TELECOM Bretagne

Orange Labs Lannion

Engineering end-of-study internship

Option 4: Information Processing Systems

Academic Year: 2015 - 2016

Supervised Dimensionality Reduction for Multi-Label Nearest Neighbors

---

Author:
Réda ALAMI
reda.alami@orange.com

Orange Labs Advisors:
Frank MEYER *(R&D engineer)*
frank.meyer@orange.com
Wissam SIBLINI *(PhD student)*
wissam.siblini@orange.com

TELECOM Bretagne Advisor:
Vincent GRIPON *(Associate Professor)*
vincent.gripon@telecom-bretagne.eu

September 06 2016

Contents

List of Figures

List of Tables

Abstract

The ML-kNN algorithm is one of the most famous and most efficient multi-label classifier. Its performances are very remarkable when compared with the other state-of-art multi-label classifiers. Nevertheless, it suffers from two major drawbacks: its accuracy crucially depends on the metric function used to compute distances between instances, and when dealing with high dimensions data, the neighborhoods identification task becomes very slow. So, both metric learning and dimensionality reduction are essential to improve the ML-kNN performances. In this report, we propose a novel multi-label Mahalanobis distance learned via a supervised dimensionality reduction approach that we call ML-ARP. ML-ARP is a process that adapts random projections on a multi-label dataset to improve the ML-kNN performances. Unlike most state of art multi-label dimensionality reduction approaches that solve eigenvalue or inverse problem, our method is iterative and scales up with high dimensions. There is no eigenvalue or inverse problems to solve. Experiments show that the ML-ARP allows us to highly upgrade the ML-kNN classifier. Statistical tests assert that the ML-ARP is better than the remaining state-of-art multi-label dimensionality reduction approaches.

Keywords: ML-kNN, dimensionality reduction, random projections, distance metric learning, nearest neighbor.

Résumé

L'algorithme ML-kNN basé sur le principe des k plus proches voisins est considérée comme le classifieur le plus robuste et le plus connu dans le domaine de la classification multi-label. Toutefois, le principe des k plus proches voisins présente deux points faibles majeurs. Premièrement, les performances du ML-kNN dépendent étroitement de la métrique choisie pour l'identification des structures de voisinage. Ensuite, la recherche des voisins dans une large base de donnée n'est pas une tâche facile à faire. En effet, plus les dimensions d'entrée sont grandes, plus la recherche des voisins se fait plus lentement. C'est pour ces raisons qu'un apprentissage de métrique adaptée et une réduction de dimension efficace s'imposent. Nous proposons ainsi, une nouvelle approche d'apprentissage de métrique via une réduction de dimension supervisée. Notre méthode se nomme ML-ARP pour Multi-Label Adaptative Random Projection. ML-ARP adapte des projections aléatoires à n'importe quelle base de donnée multi-label dans le but d'améliorer les performances du ML-kNN. Contrairement à la majorité des algorithmes de l'état de l'art en réduction de dimension multi-label qui se basent généralement sur des résolutions de problèmes aux valeurs propres ou des inversements de matrices coûteuses en complexité, ML-ARP s'itère et passe facilement à l'échelle. L'étude expérimentale menée affirme que ML-ARP améliore nettement les performances du ML-kNN. Ensuite, des tests statistiques confirment que ML-ARP est largement meilleur que le reste des algorihtmes de l'art de l'art en réduction de dimension multi-label.

Mots clef : ML-kNN, réduction de dimension supervisée, projections aléatoires, apprentissage de métrique, plus proches voisins.

Chapter 1

Context

1.1 Orange group

The *Orange* group is one of the most important telecommunication operators in the world. With almost 230 million customers and 172 000 collaborators across 32 countries.
Orange is ranked at the 50th position in the global brands. Formely *France Telecom*, the group took the name of *Orange* on May 28, 2012.
France Telecom was founded on January 1,1988, in response to a European directive for competition in telecomuunication services. The company became in 1990 a statutory operator, obtaining a separate legal personality of the State, then a limited company in 1996 where the French government was the sole shareholder. France Telecom became a private company since 2004, when the State abandons its majority shareholder status.

1.2 R&D department

Orange group attaches great importance to research and development. Research and development laboratories dubbed *Orange Labs* are distributed throughout France. *Orange Labs* network has also 18 laboratories across three contiments: China, South Korea, USA, France, Japan, Poland, the United Kingdom and also, very recently, in Jordan and Egypt.
These laboratories have long maintained strong cooperation with some engineering "Grandes écoles" such as: TELECOM ParisTech, TELECOM Bretagne, TELECOM ParisSud until the France Telecom group becomes a private company.
Orange Labs' work clusters around six major R&D projects: smart cities, mobile payment, content aggregation, mobile connections, services and applications, and finally, smart networks. There are more than 5000 employees including 3700 engineers and scientists.

1.3 Social responsibility

Orange group has made many commitments with regard to the environment and its social responsibilities respect.
The list of Orange commitments in the areas of environment, employment, technology and also customer relations are available on its website.

1.3.1 Environment

Regarding the environment, Orange objectives are, to reduce emissions of greehouse gas by 20% and energy consumption by 15% between 2006 and 2020, or increase by half the collection of used mobile phones. Orange also wants to reduce the impact generated by its products to its customers through eco-design approach.
In its main entities, the group deploys an Environmental Management System (EMS) following the ISO 14001 standard. They aim at covering 60% of the group with this system (an then with the 14001 certification).

1.3.2 Working conditions and diversity

Orange aims at improve the working conditions of its employees with the implementation of the new social contact in 2010 in France, which is deployed since 2011 in 23 countries. This charter sets a new Fashion Group's relationship with its employees. These commitments have enabled Orange to record a significant improvement of its image among its employees. Since 2015, Orange has been recognized as one of the preferred employers in the main countries where the Group operates.
It may be noted that Orange has been awarded the "Top Employers Europe 2013" in March 2013 in London, the "Top employer Africa" in August 2013, and has also received the "great place to work" label in Brazil. Orange Business Service (OBS) has been awarded the "Top Employer" in India in January 2014.
Orange aims at have 35% women in its leadership and ensure equal pay by conducting regular diagnoses. Orange was the first French group to receive the 2011 European label on professional equality.

1.3.3 Social responsibility

From the social point of view, one may cite the supporting of local developments. In France, for instance, Orange sells phones at very low prices and conducts workshops to educate people in order to reduce their costs related to ICT (Information and Communication Technologies). In Africa, Orange has developped new mobile services like text-to-speech (allows audio playback of a written text) or audio messaging to adapt to the problems of illiteracy.
To fight against the digital divide, Orange sets several goals such as covering 80% of the AMEA (Africa, Middle-East, Asia) by the end of 2015 and the deployment of solar radio and community stations for phones collective access to the internet in the most remote areas.
The Orange Healthcare division develops for over 10 years solutions to the health problems and addiction.

1.4 Orange labs team

This work was done within the "Profiling and Data-Mining" research team of Orange Labs Lannion under the direction of the manager Fabrice CLEROT and under the supervision of Frank MEYER, R&D engineer and Wissam SIBLINI, PhD student.

1.5 Big Data

Nowadays, about 90% of data stored in the world has been created in the last two years (according to the IBM website). This finding gives us an idea about the importance and the exponential growth of data volumes involved in our society.
The term "Big Data" refers to the problems and the solutions ecosystem for the storage and analysis of these gigantic masses of data. The data sources are numerous: climate sensors, spatial information sensors, online shopping transactions and finally messages through social networks. Big Data includes several analytic applications which aim at make sense to the data. Among these applications, one may found the *automatic recommendation*. The work presented in this document is located in this area. It is an engineering end-of-study internship in the domain of machine learning.

