, and their definitions can be derived from Equation (3.10), also in Subsection 3.5.3.

3.2.3 Weighted SVM

Prediction of the interaction between ligand i and target t is performed with the SVM score given by

$$f(\mathbf{x}_i^{\text{ch}}; \mathbf{w}_{(i,t)}, \mathbf{b}_{(i,t)}) = \mathbf{w}_{(i,t)}^{\text{T}} \boldsymbol{\psi}_{\text{ch}}(\mathbf{x}_i^{\text{ch}}) + \mathbf{b}_{(i,t)},$$

where \mathbf{x}_i^{ch} is the chemical profile of ligand i . The SVM parameters, $\mathbf{w}_{(i,t)}$ and $\mathbf{b}_{(i,t)}$, are obtained beforehand by the SVM learning algorithm. This is performed only with ligands whose interaction with the target t is known. This study employs the similarity of ligands as weights in the learning process, as presented in Subsection 3.5.4.

3.2.4 Weighting schemes

Ligands are given weights in both stages of the weighted CCA and the weighted SVM. These weights are dependent on the ligand to be predicted. Larger weights are given for ligands that are more similar to the ligand of interest. In predicting the interaction of the i th ligand, the weight of j th ligand is given by the normalization of

$$v'_j = \frac{1}{\|\bar{\mathbf{k}}_{\text{ch}}(\mathbf{x}_j^{\text{ch}}) - \bar{\mathbf{k}}_{\text{ch}}(\mathbf{x}_i^{\text{ch}})\| + \|\bar{\mathbf{k}}_{\text{in}}(\mathbf{x}_j^{\text{in}}) - \bar{\mathbf{k}}_{\text{in}}(\mathbf{x}_i^{\text{in}})\| + \epsilon}, \quad (3.2)$$

where ϵ is a positive constant and set to 10 in our analysis. Normalization is done by setting

$$v_j = \frac{v'_j}{\sum_{k=1}^n v'_k}$$

so that the sum of the weights is one.

3.3 EXPERIMENTS AND RESULTS

3.3.1 Data and Experimental Settings

The data used for this study was originally from [69]. The given interaction matrix consists of 62 mammalian odorant receptors as target proteins and 63 odorants as candidate ligands. It is binary in form and contains 340 positive interactions. The number of known positive interactions for each target protein is at least one and at most thirty-seven, while the median is three. Some randomly selected protein-ligand pairs are assumed to be unknown to test prediction methods, and the values of the cells are set to zero. Each row in the interaction matrix provides an *interaction profile* of the ligand.

From the chemical IDs supplied, we searched PubChem¹ for the chemical structures of the odorants to obtain the descriptors of the ligands. Frequent substructures are employed as descriptors of ligands. The frequent substructures are mined with a software named *gSpan* [92]. The software is applied to the 63 chemical structures, and the 60,311 binary descriptors are obtained as *chemical profiles*.

To illustrate the effectiveness of the kernel weighted CCA (KWCCA), we carried out experiments on an interaction dataset of G protein-coupled receptors and odorants as previously described. For evaluation of prediction performance, we applied a 10-fold Monte-Carlo cross validation, where data is randomly divided into 2 disjoint sets of training and test data for 10 repetitions. Data was partitioned such that for each target protein, 50% of the positive and negative interactions are used for training, and the other half for testing. KCCA, KWCCA, and the weighted SVM were implemented in Matlab, and LIBSVM [13] was used for the classical SVM.

We also performed prediction using SVM in the global model setting for comparison. The kernel function for the global model here is defined as the product of the inner product among chemical profiles and the inner product among columns of the interaction matrix.

Parameters of the local models are determined by finding respective values where the test data perform best using SVM and KCCA. Namely, the regularization parameter C and the kernel function for SVM are chosen so that SVM achieves the highest prediction performance, while the regularization parameters for CCA γ_{ch} and γ_{in} , and the number of dimensions of the canonical space m , are determined via the performance of KCCA. As a result, the values of the parameters are set as $C = 1000$, $\gamma_{\text{ch}} = \gamma_{\text{in}} = 1$, and $m = 4$. The RBF kernel is applied and the kernel width is determined as the mean of the distance within sets. These mentioned parameters are then fed into the algorithm employing KWCCA. The parameters are not tuned specifically for KWCCA. Thus, it is believed that there is a chance

¹ <http://pubchem.ncbi.nlm.nih.gov/>

Table 3.1: **Abbreviation of methods.**

Abbreviation	Description
WW	KWCCA + Weighted SVM
WU	KWCCA + Classical SVM
KW	Classical KCCA + Weighted SVM
KU	Classical KCCA + Classical SVM
S	SVM of local model
SGL	Linear SVM of global model
SGR	RBF SVM of global model

of improvement in the performance of this algorithm if careful and suitable parameter selection is done.

For the global model, the kernel which achieves the best performance is the linear kernel. The regularization parameter is chosen as $C = 10$, achieving the best performance among other values. Results for the case of the RBF kernel with the best C value obtained are also reported for comparison.

The methods based on KWCCA involve two stages upon implementation. First, we exploit KWCCA to extract a set of features for each compound. Second, we use them for training a machine learning algorithm employing SVMs before testing them to make predictions. In total, seven methods are implemented in the experiments: two using SVM in the global model setting, and the other five following the local model. One of the two global model methods uses RBF kernel for SVM, and the other uses the linear kernel. On the other hand, the methods used for the local models are as follows: SVM, KCCA with classical SVM, KCCA with weighted SVM, KWCCA with classical SVM, and KWCCA with weighted SVM. For simplicity of notation, we shall refer to each of the seven methods using the abbreviations in Table 3.1.

3.3.2 Performance Evaluation Criteria

The following criteria were used to compare the seven prediction methods:

1. Area under the ROC curve (AUC) – The receiver operating characteristic (ROC) curve is a plot of the true positive rate (TPR) versus the false positive rate (FPR) where

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

and TP, FN, FP, and TN are the number of true positives, false negatives, false positives, and true negatives, respectively. For performance comparison using the ROC, the AUC value is further computed.

2. F-measure – A value which is given by the harmonic mean of precision and recall:

$$F = \frac{2\text{Prec} \times \text{Recall}}{\text{Prec} + \text{Recall}}$$

where the precision Prec and the recall Recall are defined by

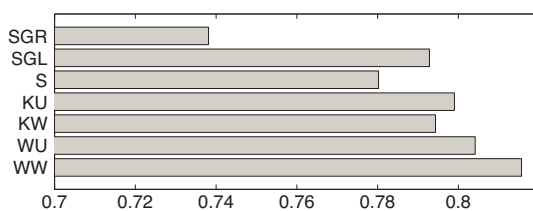
$$\text{Prec} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \text{and} \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

respectively. Since the problem is presented as a binary classification problem, only the maximum value of the F-measure values for each target is considered. The scores obtained via SVM are used as confidence levels, thus, changing the threshold yields different predictions.

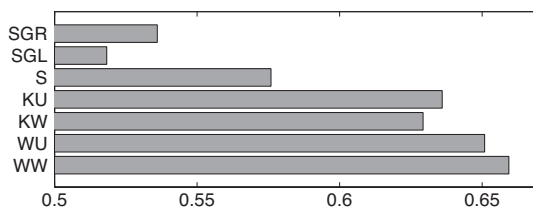
These values are calculated for each target protein and averaged over the ten data divisions. However, there are instances when the test set does not contain a true positive interaction, hence AUC and F-measures cannot be computed. Therefore, these values were disregarded and, out of 62 target proteins, AUC and F-measures were computed for 49 of them. The Wilcoxon signed test was used for the statistical significance of the difference among the values of the evaluation measures.

3.3.3 *Effects of Using CCA and Weighting*

The average AUCs and F-measures are reported in Figures 3.3a and 3.3b, respectively. In comparison with the local models, four CCA-based methods, WW, WU, KW, and KU, achieve remarkably better AUCs and F-measures compared to those of S: the differences between the AUCs and F-measures of KW, the worst among the four CCA-based methods, and S are 0.014 and 0.053, respectively, (P-values: 5.81×10^{-7} and 9.49×10^{-9} respectively). The AUC of the global model SGL is comparable to some of the local models, whereas the F-measure is not worse than that of S. A closer inspection on the results of SGL indicate that it has the lowest average number of true positives over all cross-validations among all models, around 161, which may be the reason behind a very small F-measure value.



(a) Average AUC values.

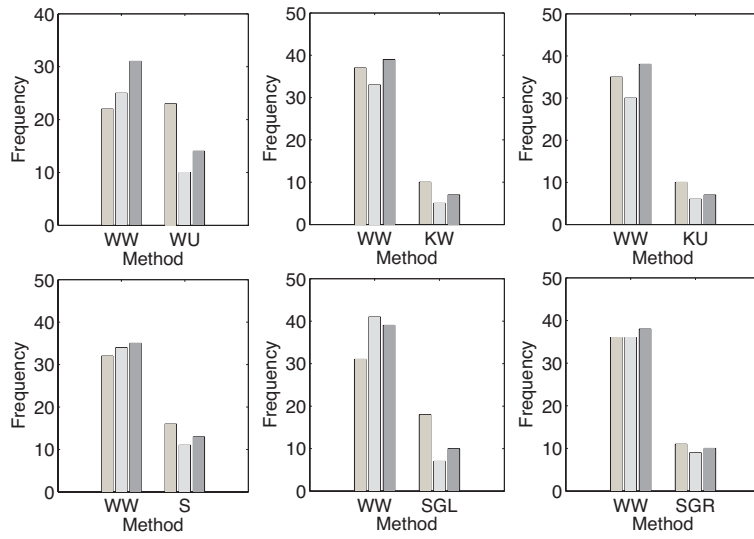


(b) Average F-measure values.

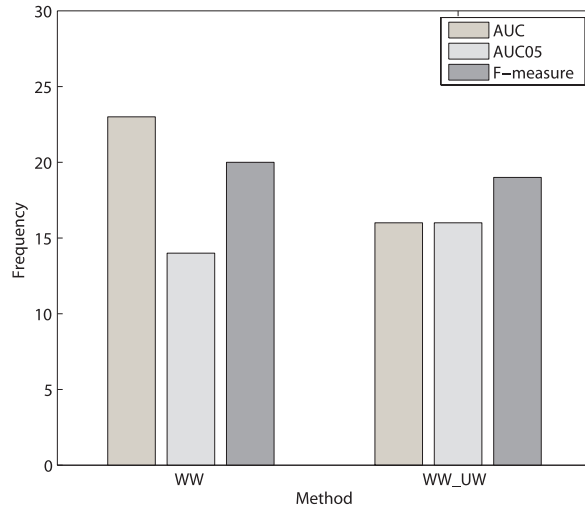
Figure 3.3: **Average performance of the methods.** Data was randomly split into training and test sets, and 10 training-testing data divisions were used for each method. Following the local model, AUC and F-measure were computed for each of the 62 targets. The bar plots represent the average AUC and average F-measure values, respectively, over the 10 cross validation sets and the 49 targets containing true positives. The two KWCCA-based methods, WW and WU, and the other methods were implemented for comparison. The difference of the performances of WW and WU from the other five methods showed to be statistically significant in terms of the P-values (by Wilcoxon signed rank test).

The effects of the weighted extension of CCA are manifested via comparison among four CCA-based methods. WW achieves significantly higher AUC and F-measures in average compared to KW and KU, where the P-values for the difference in the AUCs are 4.85×10^{-11} and 6.91×10^{-10} , and the P-values for the F-measures are 3.59×10^{-7} and 3.96×10^{-6} , respectively.

The frequencies of WW besting the AUC or F-measure values of the other methods in predicting interactions for a certain target protein are shown in the histograms in Figure 3.4a. These values represent the number of target proteins such that the evaluated AUC and F-measure values for the method WW is better than the AUC and F-measure values of the other method in comparison. Instances when there are ties between the methods were unaccounted. For the evaluated AUC and F-measure values, WW outputs are more desirable than most of the others which indicates higher quality of prediction performance.



(a) Histogram comparisons of the proposed method WW vs. other methods.



(b) Histogram comparisons of Weighted SVM using the weighting scheme in Equation (3.3) vs. Classical SVM.

Figure 3.4: **Histograms illustrating the comparisons between the proposed method WW and other methods.** Frequencies when the AUC, AUC between 0 and 0.05, and F-measure values of WW outperform the other methods, and vice-versa, are illustrated. It can be observed that AUC and F-measure values histograms for WW are more desirable than the rest.

3.3.4 *Weighted SVM vs Classical SVM*

Method WU yields interesting results in the histogram (Figure 3.4a): The frequency of WW yielding better AUCs are comparable to that of WU's, although frequency of better F-measures are relatively higher for WW than WU. To further investigate the comparison between WW and WU, we compute the area under the curve of the region of FPR between 0 and 0.05. This area, which we shall refer to as AUC05, allows us to evaluate the true positive rate with higher confidence. The histogram on AUC05 shows WW bests WU more frequently than WU does, which implies the use of weights in the SVM stage can find more true positives confidently than the classical SVM.

The motivation to endow the weights with training data in SVM learning is that the projections in the canonical space from chemical profiles with larger weights are expected to be better approximations of the projections from interaction profiles. It is possible to directly evaluate how good the approximations are by computing the distances among the projections. This motivation leads to another weighting scheme using the normalization of

$$v'_j = \frac{1}{\|\Phi_{\text{ch}}(\mathbf{x}_j^{\text{ch}}) - \Phi_{\text{in}}(\mathbf{x}_j^{\text{in}})\| + \epsilon} \quad (3.3)$$

instead of that in (3.2) during the SVM learning stage. We investigate the performance when the weighting scheme is changed to (3.3) in the SVM learning stage, and refer to this approach as WW_{UW} hereinafter. The average AUC and F-measure of WW_{UW} are 0.802 and 0.649, respectively, which are slightly worse than those of WW. The number of target proteins, for which the prediction performance of WW_{UW} is better than that of WW is not larger than the number of WW besting WW_{UW} , as depicted in Figure 3.4b. These facts imply that the changing weighting scheme in SVM learning does not achieve significant improvements.

3.3.5 *Using Interaction Profiles*

When a sufficient number of known positive and negative interactions are given for a certain ligand, the image of the interaction profile in the canonical image can provide good descriptors for predicting the remaining interactions. We further implemented two methods, herein referred to as WWI and WWIC, to investigate the performance of the interaction profile. WWI replaces the image of a chemical profile with the image of the interaction profile in the SVM stage, while WWIC concatenates the two images to feed them to the weighted SVM. The two methods achieved significant improvement. WWI achieved an average AUC of 0.857 and average F-measure of 0.699, while WWIC obtained a 0.835 average AUC and a 0.692 average F-measure. The

P-values of the differences on AUC from WW are 5.27×10^{-9} and 0.021, respectively, and P-values on F-measures are 1.05×10^{-5} and 9.17×10^{-7} , respectively. Figure 3.5 shows the average ROC curves of WW, WWI, and WWIC. The curves of WWI and WWIC are evidently higher than that of WW, which supports the claim that introducing the interaction profiles improves the prediction performance.

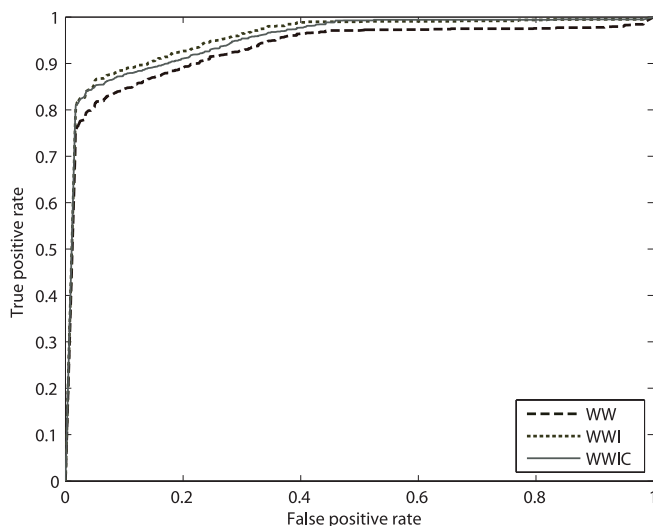


Figure 3.5: **ROC curves.** WWI uses the projections from interaction profiles in the SVM stage, and WWIC uses the projections from both chemical and interaction profiles for SVM.

3.4 SUMMARY

A kernel version of weighted canonical correlation analysis is proposed, which is implemented using a derived form of the generalized eigenvalue problem. Similar to the linear CCA and its kernelized version, this can be applied to machine learning problems for dimension reduction and feature extraction. The paper presents an application to improving the prediction quality obtained in the protein-ligand interaction problem setting. By adding bias to more similar samples, better prediction can be made which is evident on the higher AUC and F-measure values obtained. Weighting scheme on SVM based on CCA outputs were also explored and are judged to be better than classical SVM.

3.5 SUPPORTING THEORIES AND PROOFS

Here we present the details of some of the concepts employed in this chapter.

3.5.1 Generalized Eigendecomposition

The generalized eigendecomposition is defined as follows.

Theorem 3.1 (Generalized Eigendecomposition). *Let $n \in \mathbb{N}$. For any $\mathbf{A} \in \mathbb{S}^n$ and any $\mathbf{B} \in \mathbb{S}_{++}^n$,*

$$\exists \mathbf{U} \in \mathbb{R}^{n \times n}, \exists \boldsymbol{\lambda} \in \mathbb{R}^n, \text{ such that } \mathbf{AU} = \mathbf{BU}\boldsymbol{\lambda}, \\ \mathbf{U}^T \mathbf{BU} = \mathbf{I}_n.$$

The entries of $\boldsymbol{\lambda}$ are called *generalized eigenvalues*, and the columns of \mathbf{U} are called *generalized eigenvectors*.