1.6 Automatic recommendation

Automatic recommendation systems aim at facilitating the user choices by applying a first filter to the products offered to him. These products can be of different types (movies, books, videos, music, ...). We call them "items". These systems are mainly found in video on demand (VoD) applications and also in online sales of movie, books, clothing,...etc.
The domain of automatic recommendation has experienced a significant advance since 2006 when NetFlix, the first american DVD rental market, has organized a competition in order to improve its recommendation engine. For this purpose, NetFlix has proposed to offer one million dollars price to any system offering higher performances than their original. This event has then allowed much progress in the automatic recommendation domain.
These recommendation engines are crucial for online retail websites. There are three main interests for this purpose:

- Helping the user to deal with a huge catalog. With an efficient recommendation engine, we can offer a large range of products. Thus, everyone can find precisely what they are seeking for.
- Creating a customer loyalty by offering them relevant product without discomforting him.
- Making sophisticated recommendations by proposing not very well-known items from the catalog. This allows to provide the best possible user experience, and also facilitates the online sales of products considered less "maintream".

The success of this last point is a very important issue for recommendation systems. The added value of these engines is their capacity to valorize all products taken from the catalog. The user is more likely to make discoveries if the recommendation engine manages to recommand products of the "long tail".
The "long tail" (Fig.1.1) represents all products purchassed or viewed whose combined sales volume represents a significant market share. Successfully discovering these products to customers is a critical issue for online shopping websites.

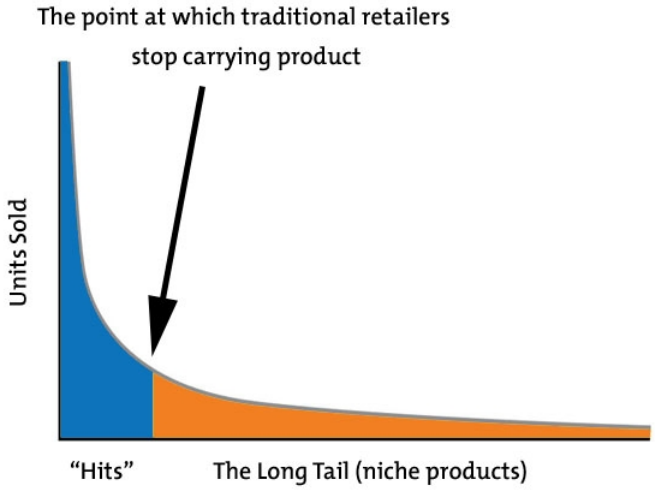


Figure 1.1: Long tail

1.7 VIPE: Visual Interactive and Personalized Exploration of data

VIPE is a multi-label interactive classification system developped in Orange Labs. It is a multi-user system hosted in a web application accessed from any browser (See http://prof.orange-labs.fr/vipe/index.php). In an interactive way, the user creates target concepts and explains them with a set of examples and counter-examples.
The learning algorithm integrates in real time these training examples in order to correct the learned system by performing several iterations.
The learning loop ends when the user considers that VIPE performances are satisfactory. Therefore, the systm is able to classify new unseen data.
In Orange Labs, two use-cases are targeted:

- Movies interactive multi-label classification.

- Tweets interactive multi-label classification.

Currently, VIPE is a demonstrator: the current version uses a learning algorithm based on Gravity fast matrix factorisation [Takacs et al., 2007] which is not the final target algorithm.

1.7.1 Movies interactive multi-label classification

To choose a "good" movie for the evening, users who have no specific title may be puzzled by the huge number of movies available in the current catalogs of VoD. Therefore, a user needs to be assisted in his choice of television programs for which he will spend time and money. VIPE allows us to create customized movies classifier which proposes suitable movies without much effort from the user. First of all, the user defines its set of preferred labels (e.g. Funny, I like, Beautiful music, sad) then annotates a small set of movies conveying his preferences to the learning algorithm. For example, he annotates Titanic positively with three labels: "Like", "Beautiful Music" and "sad". The learning algorithm can predict the most probable labels for a selected movie or the most likely movies for a selected label or a combination of them.

1.7.2 Tweets interactive multi-label classification

With the popularization of social media, the analysis of opinions has become a challenge for companies aiming at constantly improving their customer relationships. Many efforts are now on the development of automatic processing [Liu, 2012]. Moreover, we see the emergence of new systems of customer relantionship management [Ajmera et al., 2013]. The major drawback of social media is the unstructured and noisy properties of circulating data. These characteristics are especially found on Twitter which, despite a very recent attractiveness decline, remains a major site for opinions dissemination.
Recetly, VIPE system has been adapted to classify tweets by detecting the most relevant messages. To do this, the user defines its set of desired labels (Efficiency, Innovation, Problems, Negative), then he proceeds to the labeling of a small set of tweets to give positive and negative examples to the system. For example, he annotates the tweet "this is the # 4G:D" positively with two labels: Efficiency and Innovation and implicitly endorses negative with the other two labels: Problems and Negative. Based on this set of labeled messages, the learning algorithm helps the agents in foretelling the most probable labels for a selected tweet or the most likely tweets for a selected labels or a combination of them.