This subsection deals with the case when two matrices $\mathbf{A} \in \mathbb{S}^{n+m}$ and $\mathbf{B} \in \mathbb{S}_{++}^{n+m}$ are of the form

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{m \times m} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{0}_{n \times n} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{B}_x & \mathbf{0}_{m \times n} \\ \mathbf{0}_{n \times m} & \mathbf{B}_y \end{bmatrix},$$

where $\mathbf{C} \in \mathbb{R}^{m \times n}$ with $r \equiv \text{rank}(\mathbf{C})$. Let us denote the generalized eigendecomposition of (\mathbf{A}, \mathbf{B}) by

$$\mathbf{AU}_{\text{all}} = \mathbf{BU}_{\text{all}}\boldsymbol{\Lambda}_{\text{all}},$$

where $\mathbf{U}_{\text{all}}^T \mathbf{BU}_{\text{all}} = \mathbf{I}_{m+n}$, $\boldsymbol{\Lambda}_{\text{all}} = \text{diag}(\boldsymbol{\lambda}_{\text{all}})$, and $\boldsymbol{\lambda}_{\text{all}} = [\lambda_1, \lambda_2, \dots, \lambda_{m+n}]^T$, such that $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_{m+n}$. We denote the columns in \mathbf{U}_{all} by

$$\mathbf{U}_{\text{all}} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{m+n}] = \begin{bmatrix} \mathbf{u}_{x,1}, \mathbf{u}_{x,2}, \dots, \mathbf{u}_{x,m+n} \\ \mathbf{u}_{y,1}, \mathbf{u}_{y,2}, \dots, \mathbf{u}_{y,m+n} \end{bmatrix},$$

where $\forall i \in \mathbb{N}_{m+n}$, $\mathbf{u}_{x,i} \in \mathbb{R}^m$, $\mathbf{u}_{y,i} \in \mathbb{R}^n$, and define

$$\mathbf{U}_x \equiv [\mathbf{u}_{x,1}, \mathbf{u}_{x,2}, \dots, \mathbf{u}_{x,r}], \quad \mathbf{U}_y \equiv [\mathbf{u}_{y,1}, \mathbf{u}_{y,2}, \dots, \mathbf{u}_{y,r}].$$

The following theorem is the main result of this subsection.

Theorem 3.1. *Consider the following optimization problem*

$$\begin{aligned} & \text{maximize} && \text{tr}(\mathbf{X}^T \mathbf{C} \mathbf{Y}) \\ & \text{wrt} && \mathbf{X} \in \mathbb{R}^{m \times k}, \mathbf{Y} \in \mathbb{R}^{n \times k}, \\ & \text{subject to} && \mathbf{X}^T \mathbf{B}_x \mathbf{X} = \mathbf{Y}^T \mathbf{B}_y \mathbf{Y} = \mathbf{I}_k. \end{aligned}$$

An optimal solution is given by

$$\mathbf{X} = \sqrt{2}[\mathbf{u}_{x,1}, \mathbf{u}_{x,2}, \dots, \mathbf{u}_{x,k}], \quad \mathbf{Y} = \sqrt{2}[\mathbf{u}_{y,1}, \mathbf{u}_{y,2}, \dots, \mathbf{u}_{y,k}].$$

To prove Theorem 3.1, we will use the following lemma.

Lemma 3.1.

$$\mathbf{U}_x^T \mathbf{B}_x \mathbf{U}_x = \mathbf{U}_y^T \mathbf{B}_y \mathbf{U}_y = \frac{1}{2} \mathbf{I}_r.$$

Proof. Let

$$\mathbf{\Lambda} \equiv \text{diag}\{\lambda_1, \lambda_2, \dots, \lambda_r\}.$$

Here we assume that $\lambda_r > 0$. From this assumption,

$$\mathbf{C} \mathbf{U}_y = \mathbf{B}_x \mathbf{U}_x \mathbf{\Lambda}, \quad \text{and} \quad \mathbf{C}^T \mathbf{U}_x = \mathbf{B}_y \mathbf{U}_y \mathbf{\Lambda}.$$

Pre-multiplying the former equation by \mathbf{U}_x^T and post-multiplying the transpose of the latter equation by \mathbf{U}_y yield

$$\mathbf{U}_x^T \mathbf{B}_x \mathbf{U}_x \mathbf{\Lambda} = \mathbf{U}_x^T \mathbf{C} \mathbf{U}_y = \mathbf{\Lambda} \mathbf{U}_y^T \mathbf{B}_y \mathbf{U}_y.$$

For the diagonal entries of the above equality,

$$\lambda_i \mathbf{u}_{x,i}^T \mathbf{B}_x \mathbf{u}_{x,i} = \lambda_i \mathbf{u}_{y,i}^T \mathbf{B}_y \mathbf{u}_{y,i}, \quad \forall i \in \mathbb{N}_r,$$

resulting in

$$\mathbf{u}_{x,i}^T \mathbf{B}_x \mathbf{u}_{x,i} = \mathbf{u}_{y,i}^T \mathbf{B}_y \mathbf{u}_{y,i} = \frac{1}{2} \quad (3.4)$$

since $\lambda_i > 0$ and from the assumption that $\mathbf{U}_{\text{all}}^T \mathbf{B} \mathbf{U}_{\text{all}} = \mathbf{I}_{m+n}$. For the off-diagonal entries,

$$\mathbf{u}_{x,i}^T \mathbf{B}_x \mathbf{u}_{x,j} - \frac{\lambda_i}{\lambda_j} \mathbf{u}_{y,i}^T \mathbf{B}_y \mathbf{u}_{y,j} = 0. \quad (3.5)$$

Again from $\mathbf{U}_{\text{all}}^T \mathbf{B} \mathbf{U}_{\text{all}} = \mathbf{I}_{m+n}$, we also have

$$\begin{bmatrix} \mathbf{u}_{x,i} \\ \mathbf{u}_{y,i} \end{bmatrix}^T \mathbf{B} \begin{bmatrix} \mathbf{u}_{x,j} \\ \mathbf{u}_{y,j} \end{bmatrix} = \mathbf{u}_{x,i}^T \mathbf{B}_x \mathbf{u}_{x,j} + \mathbf{u}_{y,i}^T \mathbf{B}_y \mathbf{u}_{y,j} = 0. \quad (3.6)$$

From the two equations (3.5) and (3.6), we obtain

$$\mathbf{u}_{x,i}^T \mathbf{B}_x \mathbf{u}_{x,j} = \mathbf{u}_{y,i}^T \mathbf{B}_y \mathbf{u}_{y,j} = 0. \quad (3.7)$$

Thus, equations (3.4) and (3.7) establish Lemma 3.1. \square

Proof. (of Theorem 3.1) Let

$$\mathbf{Z} \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \end{bmatrix}.$$

There exists $\mathbf{R} \in \mathbb{R}^{(m+n) \times k}$ such that $\mathbf{Z} = \mathbf{U}_{\text{all}} \mathbf{R}$, and matrix \mathbf{R} is orthonormal, i.e. $\mathbf{R} \in \mathbf{O}^{(m+n) \times k}$, since

$$\begin{aligned} \mathbf{R}^T \mathbf{R} &= \mathbf{R}^T \mathbf{I}_{(m+n)} \mathbf{R} = \mathbf{R}^T \mathbf{U}_{\text{all}}^T \mathbf{B} \mathbf{U}_{\text{all}} \mathbf{R} = \mathbf{Z}^T \mathbf{B} \mathbf{Z} \\ &= \frac{1}{2} \mathbf{X}^T \mathbf{B}_x \mathbf{X} + \frac{1}{2} \mathbf{Y}^T \mathbf{B}_y \mathbf{Y} = \mathbf{I}_k. \end{aligned}$$

Let $\mathbf{r}_i \in \mathbb{R}^k$ denote the i th row vector, i.e.

$$\mathbf{R} = \begin{bmatrix} \mathbf{r}_1^\top \\ \mathbf{r}_2^\top \\ \vdots \\ \mathbf{r}_{m+n}^\top \end{bmatrix}.$$

Then we have

$$\begin{aligned} \text{tr}(\mathbf{X}^\top \mathbf{C} \mathbf{Y}) &= \frac{1}{2} \text{tr}(\mathbf{X}^\top \mathbf{C} \mathbf{Y}) + \frac{1}{2} \text{tr}(\mathbf{Y}^\top \mathbf{C}^\top \mathbf{X}) \\ &= \frac{1}{2} \text{tr} \left([\mathbf{X}^\top, \mathbf{Y}^\top] \begin{bmatrix} \mathbf{0}_{m \times m} & \mathbf{C} \\ \mathbf{C}^\top & \mathbf{0}_{n \times n} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \end{bmatrix} \right) \\ &= \text{tr}(\mathbf{Z}^\top \mathbf{A} \mathbf{Z}) = \text{tr}(\mathbf{R}^\top \mathbf{U}_{\text{all}}^\top \mathbf{A} \mathbf{U}_{\text{all}} \mathbf{R}) \\ &= \text{tr}(\mathbf{R}^\top \mathbf{U}_{\text{all}}^\top \mathbf{B} \mathbf{U}_{\text{all}} \mathbf{D}_{\lambda_{\text{all}}} \mathbf{R}) = \text{tr}(\mathbf{R}^\top \mathbf{D}_{\lambda_{\text{all}}} \mathbf{R}) \\ &= \sum_{i=1}^{m+n} \lambda_i \|\mathbf{r}_i\|^2 = \langle \mathbf{w}, \lambda_{\text{all}} \rangle, \end{aligned} \quad (3.8)$$

where $\mathbf{w} \in \mathbb{R}_+^{m+n}$ is a nonnegative vector in which the i th entry is defined by $w_i = \|\mathbf{r}_i\|^2$. To conclude the proof, we shall make use of the following two properties:

First, note that there exists $\mathbf{S} \in \mathbf{O}^{(m+n) \times (m+n-k)}$ such that $\mathbf{R}^\top \mathbf{S} = \mathbf{0}_{k \times (m+n-k)}$. Then $[\mathbf{R}, \mathbf{S}] \in \mathbf{O}^{(m+n) \times (m+n)}$. Let $\mathbf{s}_i \in \mathbb{R}^{(m+n-k)}$ denote the i th row vector, i.e.

$$\mathbf{S} = \begin{bmatrix} \mathbf{s}_1^\top \\ \mathbf{s}_2^\top \\ \vdots \\ \mathbf{s}_{m+n}^\top \end{bmatrix}.$$

Then $\forall i \in \mathbb{N}_n$ we have

$$1 = \left\| \begin{bmatrix} \mathbf{r}_i \\ \mathbf{s}_i \end{bmatrix} \right\|^2 = \|\mathbf{r}_i\|^2 + \|\mathbf{s}_i\|^2 \geq \|\mathbf{r}_i\|^2 = w_i,$$

where the first equality follows from the property of the square orthonormal matrix: $[\mathbf{R}, \mathbf{S}][\mathbf{R}, \mathbf{S}]^\top = \mathbf{I}_n$. Second, observe that

$$\|\mathbf{w}\|_1 = \sum_{i=1}^n w_i = \sum_{i=1}^n \|\mathbf{r}_i\|^2 = \text{tr}(\mathbf{R} \mathbf{R}^\top) = \text{tr}(\mathbf{R}^\top \mathbf{R}) = \text{tr}(\mathbf{I}_k) = k.$$

Now the objective function $\text{tr}(\mathbf{X}^\top \mathbf{C} \mathbf{Y})$ is maximized when

$$w_i = \begin{cases} 1, & \text{if } 1 \leq i \leq k, \\ 0, & \text{if } k < i \leq n, \end{cases}$$

and so the value of (3.8) is bounded above by

$$\langle \mathbf{w}, \boldsymbol{\lambda} \rangle \leq \sum_{i=1}^k \lambda_i.$$

Finally, we show that equality holds when

$$\mathbf{R} = \begin{bmatrix} \mathbf{I}_k \\ \mathbf{0}_{(m+n-k) \times k} \end{bmatrix}.$$

Since $\mathbf{Z} = \mathbf{UR} = \mathbf{U}_{\text{ma}}$, where $\mathbf{U}_{\text{ma}} \equiv [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k]$, we have

$$\text{tr}(\mathbf{Z}^T \mathbf{AZ}) = \text{tr}(\mathbf{U}_{\text{ma}}^T \mathbf{AU}_{\text{ma}}) = \sum_{i=1}^k \lambda_i.$$

Thus, $\mathbf{Z} = \mathbf{U}_{\text{ma}}$ is an optimum, which implies

$$\mathbf{X} = \sqrt{2}[\mathbf{u}_{x,1}, \mathbf{u}_{x,2}, \dots, \mathbf{u}_{x,k}], \quad \mathbf{Y} = \sqrt{2}[\mathbf{u}_{y,1}, \mathbf{u}_{y,2}, \dots, \mathbf{u}_{y,k}].$$

□

3.5.2 Linear Weighted CCA

Algorithm 3.1 (Linear Weighted CCA). *Define two matrices,*

$$\mathbf{X}_{ch} \equiv [\mathbf{x}_1^{ch}, \mathbf{x}_2^{ch}, \dots, \mathbf{x}_n^{ch}] \quad \text{and} \quad \mathbf{X}_{in} \equiv [\mathbf{x}_1^{in}, \mathbf{x}_2^{in}, \dots, \mathbf{x}_n^{in}],$$

and let $\mathbf{v} = [v_1, v_2, \dots, v_n]^T$. Then the optimal offsets, $\boldsymbol{\mu}_{ch}$ and $\boldsymbol{\mu}_{in}$, are computed as

$$\boldsymbol{\mu}_{ch} = \mathbf{X}_{ch} \mathbf{v} \quad \text{and} \quad \boldsymbol{\mu}_{in} = \mathbf{X}_{in} \mathbf{v}.$$

We use the optimal offsets to define $\mathbf{C}_{ch,ch}$, $\mathbf{C}_{ch,in}$, $\mathbf{C}_{in,ch}$, $\mathbf{C}_{in,in}$ as

$$\begin{aligned} \mathbf{C}_{ch,ch} &\equiv \mathbf{X}_{ch} \mathbf{D}_v \mathbf{X}_{ch}^T - \boldsymbol{\mu}_{ch} \boldsymbol{\mu}_{ch}^T, & \mathbf{C}_{ch,in} &\equiv \mathbf{X}_{ch} \mathbf{D}_v \mathbf{X}_{in}^T - \boldsymbol{\mu}_{ch} \boldsymbol{\mu}_{in}^T, \\ \mathbf{C}_{in,ch} &\equiv \mathbf{X}_{in} \mathbf{D}_v \mathbf{X}_{ch}^T - \boldsymbol{\mu}_{in} \boldsymbol{\mu}_{ch}^T, & \mathbf{C}_{in,in} &\equiv \mathbf{X}_{in} \mathbf{D}_v \mathbf{X}_{in}^T - \boldsymbol{\mu}_{in} \boldsymbol{\mu}_{in}^T, \end{aligned}$$

and consider the following generalized eigendecomposition problem:

$$\begin{bmatrix} \mathbf{0}_{p_{ch} \times p_{ch}} & \mathbf{C}_{ch,in} \\ \mathbf{C}_{in,ch} & \mathbf{0}_{p_{in} \times p_{in}} \end{bmatrix} \begin{bmatrix} \mathbf{w}^{ch} \\ \mathbf{w}^{in} \end{bmatrix} = \begin{bmatrix} \mathbf{C}_{in,in} & \mathbf{0}_{p_{in} \times p_{ch}} \\ \mathbf{0}_{p_{ch} \times p_{in}} & \mathbf{C}_{ch,ch} \end{bmatrix} \begin{bmatrix} \mathbf{w}^{ch} \\ \mathbf{w}^{in} \end{bmatrix}.$$

Denote the h th major eigenvector by $\begin{bmatrix} \mathbf{w}_h^{ch} \\ \mathbf{w}_h^{in} \end{bmatrix}$. The optimal loading matrices

\mathbf{W}_{ch} and \mathbf{W}_{in} are computed by setting the h th columns of \mathbf{W}_{ch} and \mathbf{W}_{in} to \mathbf{w}_h^{ch} and \mathbf{w}_h^{in} , respectively.

Theorem 3.2. Algorithm 3.1 yields the parameters of the mapping functions $(\Phi_{\text{ch}}, \Phi_{\text{in}})$ which minimize the expected deviation subject to the scaling constraints that the second moment matrices are the identity matrix.

To prove this theorem, we employ the following result.

Let two affine mapping functions $\Phi_{\text{ch}} : \mathbb{R}^{p_{\text{ch}}} \rightarrow \mathbb{R}^m$ and $\Phi_{\text{in}} : \mathbb{R}^{p_{\text{in}}} \rightarrow \mathbb{R}^m$ be parametrized by

$$\Phi_{\text{ch}}(\mathbf{x}_{\text{ch}}) = \mathbf{W}_{\text{ch}}^T (\mathbf{x}_{\text{ch}} - \boldsymbol{\mu}_{\text{ch}}) \quad \text{and} \quad \Phi_{\text{in}}(\mathbf{x}_{\text{in}}) = \mathbf{W}_{\text{in}}^T (\mathbf{x}_{\text{in}} - \boldsymbol{\mu}_{\text{in}}),$$

respectively. For a probabilistic distribution $p(\mathbf{x}^{\text{ch}}, \mathbf{x}^{\text{in}})$, define the expected deviation by

$$J = \mathbb{E}_{p(\mathbf{x}^{\text{ch}}, \mathbf{x}^{\text{in}})} [\|\Phi_{\text{ch}}(\mathbf{x}^{\text{ch}}) - \Phi_{\text{in}}(\mathbf{x}^{\text{in}})\|^2].$$

If we consider the optimization problem for minimizing the expected deviation with respect to the parameters $(\mathbf{W}_{\text{ch}}, \boldsymbol{\mu}_{\text{ch}}, \mathbf{W}_{\text{in}}, \boldsymbol{\mu}_{\text{in}})$ subject to

$$\mathbb{E}_{p(\mathbf{x}^{\text{ch}})} [\Phi_{\text{ch}}(\mathbf{x}^{\text{ch}}) \Phi_{\text{ch}}(\mathbf{x}^{\text{ch}})^T] = \mathbb{E}_{p(\mathbf{x}^{\text{in}})} [\Phi_{\text{in}}(\mathbf{x}^{\text{in}}) \Phi_{\text{in}}(\mathbf{x}^{\text{in}})^T] = \mathbf{I}_m,$$

a minimizer of the optimization problem is found by the following optimization problem.

Set the offset vectors as

$$\boldsymbol{\mu}_{\text{ch}} = \mathbb{E}_{p(\mathbf{x}^{\text{ch}})} [\mathbf{x}^{\text{ch}}] \quad \text{and} \quad \boldsymbol{\mu}_{\text{in}} = \mathbb{E}_{p(\mathbf{x}^{\text{in}})} [\mathbf{x}^{\text{in}}].$$

The optimal loading matrices \mathbf{W}_{ch} and \mathbf{W}_{in} are computed by setting the h th columns of \mathbf{W}_{ch} and \mathbf{W}_{in} to \mathbf{w}_h^{ch} and \mathbf{w}_h^{in} , respectively, and denoting by $\begin{bmatrix} \mathbf{w}_h^{\text{ch}} \\ \mathbf{w}_h^{\text{in}} \end{bmatrix}$ the h th major eigenvector of the generalized eigendecomposition:

$$\begin{aligned} & \begin{bmatrix} \mathbf{0}_{p_{\text{ch}} \times p_{\text{ch}}} & \check{\mathbf{C}}_{\text{ch}, \text{in}} \\ \check{\mathbf{C}}_{\text{in}, \text{ch}} & \mathbf{0}_{p_{\text{in}} \times p_{\text{in}}} \end{bmatrix} \begin{bmatrix} \mathbf{w}_h^{\text{ch}} \\ \mathbf{w}_h^{\text{in}} \end{bmatrix} \\ &= \begin{bmatrix} \check{\mathbf{C}}_{\text{in}, \text{in}} & \mathbf{0}_{p_{\text{in}} \times p_{\text{ch}}} \\ \mathbf{0}_{p_{\text{ch}} \times p_{\text{in}}} & \check{\mathbf{C}}_{\text{ch}, \text{ch}} \end{bmatrix} \begin{bmatrix} \mathbf{w}_h^{\text{ch}} \\ \mathbf{w}_h^{\text{in}} \end{bmatrix}, \end{aligned} \quad (3.9)$$

where the covariance matrices are given by

$$\begin{aligned} \check{\mathbf{C}}_{\text{ch}, \text{ch}} &\equiv \mathbb{E}_{p(\mathbf{x}^{\text{ch}})} [(\mathbf{x}^{\text{ch}} - \boldsymbol{\mu}_{\text{ch}})(\mathbf{x}^{\text{ch}} - \boldsymbol{\mu}_{\text{ch}})^T], \\ \check{\mathbf{C}}_{\text{ch}, \text{in}} &\equiv \mathbb{E}_{p(\mathbf{x}^{\text{ch}}, \mathbf{x}^{\text{in}})} [(\mathbf{x}^{\text{ch}} - \boldsymbol{\mu}_{\text{ch}})(\mathbf{x}^{\text{in}} - \boldsymbol{\mu}_{\text{in}})^T], \\ \check{\mathbf{C}}_{\text{in}, \text{ch}} &\equiv \check{\mathbf{C}}_{\text{ch}, \text{in}}^T, \\ \check{\mathbf{C}}_{\text{in}, \text{in}} &\equiv \mathbb{E}_{p(\mathbf{x}^{\text{in}})} [(\mathbf{x}^{\text{in}} - \boldsymbol{\mu}_{\text{in}})(\mathbf{x}^{\text{in}} - \boldsymbol{\mu}_{\text{in}})^T]. \end{aligned}$$

To verify this, we let

$$\mathbf{x} \equiv \begin{bmatrix} \mathbf{x}^{\text{ch}} \\ \mathbf{x}^{\text{in}} \end{bmatrix}, \quad \text{and} \quad \boldsymbol{\mu}_{\text{tot}} \equiv \begin{bmatrix} \boldsymbol{\mu}_{\text{ch}} \\ \boldsymbol{\mu}_{\text{in}} \end{bmatrix}.$$

Then the second order moment matrices are rewritten as

$$\begin{aligned}\mathbb{E}[\boldsymbol{\Phi}_{\text{ch}}(\mathbf{x}^{\text{ch}})\boldsymbol{\Phi}_{\text{ch}}(\mathbf{x}^{\text{ch}})^{\text{T}}] &= \mathbf{W}_{\text{ch}}^{\text{T}}\mathbb{E}[(\boldsymbol{\mu}_{\text{ch}} - \mathbf{x}^{\text{ch}})(\boldsymbol{\mu}_{\text{ch}} - \mathbf{x}^{\text{ch}})^{\text{T}}]\mathbf{W}_{\text{ch}}, \\ \mathbb{E}[\boldsymbol{\Phi}_{\text{in}}(\mathbf{x}^{\text{in}})\boldsymbol{\Phi}_{\text{in}}(\mathbf{x}^{\text{in}})^{\text{T}}] &= \mathbf{W}_{\text{in}}^{\text{T}}\mathbb{E}[(\boldsymbol{\mu}_{\text{in}} - \mathbf{x}^{\text{in}})(\boldsymbol{\mu}_{\text{in}} - \mathbf{x}^{\text{in}})^{\text{T}}]\mathbf{W}_{\text{in}},\end{aligned}$$

respectively, and the expected deviation is arranged as

$$\begin{aligned}J &= \mathbb{E}[\|\boldsymbol{\Phi}_{\text{ch}}(\mathbf{x}^{\text{ch}}) - \boldsymbol{\Phi}_{\text{in}}(\mathbf{x}^{\text{in}})\|^2] \\ &= \mathbb{E}[\|\mathbf{W}_{\text{ch}}^{\text{T}}(\mathbf{x}^{\text{ch}} - \boldsymbol{\mu}_{\text{ch}}) - \mathbf{W}_{\text{in}}^{\text{T}}(\mathbf{x}^{\text{in}} - \boldsymbol{\mu}_{\text{in}})\|^2] \\ &= \text{tr}(\mathbf{W}_{\text{ch}}^{\text{T}}\mathbb{E}[(\mathbf{x}^{\text{ch}} - \boldsymbol{\mu}_{\text{ch}})(\mathbf{x}^{\text{ch}} - \boldsymbol{\mu}_{\text{ch}})^{\text{T}}]\mathbf{W}_{\text{ch}}) \\ &\quad + \text{tr}(\mathbf{W}_{\text{in}}^{\text{T}}\mathbb{E}[(\mathbf{x}^{\text{in}} - \boldsymbol{\mu}_{\text{in}})(\mathbf{x}^{\text{in}} - \boldsymbol{\mu}_{\text{in}})^{\text{T}}]\mathbf{W}_{\text{in}}) \\ &\quad - 2\text{tr}(\mathbf{W}_{\text{ch}}^{\text{T}}\mathbb{E}[(\mathbf{x}^{\text{ch}} - \boldsymbol{\mu}_{\text{ch}})(\mathbf{x}^{\text{in}} - \boldsymbol{\mu}_{\text{in}})^{\text{T}}]\mathbf{W}_{\text{in}}) \\ &= 2m - 2\text{tr}(\mathbf{W}_{\text{ch}}^{\text{T}}\mathbb{E}[(\mathbf{x}^{\text{ch}} - \boldsymbol{\mu}_{\text{ch}})(\mathbf{x}^{\text{in}} - \boldsymbol{\mu}_{\text{in}})^{\text{T}}]\mathbf{W}_{\text{in}}).\end{aligned}$$

From here, we will first derive the optimal value of $\boldsymbol{\mu}_{\text{tot}}$, and then give the algorithm to find the optimal \mathbf{W}_{ch} and \mathbf{W}_{in} . Introducing the Lagrangian multipliers $\boldsymbol{\Lambda}_{\text{ch}} \in \mathbb{S}^{\text{p}_{\text{ch}}}$ and $\boldsymbol{\Lambda}_{\text{in}} \in \mathbb{S}^{\text{p}_{\text{in}}}$, the Lagrangian function is written as

$$\begin{aligned}\mathcal{L}_{\Lambda} &= 2m - 2\text{tr}(\mathbf{W}_{\text{ch}}^{\text{T}}\mathbb{E}[(\boldsymbol{\mu}_{\text{ch}} - \mathbf{x}^{\text{ch}})(\boldsymbol{\mu}_{\text{in}} - \mathbf{x}^{\text{in}})^{\text{T}}]\mathbf{W}_{\text{in}}) \\ &\quad - \langle \boldsymbol{\Lambda}_{\text{ch}}, \mathbf{I}_m - \mathbf{W}_{\text{ch}}^{\text{T}}\mathbb{E}[(\boldsymbol{\mu}_{\text{ch}} - \mathbf{x}^{\text{ch}})(\boldsymbol{\mu}_{\text{ch}} - \mathbf{x}^{\text{ch}})^{\text{T}}]\mathbf{W}_{\text{ch}} \rangle \\ &\quad - \langle \boldsymbol{\Lambda}_{\text{in}}, \mathbf{I}_m - \mathbf{W}_{\text{in}}^{\text{T}}\mathbb{E}[(\boldsymbol{\mu}_{\text{in}} - \mathbf{x}^{\text{in}})(\boldsymbol{\mu}_{\text{in}} - \mathbf{x}^{\text{in}})^{\text{T}}]\mathbf{W}_{\text{in}} \rangle.\end{aligned}$$

To obtain the values of $\boldsymbol{\mu}_{\text{ch}}$ and $\boldsymbol{\mu}_{\text{in}}$ at the saddle point, we set the derivative of the Lagrangian to zero:

$$\begin{aligned}\frac{\partial \mathcal{L}_{\Lambda}}{\partial \boldsymbol{\mu}_{\text{ch}}} &= 2\mathbf{W}_{\text{ch}}\mathbf{W}_{\text{in}}^{\text{T}}(\boldsymbol{\mu}_{\text{in}} - \mathbb{E}[\mathbf{x}^{\text{in}}]) + 2\mathbf{W}_{\text{ch}}\boldsymbol{\Lambda}_{\text{ch}}\mathbf{W}_{\text{ch}}^{\text{T}}(\boldsymbol{\mu}_{\text{ch}} - \mathbb{E}[\mathbf{x}^{\text{ch}}]), \\ \frac{\partial \mathcal{L}_{\Lambda}}{\partial \boldsymbol{\mu}_{\text{in}}} &= 2\mathbf{W}_{\text{in}}\mathbf{W}_{\text{ch}}^{\text{T}}(\boldsymbol{\mu}_{\text{ch}} - \mathbb{E}[\mathbf{x}^{\text{ch}}]) + 2\mathbf{W}_{\text{in}}\boldsymbol{\Lambda}_{\text{in}}\mathbf{W}_{\text{in}}^{\text{T}}(\boldsymbol{\mu}_{\text{in}} - \mathbb{E}[\mathbf{x}^{\text{in}}]).\end{aligned}$$

The two derivatives vanish simultaneously when

$$\boldsymbol{\mu}_{\text{ch}} = \mathbb{E}[\mathbf{x}^{\text{ch}}] \quad \text{and} \quad \boldsymbol{\mu}_{\text{in}} = \mathbb{E}[\mathbf{x}^{\text{in}}].$$

Next we derive the optimal \mathbf{W}_{ch} and \mathbf{W}_{in} . Substituting the definitions of the covariance matrices into the expected deviation and the second moments of the affine transformations, we have

$$J = 2m - 2\text{tr}(\mathbf{W}_{\text{ch}}^{\text{T}}\check{\mathbf{C}}_{\text{ch,in}}\mathbf{W}_{\text{in}})$$

and

$$\begin{aligned}\mathbb{E}_{\text{p}(\mathbf{x}^{\text{ch}})}[\boldsymbol{\Phi}_{\text{ch}}(\mathbf{x}^{\text{ch}})\boldsymbol{\Phi}_{\text{ch}}(\mathbf{x}^{\text{ch}})^{\text{T}}] &= \mathbf{W}_{\text{ch}}^{\text{T}}\check{\mathbf{C}}_{\text{ch,ch}}\mathbf{W}_{\text{ch}}, \\ \mathbb{E}_{\text{p}(\mathbf{x}^{\text{in}})}[\boldsymbol{\Phi}_{\text{in}}(\mathbf{x}^{\text{in}})\boldsymbol{\Phi}_{\text{in}}(\mathbf{x}^{\text{in}})^{\text{T}}] &= \mathbf{W}_{\text{in}}^{\text{T}}\check{\mathbf{C}}_{\text{in,in}}\mathbf{W}_{\text{in}}.\end{aligned}$$

Then, by omitting the constants, the problem for finding the optimal \mathbf{W}_{ch} and \mathbf{W}_{in} is reduced to the following optimization problem:

$$\begin{aligned}\text{maximize} &\quad \text{tr}(\mathbf{W}_{\text{ch}}^{\text{T}}\check{\mathbf{C}}_{\text{ch,in}}\mathbf{W}_{\text{in}}) \\ \text{wrt} &\quad \mathbf{W}_{\text{ch}} \in \mathbb{R}^{\text{p}_{\text{ch}} \times m}, \mathbf{W}_{\text{in}} \in \mathbb{R}^{\text{p}_{\text{in}} \times m}, \\ \text{subject to} &\quad \mathbf{W}_{\text{ch}}^{\text{T}}\check{\mathbf{C}}_{\text{ch,ch}}\mathbf{W}_{\text{ch}} = \mathbf{W}_{\text{in}}^{\text{T}}\check{\mathbf{C}}_{\text{in,in}}\mathbf{W}_{\text{in}} = \mathbf{I}_m.\end{aligned}$$

From Theorem 3.1, the optimization problem is solved by generalized eigendecomposition (3.9) previously described. Hence, the given algorithm finds a minimizer of the optimization problem.

Now we present the proof of Theorem 3.2.

Proof. Observe that if we substitute the probabilistic distribution $p(\mathbf{x}^{\text{ch}}, \mathbf{x}^{\text{in}})$ to the empirical distribution $q(\mathbf{x}^{\text{ch}}, \mathbf{x}^{\text{in}})$ defined in the main text, the first moments and the second moments are expressed as

$$\begin{aligned} \mathbb{E}_{p(\mathbf{x}^{\text{ch}})}[\mathbf{x}^{\text{ch}}] &= \mathbf{X}_{\text{ch}}\mathbf{v}, & \mathbb{E}_{p(\mathbf{x}^{\text{in}})}[\mathbf{x}^{\text{in}}] &= \mathbf{X}_{\text{in}}\mathbf{v}, \\ \check{\mathbf{C}}_{\text{ch,ch}} &= \mathbf{X}_{\text{ch}}\mathbf{D}_{\text{v}}\mathbf{X}_{\text{ch}}^{\text{T}} - \boldsymbol{\mu}_{\text{ch}}\boldsymbol{\mu}_{\text{ch}}^{\text{T}}, & \check{\mathbf{C}}_{\text{ch,in}} &= \mathbf{X}_{\text{ch}}\mathbf{D}_{\text{v}}\mathbf{X}_{\text{in}}^{\text{T}} - \boldsymbol{\mu}_{\text{ch}}\boldsymbol{\mu}_{\text{in}}^{\text{T}}, \\ \check{\mathbf{C}}_{\text{in,ch}} &= \check{\mathbf{C}}_{\text{ch,in}}^{\text{T}}, & \check{\mathbf{C}}_{\text{in,in}} &= \mathbf{X}_{\text{in}}\mathbf{D}_{\text{v}}\mathbf{X}_{\text{in}}^{\text{T}} - \boldsymbol{\mu}_{\text{in}}\boldsymbol{\mu}_{\text{in}}^{\text{T}}, \end{aligned}$$

which implies that the optimization algorithm given in the result above is equivalent to Algorithm 3.1 in this case. Thus, Theorem 3.2 is established. \square

3.5.3 Kernel Weighted CCA

Suppose we are given n drug samples. The kernel matrices of the chemical and interaction kernels $\mathbf{K}_{\text{ch}} \in \mathbb{S}^{n \times n}$ and $\mathbf{K}_{\text{in}} \in \mathbb{S}^{n \times n}$ are defined so that their respective entries are the values of the kernel functions, as given by

$$\mathbf{K}_{i,j}^{\text{ch}} = \mathbf{K}_{\text{ch}}(\mathbf{x}_i^{\text{ch}}, \mathbf{x}_j^{\text{ch}}), \quad \mathbf{K}_{i,j}^{\text{in}} = \mathbf{K}_{\text{in}}(\mathbf{x}_i^{\text{in}}, \mathbf{x}_j^{\text{in}}), \quad \forall i, j \in \mathbb{N}_n.$$

The kernel empirical mapping of the two information sources are defined as

$$\begin{aligned} \mathbf{k}_{\text{ch}}(\mathbf{x}^{\text{ch}}) &\equiv [\mathbf{K}_{\text{ch}}(\mathbf{x}_1^{\text{ch}}, \mathbf{x}^{\text{ch}}), \mathbf{K}_{\text{ch}}(\mathbf{x}_2^{\text{ch}}, \mathbf{x}^{\text{ch}}), \dots, \mathbf{K}_{\text{ch}}(\mathbf{x}_n^{\text{ch}}, \mathbf{x}^{\text{ch}})]^{\text{T}}, \\ \mathbf{k}_{\text{in}}(\mathbf{x}^{\text{in}}) &\equiv [\mathbf{K}_{\text{in}}(\mathbf{x}_1^{\text{in}}, \mathbf{x}^{\text{in}}), \mathbf{K}_{\text{in}}(\mathbf{x}_2^{\text{in}}, \mathbf{x}^{\text{in}}), \dots, \mathbf{K}_{\text{in}}(\mathbf{x}_n^{\text{in}}, \mathbf{x}^{\text{in}})]^{\text{T}}. \end{aligned}$$

KWCCA needs the kernel values among shifted vectors in the kernel Hilbert spaces. The shifted kernels for chemical profiles can be computed as

$$\begin{aligned} \bar{\mathbf{K}}_{\text{ch}}(\mathbf{x}_i^{\text{ch}}, \mathbf{x}_j^{\text{ch}}) &\equiv \mathbf{K}_{\text{ch}}(\mathbf{x}_i^{\text{ch}}, \mathbf{x}_j^{\text{ch}}) - \mathbf{v}^{\text{T}}(\mathbf{k}_{\text{ch}}(\mathbf{x}_i^{\text{ch}}) - \mathbf{k}_{\text{ch}}(\mathbf{x}_j^{\text{ch}})) + \mathbf{v}^{\text{T}}\mathbf{K}_{\text{ch}}\mathbf{v}, \\ \bar{\mathbf{K}}_{\text{ch}}(\mathbf{x}^{\text{ch}}) &\equiv (\mathbf{I}_n - \mathbf{1}_n\mathbf{v}^{\text{T}})(\mathbf{k}_{\text{ch}}(\mathbf{x}^{\text{ch}}) - \mathbf{K}_{\text{ch}}\mathbf{v}), \\ \bar{\mathbf{K}}_{\text{ch}} &\equiv (\mathbf{I}_n - \mathbf{1}_n\mathbf{v}^{\text{T}})\mathbf{K}_{\text{ch}}(\mathbf{I}_n - \mathbf{v}\mathbf{1}_n^{\text{T}}), \end{aligned} \quad (3.10)$$

and $\bar{\mathbf{K}}_{\text{in}}(\mathbf{x}_i^{\text{in}}, \mathbf{x}_j^{\text{in}})$, $\bar{\mathbf{K}}_{\text{in}}(\mathbf{x}^{\text{in}})$, and $\bar{\mathbf{K}}_{\text{in}}$ are defined similarly.