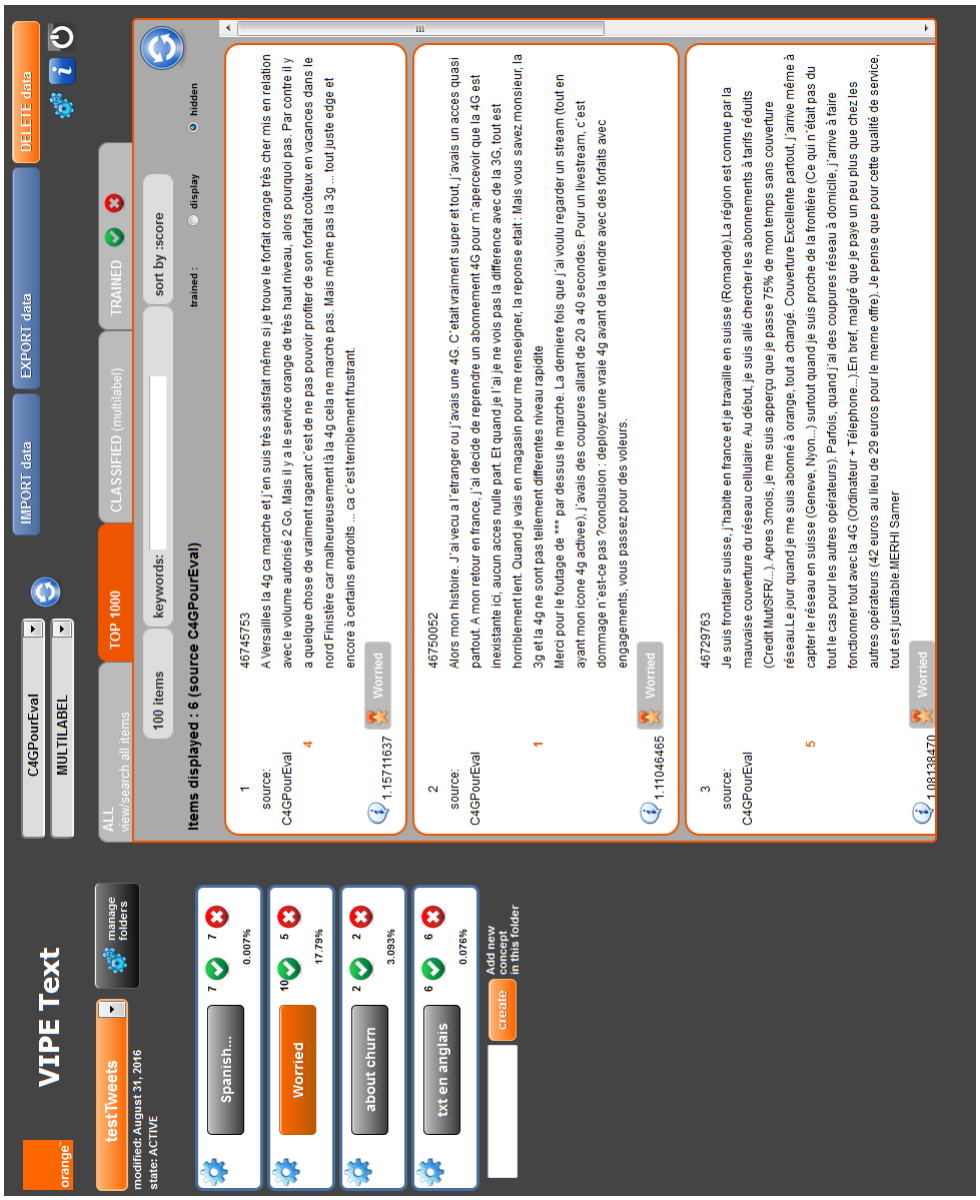


Figure 1.2: 4G feeling analysis

Figure 1.3: Vipe Interface for tweet classification

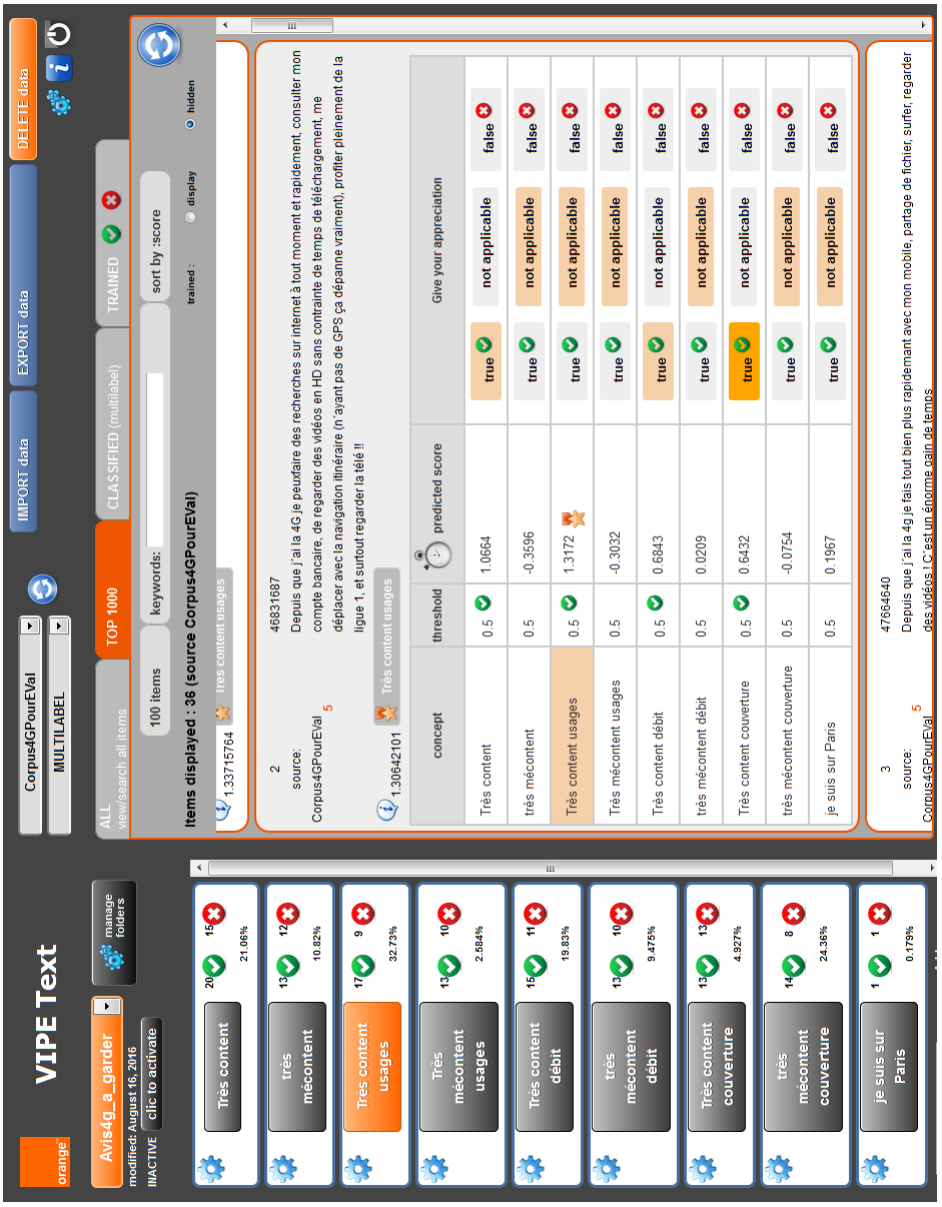


Figure 1.4: 4G tweets classification

Chapter 2

Introduction

In various real-world problems, objects mostly tend to be associated with multiple notions simultaneously instead of one. For instance, for a Video-On-Demand application, a movie might be annotated as both *romantic*, *funny*, *emotional* and *dramatic*. In image categorization tasks, a picture might contain different notions simultaneously like *mountains, trees, desert*. Then, in text mining, a document may treat several topics, such as *litterature*, *law* and *history*. This domain is known as *multi-label learning*. To address this issue, conventional classification based on a single label association has been extended to multi-label. Multi-label classification is getting a growing attention today due, in particular, to the expansion in online labeling services and the inflation in the volume of data.

Associated to the increasing importance of applications, various algorithms have been proposed (Zhang and Zhou, 2013; Madjarov et al.,2012; Tsoumakas and Katakis, 2007). Roughly speaking, they can be categorized in three main families:

- *problem transformation methods* transform the multi-label learning problem into one or several single label classification or regression problems.
- *ensemble methods* apply a collection of learners.
- *algorithm adaptation* methods adapt existing learning algorithms to learn from multi-label data.

The problem transformation methods were the first most popular approaches but the ensemble and the adaptation methods have proved interest. Among the latter, the adaptation of the well-known k-nearest neighbor algorithm is probably the most successful. Based on the maximum a posteriori principle, the multi-label k-nearest neighbor ML-kNN is one of the most efficient algorithms [Zhang and Zhou, 2007]. It operates instance-based learning and is able to outrun many model-based methods from the state of art approaches. However, the kNN principle suffers from a major drawback: its accuracy crucially depends on the metric function used to compute distances between instances. Typically, when no prior knowledge is assumed, kNN computes simple Euclidean distances. Unfortunately, such distance may overlook the usefulness of different irrelevant features

describing the handling instances. Metric learning has been integrated in the learning process to overcome this limit with significant improved results.

In this report, we propose a Mahalanobis distance metric learning method based on a kNN principle. We impose the matrix to be of low rank and thus considered as a dimensionality reduction metric. Our dimensionality reduction method is then combined with the ML-kNN algorithm. Our proposal solves an optimization problem by using a metaheuristic called Reduced Variable Neighborhood Search principle. To highlight our algorithm's efficiency, we make a comparative study with the state-of-art multi-label dimensionality reduction algorithm. We compare the impact of our dimensionality reduction with many approaches which cover the main different families of dimension reduction algorithms: Canonical Correlation Analysis (CCA), Multi-label Dimensionality reduction via Dependencies Maximization (MDDM) [Zhang and Zhou, 2010], the Partial Least Squares regression (PLS) [Maitra and Yan, 2008], or its Orhtonormal version (OPLS). The remainder of this report is as follow. Chapter III reviews previous approaches standing for the state-of-art multi-label dimensionality redution. We then introduce our proposal in chapter IV. Chapter V stands for an experimental study on several data sets. Chapter VI dicusses the results obtained according to several performance measures taken from the litterature. Chapter VII concludes by summarizing our main contributions and sketching many directions of ongoing research.

Chapter 3

Related Works

Over the last decade, researchers have focused on dimensionality reduction for multi-label classification problems. Several reduction methods were developped and coupled with classifier [Li et al., 2013] [Zhang and Zhou, 2010] [Yu et al., 2005], especially ML-kNN [Zhang and Zhou, 2007] and SVM [Godbole and Sarawagi, 2004], to improve their classification performance. In this section, we give a brief review on multi-label dimensionality reduction methods that were coupled with a kNN classifier. There are approaches in which dimensionality reduction methods is applied as an independent preprocessing on data before classification and others in which the two tasks are accomplished simultaneously.

3.1 Independent Reduction Methods

The independent methods can be categorized into three main types: The unsupervised reduction methods that usually tend to summarize the feature space while keeping a maximum of its structural information (such as features' covariance or co-occurence). The supervised reduction methods that aim at emphasizing the link between the projected features and the labels (based on dependence or covariance criteria). And finally, the weakly-supervised methods whose purpose is to reduce the feature space while preserving high order relationship between variables and/or satisfying distance constraints between instances.

In this report, we describe five methods that represent the three previous categories and that are widely used in multi-label dimensionality reduction comparative studies: Principal Component Analysis (PCA) [Abdi and Williams, 2010], Canonical Correlation Analysis (CCA) [Sun et al., 2011, Hotelling, 1936], Orthonormal Partial Least Square (OPLS) [Rosipal and Krämer, 2006], Multi-label Dimensionality reduction via Dependence Maximization (MDDM) [Zhang and Zhou, 2010] and Variable Pairwise Constraint projection for Multi-label Ensemble (VPCME) [Li et al., 2013].

Unsupervised feature space reduction methods can be borrowed from single-label problems as they only operate and focus on the feature space. There are many of them and the most used is

PCA [Abdi and Williams, 2010]. It seeks the principal components of the features' covariance and use them to project the feature space. This kind of approach, however, do not consider the links between features and labels. When the main purpose is classification, it is more common to guide dimensionality reduction with supervised information such as pairwise constraints or label informations.

To address this, supervised methods emerged such as CCA [Sun et al., 2011, Hotelling, 1936] and Partial Least Square (PLS) [Bishop, 2006]. These two approaches are commonly used in multi-label problems since they analyze and maximize correlation between two sets of variables.

PLS seeks the directions, in the feature space, that have a maximum covariance with the label space while CCA seeks directions in both label and feature space which have a maximum "canonical" correlations with each other. We consequently project the feature space into its best directions. These two methods have a close connection and their equivalence has been recently proved [Sun et al., 2009]. In this paper, we did not use PLS but Orthonormal PLS (OPLS), a variant of Partial Least Square that aims at obtaining orthonormal directions [Rosipal and Krämer, 2006].

Besides, MDDM, an approach based on Hilbert-Schmidt Independence Criterion (HSIC) [Zhang and Zhou, 2010], is also supervised. This algorithm finds a projection of the feature space that maximizes the HSIC between the projected data features and the labels.

Finally, VPCME is a weakly supervised method. This type of methods focus on high-order relationship between variables [Chen et al., 2009] or between instances [Li et al., 2013]. For example, VPCME computes a projection on the feature space that preserves a small distance between "Must-link" instances and a large distance between "Cannot-link" instances. The author proposed an automatic build of these constraints based on the percentage of common label between two instances. Formally, the algorithm seeks the projection that optimizes a tradeoff between minimizing the must-link set scatter and maximizing the cannot-link set scatter. The authors extended this method with a boosting-link strategy by learning several projection on bootstraps.

The five previously listed methods have been coupled with a kNN classifier and performed successfully in many real-world problems. Nevertheless, as they were used as an independant preprocessing before kNN classification, their optimization did not take the kNN performance into account. These methods' only objective is to optimize their own criteria (covariance, dependence, co-occurence) although it might be partially inconsistant with the principle of the classifier. Thus, the performance of the classification can be penalized.

3.2 Simultaneous Dimensionality Reduction and Multi-Label Classification

To tackle this problem, a few studies proposed a dimensionality reduction method that is aware of the classifier criterion [Guo and Schuurmans, 2012] [Ji and Ye, 2009]. These studies proposed approaches that learn dimensionality reduction and a classifier simultaneously in a global optimization problem. They either used an SVM classifier [Ji and Ye, 2009] or a large margin classifier [Guo and Schuurmans, 2012]. Their classification performances were increased compared to an approach with independant dimensionality reduction. However, we did not focus on these methods. We specifically targeted a kNN classifier since it does not need to learn, it is online

and it is naturally multi-label. Moreover, there might remain a weakness in their proposal. In [Guo and Schuurmans, 2012], in the optimization problem, they expressed the loss function as a sum of two reconstruction errors: dimensionality reduction and classification. As they jointly optimize these two inconsistant objectives, each objective cannot be optimally satistfied. In [Ji and Ye, 2009], they combined the two formulations which led to a two-parameter optimization problem that they solved alternatively.

In this report, we propose a novel dimensionality reduction approach with the projection as our unique parameter and the ML-kNN performance as our sole purpose. We compare our reduction method to the four methods PCA, CCA, OPLS and MDDM coupled with the ML-kNN classification.

Chapter 4

Proposal

4.1 Notations and Definitions

4.1.1 Multi-label Mining

Let us introduce notations first.
We consider $\mathcal{X} = \mathbb{R}^{d_x}$ (or $\mathbb{Z}^{d_x}$) to be the d_x-dimensional feature space and $\mathcal{Y} = \{0,1\}^{d_y}$ the labels' finite set which consists in d_y possible class labels.
An instance $x \in \mathcal{X}$, typically represented by a features vector $x = (x_1, x_2, ..., x_{d_x})$, is linked to a binary vector $Y = (y_1, y_2, ..., y_{d_y}) \in \mathcal{Y}$.
In order to build a predictive model and then evaluate it in term of performances, one needs to deal with two kinds of data sets:

- the training set $\mathcal{L} = \left\{\left(x_i^{\mathcal{L}}, Y_i^{\mathcal{L}}\right) \in \mathcal{X} \times \mathcal{Y} \mid i \in \{1, ..., N_{\mathcal{L}}\}\right\}$ of cardinality $N_{\mathcal{L}}$ used to tain the model.
- the testing one $\mathcal{T} = \left\{\left(x_i^{\mathcal{T}}, Y_i^{\mathcal{T}}\right) \in \mathcal{X} \times \mathcal{Y} \mid i \in \{1, ..., N_{\mathcal{T}}\}\right\}$ of cardinality $N_{\mathcal{T}}$ used to compute the performances of the model.