If we define

$$\tilde{\mathbf{K}}_{\text{ch}} \equiv \mathbf{D}_{\text{v}}^{1/2}\bar{\mathbf{K}}_{\text{ch}}\mathbf{D}_{\text{v}}^{1/2} \quad \text{and} \quad \tilde{\mathbf{K}}_{\text{in}} \equiv \mathbf{D}_{\text{v}}^{1/2}\bar{\mathbf{K}}_{\text{in}}\mathbf{D}_{\text{v}}^{1/2},$$

then the expected deviation between the images in \mathbb{R}^m is expressed as

$$\mathbb{E}[\|\boldsymbol{\psi}_{\text{ch}}(\mathbf{x}^{\text{ch}}) - \boldsymbol{\psi}_{\text{in}}(\mathbf{x}^{\text{in}})\|^2] = 2m - 2\text{tr}(\mathbf{A}_{\text{ch}}^{\text{T}}\tilde{\mathbf{K}}_{\text{ch}}\tilde{\mathbf{K}}_{\text{in}}\mathbf{A}_{\text{in}}). \quad (3.11)$$

The second moment matrices can be written as

$$\begin{aligned}\mathbb{E}[\boldsymbol{\psi}_{\text{ch}}(\mathbf{x}^{\text{ch}})\boldsymbol{\psi}_{\text{ch}}(\mathbf{x}^{\text{ch}})^{\top}] &= \mathbf{A}_{\text{ch}}^{\top} \tilde{\mathbf{K}}_{\text{ch}}^2 \mathbf{A}_{\text{ch}}, \\ \mathbb{E}[\boldsymbol{\psi}_{\text{in}}(\mathbf{x}^{\text{in}})\boldsymbol{\psi}_{\text{in}}(\mathbf{x}^{\text{in}})^{\top}] &= \mathbf{A}_{\text{in}}^{\top} \tilde{\mathbf{K}}_{\text{in}}^2 \mathbf{A}_{\text{in}}.\end{aligned}$$

Hence, the algorithm can be reduced to the following maximization problem:

$$\begin{aligned}\text{maximize} \quad & \text{tr}(\mathbf{A}_{\text{ch}}^{\top} \tilde{\mathbf{K}}_{\text{ch}} \tilde{\mathbf{K}}_{\text{in}} \mathbf{A}_{\text{in}}) \\ \text{wrt} \quad & \mathbf{A}_{\text{ch}}, \mathbf{A}_{\text{in}} \in \mathbb{R}^{n \times m}, \\ \text{subject to} \quad & \mathbf{A}_{\text{ch}}^{\top} \tilde{\mathbf{K}}_{\text{ch}}^2 \mathbf{A}_{\text{ch}} = \mathbf{A}_{\text{in}}^{\top} \tilde{\mathbf{K}}_{\text{in}}^2 \mathbf{A}_{\text{in}} = \mathbf{I}_m.\end{aligned}$$

Since the rank of $\mathbf{I}_n - \mathbf{v}\mathbf{1}_n^{\top}$ is $n - 1$, the two shifted kernel matrices, $\tilde{\mathbf{K}}_{\text{ch}}$ and $\tilde{\mathbf{K}}_{\text{in}}$, are always singular. To avoid overfitting, we introduce two regularization terms $\gamma_{\text{ch}}\mathbf{A}_{\text{ch}}^{\top}\mathbf{A}_{\text{ch}}$ and $\gamma_{\text{in}}\mathbf{A}_{\text{in}}^{\top}\mathbf{A}_{\text{in}}$ to scale constraints as

$$\mathbf{A}_{\text{ch}}^{\top} \tilde{\mathbf{K}}_{\text{ch}}^2 \mathbf{A}_{\text{ch}} + \gamma_{\text{ch}}\mathbf{A}_{\text{ch}}^{\top}\mathbf{A}_{\text{ch}} = \mathbf{A}_{\text{in}}^{\top} \tilde{\mathbf{K}}_{\text{in}}^2 \mathbf{A}_{\text{in}} + \gamma_{\text{in}}\mathbf{A}_{\text{in}}^{\top}\mathbf{A}_{\text{in}} = \mathbf{I}_m, \quad (3.12)$$

where γ_{ch} and γ_{in} are constants. The following algorithm finds \mathbf{A}_{ch} and \mathbf{A}_{in} minimizing the expected deviation subject to the regularized constraints.

Algorithm 3.2 (Kernel Weighted CCA). *Solve the following generalized eigendecomposition:*

$$\begin{bmatrix} \mathbf{0}_{n \times n} & \tilde{\mathbf{K}}_{\text{ch}} \tilde{\mathbf{K}}_{\text{in}} \\ \tilde{\mathbf{K}}_{\text{in}} \tilde{\mathbf{K}}_{\text{ch}} & \mathbf{0}_{n \times n} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}^{\text{ch}} \\ \boldsymbol{\alpha}^{\text{in}} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{K}}_{\text{ch}}^2 + \gamma_{\text{ch}}\mathbf{I}_n & \mathbf{0}_{n \times n} \\ \mathbf{0}_{n \times n} & \tilde{\mathbf{K}}_{\text{in}}^2 + \gamma_{\text{in}}\mathbf{I}_n \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}^{\text{ch}} \\ \boldsymbol{\alpha}^{\text{in}} \end{bmatrix}.$$

The matrix $\begin{bmatrix} \mathbf{A}_{\text{ch}} \\ \mathbf{A}_{\text{in}} \end{bmatrix}$ is set so that its h th column is set to the h th major generalized eigenvector.

Theorem 3.3. Algorithm 3.2 yields the parameters of the mapping functions $(\boldsymbol{\psi}_{\text{ch}}, \boldsymbol{\psi}_{\text{in}})$ which minimize the expected squared deviation (3.11) subject to the scaling constraints given in (3.12).

Proof. The problem of minimizing (3.11) subject to (3.12) can be rewritten as

$$\begin{aligned}\text{maximize} \quad & \text{tr}(\mathbf{A}_{\text{ch}}^{\top} \tilde{\mathbf{K}}_{\text{ch}} \tilde{\mathbf{K}}_{\text{in}} \mathbf{A}_{\text{in}}) \\ \text{wrt} \quad & \mathbf{A}_{\text{ch}} \in \mathbb{R}^{n \times m}, \mathbf{A}_{\text{in}} \in \mathbb{R}^{n \times m}, \\ \text{subject to} \quad & \mathbf{A}_{\text{ch}}^{\top} \tilde{\mathbf{K}}_{\text{ch}}^2 \mathbf{A}_{\text{ch}} + \gamma_{\text{ch}}\mathbf{A}_{\text{ch}}^{\top}\mathbf{A}_{\text{ch}} = \mathbf{A}_{\text{in}}^{\top} \tilde{\mathbf{K}}_{\text{in}}^2 \mathbf{A}_{\text{in}} + \gamma_{\text{in}}\mathbf{A}_{\text{in}}^{\top}\mathbf{A}_{\text{in}} = \mathbf{I}_m.\end{aligned}$$

From Theorem 3.1, this optimization problem is solved by the generalized eigendecomposition in Algorithm 3.2. \square

3.5.4 Weighted SVM

Let us denote the protein-ligand interaction table by $\mathbf{Y} \in \{\pm 1, 0\}^{n \times p_{in}}$, where each row represents a ligand, and each column represents a target protein. Each cell in the table \mathbf{Y} takes one of three values, ± 1 and 0 : $+1$ and -1 indicate the existence and the absence of the interaction, respectively, and unknowns are 0 .

SVM learning algorithm finds a classification boundary minimizing the violations for the constraints that training points are kept out of the margin. The weighted SVM employed in this study counts the violations with weights v_j (See Subsections 3.2.4 and 3.3.4) given to each training data.

When the interaction between a ligand i and a target t is predicted, ligands whose interactions with the target t are known are selected as training data. If the index set of the ligands for training is denoted by J_i , then the weighted SVM minimizes

$$\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{j \in J_i} v_j \max(0, 1 - Y_{j,t} f(\mathbf{x}_j^{\text{ch}}; \mathbf{w}, \mathbf{b})),$$

where C is constant. This algorithm is reduced to the classical SVM if all v_j 's are set to be equal in value. The dual problem of the weighted SVM learning algorithm can be derived as a quadratic program with box constraints and a single equality linear constraint, enabling fast learning with kernels.

ENZYME ACTIVE SITE PREDICTION

Enzymes are protein molecules, each of which plays an important role in several metabolic reactions in the body needed to sustain life. These proteins consist of atoms, and can be modeled using 3-dimensional structures such as that shown in Figure 4.1b. In fact, there are many proteins for which the corresponding structures are known, that is, the 3D coordinates of their atoms have been measured and documented. However, among these, many still have unknown functions. Studying enzymes and their structure aids in determining their function in the body. And if their functions should be uncovered, they would be very useful to scientists and pharmacists. Therefore, searching for proteins with specific enzyme function or reaction, for instance, has been an active research theme [6, 15, 19, 31, 43, 47, 36, 37, 77, 79, 84].

Aside from predicting protein and ligand interactions, determining active sites of enzyme proteins are also useful for drug discovery and development. Active sites are localized positions among the atoms that comprise an enzyme molecule responsible for the catalytic reactions. As in Figure 4.1c, these localized regions are very small compared to the entire protein molecule. In spite of this, two enzymes with different overall molecular shape may also have common enzymatic reactions as long as they have active sites with similar forms. Aside from determining their functionality, by analyzing the structures of enzymes and identifying the active sites, drugs can be designed by identifying which substrates or chemical compounds can fit into the active sites to produce some desired reaction or product.

Recent results imply that cases where analogous enzymes share active sites are not rare [22]. Thus, it is necessary to examine explicit local structures of active sites that may possibly reveal enzyme functionality, rather than global structures [50]. There have been several “template-based” methods performing local structure comparison for detecting similar active sites [84, 19, 43, 77, 6, 15, 31, 47, 79]. Using a predefined template, these methods search for occurrences of the active site residue atoms contained in the template within the structure of the target protein. However, there remain some difficulties and questions that need to be challenged in relation to these template-based methods. One complication is that prediction accuracy may be dependent on the atom type and number of atoms in the template. Determining which atoms in the catalytic site should be included is often very difficult, and even experts on enzyme structure and function might resolve to trial and error to create the best template. In addition to the aforementioned, template-based methods not only

yield many site matches, but also a huge number of mismatches. Thus, the possibility of reducing the number of mismatches is always contested.

Two well-known template-based methods are TESS [84] and JESS [6]. While TESS uses geometric hashing to search for local structures [84] and JESS uses kd-tree data structures [6], both methods compute the mean square deviation for the search results. In this work, we give a weight coefficient to each atom to employ the weighted squared deviation, and developed a new machine learning algorithm for determining the weights. While Kato and Nagano have previously proposed another algorithm based on a linear program [36], the new algorithm presented in this chapter applies a more natural formulation in terms of machine learning. The mainstream of modern machine learning methods involves minimizing an objective function defined by the sum of a loss function and a regularization function to determine the values of the model parameters, which are the weights for the atoms for the task at hand. The previous method [36], however, is different from this approach, from which we pose some questions: Is the active site prediction task so particular that a specialized method which does not follow the typical modern approach is necessary? Or, is it possible to replace the learning algorithm with one developed in a more natural fashion?

One contribution of this work is to show that comparable prediction performance can be achieved even with algorithms that adopt modern machine learning techniques. A regularization function, which is combined with a loss function to form the objective function is similar, in some sense, to the prior distribution in maximum a posteriori (MAP) estimation. The regularization function is chosen so that the model parameter is set to a non-informative point when minimizing only the regularization function. In our task, the non-informative parameters are equal weights. To incorporate the differences with the equal weights in the regularization, this work employs two Bregman divergences [40]: the squared Euclidean distance and the Kullback-Leibler divergence. The learning algorithm with either of the two regularization functions is an instance of Bregman divergence regularized machine (BDRM) [40], and its framework can be used for learning.

The rest of this chapter is organized as follows. The succeeding section reviews a typical procedure for active site search, which is also employed in this study. The procedure for active site search quantifies how much the candidate local sites are deviated from the template. For the quantification of the deviation, weight coefficients are adopted, and algorithms that automatically determine the weight coefficients are presented in Section 4.2. Experimental results are reported in Section 4.3.

4.1 ACTIVE SITE SEARCH METHOD

This work addresses the active site search problem described in Figure 4.1. The input in the active site search problem is composed of the template describing an active site of interest and function-unknown protein structures. The task is to identify the active sites if they exist; otherwise, the search algorithm will output “Not Found.”

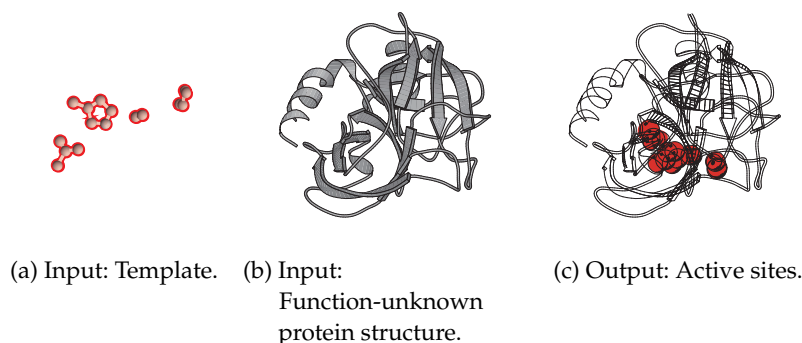


Figure 4.1: **The active site search problem.** In this problem, the inputs are a template and a function-unknown protein structure, and the output is a set of active sites in the protein structure. (a) A template is a set of atoms in an active site of interest. (b) The 3D coordinates of the atoms and the atom types are given for the function-unknown protein structure. (c) Estimated active sites have a similar shape to the template.

The typical procedure for the active site search problem consists of two stages: Local Site Search stage and Deviation Computation stage. In the Local Site Search stage, an algorithm such as TESS [84] or JESS [6] is used to enumerate local sites whose shapes are possibly similar to the template. These local sites also contain the same atoms corresponding to the ones in the template.

Conventionally, the mean square deviation between the template and each candidate of local sites is computed in the Deviation Computation stage. Suppose a template has n atoms. Then each local site candidate also contains exactly n atoms. These two sets of n atoms have a one-to-one correspondence, as depicted in Figure 4.2.

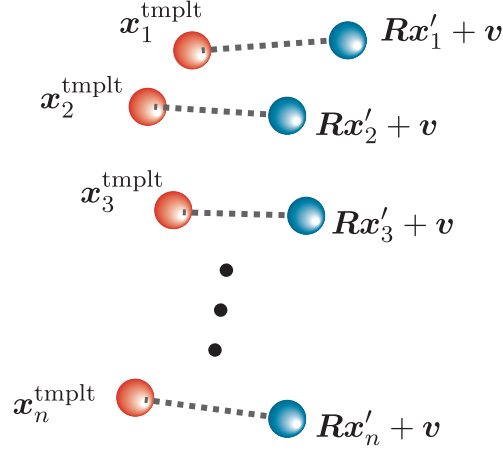


Figure 4.2: **Comparison between a local site and a template.** Basically, each atom in a local site is of the same type as the corresponding atom in the template. For example, a carbon atom in the local site corresponds to a carbon atom in the template, and an oxygen atom in the local site to an oxygen atom in the template. Atoms in local sites outputted from the local site search stage have a one-to-one correspondence with the atoms in the template. For each atom i in the template located at $\mathbf{x}_i^{\text{tmplt}} \in \mathbb{R}^3$, the corresponding atom in a local site candidate located at $\mathbf{x}'_i \in \mathbb{R}^3$ is rotated and shifted, and the deviation $\|\mathbf{x}_i^{\text{tmplt}} - \mathbf{R}\mathbf{x}'_i - \mathbf{v}\|$ is assessed.

Essentially, the type of the i th atom in a local site candidate is the same as the type of the i th atom in the template. Let us denote by $\mathbf{x}_i^{\text{tmplt}} \in \mathbb{R}^3$ the 3D coordinates of the i th atom in the template, and by $\mathbf{x}'_i \in \mathbb{R}^3$ the 3D coordinates of the i th atom in the corresponding local site candidate. Let

$$\begin{aligned} \mathbf{X}^{\text{tmplt}} &:= [\mathbf{x}_1^{\text{tmplt}}, \mathbf{x}_2^{\text{tmplt}}, \dots, \mathbf{x}_n^{\text{tmplt}}] \in \mathbb{R}^{3 \times n} \quad \text{and} \\ \mathbf{X}' &:= [\mathbf{x}'_1, \mathbf{x}'_2, \dots, \mathbf{x}'_n] \in \mathbb{R}^{3 \times n}. \end{aligned}$$

In case of unweighted atoms, the mean square deviation is given by

$$D_{\text{unwei}}(\mathbf{X}', \mathbf{X}^{\text{tmplt}}; \mathbf{R}, \mathbf{v}) := \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i^{\text{tmplt}} - (\mathbf{R}\mathbf{x}'_i + \mathbf{v})\|^2$$

where $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ and $\mathbf{v} \in \mathbb{R}^3$ are parameters for rigid-body transformation; \mathbf{R} and \mathbf{v} represent a rotation matrix and a translation vector, respectively, with the rotation matrix satisfying $\mathbf{R}^T \mathbf{R} = \mathbf{I}$. The two parameters (\mathbf{R}, \mathbf{v}) that minimize the mean square deviation are used, and the minimizer can be found in $O(1)$ computation.