The predictive model wished to be well-learned consists in a multi-label classifier. It aims at mapping an instance x taken from $\mathcal{X}$ with the label vector $Y \in \mathcal{Y}$ that optimizes a specific loss function.
To do this, we need to compute a kind of confidence that the label y_l is a proper label of x. This can be done by defining a real-valued function $f : \mathcal{X} \times \mathcal{Y} \to [0,1]$ where $f(x,y)$ traduces the probability that the instance x is associated with the label y.
We denote by $f(x,\cdot)$, the real valued vector of dimension d_y carrying out the values $f(x,y)$ for a given instance x:

$$f(x,\cdot) = \left(f(x,y_1), ..., f(x,y_{d_y})\right) \in [0,1]^{d_y} \tag{4.1}$$

A successful learning classifier system will tend to output greater values for the correct labels than the incorrect ones, i.e:

$$f(x_i, y) \succ f(x_i, z) \Leftrightarrow y \in Y_i \wedge z \notin Y_i \tag{4.2}$$

Moreover, the fonction f can be transformed to a ranking bijection $rank(\cdot, \cdot)$, which maps the outputs given by $f(x_i, y)$ for any label y with an element from $\{1, 2, ..., d_y\}$ such that:

$$f(x_i, y) \succeq f(x_i, z) \Leftrightarrow rank(x_i, y) \preceq rank(x_i, z) \tag{4.3}$$

Furthermore, a multi-label classifier $h(\cdot)$ can be derived from the function f by choosing a threshold function $t(\cdot)$, where:

$$h(x_i) = \{y \mid f(x_i, y) \geq t(x_i) \wedge y \in \mathcal{Y}\} \tag{4.4}$$

Finally, we predict the labels vector $Z_i \in \mathcal{Y}$ of an unseen instance by exploiting the outputs of the classifier h:

$$Z_i = \big(\lfloor y_1 \in h(x_i) \rfloor, \lfloor y_2 \in h(x_i) \rfloor, ..., \lfloor y_{d_y} \in h(x_i) \rfloor\big) \tag{4.5}$$

where:

$$\lfloor \xi \rfloor = \begin{cases} 1 & \textit{when the event } \xi \textit{ holds} \\ 0 & \textit{otherwise} \end{cases}$$

4.1.2 Metric Learning via Supervised Dimensionality Reduction

Through supervised dimensionality reduction framework, one aims at finding a low dimensional representation maximizing the correlated information between the instances space $\mathcal{X}$ and the labels space $\mathcal{Y}$. This low dimensional representation can be expressed with a projection matrix $P \in \mathbb{R}^{r \times d_x}$ of low rank $r \ll d_x$.
So, let $\mathcal{P} = \big\{M \in \mathbb{R}^{r \times d_x} \mid rank(M) = r\big\}$ be the space containing all projection matrices of rank r. Then, we define the *projected* version of any collection of instances $\mathcal{I} = \{x_i \in \mathcal{X} \mid i \in [\![1, N_{\mathcal{I}}]\!]\}$ with regards to the r-rank matrix P: $\mathcal{I}^P = \{x \cdot P^\tau \in \mathbb{R}^r \mid x \in \mathcal{I}\}$.
To do this, we introduce the following operator:

$$\begin{array}{rccc} \langle \cdot \rangle_P & : & \mathbb{R}^{d_x} & \rightarrow & \mathbb{R}^r \\ & & \mathcal{I} = \{x_i \mid i \in [\![1, N_{\mathcal{I}}]\!]\} & \mapsto & \langle \mathcal{I} \rangle_P = \mathcal{I}^P = \{x \cdot P^\tau | x \in \mathcal{I}\} \end{array} \tag{4.6}$$

Based on the low rank matrix P, one can define a Mahalanobis (pseudo) distance [Joseph et al., 2013] $\| \cdot, \cdot \|_{\mathcal{M}}$ between two instances x and x' taken from $\mathcal{X}$:

$$\| x, x' \|_{\mathcal{M}} = \sqrt{(x - x') \cdot \mathcal{M} \cdot (x - x')^\tau} \tag{4.7}$$

where: $\mathcal{M} = P^\tau \cdot P \in \mathbb{R}^{d_x \times d_x}$ is a symmetric positive semi-definite (PSD) matrix of rank $r \prec d_x$.
We summarize all symbols and notations used in this report in Table 4.1.

4.2 ML-kNN Classifier

The main idea of this algorithm consists in adapting the k-nearest neighbor principle in order to deal with multi-label data, where a maximum a posteriori rule (MAP) is used to predict labels by exploiting the labeling pattern embodied in the instance neighborhood.

Notations	Mathematical Meanings
$\mathcal{X}$	d_x-dimensional instance space $\mathbb{R}^{d_x}$ (or $\mathbb{Z}^{d_x}$)
$\mathcal{Y}$	d_y possible class labels space $\{0,1\}^{d_y}$
x	d_x-dimensional feature vector $x = (x_1, x_2, ..., x_{d_x}) \in \mathcal{X}$
Y	correct label vector $Y = (y_1, y_2, ..., y_{d_y})$ associated with x
$\overline{Y}$	complementary vector of Y in $\mathcal{Y}$: $\overline{Y} = \underbrace{(1,1,..,1)}_{d_y} - Y$
d_x	number of features
d_y	number of labels (classes)
$x_i^{\mathcal{L}}$	training instance taken from $\mathcal{X}$
$Y_i^{\mathcal{L}}$	training label vector associated with $x_i^{\mathcal{L}}$
$x_i^{\mathcal{T}}$	testing instance taken from $\mathcal{X}$
$Y_i^{\mathcal{T}}$	testing label vector associated with $x_i^{\mathcal{T}}$
$\mathcal{L}$	multi-label training dataset $\left\{\left(x_i^{\mathcal{L}}, Y_i^{\mathcal{L}}\right) \middle\vert 1 \leq i \leq N_{\mathcal{L}}\right\}$
$\mathcal{S}_x^{\mathcal{L}}$	multi-label training instance set $\left\{x_i^{\mathcal{L}} \middle\vert 1 \leq i \leq N_{\mathcal{L}}\right\}$.
$\mathcal{S}_y^{\mathcal{L}}$	multi-label training label set $\left\{Y_i^{\mathcal{L}} \middle\vert 1 \leq i \leq N_{\mathcal{L}}\right\}$.
$\mathcal{T}$	multi-label testing dataset $\left\{\left(x_i^{\mathcal{T}}, Y_i^{\mathcal{T}}\right) \middle\vert 1 \leq i \leq N_{\mathcal{T}}\right\}$
$\mathcal{S}_x^{\mathcal{T}}$	multi-label testing instances set $\left\{x_i^{\mathcal{T}} \middle\vert 1 \leq i \leq N_{\mathcal{T}}\right\}$.
$\mathcal{S}_y^{\mathcal{T}}$	multi-label testing labels set $\left\{Y_i^{\mathcal{T}} \middle\vert 1 \leq i \leq N_{\mathcal{T}}\right\}$.
$N_{\mathcal{L}}$	training dataset cardinality
$N_{\mathcal{T}}$	test dataset cardinality
$h^{ML}(\cdot)$	ML-kNN classifier $h^{ML} : \mathcal{X} \to 2^{\mathcal{Y}}$, where $h^{ML}(x)$ returns the predicted labels vector for x
$h(\cdot)$	kNN classifier $h : \mathcal{X} \to 2^{\mathcal{Y}}$, where $h(x)$ returns the predicted labels vector for x
$f^{ML}(\cdot,\cdot)$	real-valued fonction resulting from the ML-kNN maximum a posteriori principle $f^{ML} : \mathcal{X} \times \mathcal{Y} \to [0,1]$ where $f^{ML}(x,y)$ returns the probability of y being a correct label of x
$f(\cdot,\cdot)$	real-valued fonction resulting from the kNN classifier $f : \mathcal{X} \times \mathcal{Y} \to [0,1]$ where $f(x,y)$ returns the probability of y being a correct label of x
$rank(\cdot,\cdot)$	ranking function which maps, based on the descending order induced from the outputs of $f(x_i,y)$ or $f^{ML}(x_i,y)$, any $y \in \mathcal{Y}$ to $\{1,2,...,d_y\}$
Z	predicted label vector $(z_1, z_2, ..., z_{d_y}) \in \mathcal{Y}$ associated with x by applying the ML-kNN or the kNN classifier
$\lfloor\cdot\rfloor$	where, for any event ξ, $\lfloor\xi\rfloor = 1$ *if* ξ *holds and* 0 *otherwise*
$\vert\cdot\vert$	$\vert Y = \left(y_1, ..., y_{d_y}\right)\vert$ returns the number of entries y_l equal to 1, i.e. $\vert Y\vert = \sum_{l=1}^{d_y} \lfloor y_l = 1 \rfloor$
$\{y \in Y\}$	for any $Y = \left(y_1, ..., y_{d_y}\right)$, return the labels $\{y_i \mid y_i = 1, i \in [\![1, d_y]\!]\}$
$\mathcal{P}$	Space containing all projection matrices of rank r: $\left\{M \in \mathbb{R}^{r \times d_x} \mid rank(M) = r\right\}$
P	$r \times d_x$ projection matrix of rank $r \prec d_x$
$\mathcal{M}$	$\mathcal{M} = P^\tau \cdot P$ Mahalanobis PSD matrix of rank $r \prec d_x$
$\Vert\cdot,\cdot\Vert_{\mathcal{M}}$	Mahalanobis distance with regards to the PSD matrix $\mathcal{M}$
$\langle\mathcal{I}\rangle_P$	*Projected* version of any collection of instances $\mathcal{I}$ taken from $\mathcal{X}$.

Table 4.1: Notations Summary

Let $\mathfrak{N}(x)$ describes the set containing the k nearest neighbors of an unseen instance x taken from the training data set $\mathcal{L}$. To identify the neighborhood, we generally deal with the well-known Euclidean distance with regards to the Euclidean properties of the instance space. ML-kNN calculates for any class label $l \in \{1, 2, ..., d_y\}$ the following amount:

$$C_l = \sum_{(x^+, Y^+) \in \mathfrak{N}(x)} \lfloor y_l \in Y^+ \rfloor \tag{4.8}$$

Basically, the quantity C_l records the number of x's neighbors with label l.

We denote by H_l^1 the event that the instance x has label l and $\mathbb{P}(H_l^1|C_l)$ represents the posterior probability that H_l^1 is true when x has exactly C_l neighbors associated to the label l.
Subsequently, H_l^0 represents the event that the instance x don't have the label l and $\mathbb{P}(H_l^0|C_l) = 1 - \mathbb{P}(H_l^1|C_l)$ represents the posterior probability that H_l^0 holds under the same condition.
To predict the label vector Y of an instance x, a decision is done by applying the *maximum a posteriori* rule (MAP) for each label y_i:

$$\begin{cases} Y = \left[y_1, ..., y_{d_y}\right] \in \mathcal{Y} \\ y_l = \lfloor \frac{\mathbb{P}(H_l^1|C_l)}{\mathbb{P}(H_l^0|C_l)} \succ 1 \rfloor \quad \forall l \in \{1, ..., d_y\} \end{cases} \tag{4.9}$$

According to Bayes theorem, we deduce that:

$$\frac{\mathbb{P}(H_l^1|C_l)}{\mathbb{P}(H_l^0|C_l)} = \frac{\mathbb{P}(H_l^1) \cdot \mathbb{P}(C_l|H_l^1)}{\mathbb{P}(H_l^0) \cdot \mathbb{P}(C_l|H_l^0)} \tag{4.10}$$

In Eq (4.10), $\mathbb{P}(H_l^1)$ (or $\mathbb{P}(H_l^0)$) represents the prior probability that the event H_l^1 holds (or H_l^0 holds). These prior probabilities are estimated by counting the number of training examples associated with each label l according to equations (4.11) and (4.12):

$$\mathbb{P}(H_l^1) = \frac{s + \sum_{i=1}^{i=N_\mathcal{L}} \lfloor y_l \in Y_i \rfloor}{s \times 2 + N_\mathcal{L}} \tag{4.11}$$

$$\mathbb{P}(H_l^0) = 1 - \mathbb{P}(H_l^1) \quad \forall l \in \{1, .., d_y\} \tag{4.12}$$

Where s is the Laplace smoothing parameter controlling the uniform prior effect on the estimation. This is the first step of the ML-kNN algorithm.

The second step consists in computing the two quantities $\mathbb{P}(C_l|H_l^j)$ $(j \in \{0, 1\})$ representing the likelihood that the instance x has exactly C_l neighbors with label l when the event H_l^j holds.
To do this, we define two $d_y \times (k+1)$ *frequency matrices* Υ^0 and Υ^1 within the equations (4.13) and (4.14):

$\forall (l, \kappa) \in \{1, ..., d_y\} \times \{0, ..., k\}$

$$\Upsilon_{l,\kappa}^1 = \sum_{i=1}^{N_\mathcal{L}} \lfloor y_l \in Y_i \rfloor \cdot \lfloor \epsilon_l(x_i) = \kappa \rfloor \tag{4.13}$$

$$\Upsilon^0_{l,\kappa} = \sum_{i=1}^{N_{\mathcal{L}}} \lfloor y_l \notin Y_i \rfloor \cdot \lfloor \epsilon_l(x_i) = \kappa \rfloor \tag{4.14}$$

where $\epsilon_l(x_i) = \sum_{(x^+,Y^+)\in\mathfrak{N}(x_i)} \lfloor y_l \in Y^+ \rfloor$

The quantities $\epsilon_l(x_i)$ mark the number of x_i's neighbors associated with the label y_l. By this way, $\Upsilon^1_{l,\kappa}$ records the exact number of training instances associated with the label y_l having exactly κ neighbors with label y_l.
Correspondingly, $\Upsilon^0_{l,\kappa}$ records the exact number of training instances not associated with the label y_l having exactly κ neighbors with label y_l.
Subsequently, the likelihoods $\mathbb{P}(C_l|H^j_l)$ $(j \in \{0,1\})$ are computed according to the equation (4.15):

$$\mathbb{P}(C_l|H^j_l) = \frac{s + \Upsilon^j_{l,C_l}}{s \times (k+1) + \sum_{\kappa=0}^{k} \Upsilon^j_{l,\kappa}} \qquad (j,l,C_l) \in \{0,1\} \times \{1,...,d_y\} \times \{0,...,k\} \tag{4.15}$$

This step conclude the ML-kNN training phase.
Then, the ML-kNN testing phase only consists in substituting Eq (4.12), Eq (4.11) and Eq (4.15) into Eq (4.10).
Therefore, the labels vector prediction step is naturally-done by following equation (4.9). We summarize the ML-kNN algorithm in Fig. 4.1 and Fig. 4.2.

4.3 ML-ARP: Multi-Label Adaptative Random Projections

4.3.1 Backgrounds

Traditionally, according to Johnson-Lindenstrauss lemma [Dasgupta and Gupta, 2003], random projection methods are used in order to preserve - up to a small distorsion - pairwise l_2 distances between instances.
In mono-label classification, these techniques have been called several times in processing of both noisy and noiseless images and also in text documents information retrieval [Bingham and Mannila, 2001].
In multi-label mining, projection techniques have mainly been used in order to reduce the space representation dimentionality while the pairwise distances are carefully preserved. Among these applications, one may found:

- ***labels random projection*** (output space), where in [Wan et al., 2016], small random subset of labels are coupled with a single-label classifier in order to build the RAKEL (RAndom k-labELsets) multi-label classifier.
- ***features random projection*** (input space), where in [Ran and Oh, 2015] sparse random projections are used in order to optimize wireless sensor networks.