If we give weight coefficients $\mathbf{w} = [w_1, w_2, \dots, w_n]^T$ to n atoms, the weighted square deviation is given by the linear combination

$$D(\mathbf{X}', \mathbf{X}^{\text{tplt}}; \mathbf{w}, \mathbf{R}, \mathbf{v}) := \sum_{i=1}^n w_i \|\mathbf{x}_i^{\text{tplt}} - (\mathbf{R}\mathbf{x}'_i + \mathbf{v})\|^2, \quad (4.1)$$

which contains the mean square deviation with the weights $\mathbf{w} = \mathbf{1}/n$.

4.2 LEARNING ALGORITHM

In this work, the weighted square deviation (4.1) is employed, and the weight coefficients \mathbf{w} are determined by learning from a training data set.

A data set for training (Figure 4.3) is constructed as follows.

1. Collect the protein structures of enzymes whose active sites are known.
2. Apply the TESS algorithm to the collected protein structures to get the local sites that are possibly similar to the template.
3. For each local site obtained in the previous step, give a class label $y_i \in \{\pm 1\}$. If the local site is actually an active site represented by the template, set $y_i = +1$; otherwise, $y_i = -1$.













Site 1	Site 2	• • •	Site ℓ
$\mathbf{x}_{1,1}$ 	$\mathbf{x}_{1,2}$ 		 $\mathbf{x}_{1,\ell}$
$\mathbf{x}_{2,1}$ 	$\mathbf{x}_{2,2}$ 		$\mathbf{x}_{2,\ell}$ 
 $\mathbf{x}_{3,1}$	$\mathbf{x}_{3,2}$ 	• • •	 $\mathbf{x}_{3,\ell}$
•	•		•
•	•		•
$\mathbf{x}_{n,1}$ 	$\mathbf{x}_{n,2}$ 		$\mathbf{x}_{n,\ell}$ 
y_1	y_2	• • •	y_ℓ

Figure 4.3: **Dataset for training.** The proposed method weights atoms in a template, and the weights are determined via machine learning using a training dataset. Protein structures whose active sites are known are used for the construction of a training dataset. For these protein structures, TESS is used to search local sites similar to the template. Class label $y_i = +1$ is assigned to local sites which are true active sites, and class label $y_i = -1$ is given to non-active sites.

The number of atoms in each local site is equal to the number of atoms in the template, n . Moreover, each atom in the local site bijectively corresponds to one of the n atoms in the template. Suppose that ℓ local sites are obtained. Then the 3D coordinates of the k th local site are expressed as $\mathbf{x}_{1,k}, \mathbf{x}_{2,k}, \dots, \mathbf{x}_{n,k} \in \mathbb{R}^3$, and for future reference, we define

$$\mathbf{X}^{(k)} := [\mathbf{x}_{1,k}, \mathbf{x}_{2,k}, \dots, \mathbf{x}_{n,k}].$$

This work employs the regularized loss minimization approach using a regularization function and a loss function. The regularized loss minimization is similar to the MAP estimation [8] that maximizes the product of the likelihood function and the prior density function, where the likelihood function corresponds to the loss function and the prior density function to the regularization function. The loss function is designed so that, for positive sites, a nonzero loss is given if the weighted square deviation is smaller than a threshold θ , and for negative sites, a nonzero loss is given if the weighted square deviation is greater than the threshold θ . The definition of the loss function is given by

$$\text{loss}(\mathbf{w}, \theta) := \sum_{k=1}^{\ell} c_k (y_k (\theta - D(\mathbf{X}^{(k)}, \mathbf{X}^{\text{tmplt}}; \mathbf{w}, \mathbf{R}^{(k)}, \mathbf{v}^{(k)})))_+^2$$

where c_k is a positive constant. The rigid-body parameters $(\mathbf{R}^{(k)}, \mathbf{v}^{(k)})$ are set to the minimizers of $D(\mathbf{X}^{(k)}, \mathbf{X}^{\text{tmplt}}; \mathbf{1}/n, \mathbf{R}^{(k)}, \mathbf{v}^{(k)})$, and the operator $(\cdot)_+$ is defined as $(x)_+ = \max(x, 0)$, $\forall x \in \mathbb{R}$.

The regularization function is designed as follows. The prior distribution in the MAP estimation is often chosen so that the estimated model is non-informative if the training data is not given. Similarly, the regularization function employed in this study is minimized when no data is provided. In this work, the weight coefficients are the model parameters, and the most non-informative parameters are equal weights:

$$w_1 = w_2 = \dots = w_n = 1/n$$

Hence, we constructed the regularization function so that it is minimized at

$$\mathbf{w}_0 = [w_{1,0}, w_{2,0}, \dots, w_{n,0}]^T := \mathbf{1}_n/n.$$

The following two regularization functions are employed in this analysis:

- ℓ_2 regularization

$$r_2(\mathbf{w}) := \frac{1}{2} \|\mathbf{w} - \mathbf{w}_0\|^2.$$

- *KL (Kullback-Leibler) regularization*

$$r_{\text{KL}}(\mathbf{w}) := \sum_{i=1}^n w_i \log \frac{w_i}{w_{i,0}} - w_i + w_{i,0}.$$

As a result, we obtain the following minimization problem:

$$\begin{aligned} \text{minimize} \quad & r(\mathbf{w}) + \frac{\lambda}{2}(\theta - \theta_0)^2 + \text{loss}(\mathbf{w}, \theta), \\ \text{wrt} \quad & \mathbf{w} \in \mathbb{R}_+^n, \theta \in \mathbb{R}_+, \end{aligned} \quad (4.2)$$

where either ℓ_2 regularization or KL regularization is used as the regularization function $r(\cdot)$, λ is a positive constant, and θ_0 is set to 1 in the experiments described in the succeeding section.

4.2.1 Optimization Algorithm

Both ℓ_2 regularization function and KL regularization function are examples of Bregman divergence, which is defined as follows.

Let φ be a seed function which is continuously-differentiable, real-valued, and strictly convex. A *Bregman divergence* $D_\varphi : \text{dom}\varphi \times \text{ri}(\text{dom}\varphi) \rightarrow [0, +\infty)$ is constructed with φ as

$$D_\varphi(\mathbf{x}; \mathbf{y}) := \varphi(\mathbf{x}) - \varphi(\mathbf{y}) - \langle \mathbf{x} - \mathbf{y}, \nabla \varphi(\mathbf{y}) \rangle,$$

where $\text{dom}\varphi$ is the domain of φ , and $\text{ri}(\text{dom}\varphi)$ is the relative interior of $\text{dom}\varphi$. For example, the seed function $\varphi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$ yields $D_\varphi(\mathbf{w}; \mathbf{w}_0) = r_2(\mathbf{w})$, and $\varphi(\mathbf{w}) = \sum_{i=1}^n w_i \log w_i$ generates $D_\varphi(\mathbf{w}; \mathbf{w}_0) = r_{\text{KL}}(\mathbf{w})$.

The learning algorithm from problem (4.2) is an instance of BDRM [40] with either of the two regularization functions, $r_2(\cdot)$ and $r_{\text{KL}}(\cdot)$, but with an additional constraint that the weight coefficients have to be non-negative when using $r_2(\cdot)$. The resulting algorithms are shown in Algorithm 4.1 for ℓ_2 regularization and in Algorithm 4.2 for KL regularization, where $\mathbf{a}_k = [a_{1,k}, a_{2,k}, \dots, a_{n,k}]^\top$ is an n -dimensional vector whose entry is defined as

$$a_{i,k} := y_k \|\mathbf{x}_i^{\text{tmp}t} - (\mathbf{R}\mathbf{x}'_i + \mathbf{v})\|^2, \quad i = 1, 2, \dots, n.$$

Algorithm 4.1 ℓ_2 Regularized Learning Algorithm

- 1: $(\boldsymbol{\alpha}, \boldsymbol{\beta}) := (\mathbf{0}_\ell, \mathbf{0}_d); (\mathbf{w}, \theta) := (\mathbf{w}_0, \theta_0);$
 - 2: **for** $t = 1, 2, \dots$ **do**
 - 3: **for** $k = 1, 2, \dots, \ell$ **do**
 - 4: $\bar{\mathbf{w}} := \mathbf{w} + \alpha_k \mathbf{a}_k; \quad \bar{\theta} := \theta - \frac{1}{\lambda} \alpha_k y_k;$
 - 5: $\alpha_k := \frac{(\langle \mathbf{a}_k, \bar{\mathbf{w}} \rangle - y_k \bar{\theta})_+}{1/\lambda + 1/c_k + \|\mathbf{a}_k\|^2};$
 - 6: $\mathbf{w}_{-1/2} := \bar{\mathbf{w}} - \alpha_k \mathbf{a}_k; \quad \theta := \bar{\theta} + \frac{1}{\lambda} \alpha_k y_k;$
 - 7: $\mathbf{w} := (\mathbf{w}_{-1/2} - \boldsymbol{\beta})_+; \quad \boldsymbol{\beta} := \mathbf{w}_{-1/2}^t - \boldsymbol{\beta} - \mathbf{w};$
 - 8: **end for**
 - 9: **end for**
-

Algorithm 4.2 KL Regularized Learning Algorithm.

```

1:  $\alpha := 0_\ell; \mathbf{v} := \log(\mathbf{w}_0); \theta := \theta_0;$ 
2: for  $t = 1, 2, \dots$  do
3:   for  $k = 1, 2, \dots, \ell$  do
4:      $\bar{\mathbf{v}} := \mathbf{v} + \alpha_k \mathbf{a}_k;$ 
5:      $\bar{\theta} := \theta - \frac{1}{\lambda} \alpha_k y_k;$ 
6:     if  $\langle \mathbf{a}_k, \exp(\bar{\mathbf{v}}) \rangle \leq y_k \bar{\theta}$  then
7:        $\alpha_k := 0;$ 
8:     else
9:       Find positive  $\alpha_k$  satisfying a nonlinear equation
           
$$\langle \mathbf{a}_k, \exp(\bar{\mathbf{v}} - \alpha_k \mathbf{a}_k) \rangle = y_k \bar{\theta} + (1/\lambda + 1/c_k) \alpha_k.$$

10:    end if
11:     $\mathbf{v} := \bar{\mathbf{v}} - \alpha_k \mathbf{a}_k; \quad \theta := \bar{\theta} + \frac{1}{\lambda} \alpha_k y_k;$ 
12:  end for
13: end for

```

4.3 EXPERIMENTS AND RESULTS

To investigate the performance of our methods, experiments were conducted over PDB datasets. We used PDB entries that are annotated in EzCatDB [53, 54], which provides information of enzyme classifications and the amino acid residues of active sites. A total of 31 enzyme classes having a sufficient number of proteins were chosen, and these templates were generated from the amino acid residues in their active sites. The resulting dataset contains 5,538 protein structures, some of them containing multiple active sites.

Each amino acid in the active site is classified into one of four types: catalytic-site residues, cofactor binding site residues, modified residues, and mainchain catalytic residues. For cofactor binding site residues, all atoms are included in the template, whereas atoms from the sidechains of residues were included in the template for modified residues and catalytic-site residues. For mainchain catalytic residues, only mainchain atoms were included in the template.

Two proposed methods, ℓ_2 -regularized and KL-regularized BDRMs (abbreviated to L2 and KL), were examined. These methods were compared with the (unweighted) mean square deviation (UD) and Kato and Nagano’s method (KND) [36].

Half of the 5,538 protein structures in the data set were chosen randomly and used for training, and the rest is used for testing. This is repeated three times to get three divisions. Sensitivity at 95% specificity and AUC values were obtained from each of testing data set. Sensitivity is defined as the ratio of true positive sites that are correctly identified whereas specificity is the ratio of true negative sites that are correctly identified. The AUC value is the area under

the ROC curve, which plots the ratio of true positives against the ratio of false positives for different possible thresholds. Since 31 templates are used in three divisions, 93 sensitivities and 93 AUC scores were obtained. The averages are reported hereinafter.

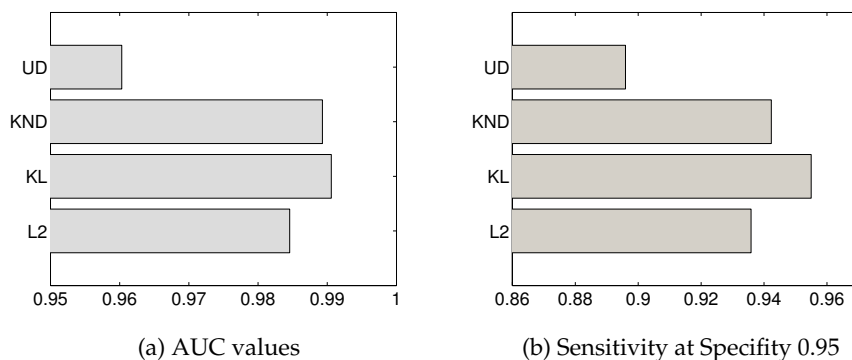


Figure 4.4: Average performance of all methods.

Figures 4.4a and 4.4b illustrate the average AUC and sensitivity scores for all methods, respectively. For both evaluation measures, KL achieved the highest performance, and the second highest was KND. The performance of UD was lower than any weighted square deviation method. To detect the statistical difference, one-sample t-test was performed. P-value for the difference of AUC scores between L2 and UD was $4.88 \cdot 10^{-4}$ and P-value for KL and UD was $1.30 \cdot 10^{-4}$. For sensitivity, P-values for the difference of UD from L2 and KL were $2.68 \cdot 10^{-3}$ and $1.26 \cdot 10^{-3}$, respectively. These facts imply that the weighted square deviations are significantly better than UD. The statistical test also allows us to conclude that the performance of L2 is not significantly worse than KND, since the P-values for the AUC score and sensitivity, which are 0.137 and 0.204, respectively, are not small enough. P-values between KL and KND are 0.0427 and 0.0810 for AUC score and sensitivity, respectively, making it hard to insist that KL is significantly better than KND. However, these results suggest that KL and L2 achieves predictive performances that are comparable to KND.

4.4 SUMMARY

In this chapter, we presented a new machine learning algorithm that determines the values of weight coefficients for atoms in the template used for enzyme active site search. An existing state-of-the-art method [36] has already proposed introducing weights for atoms and determining the weights automatically, although their algorithm

was not formulated in the framework of regularized loss minimization which is a recent trend in the machine learning community [29]. Experimental results showed that the proposed methods possess comparable prediction performance to the existing state-of-the-art method, suggesting that there is no more need to resort to the learning algorithm specialized to the enzyme active site search problem at the sacrifice of the typical machine learning framework.

Among currently trending fields, research efforts particularly related to biometrics and brain-computer interface (BCI) have been aimed at modeling data either as a low-dimensional subspace or a sequence of vectors. There have been studies in these areas dedicated to algorithms for such type of input [5, 9, 17, 32, 60, 86]. This may be induced by the nature of the data, which is commonly time series, such as electroencephalography (EEG) signals collected while subjects perform motor tasks or during induction of visual stimuli. EEG data is generated by placing several sensors accordingly on the head of the subject as shown in Figure 5.1, and each sensor records neural activity depicted by the signals. The vector sequence illustrated in Figure 5.2 corresponds to the signals collected from all sensors.

Appropriate data representation has been considered as one of the most important challenges in dealing with classification tasks. Vector form may be the simplest and most common representation of samples in existing literatures, especially when using popular techniques such as support vector machines and kernel methods. However, this may not be the best representation to encompass significant, if not all, attributes and information useful for discrimination. To address this problem, new modes of data representation are constantly being explored [18, 25, 27, 35, 41, 44, 45, 46, 48, 49, 55, 59, 75, 76, 81, 83]. Along with this, various feature extraction techniques and discrimination methods are also being investigated, and several studies have proven kernel methods to be a flexible technique in supporting various data structures, such as graphs [35, 45, 46, 55, 75, 83], strings [48, 49, 59, 81], and even subspaces and sets of vectors [18, 25, 27, 41, 76].

Kernel-based algorithms [16, 72, 74] have become prevalent in recent years. Aside from yielding promising results, they have been proven suitable for problems with complex data patterns, and those that suffer from the curse of dimensionality. Along with SVM, kernel methods have become one of the most valuable tools in machine learning.

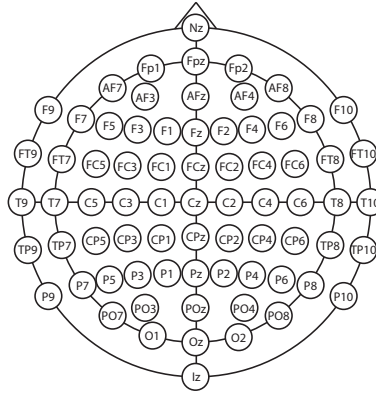


Figure 5.1: Sample position map of EEG sensors.

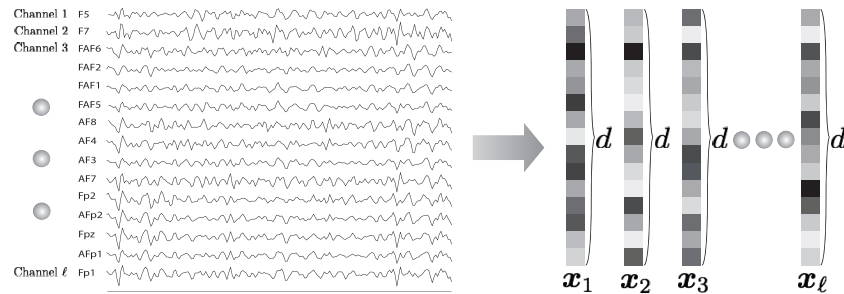


Figure 5.2: **EEG signals as a sequence of vectors.** For BCI, EEG signals are recorded over a certain time interval using several channels or sensors. Each vector in the sequence corresponds to a channel used in the procedure, and vector entry represents an instantaneous signal intensity. The vector sequence is usually concatenated to represent the vector set input.

In this chapter, we focus on data represented as sets of vectors. Different algorithms have been formulated in such a way that data are approximated by low-dimensional linear subspaces [20, 26, 27, 41, 70, 76, 88, 89]. However, as previously pointed out [26], the task of appropriately handling data has become an issue, such as inconsistency in strategy when feature extraction is done in a Euclidean space while non-Euclidean metrics are used. For this purpose, they proposed a unified framework for subspace-based approaches by formulating the problem on the Grassmann manifold, a space of linear subspaces with a fixed dimension. On the other hand, these methods involve dimension reduction, and even with the use of the usual dimension reduction techniques such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), there is always a possibility of information loss. This makes the selection of the subspace dimension a crucial step. Furthermore,

methods such as PCA and LDA usually employ eigendecomposition, and hence, may be very time consuming especially for high dimensional data.