Algorithm 1 Y=ML-kNN($\mathcal{L},\ k,\ x$)

Inputs:

- $\mathcal{L} = \left\{\left(x_i^{\mathcal{L}}, Y_i^{\mathcal{L}}\right) \in \mathcal{X} \times \mathcal{Y} \mid i \in \{1, ..., N_{\mathcal{L}}\}\right\}$: multi-label training dataset of cardinality $N_{\mathcal{L}}$
- k: Neighborhood size
- $x \in \mathcal{X}$: testing instance

Training Step

- *Computing Prior Probabilities*

  for $l = 1$ **to** d_y **do**
  Estimate $\mathbb{P}(H_l^1)$ and $\mathbb{P}(H_l^0)$ according to Eq.(4.11) and Eq.(4.12)
  endfor

- *Computing neighborhood*

  for $i = 1$ **to** $N_{\mathcal{L}}$ **do**
  Identify the k nearest neighbors $\mathfrak{N}(x_i^{\mathcal{L}})$ for $x_i^{\mathcal{L}}$
  endfor

- *Computing Posterior Probabilities*

  for $\kappa = 0, l = 1$ to k, d_y **do**
  Update frequency matrices $\Upsilon_{l,\kappa}^0$ and $\Upsilon_{l,\kappa}^1$ according to Eq.(4.14) and (4.13).
  endfor

Testing Step

- Identify the k nearest neighbors $\mathfrak{N}(x)$ for x.
- **for** $l = 1$ **to** d_y **do**
  Calculate the statistics C_l according to Eq.(4.8)
  endfor
- Compute Y by applying Eq.(4.9) (in conjuction with Eq.(4.11), Eq.(4.12), Eq.(4.15) and Eq.(4.10))

Output:

- $Y \in \mathcal{Y}$: Predicted label vector for the instance x.

Figure 4.1: Pseudo-code of the ML-kNN algorithm applied on an unseen instance x

Algorithm 2 $\mathcal{S}_z$=ML-kNN($\mathcal{L},\ k,\ \mathcal{S}_x$)

Inputs:

- $\mathcal{L} = \{(x_i^{\mathcal{L}}, Y_i^{\mathcal{L}}) \in \mathcal{X} \times \mathcal{Y} \mid i \in \{1, ..., N_{\mathcal{L}}\}\}$: multi-label training dataset of cardinality $N_{\mathcal{L}}$
- k: Neighborhood size
- $\mathcal{S}_x = \{x_i \mid i \in \{1, ..., N_{\mathcal{S}}\}\} \subset \mathcal{X}$: testing set

Process:

for $i = 1$ **to** $N_{\mathcal{S}}$ **do**
Y_i = ML-kNN($\mathcal{L},\ k,\ x_i$) (according to Fig.4.1)
endfor

Output:

- $\mathcal{S}_z = \{Y_i \mid i \in \{1, ..., N_{\mathcal{S}}\}\} \subset \mathcal{Y}$: Predicted label vector set

Figure 4.2: Pseudo-code of the ML-kNN algorithm applied on a set of unseen instances $\mathcal{S}_x$

4.3.2 ML-ARP

In order to improve the performances of the ML-kNN classifier, we need to find a better instance space representation than the original one. In other words, we seek to build a new instance space in which the ML-kNN classifier performances become optimal.
For this issue, we propose to resolve the following problem:

$$\min_{P \in \mathcal{P}} \Theta(P) = \min_{P \in \mathcal{P}} \parallel \mathcal{S}_y^{\mathcal{L}}, \mathcal{S}_z^{\mathcal{L}} = MLkNN(\mathcal{L}^P, k, \langle \mathcal{S}_x^{\mathcal{L}} \rangle_P) \parallel \tag{4.16}$$

where:

- Θ: denotes the reconstruction error between the true training set of labels $\mathcal{S}_y^{\mathcal{L}}$ and $\mathcal{S}_z^{\mathcal{L}}$ the set of labels predicted with Ml-kNN on the P-projected space.
- P: the low rank projection matrix $P \in \mathcal{P}$ traducing the transition between the original space $\mathcal{X}$ and the reduced dimensional space $\mathcal{X}^P = \mathbb{R}^r$.
- $\parallel \cdot, \cdot \parallel$: denote the Hamming Loss measure (Eq.(5.1)).
- k: the instances neighborhood size used as the second input parameter in the ML-kNN algorithm (see 4.2).
- $\langle \mathcal{S}_x^{\mathcal{L}} \rangle_P$: projected version of the training instance set $\mathcal{S}_x^{\mathcal{L}}$ with regards to the projection matrix P according to Eq.(4.6).
- $\mathcal{L}^P$: denotes the projected version of the training set $\mathcal{L}$ according to Eq.(4.17):

$$\mathcal{L}^P = \{(x_i^{\mathcal{L}} \cdot P^\tau, Y_i^{\mathcal{L}}) \in \mathcal{X}^P \times \mathcal{Y} \mid i \in \{1, ..., N_{\mathcal{L}}\}\} \tag{4.17}$$

Because of the nonconvex properties of the problem 4.16, we propose to follow an heuristic optimization way via the Reduced Variable Neighborhood Search (RVNS) [Xiao et al., 2011].
The RVNS principle revolves around iteratively and ramdomly changing the parameter P and validating the changes that improve the objective function. We can summarize the main steps of the RVNS principle applied on the problem 4.16 as follows:

1. Draw a current solution P from $\mathcal{P}$.
2. Make a slightly modification of the solution P into a new solution P' using a sparse speed matrix ϑ (See Algorithm 4.3.2).
3. Evaluate both P and P' according to the reconstruction error $\Theta(P)$ and $\Theta(P')$.
4. If $\Theta(P')$ is lesser than $\Theta(P)$, then P' become the new current solution, otherwise P is kept.
5. Repeat the steps 2., 3. and 4. until convergence.

This process is repeated many times, generally, several hundred times. In our framework, the RVNS principle is applied to the search of a good projection according to the performances of the ML-kNN classifier using this projection as a new representation space. Thus, we learn the projection globally and naturally in a supervised way.
We choose the first solution of the optimization process to be a random projection drawn from a zero-mean, unit-variance gaussian distribution. By this way, we progressively adapt the initial random projection to the handled multi-label dataset. We call this method the Multi-Label Adaptative Random Projection (ML-ARP).
It should be noted that ML-ARP don't rely on the Johnson-Lindenstrauss lemma. Its objective is far from preserving geometric distances of the original space $\mathcal{X}$. Contrariwise, ML-ARP uses random projections are used to deform the original space in furtherance of improving the ML-kNN accuracy. Thusly, we can easily override the k-nearest neighbors methods drawback.
Effectively, ML-ARP approach provides us with a new instance space representation caracterized by the transition matrix P which can be used to build a Mahalanobis pseudo-distance. Instead of computing a low accuracy Euclidean distance between two instances x and x' taken from the original space $\mathcal{X}$, we propose to compute the distance between their respective projected versions $x \cdot P^\tau$ and $x' \cdot P^\tau$ according to Eq.(4.7).
Thus, this guarantees that ML-kNN accuracy is widely improved.

Algorithm 3 $\vartheta = SpeedMatrix(r,\ d_x)$

Inputs:

- r: Row number
- d_x: Column number

Process:

$MutationRate \sim \mathcal{U}_{[0,1]}$
for $i = 1$ **to** r **do**
  for $j = 1$ **to** d_x **do**
  $val \sim \mathcal{U}_{[0,1]}$
    if$(val < MutationRate)$ **do**
      $\vartheta_{i,j} \sim \mathcal{N}(0, 1)$
    else
      $\vartheta_{i,j} \leftarrow 0$
    endIf
  endFor
endFor

Output:

- $\vartheta \in \mathcal{P}$: Sparse speed matrix.