With the aforementioned issues in mind, the goal of this work is to examine a kernel function, which we refer to as the mean polynomial kernel, that can retain data information while being computationally inexpensive. Also, as a more general approach than kernels for subspaces, we treat data as a common collection of vectors, instead of a linear subspace. The kernel is invariant of the permutation order of the vectors in the set. In addition, we present an interesting relationship between this kernel and the Projection kernel, which is a known Grassmann kernel. We give emphasis to BCI applications posed as a binary classification problem, which are of particular interest due to their practicality in various areas, biometrics and cognitive training and improvement, among others.

5.1 PRELIMINARIES

Consider a set of data $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_\ell \in \mathbb{R}^d$, where ℓ is the number of data points. Let us denote the h th entry in the i th data point \mathbf{x}_i by $x_{h,i}$. A sample statistic,

$$\frac{1}{\ell} \sum_{i=1}^{\ell} \prod_{h=1}^d x_{h,i}^{p_h},$$

is said to be the q th order moment if the d -dimensional vector $\mathbf{p} \in (\mathbb{N} \cup \{0\})^d$ satisfies $p_1 + p_2 + \dots + p_d = q$. The *uncentered covariance matrix* defined by

$$\frac{1}{\ell} \sum_{i=1}^{\ell} \mathbf{x}_i \mathbf{x}_i^T$$

contains all the second order moments. Indeed, the (h, k) th entry in the uncentered covariance matrix is the the second order moment with $\mathbf{p} = \mathbf{e}_h + \mathbf{e}_k$, where \mathbf{e}_h is a unit vector whose h th entry is one and the rest of the entries are zero. Let $\bar{\mathbf{x}} = [\bar{x}_1, \bar{x}_2, \dots, \bar{x}_d]^T$ be the mean vector of the data points. With the d -dimensional vector \mathbf{p} satisfying $p_1 + p_2 + \dots + p_d = q$, the q th order central moment is defined as

$$\frac{1}{\ell} \sum_{i=1}^{\ell} \prod_{h=1}^d (x_{h,i} - \bar{x}_h)^{p_h}.$$

Every second order central moment is included in the *central covariance matrix*

$$\frac{1}{\ell} \sum_{i=1}^{\ell} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T,$$

which is usually referred to simply as the *covariance matrix*.

For succeeding sections, we refer to a matrix \mathbf{U} as *orthonormal* if $\mathbf{U}^T \mathbf{U} = \mathbf{I}$, and define the *vectorization* of an $m \times n$ matrix \mathbf{A} as the column vector $\text{vec}(\mathbf{A}) = [a_{11}, a_{12}, \dots, a_{1n}, \dots, a_{m1}, a_{m2}, \dots, a_{mn}]^T$.

5.2 GRASSMANN KERNELS AND RELATED METHODS

We give a concise discussion of the Grassmann kernels [27, 26, 82], their analogy with the mean polynomial kernel, and some related methods.

A Grassmann manifold, or Grassmannian, is defined as a set of linear subspaces with a fixed number of dimensions, say, m . Several metrics used in literatures have been specified in this manifold, mostly incorporating principal angles or angles between subspaces in their characterization [20, 26, 27, 70, 76, 82, 88, 89]. Moreover, kernels over these manifolds have also been introduced. In particular, we are interested in the following kernels:

Definition 5.1. Let \mathbf{U}_x and \mathbf{U}_y be orthonormal matrices whose columns are bases of linear subspaces. The *Projection kernel* is defined as

$$k_{\text{PROJ}}(\mathbf{U}_x, \mathbf{U}_y) = \|\mathbf{U}_x^T \mathbf{U}_y\|_F^2,$$

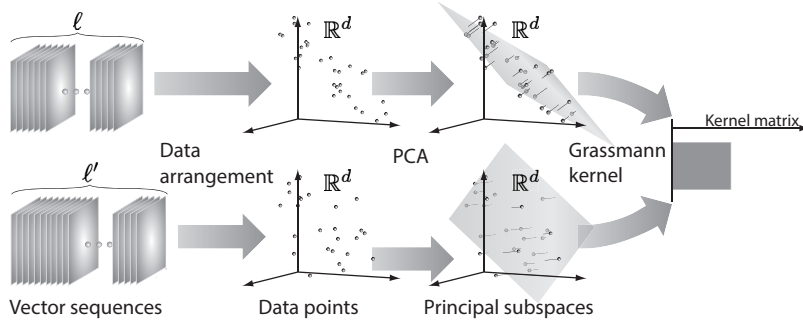
where $\|\cdot\|_F$ denotes the Frobenius norm, and the *Binet-Cauchy kernel* is given by

$$k_{\text{BC}}(\mathbf{U}_x, \mathbf{U}_y) = (\det \mathbf{U}_x^T \mathbf{U}_y)^2 = \det \mathbf{U}_x^T \mathbf{U}_y \mathbf{U}_y^T \mathbf{U}_x.$$

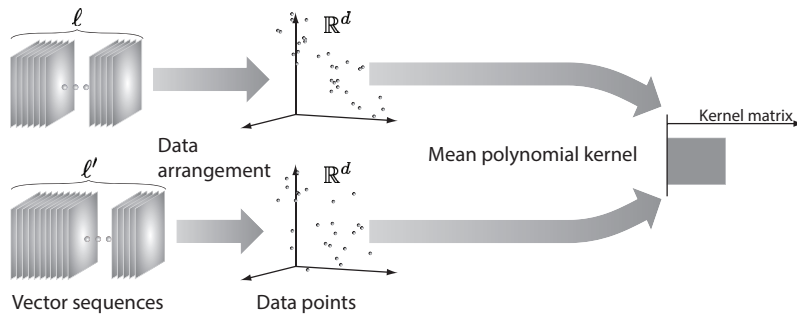
Many existing problems can be realized on nonlinear manifolds such as the Grassmannian. This being said, various methods in the Grassmannian setting have been proposed. One such technique is the use of Grassmann kernels in conjunction with support vector machines (GK-SVM) [76]. This approach entails the computation of kernel matrices, which then proceed as the SVM input. Analogously, the mean polynomial kernel given in Section 5.3 is applied in this manner when SVM is the classifier. Figure 5.3 gives a general illustration of the flow of computation of the Grassmann kernels and the mean polynomial kernel, and also highlights the difference between the two kernels.

Another comparable method is the Grassmann Distance Mutual Subspace Method (GD-MSM) [76]. This technique integrates the Grassmann metrics in the Mutual Subspace Method (MSM) [89]. Furthermore, the task of subspace classification can be approached in two ways. The first one, which is referred to as the *subject-wise dictionary*, is done by assuming that one subject or object corresponds to one principal subspace. During the training stage, the total of principal subspaces calculated is the same as the number of subjects. These serve as the bases to which the unlabeled principal subspaces of test subjects are compared to, and the subspace with the minimal Grassmann distance from the unlabeled subspace is determined. The second approach is done by assuming one principal subspace per class. The principal subspaces, which in this case is referred to as the *class-wise dictionary*, are derived from each class among the training

data. This being said, we have only two principal subspaces in the case of binary classification, regardless of the number of subjects. In the testing stage, unlabeled principal subspaces are classified according to which subspace they are closer to in terms of metric.



(a) Flow for Grassmann kernels.



(b) Flow for Mean Polynomial kernel.

Figure 5.3: **Flow of methodology for computing the kernel value for the Grassmann kernels and the mean polynomial kernel.** Grassmann kernels are defined on a Grassmann manifold which is a set of linear subspaces. When employing these kernels, each vector sequence, represented by a set of data points on space, is approximated by a principal subspace obtained via PCA. However, this poses a threat of some degree of information loss, and is more likely to consume more time due to eigendecomposition. The mean polynomial kernel, on the other hand, can be directly applied to compute the kernel value between the sets of data points. It can avoid information loss while being more time efficient.

The score function can be considered for the two aforementioned mutual subspace methods. The SVM score, mentioned in Section 2.3, can serve as a confidence level. Namely, a higher score may provide higher certainty of assigning the data to the positive class. For the *class-wise dictionary*, the difference between the distance to the subspace of the negative class, d_- , and the distance to the subspace of the positive class, d_+ , represents how confidently unknown labels are classified

as positive. Hence, we define the score function as $d_- - d_+$. For the *subject-wise dictionary*, we define the score function by the difference between the minimal distance to negative class subspaces and the minimal distance to the positive class subspaces.

5.3 MEAN POLYNOMIAL KERNEL

In this section, we discuss the details of the *mean polynomial kernel*, which can be directly applied to data in the form of vector sets.

Consider two sets of vectors $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^\ell$ and $\mathcal{Y} = \{\mathbf{y}_j\}_{j=1}^{\ell'}$, where $\mathbf{x}_i, \mathbf{y}_j \in \mathbb{R}^d$. To define a kernel for such types of data, we introduce a notation of a set of vector sequences as

$$\mathcal{S} = \{\{\mathbf{z}_i\}_{i=1}^n \mid n \in \mathbb{N} \text{ and } \forall i \in \mathbb{N}_n, \mathbf{z}_i \in \mathbb{R}^d\},$$

where \mathbb{N} is the set of natural numbers, and $\mathbb{N}_n = \{i \in \mathbb{N} \mid i \leq n\}$, such that \mathcal{S} is the input domain for the kernel defined as follows.

Definition 5.2. Let $k_q : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$ such that

$$k_q(\mathcal{X}, \mathcal{Y}) = \frac{1}{\ell \ell'} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell'} \langle \mathbf{x}_i, \mathbf{y}_j \rangle^q,$$

where $\mathcal{X}, \mathcal{Y} \in \mathcal{S}$ and $q \in \mathbb{N}$. We shall refer to k_q as the *qth order mean polynomial kernel*.

It can be shown that this kernel is a special case of the multi-instance kernels [21] when instances involve linear kernels or polynomial kernels with constant $c = 0$. With regards to its characterization, we can easily confirm that for the case $q = 2$, the covariance matrix is directly used as a feature vector. For instance, consider two matrices, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_\ell]$ and $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_{\ell'}]$, for the set of vectors \mathcal{X} and \mathcal{Y} , respectively. Then their respective uncentered covariance matrices are given by

$$\mathbf{\Sigma}_x = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbf{x}_i \mathbf{x}_i^\top \quad \text{and} \quad \mathbf{\Sigma}_y = \frac{1}{\ell'} \sum_{j=1}^{\ell'} \mathbf{y}_j \mathbf{y}_j^\top.$$

By defining a feature map $\phi(\mathbf{X}) = \text{vec}(\mathbf{\Sigma}_x)$, we have

$$\begin{aligned} \langle \phi(\mathbf{X}), \phi(\mathbf{Y}) \rangle &= \langle \text{vec}(\mathbf{\Sigma}_x), \text{vec}(\mathbf{\Sigma}_y) \rangle = \text{tr}(\mathbf{\Sigma}_x \mathbf{\Sigma}_y) \\ &= \frac{1}{\ell \ell'} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell'} \text{tr}(\mathbf{x}_i \mathbf{x}_i^\top \mathbf{y}_j \mathbf{y}_j^\top) \\ &= \frac{1}{\ell \ell'} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell'} \text{tr}(\mathbf{y}_j^\top \mathbf{x}_i \mathbf{x}_i^\top \mathbf{y}_j) \\ &= \frac{1}{\ell \ell'} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell'} \langle \mathbf{x}_i, \mathbf{y}_j \rangle^2. \end{aligned} \tag{5.1}$$

Hence, the Euclidean inner product of vectorized covariance matrices is precisely the second order mean polynomial. Furthermore, all information contained within the uncentered matrices are preserved and can be exploited.

If we rewrite the definition of the kernel as

$$\bar{k}_q(\mathbf{X}, \mathbf{Y}) = \frac{1}{\ell \ell'} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell'} \langle \mathbf{x}_i - \bar{\mathbf{x}}, \mathbf{y}_j - \bar{\mathbf{y}} \rangle^q, \quad (5.2)$$

where $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ are the mean vectors of \mathbf{X} and \mathbf{Y} , respectively, then the kernel is the inner product among centered covariance matrices when $q = 2$.

More generally, we can say that the q th order mean polynomial kernel contains all q th order moments as feature vectors. Indeed, if we let $\mathbb{P}_q = \{\mathbf{p} \in (\mathbb{N} \cup \{0\})^d \mid \mathbf{p}^T \mathbf{1} = q\}$ and $x_{h,i}$ be the h th entry in \mathbf{x}_i , enumerating all q th order moments allows us to define

$$\phi_{\mathbf{p}}(\mathbf{X}) = \frac{1}{\ell} \sqrt{\frac{q!}{p_1! p_2! \cdots p_d!}} \sum_{i=1}^{\ell} \prod_{h=1}^d x_{h,i}^{p_h}.$$

By using the feature map given by $\phi(\mathbf{X}) = [\phi_{\mathbf{p}}(\mathbf{X})]_{\mathbf{p} \in \mathbb{P}_q}$, we can derive the following equality

$$k_q(\mathbf{X}, \mathbf{Y}) = \langle \phi(\mathbf{X}), \phi(\mathbf{Y}) \rangle, \quad (5.3)$$

as given in Subsection 5.7.1. Existence of a feature vector ensures the positive semidefiniteness of the mean polynomial kernel. Similarly for the centered version of the mean polynomial kernel, the features can be explicitly expressed as a set of all the q th order central moments (See Subsection 5.7.3).

5.4 MEAN POLYNOMIAL KERNEL AND PROJECTION KERNEL RELATIONSHIP

We now establish a relationship between the proposed mean polynomial kernel and the existing Projection kernel. In principle, Grassmann kernels are considered as kernel functions for principal subspaces. Eigendecomposition of two symmetric matrices Σ_x and Σ_y is essential for the computation of the Projection kernel value between two vector sequences \mathcal{X} and \mathcal{Y} . Moreover, it can be shown that the bases of the principal subspaces are exactly the m major eigenvectors. To obtain the value of the Projection kernel between two subspaces \mathcal{X} and \mathcal{Y} , the m eigenvectors are initially stored in the matrices \mathbf{U}_x and \mathbf{U}_y . Let us define $\Sigma'_x = \mathbf{U}_x \mathbf{U}_x^T$ and $\Sigma'_y = \mathbf{U}_y \mathbf{U}_y^T$, the uncentered covariance matrices where the m major eigenvalues are replaced with ones and the rest of the eigenvalues are disregarded. Then the two kernels are related by the equality

$$k_{\text{PROJ}}(\mathbf{U}_x, \mathbf{U}_y) = \langle \text{vec}(\Sigma'_x), \text{vec}(\Sigma'_y) \rangle. \quad (5.4)$$

Details of the derivation are given in Subsection 5.7.2.

An assessment of both equations (5.1) and (5.4) suggests that while the second order mean polynomial kernel preserves every bit of information in the uncentered covariance matrices, the Projection kernel possesses the possibility to disregard and lose information of each dimension of the principal subspaces, and all the information on their orthogonal complements. A similar case can be said for the centered version of the mean polynomial kernel (5.2) versus the Projection kernel, by using the centered covariance matrices. Although the first dilemma of the Projection kernel has been addressed by Hamm and Lee [27] by extending the kernel, resulting to the scaling of information of each dimension in linear subspaces and their preservation, data on the orthogonal complement are still overlooked. As with all dimension reduction techniques, there is always a risk of losing information when employing the Grassmann kernel. Though the hope is to retain the dimensions that are most discriminant, dimension number selection must be done with care and has become a critical stage in the implementation process. Furthermore, implementation via eigenvalue decomposition adds to the computational cost of k_{PROJ} , and also k_{BC} , giving the mean polynomial kernel an efficiency advantage, especially when presented with very high dimensional data.

5.5 EXPERIMENTS AND RESULTS

We evaluate the performance of the mean polynomial kernel in binary classification tasks using data with underlying subspace structures. Techniques using the Grassmann kernels and Grassmann Distance Mutual Subspace method (GD-MSM) were also performed for comparison.

5.5.1 EEG Signal Task Classification

We compared the performances of the proposed kernel and the Grassmann kernels for EEG task identification in a binary setting using the BCI competition III-IVa dataset [9]. The data contains recorded measurements of five subjects (aa, al, av, aw, and ay) during motor imagery tasks (right hand and right foot movement) using 118 channels of electrodes. The EEG signals were recorded for 3.5 seconds with 1000 Hz sampling rate for each trial. However, we used the available downsampled version (at 100 Hz) of the data, and utilized the 0.5 to 3.5-second interval from the visual cues for each trial, resulting to a time range of 3.0 sec per trial. For data preprocessing, frequency band selection was done, and data was filtered between frequencies of 10 to 35 Hz. For each subject, 140 trials were conducted for each task, for a total of 280 trials per subject.

Two methods were employed: one using kernels with SVM and the other one using GD-MSM. For the first method, three types of kernel functions were utilized: the Grassmann kernels, Projection (PROJ) and Binet-Cauchy (BC) kernels, and the mean polynomial kernel (MP). For the GD-MSM, eight metrics were used for comparison: average distance, Binet-Cauchy metric, Geodesic distance, maximum correlation, minimum correlation, Frobenius norm based Procrustes distance, 2-norm based Procrustes distance, and Projection metric, as defined in [76]. For the SVM setting, 6-fold cross-validation was employed to evaluate the performance of the kernels such that one session per subject is used as test data while the remaining five sessions are used for training. On the other hand, class-wise (GDMSM-CD) and subject-wise (GDMSM-SD) dictionaries were implemented for the GD-MSM, as described in Section 5.2.

As for the parameters of the kernel methods, the value of q for the MP kernel was varied from 1 to 5, while the dimension of the subspace, m , was varied from 1 to 10. The regularization parameter C for SVM was varied over the set $\{10^0, 10^1, 10^2, 10^3, 10^4, 10^5\}$. To optimize the tuning of the said parameters, we implemented a 3-fold cross-validation grid search of the pairs (q, C) and (m, C) on the training data, for each cross-validation set. Values of the pairs were chosen such that the highest accuracy value is obtained. Variation and selection of the value of m for GDMSM was also done in a similar manner. The area under the ROC curve (AUC), accuracy, and F-measure values were considered for evaluating the performance of each method.

Figure 5.4 illustrates the average accuracy, AUC, and F-measure values of each method for all 5 cross-validation sets. From the graph, it is evident that the MP kernel outperforms the other methods on all three benchmarks (with accuracy, AUC and F-measure values of 84.0%, 0.896, and 0.876, respectively). This is followed by the PROJ method, with values 82.2%, 0.881, and 0.863, respectively. The values presented here for the two GDMSM's are the highest obtained among all eight metrics used, which, interestingly, is the maximum correlation. We can therefore conclude that the method employing the MP kernel plus SVM is better than the GD-MSM regardless of the selected metric.

The cumulative ratio distributions of the eigenvalues of the EEG signal sequences as the dimension parameter m of the Grassmann methods is varied from $m = 1$ to 10 are presented as box plots in Figure 5.5. Apparently, the values for the EEG data are very low. At minimum, the median is 0.127 ($m = 1$), and maximum is 0.544 ($m = 10$). The cumulative ratios also vary significantly as the dimension changes. Outliers can be observed above the fourth quartile, but not as much as in the face video data. Moreover, the difference between the maximum (outlier) and the minimum in each respective box plot is at

least 0.157 and at most 0.263, which are significantly lower than those in the previous dataset. This may explain why the Projection kernel and most of the other Grassmann-based methods perform better in terms of AUC and F-measure values on this data set.

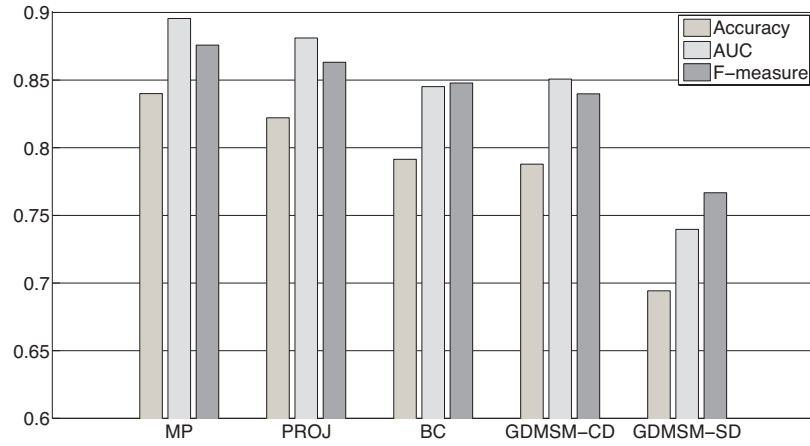


Figure 5.4: **Average performance of all methods.** The bar plot represents the average accuracy, average AUC, and average F-measure values computed.

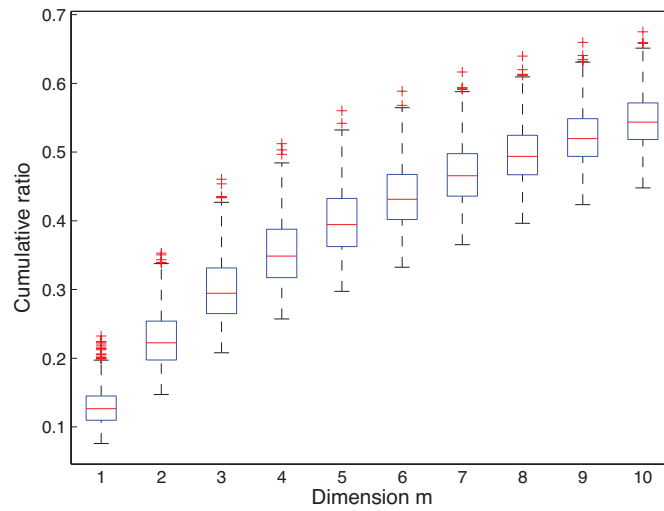


Figure 5.5: **Cumulative ratio distribution of the eigenvalues of the EEG signal sequences as dimension m is varied.**

5.5.2 Efficiency Comparison

Table 5.1: Time complexity comparison of the kernels.

	Training Stage (For kernel matrix computation)	Testing Stage (For prediction of a single sequence)
MP	$\mathcal{O}(n_{\text{tra}}^2 \ell^2 d \log_2 q)$	$\mathcal{O}(n_{\text{sv}} \ell^2 d \log_2 q)$
PROJ	For covariance matrix computation	
	$\mathcal{O}(d^2 \ell n_{\text{tra}})$	$\mathcal{O}(d^2 \ell)$
	Eigendecomposition	
BC	For covariance matrix computation	
	$\mathcal{O}(d^2 \ell n_{\text{tra}})$	$\mathcal{O}(d^2 \ell)$
	Eigendecomposition	
	Kernel value computation	
	$\mathcal{O}(dm^2 n_{\text{tra}}^2)$	$\mathcal{O}(d^2 mn_{\text{sv}})$
	Kernel value computation	
	$\mathcal{O}(k^3 n_{\text{tra}})$	$\mathcal{O}(k^3)$
	Kernel value computation	
	$\mathcal{O}(m^3 n_{\text{tra}}^2)$	$\mathcal{O}(d^2 mn_{\text{sv}})$

We investigated the time complexity of the MP kernel, and compared it with the Grassmann kernels. Suppose we are given n_{tra} number of training samples, and n_{sv} number of support vectors. For simplicity, we will assume that every (feature) vector sequence has length ℓ , and that each vector has length d . Moreover, we denote the dimension of the principal subspace as m for the Grassmann kernels, and let $k = \min(\ell, d)$. In Table 5.1, we give the computation time for each step in the calculation of the kernels. From this table, we conclude that the MP kernel is not only better in terms of performance, but it is also more efficient in terms of computational cost compared to the Grassmann kernels. This was confirmed empirically, as the average CPU time recorded for the MP kernel, for any value of q , is around 58 sec for the EEG data. On the other hand, computation of both Grassmann kernel matrices is about 1.20×10^3 for any m , while the BC takes 1.22×10^3 and 1.23×10^3 when $m = 5$ and $m = 10$, respectively. It is also worth mentioning that should the number of features d increase, the computational time for the Grassmann kernels will drastically increase, whereas the increase with the MP kernel is only linear.

5.5.3 Discussion

We conclude this section by considering an extension of the mean polynomial kernel. There are many possible extensions, one of which is by replacing the sample mean $\langle \mathbf{x}_i, \mathbf{y}_j \rangle^q$ with the expected value with respect to a probabilistic distribution: $k'_q(\mathbf{X}, \mathbf{Y}) = \mathbb{E}(\langle \mathbf{x}, \mathbf{y} \rangle^q)$. From this, the mean polynomial kernel can be derived as a special case when

$$p_x(\mathbf{x}) = \frac{1}{\ell} \sum_{i=1}^{\ell} \delta(\mathbf{x} - \mathbf{x}_i) \quad \text{and} \quad p_y(\mathbf{y}) = \frac{1}{\ell'} \sum_{i=1}^{\ell'} \delta(\mathbf{y} - \mathbf{y}_i),$$

where $\delta(\cdot)$ is the Dirac delta function.

Another choice of a probabilistic distribution can be Gaussian mixture. Suppose we are given two Gaussian mixtures

$$p_x(\mathbf{x}) = \sum_{i=1}^{\ell} \pi_{x,i} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{x,i}, \boldsymbol{\Sigma}_{x,i}) \quad \text{and}$$

$$p_y(\mathbf{y}) = \sum_{i=1}^{\ell'} \pi_{y,i} \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_{y,i}, \boldsymbol{\Sigma}_{y,i}),$$

where ℓ and ℓ' are the number of Gaussian components for the two probabilistic distributions p_x and p_y , respectively, $\pi_{z,i}$ is the mixing coefficient satisfying $\sum_{i=1}^n \pi_{z,i} = 1$, and $\boldsymbol{\mu}_{z,i}$ and $\boldsymbol{\Sigma}_{z,i}$ are the mean vector and covariance matrix of the i th Gaussian component, respectively. The second order mean polynomial kernel can be readily computed as

$$k_2(p_x, p_y) = \sum_{i=1}^{\ell} \sum_{j=1}^{\ell'} \pi_{x,i} \pi_{y,j} \left((\boldsymbol{\mu}_{x,i}^T \boldsymbol{\mu}_{y,j})^2 + \text{tr}(\boldsymbol{\Sigma}_{x,i} \boldsymbol{\Sigma}_{y,j}) + \boldsymbol{\mu}_{x,i}^T \boldsymbol{\Sigma}_{y,j} \boldsymbol{\mu}_{x,i} + \boldsymbol{\mu}_{y,j}^T \boldsymbol{\Sigma}_{x,i} \boldsymbol{\mu}_{y,j} \right).$$

This example includes the original definition of the mean polynomial kernel in Definition 5.2, which can be shown by letting

$$\begin{aligned} \pi_{x,i} &= 1/\ell, & \boldsymbol{\mu}_{x,i} &= \mathbf{x}_i, & \boldsymbol{\Sigma}_{x,i} &= \sigma_{x,i}^2 \mathbf{I}, \\ \pi_{y,j} &= 1/\ell', & \boldsymbol{\mu}_{y,j} &= \mathbf{y}_j, & \boldsymbol{\Sigma}_{y,j} &= \sigma_{y,j}^2 \mathbf{I}, \end{aligned}$$

for all $i \in \mathbb{N}_\ell$ and $j \in \mathbb{N}_{\ell'}$, and taking the limit as $\sigma^2 \rightarrow 0$. When one wishes to weight each frame in image sequences, the weights can be set to $\pi_{x,i}$ or $\pi_{y,j}$. Positive $\sigma_{x,i}^2$ or positive $\sigma_{y,j}^2$ can be used to represent uncertainties in observations,

Similar to the original mean polynomial kernel, we can explicitly represent features of the extended mean polynomial kernel, as given in Subsection 5.7.3.

5.6 SUMMARY

In this work, we have examined the mean polynomial kernel as a kernel for binary classification of data modeled as vector sets or sequences. Analogy and connection to related methods, Grassmann Projection kernel in particular, have also been drawn. The effectiveness of the MP kernel was empirically supported using data of motor imagery EEG recordings. Furthermore, we present a comparison of computational costs between methods, and some interesting extensions of the MP kernel by considering the probabilistic distribution of the data. In brief, the mean polynomial kernel excels known methods from literature, both in performance and efficiency.

5.7 PROOFS AND DISCUSSION

5.7.1 Derivation of Equation (5.3)

Proposition 5.1. *A mapping function of mean polynomial kernel is*

$$\phi(\mathbf{X}) = \left[\frac{1}{\ell} \sqrt{\frac{q!}{p_1! p_2! \cdots p_d!}} \sum_{i=1}^{\ell} \prod_{h=1}^d x_{h,i}^{p_h} \right],$$

where $\mathbf{p} \in (\mathbb{N} \cup \{0\})^q$ such that $\mathbf{p}^T \mathbf{1} = q$.

Proof. Let $x_{h,i}$ and $y_{h,i}$ be the (h, i) th entries in \mathbf{X} and \mathbf{Y} , respectively.

Let d be the number of rows in \mathbf{X} and \mathbf{Y} ,

$$k_q(\mathbf{X}, \mathbf{Y}) = \frac{1}{\ell \ell'} \sum_{i,j} \langle \mathbf{x}_i, \mathbf{y}_j \rangle^q = \frac{1}{\ell \ell'} \sum_{i,j} \left(\sum_{h=1}^d x_{h,i} y_{h,j} \right)^q.$$

Using the multinomial theorem, we get

$$\begin{aligned} k_q(\mathbf{X}, \mathbf{Y}) &= \frac{1}{\ell \ell'} \sum_{i,j} \sum_{\mathbf{p}} \frac{q!}{p_1! \cdots p_d!} \prod_{h=1}^d x_{h,i}^{p_h} y_{h,j}^{p_h} \\ &= \sum_{\mathbf{p}} \left(\frac{1}{\ell} \sqrt{\frac{q!}{p_1! p_2! \cdots p_d!}} \sum_{i=1}^{\ell} \prod_{h=1}^d x_{h,i}^{p_h} \right) \\ &\quad \times \left(\frac{1}{\ell'} \sqrt{\frac{q!}{p_1! p_2! \cdots p_d!}} \sum_{j=1}^{\ell'} \prod_{h=1}^d y_{h,j}^{p_h} \right) \\ &= \langle \phi(\mathbf{X}), \phi(\mathbf{Y}) \rangle, \end{aligned}$$

where $\mathbf{p} \in (\mathbb{N} \cup \{0\})^q$ such that $\mathbf{p}^T \mathbf{1} = q$. □

5.7.2 Derivation of Equation (5.4)

Suppose the transformed covariance matrices are given by

$$\Sigma'_x = \mathbf{U}_x \Lambda_x \mathbf{U}_x^T = \mathbf{U}_x \mathbf{U}_x^T \quad \text{and} \quad \Sigma'_y = \mathbf{U}_y \Lambda_y \mathbf{U}_y^T = \mathbf{U}_y \mathbf{U}_y^T,$$

obtained via eigendecomposition of the covariance matrices Σ_x and Σ_y , and setting the major eigenvalues to one and the minor eigenvalues to zero. Then we can write

$$\begin{aligned} \kappa_{\text{PROJ}}(\mathbf{U}_x, \mathbf{U}_y) &= \|\mathbf{U}_x^T \mathbf{U}_y\|_F^2 = \text{tr}(\mathbf{U}_x^T \mathbf{U}_y \mathbf{U}_y^T \mathbf{U}_x) = \text{tr}(\mathbf{U}_x \mathbf{U}_x^T \mathbf{U}_y \mathbf{U}_y^T) \\ &= \text{tr}(\Sigma'_x \Sigma'_y) = \langle \text{vec}(\Sigma'_x), \text{vec}(\Sigma'_y) \rangle. \end{aligned}$$

This concludes the derivation of equation (5.4).

5.7.3 Explicit Representation of Features

In Section 5.3, we have shown that the features of the mean polynomial kernel can be represented explicitly. Features of the centered mean polynomial kernel are represented by the q th central moments:

$$\bar{\Phi}_{\mathbf{p}}(\mathbf{X}) = \frac{1}{\ell} \sqrt{\frac{q!}{p_1! p_2! \cdots p_d!}} \sum_{i=1}^{\ell} \prod_{h=1}^d (x_{h,i} - \bar{x}_h)^{p_h}.$$

The features that produce the extended mean polynomial kernel are given by

$$\Phi_{\mathbf{p}}(p_x) = \sqrt{\frac{q!}{p_1! p_2! \cdots p_d!}} \mathbb{E} \left(\prod_{h=1}^d x_{h,i}^{p_h} \right)$$

for all $\mathbf{p} \in (\mathbb{N} \cup \{0\})^d$ such that $\mathbf{p}^T \mathbf{1} = q$, and the features for the extended centered mean polynomial kernel are given by

$$\bar{\Phi}_{\mathbf{p}}(p_x) = \sqrt{\frac{q!}{p_1! p_2! \cdots p_d!}} \mathbb{E} \left(\prod_{h=1}^d (x_h - \mathbb{E}(x_h))^{p_h} \right).$$

CONCLUSION AND FUTURE DIRECTIONS

In brief, this thesis deals with machine learning applications for bioinformatics and brain-computer interface. In particular, we challenged conventional methods in predicting drug-protein interactions, enzyme active sites, and EEG signal tasks, which are deemed very useful in drug design and discovery, cognitive sciences and neuroinformatics.

In Chapter 3, we proposed the kernel weighted CCA which was executed via the eigendecomposition method. We showed that the performance of SVMs in the task of protein-ligand could further be improved when KWCCA is employed. We also compared effects on the performance when different features were used for the weights (e.g., features from the interaction profiles, chemical profiles, or both). Even in the field of computational biology, CCA for more than two data sources has been widely used [58, 78, 90] and their usual objectives involve maximizing the sum of correlations for every pair of data sources. One possible extension of this work is to explore the case where multiple data sets are simultaneously analyzed using CCA. It could also be interesting to investigate the effectiveness of applying the proposed method to other biological problems aside from protein-ligand interaction prediction.

In Chapter 4, we proposed an algorithm for learning the weights of atoms of the enzymes used for computing the deviation in searching for common active sites between a template and an enzyme query. Aside from modeling the algorithm in a more natural fashion compared to existing techniques, we also introduced the use of Bregman divergences in the regularization functions. Since, the proposed algorithms in this work were designed to minimize the sum of the regularization function and the loss function, this allows us to consider several extensions, such as multi-task learning [38], transfer learning [39], and structural learning [34], developed well in the field of machine learning. Hence, this study could be a great step to facilitate further developments of algorithms for the active site search task.

Finally, in Chapter 5, we investigated a special case of the multi-instance kernel and its applicability to physiological data such as EEG signals. Furthermore, we also analyzed its relation to Grassmann Projection kernel, and compared their performances in task classification. An interesting future direction of this work, as well as the proposed algorithms in Chapters 3 and 4, would be to extend its practice to multi-classification tasks.

BIBLIOGRAPHY

- [1] S. Akaho. Kernel method for canonical correlation analysis. In *Proceedings of International Meeting of Psychometric Society*, 2001. (Cited on page [13](#).)
- [2] I. Antes, S.W. Siu, and T. Lengauer. DynaPred: a structure and sequence based method for the prediction of MHC class I binding peptide sequences and conformations. *Bioinformatics*, 22(14):e16–24, Jul 2006. (Cited on page [11](#).)
- [3] J. Bajorath. Computational analysis of ligand relationships within target families. *Curr Opin Chem Biol*, 12(3):352–8, Jun 2008. (Cited on pages [11](#) and [12](#).)
- [4] P.J. Ballester and J.B. Mitchell. A machine learning approach to predicting protein-ligand binding affinity with applications to molecular docking. *Bioinformatics*, 26(9):1169–75, May 2010. (Cited on page [11](#).)
- [5] A. Barachant, S. Bonnet, M. Congedo, and C. Jutten. Classification of covariance matrices using a Riemannian-based kernel for BCI applications. *Neurocomputing*, 112(0):172–178, 2013. (Cited on page [45](#).)
- [6] J.A. Barker and J.M. Thornton. An algorithm for constraint-based structural template matching: application to 3D templates with statistical analysis. *Bioinformatics*, 19(13):1644–9, Sep 2003. (Cited on pages [35](#), [36](#), and [37](#).)
- [7] T. Biniashvili, E. Schreiber, and Y. Kliger. Improving classical substructure-based virtual screening to handle extrapolation challenges. *J Chem Inf Model*, 52(3):678–85, Mar 2012. (Cited on page [11](#).)
- [8] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC, New York, USA, 2006. (Cited on page [40](#).)
- [9] B. Blankertz, K.R. Müller, D. Krusienki, G. Schalk, J. Wolpaw, A. Schlögl, G. Pfurtscheller, J.d.R. Millán, M. Schröder, and N. Birbaumer. The BCI competition III: Validating alternative approaches to actual BCI problems. *IEEE Trans Neural Sys Rehab Eng*, 14(2):153–159, 2006. (Cited on pages [45](#) and [52](#).)
- [10] K. Bleakley and Y. Yamanishi. Supervised prediction of drug-target interactions using bipartite local models. *Bioinformatics*, 25(18):2397–403, Sep 2009. (Cited on page [12](#).)

- [11] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, NY, USA, 2004. (Cited on page 5.)
- [12] M. Campillos, M. Kuhn, A.-C. Gavin, L.J. Jensen, and P. Bork. Drug target identification using side-effect similarity. *Science*, 321: 263–266, 2008. (Cited on page 12.)
- [13] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. (Cited on page 19.)
- [14] B. Chen, R.F. Harrison, G. Papadatos, P. Willett, D.J. Wood, X.Q. Lewell, P. Greenidge, and N. Stiefl. Evaluation of machine-learning methods for ligand-based virtual screening. *J Comput Aided Mol Des*, 21(1-3):53–62, Jan-Mar 2007. (Cited on page 11.)
- [15] K.C. Chou and Y.D. Cai. A novel approach to predict active sites of enzyme molecules. *Proteins*, 55(1):77–82, Apr 2004. (Cited on page 35.)
- [16] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, NY, USA, 2000. (Cited on pages 1, 5, 6, and 45.)
- [17] A. Delorme and S. Makeig. EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *Journal of Neuroscience Methods*, 134:9–21, 2004. (Cited on page 45.)
- [18] F. Desobry, M. Davy, and W.J. Fitzgerald. A class of kernels for sets of vectors. In *Proc. 13th ESANN*, pages 461–466, 2005. (Cited on page 45.)
- [19] J.S. Fetrow and J. Skolnick. Method for prediction of protein function from sequence using the sequence-to-structure-to-function paradigm with application to glutaredoxins/thioredoxins and T1 ribonucleases. *J Mol Biol*, 281(5):949–68, Sep 1998. (Cited on page 35.)
- [20] K. Fukui and O. Yamaguchi. Face recognition using multi-viewpoint pattern for robot vision. In *Proc. Int. Symp. Robotics Research*, pages 192–201, 2003. (Cited on pages 46 and 48.)
- [21] T. Gärtner, P.A. Flach, A. Kowalczyk, and A.J. Smola. Multi-instance kernels. In *Proc. 19th ICML*, pages 179–186, June 2002. (Cited on page 50.)
- [22] P.F. Gherardini, M.N. Wass, M. Helmer-Citterich, and M.J. Sternberg. Convergent evolution of enzyme active sites is not a rare phenomenon. *J Mol Biol*, 372(3):817–45, Sep 2007. (Cited on page 35.)

- [23] A. Golugula, G. Lee, S.R. Master, M.D. Feldman, J.E. Tomaszewski, and A. Madabhushi. Supervised regularized canonical correlation analysis: integrating histologic and proteomic data for predicting biochemical failures. In *Proc. IEEE Eng Med Biol Soc Conf*, number -, pages 6434–7, 2011. (Cited on page 14.)
- [24] A.J. Gonzalez, L. Liao, and C.H. Wu. Predicting ligand binding residues and functional sites using multi-positional correlations with graph theoretic clustering and kernel CCA. *IEEE/ACM Trans Comput Biol Bioinform*, Oct 2011. (Cited on page 13.)
- [25] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Proc. IEEE ICCV*, volume 2, pages 1458–1465, Beijing, China, October 2005. (Cited on page 45.)
- [26] J. Hamm and D. Lee. Grassmann discriminant analysis: a unifying view on subspace-based learning. In *Proc. 25th ICML*, pages 376–383, 2008. (Cited on pages 46 and 48.)
- [27] J. Hamm and D. Lee. Extended Grassmann kernels for subspace-based learning. In *Proc. NIPS*, pages 601–608, 2008. (Cited on pages 45, 46, 48, and 52.)
- [28] D.R. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Comput*, 16(12):2639–64, Dec 2004. (Cited on page 13.)
- [29] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning*. Springer, July 2003. (Cited on pages 1, 5, and 44.)
- [30] H. Hotelling. Relation between two sets of variates. *Biometrika*, 28(3), 1936. (Cited on page 13.)
- [31] V.A. Ivanisenko, S.S. Pintus, D.A. Grigorovich, and N.A. Kolchanov. PDBSiteScan: a program for searching for active, binding and posttranslational modification sites in the 3D structures of proteins. *Nucleic Acids Res*, 32(Web Server issue): W549–54, Jul 2004. (Cited on page 35.)
- [32] G. Pfurtscheller, J. Müller-Gerking, and H. Flyvbjerg. Designing optimal spatial filters for single-trial EEG classification in a movement task. *Clinical Neurophysiology*, 110(5):787–798, 1999. (Cited on page 45.)
- [33] L. Jacob and J.-P. Vert. Protein-ligand interaction prediction: An improved chemogenomics approach. *Bioinformatics*, 24(19):2149–56, Oct 2008. (Cited on pages 11 and 12.)

- [34] T. Joachims. A support vector method for multivariate performance measures. In *Proc. 22nd ICML*, 2005. (Cited on page 59.)
- [35] H. Kashima, K. Tsuda, and A. Inokuchi. Kernels for graphs. In *Kernels and Bioinformatics*, pages 155–170, Cambridge, MA, USA, 2004. MIT Press. (Cited on page 45.)
- [36] T. Kato and N. Nagano. Metric learning for enzyme active-site search. *Bioinformatics*, 26(21):2698–2704, 2010. (Cited on pages 35, 36, 42, and 43.)
- [37] T. Kato and N. Nagano. Discriminative structural approaches for enzyme active-site prediction. *BMC Bioinformatics*, 12(Suppl 1): S49, 2011. (Cited on page 35.)
- [38] T. Kato, H. Kashima, M. Sugiyama, and K. Asai. Conic programming for multitask learning. *IEEE Trans Knowl Data Eng*, 22(7):957–968, 2010. (Cited on page 59.)
- [39] T. Kato, K. Okada, H. Kashima, and M. Sugiyama. A transfer learning approach and selective integration of multiple types of assays for biological network inference. *International Journal of Knowledge Discovery in Bioinformatics (IJKDB)*, 1(1):66–80, 2010. (Cited on page 59.)
- [40] T. Kato, W. Takei, and S. Omachi. A discriminative metric learning algorithm for face recognition. *IPSJ Transactions on Computer Vision and Applications*, 5(85–89), 2013. (Cited on pages 36 and 41.)
- [41] T.-K. Kim, J. Kittler, and R. Cipolla. Discriminative learning and recognition of image set classes using canonical correlations. *IEEE Trans Pattern Anal Mach Intell*, 29:1005–1018, 2007. (Cited on pages 45 and 46.)
- [42] T. Klabunde. Chemogenomic approaches to drug discovery: similar receptors bind similar ligands. *Br. J. Pharmacol.*, 152:5–7, 2007. (Cited on page 12.)
- [43] G.J. Kleywegt. Recognition of spatial motifs in protein structures. *J Mol Biol*, 285(4):1887–97, Jan 1999. (Cited on page 35.)
- [44] R.I. Kondor and T. Jebara. A kernel between sets of vectors. In *Proc. 20th ICML*, pages 361–368, August 2003. (Cited on page 45.)
- [45] R.I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proc. 19th ICML*, pages 315–322, July 08–12, 2002. (Cited on page 45.)

- [46] T. Kuboya, K. Hirata, and K.F. Aoiki-Kinoshita. An efficient unordered tree kernel and its application to glycan classification. In *Proc. PAKDD*, pages 184–195, 2008. (Cited on page 45.)
- [47] R.A. Laskowski, J.D. Watson, and J.M. Thornton. Protein function prediction using local 3D templates. *J Mol Biol*, 351(3):614–26, Aug 2005. (Cited on page 35.)
- [48] C. Leslie, E. Eskin, and W.S. Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Proc. of the Pacific Symposium on Biocomputing*, pages 564–575, 2002. (Cited on page 45.)
- [49] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *J. Mach. Learn. Res.*, 2:419–444, 2002. (Cited on page 45.)
- [50] Y. Loewenstein, D. Raimondo, O.C. Redfern, J. Watson, D. Frishman, M. Linial, C. Orengo, J. Thornton, and A. Tramontano. Protein function annotation by homology-based inference. *Genome Biol.*, 10(2):207, Feb 2009. (Cited on page 35.)
- [51] Y.C. Martin, J.L. Kofron, and L.M. Traphagen. Do structurally similar molecules have similar biological activity. *J. Med. Chem*, 45:4350–4358, 2002. (Cited on page 12.)
- [52] K.P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, Cambridge, MA, USA, 2012. (Cited on pages 1 and 5.)
- [53] N. Nagano. EzCatDB: the Enzyme Catalytic-mechanism Database. *Nucleic Acids Res*, 33(Database issue):D407–12, Jan 2005. (Cited on page 42.)
- [54] N. Nagano, T. Noguchi, and Y. Akiyama. Systematic comparison of catalytic mechanisms of hydrolysis and transfer reactions classified in the EzCatDB database. *Proteins*, 66(1):147–59, Jan 2007. (Cited on page 42.)
- [55] M. Neumann, N. Patricia, R. Garnett, and K. Kersting. Efficient graph kernels by randomization. In *Proc. ECML/PKDD*, pages 378–393, 2012. (Cited on page 45.)
- [56] G.V. Paolini, R.H.B. Shapland, W.P. van Hoorn, J.S. Mason, and A.L. Hopkins. Global mapping of the pharmacological space. *Nat Biotechnol*, 24:805–815, 2006. (Cited on page 11.)
- [57] F. Pazos, A. Rausell, and A. Valencia. Phylogeny-independent detection of functional residues. *Bioinformatics*, 22(12):1440–8, Jun 2006. (Cited on page 13.)

- [58] Y. Peng, D. Zhang, and J. Zhang. A new canonical correlation analysis algorithm with local discrimination. *Neural Process Lett*, 31:1–15, 2010. (Cited on page 59.)
- [59] S. Qiu, T. Lane, and L.J. Buturovic. A randomized string kernel and its application to RNA interference. In *Proc. AAAI*, pages 627–632, 2007. (Cited on page 45.)
- [60] H. Ramoser, J. Müller-Gerking, and G. Pfurtscheller. Optimal spatial filtering of single trial EEG during imagined hand movement. *IEEE transactions on rehabilitation engineering a publication of the IEEE Engineering in Medicine and Biology Society*, 8(4):441–446, 2000. (Cited on page 45.)
- [61] R. Relator, T. Kato, and R. Lemence. Improved protein-ligand prediction using kernel weighted canonical correlation analysis. *IP SJ TBIO*, 6:18–28, June 2013. (Cited on page 3.)
- [62] R. Relator, Y. Hirohashi, E. Ito, and T. Kato. Mean polynomial kernel and its application to vector sequence recognition. *IEICE Trans. Inf. and Syst.*, 97-D(7):1855–1863, 2014. (Cited on page 3.)
- [63] R. Relator, T. Kato, and N. Nagano. Enzyme active site prediction using bregman divergence regularized machine. In *IEICE Tech. Rep.*, volume 113, pages 61–66, December 2013. (In Japanese). (Cited on page 3.)
- [64] R. Relator, Y. Hirohashi, E. Ito, and T. Kato. Application of the mean polynomial kernel to video image recognition and brain-machine interface. In *IEICE Tech. Rep.*, volume 113, pages 281–286, January 2014. (In Japanese). (Cited on page 3.)
- [65] R. Relator, T. Kato, and R. Lemence. Improved protein-ligand prediction using kernel weighted canonical correlation analysis. In *IP SJ SIG Tech. Rep.*, volume 2014-BIO-37, pages 1–6, March 2014. (Cited on page 3.)
- [66] R. Relator, Y. Hirohashi, and T. Kato. Mean polynomial kernel for face membership authentication. In *Proc. MVA2013*, pages 1–3, May 2013. (Cited on page 3.)
- [67] R. Relator, Y. Hirohashi, E. Ito, and T. Kato. Mean polynomial kernel and its application to vector sequence recognition. In *IP SJ SIG Tech. Rep.*, volume 2014-MPS-100, September 2014. (Cited on page 3.)
- [68] D. Rognan. Chemogenomic approaches to rational drug design. *Br J Pharmacol*, 152(1):38–52, Sep 2007. (Cited on pages 11 and 12.)
- [69] H. Saito, M. Kubota, R.W. Roberts, Q. Chi, and H. Matsunami. RTP family members induce

- functional expression of mammalian odorant receptors. *Cell*, 119 (5):679–91, Nov 2004. (Cited on page 19.)
- [70] H. Sakano and N. Mukawa. Kernel mutual subspace method for robust facial image recognition. In *Proc. Int. Conf. on Knowledge-Based Intell. Eng. Sys. And App. Tech*, pages 245–248, 2000. (Cited on pages 46 and 48.)
- [71] D. Samarov, J.S. Marron, Y. Liu, C. Grulke, and A. Tropsha. Local kernel canonical correlation analysis with application to virtual drug screening. *Ann Appl Stat*, 5(3):2169–2196, Sep 2011. (Cited on page 13.)
- [72] B. Schölkopf and A. Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press, Cambridge, MA, USA, 2001. (Cited on pages 1, 5, 6, and 45.)
- [73] A. Schuffenhauer, P. Floersheim, P. Acklin, and E. Jacoby. Similarity metrics for ligands reflecting the similarity of the target proteins. *J. Chem. Inf. Comput. Sci.*, 43:391–405, 2003. (Cited on page 12.)
- [74] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press, NY, USA, 2004. (Cited on pages 1, 5, and 45.)
- [75] Q. Shi, J. Petterson, G. Dror, J. Langford, A. Smola, and S.V.N. Vishwanathan. Hash kernels for structured data. *J. Mach. Learn. Res.*, 10:2615–2637, 2009. (Cited on page 45.)
- [76] R. Shigenaka, B. Raytchev, T. Tamaki, and K. Kaneda. Face sequence recognition using Grassmann distances and Grassmann kernels. In *Proc. IJCNN*, pages 2630–2636, 2012. (Cited on pages 45, 46, 48, and 53.)
- [77] A. Stark and R.B. Russell. Annotation in three dimensions. PINTS: Patterns in Non-homologous Tertiary Structures. *Nucleic Acids Res*, 31(13):3341–4, Jul 2003. (Cited on page 35.)
- [78] C.S. Tang and M.A. Ferreira. A gene-based test of association using canonical correlation analysis. *Bioinformatics*, 28(6):845–50, Mar 2012. (Cited on page 59.)
- [79] J.W. Torrance, G.J. Bartlett, C.T. Porter, and J.M. Thornton. Using a library of structural templates to recognise catalytic sites and explore their evolution in homologous families. *J Mol Biol*, 347(3): 565–81, Apr 2005. (Cited on page 35.)
- [80] T. van Laarhoven, S.B. Nabuurs, and E. Marchiori. Gaussian interaction profile kernels for predicting drug-target interaction. *Bioinformatics*, 27(21):3036–43, Nov 2011. (Cited on page 11.)

- [81] S.V.N. Vishwanathan and A.J. Smola. Fast kernels for string and tree matching. In *Advances in Neural Info. Proc. Sys.*, volume 15, pages 569–576, 2003. (Cited on page 45.)
- [82] S.V.N. Vishwanathan and A.J. Smöla. Binet-cauchy kernels. In *Proc. NIPS*, pages 1441–1448, 2004. (Cited on page 48.)
- [83] S.V.N. Vishwanathan, N.N. Schraudolph, R.I. Kondor, and K.M. Borgwardt. Graph kernels. *J. Mach. Learn. Res.*, 11:1201–1242, 2010. (Cited on page 45.)
- [84] A.C. Wallace, N. Borkakoti, and J.M. Thornton. TESS: a geometric hashing algorithm for deriving 3D coordinate templates for searching structural databases. application to enzyme active sites. *Protein Sci*, 6(11):2308–2323, 1997. (Cited on pages 35, 36, and 37.)
- [85] A.M. Wassermann, H. Geppert, and J. Bajorath. Ligand prediction for orphan targets using support vector machines and various target-ligand kernels is dominated by nearest neighbor effects. *J. Chem. Inf. Model.*, 49:2155–2167, 2009. (Cited on page 12.)
- [86] R.M. Willems, I. Toni, P. Hagoort, and D. Casasanto. Body-specific motor imagery of hand actions: neural evidence from right- and left-handers. *Front Hum Neuro*, 3:1–9, 2009. (Cited on page 45.)
- [87] D.M. Witten, R. Tibshirani, and T. Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–34, Jul 2009. (Cited on page 14.)
- [88] L. Wolf and A. Shashua. Learning over sets using kernel principal angles. *J. Mach. Learn. Res.*, 4:913–931, 2003. (Cited on pages 46 and 48.)
- [89] O. Yamaguchi, K. Fukui, and K. Maeda. Face recognition using temporal image sequence. In *Proc. Int. Conf. Automatic Face and Gesture Recognition*, pages 318–323, 1998. (Cited on pages 46 and 48.)
- [90] Y. Yamanishi, J.P. Vert, A. Nakaya, and M. Kanehisa. Extraction of correlated gene clusters from multiple genomic data by generalized kernel canonical correlation analysis. *Bioinformatics*, 19 Suppl 1:i323–30, 2003. (Cited on pages 13 and 59.)
- [91] Y. Yamanishi, E. Pauwels, H. Saigo, and V. Stoven. Extracting sets of chemical substructures and protein domains governing drug-target interactions. *J Chem Inf Model*, May 2011. (Cited on page 14.)

- [92] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *Proc. ICDM '02*, pages 721–724, 2002. (Cited on page [19](#).)
- [93] S. Yu, B. De Moor, and Y. Moreau. Learning with heterogenous data sets by weighted multiple kernel canonical correlation analysis. In *Proc. IEEE MLSP 2007*, pages 81–86, August 2007. (Cited on page [13](#).)