Figure 4.3: Algorithm to draw a sparse speed matrix

4.4 ML-ARP Algorithm

Algorithm 4 P^*=ML-ARP($\mathcal{L}$, k, r, $PerfMesure$, $StopCondition$)

Inputs:

- $\mathcal{L} = \left\{\left(x_i^{\mathcal{L}}, Y_i^{\mathcal{L}}\right) \in \mathcal{X} \times \mathcal{Y} \mid i \in \{1, ..., N_{\mathcal{L}}\}\right\}$: multi-label training dataset of cardinality $N_{\mathcal{L}}$ divided into:
  - $\mathcal{S}_x^{\mathcal{L}} = \left\{x_i^{\mathcal{L}} \in \mathcal{X} \mid i \in \{1, ..., N_{\mathcal{L}}\}\right\}$: set of training instances.
  - $\mathcal{S}_y^{\mathcal{L}} = \left\{Y_i^{\mathcal{L}} \in \mathcal{Y} \mid i \in \{1, ..., N_{\mathcal{L}}\}\right\}$: set of true labels.
- k: Neighborhood size
- r: Reduced dimension
- $PerfMesure$: A multi-label performance measure taken from 5.2 denoted by $\| \cdot, \cdot \|$.
- $StopCondition$: One of the following conditions:
  - Maximum of the CPU time allowed.
  - Maximum number of iterations allowed.

Initialization

- Draw $P \in \mathcal{P}$ such that: $P_{i,j} \sim \mathcal{N}(0,1) \quad \forall (i,j) \in [\![1, r]\!] \times [\![1, d_x]\!]$
- Normalize each row of P according to an unit Euclidean norm. $(P_{i,\cdot} \cdot P_{i,\cdot}^{\tau} = 1)$
- Project the set $\mathcal{S}_x^{\mathcal{L}}$ into P to get $\langle \mathcal{S}_x^{\mathcal{L}} \rangle_P$ according to Eq:4.6.
- Get the projected version of $\mathcal{L}$ denoted by $\mathcal{L}^P$ according to Eq.4.17.
- Perform a ML-kNN process (according to Fig 4.2) with $\mathcal{L}^P$, k and $\langle \mathcal{S}_x^{\mathcal{L}} \rangle_P$ as inputs. Recover $\mathcal{S}_z^{\mathcal{L}}$ as output.
- Compute the actual reconstruction error: $actual\ error = \| \mathcal{S}_y^{\mathcal{L}}, \mathcal{S}_z^{\mathcal{L}} \|$.

Learning Step

- Draw a speed matrix ϑ according to fig. 4.3.2.
- **while**($StopCondition$ not reached) **do**
  - Compute $P^* = P + \vartheta$
  - Normalize each row of P^* according to an unit Euclidean norm. $(P_{i,\cdot}^* \cdot P_{i,\cdot}^{*\tau} = 1)$
  - Project the set $\mathcal{S}_x^{\mathcal{L}}$ into P^* to get $\langle \mathcal{S}_x^{\mathcal{L}} \rangle_{P^*}$ according to Eq:4.6.
  - Get the projected version of $\mathcal{L}$ denoted by $\mathcal{L}^{P^*}$ according to Eq.4.17.
  - Perform a ML-kNN process (according to Fig 4.2) with $\mathcal{L}^{P^*}$, k and $\langle \mathcal{S}_x^{\mathcal{L}} \rangle_{P^*}$ as inputs. Recover $\mathcal{S}_z^{\mathcal{L}*}$ as output.
  - Compute the new reconstruction error $new\ error = \| \mathcal{S}_y^{\mathcal{L}}, \mathcal{S}_z^{\mathcal{L}*} \|$.

- **if**$(new\ error < actual\ error)$ **do**
  - $actual\ error \leftarrow new\ error$.
  - $P \leftarrow P^*$
- **else do**
  - Draw a speed matrix ϑ according to fig. 4.3.2.
- **endIf**
- **endWhile**

Output:

- $P^* \in \mathcal{P}$: Projection matrix of rank r solution of the problem 4.16.

Chapter 5

Experiments

5.1 Data sets

In order to evaluate the performances of multi-label classifiers, one need to define some real-world data set taken from varions domains such as: music, images, text and medical.

5.1.1 Yeast Gene Functional Analysis

The goal of studying the Yeast dataset [Elisseeff and Weston, 2001] is to predict the gene functional classes of the Yeast *Saccharomyces cerevisiae*. Each gene is described by the concatenation of micro-array expression data and phylogenetic profile and is associated with a set of functional classes of cardinal 14. The multi-label data set contains 2417 genes (instances) represented by a 103-dimensional real-valued feature vectors.

5.1.2 Emotions

This is a small datatset [Trohidis et al., 2008] where each instance describes a piece of music by 71-dimensional real-valued feature vector. Each of the 593 instances linked to 6 possible emotions (labels): *sad-lonely, angry-aggressive, amazed-surprised, relaxing-calm, quiet-still* and *happy-pleased*.

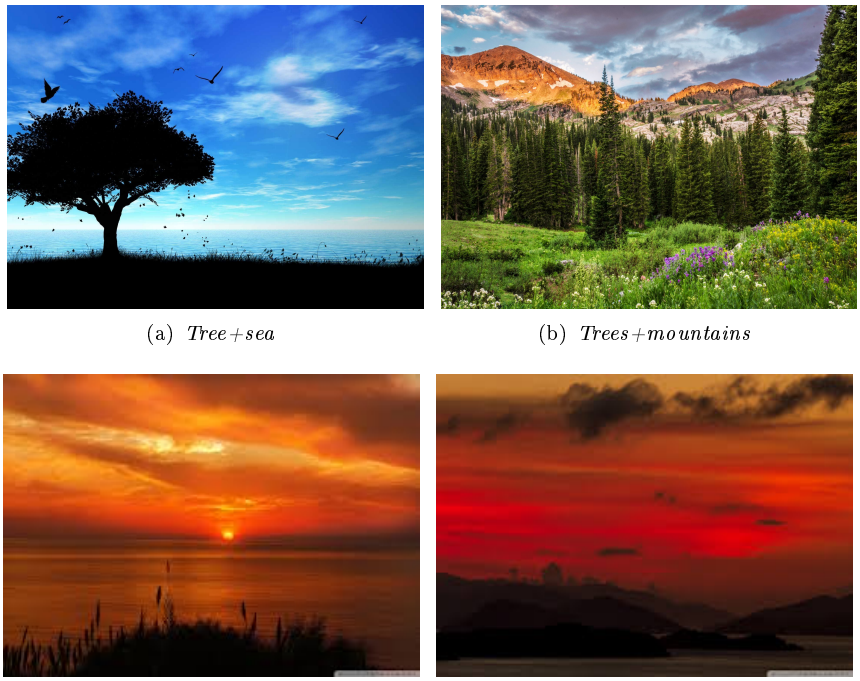
5.1.3 Mediamill

Mediamill dataset [Sorower, 2010] is a multimedia dataset firstly introduced in a generic video indexing challenge problem that decomposes the problem into unimodal and multimodal video analysis [Snoek et al., 2006]. The set contains 85 hours of international broadcast news data, divided

into 43907 instances described by 120-deimensional real-valued feature vectors, categrorized into 101 class labels.

5.1.4 Natural Scene

In natural scene categorization [Boutell et al., 2004], each scene picture belongs to several image classes simultaneously. For instance, the image in figure 5.1(a) can be classified as a tree scene as well as a sea scene, while image in figure 5.1(b) may be categorized as a tree scene as well as mountain scene. The picture in figure 5.1(c) is considered as both sunset and sea scene, while the one in figure 5.1(d) belongs to both sunset and mountain categories. The natural scene multi-label data set consists in 2407 examples represented by a 294-dimensional real-valued feature vectors which are categorized into 6 possible class labels: *sea, sunset, montain, desert, tree* and *urban*.



(a) *Tree+sea* (b) *Trees+mountains*

(c) *Sunset+Sea* (d) *Sunset+mountain*

Figure 5.1: Examples of multi-labelled images

5.1.5 Corel5k

The Corel5k [Ponce et al., 2006] data set contains images from the Stock Photo Cds. Each image is annotated with 1 to 5 keywords which are considered as labels. There is 374 possible keywords. The data set consists in 5000 real-valued feature vectors of 499 dimensions.

5.1.6 Delicious

Delicious dataset [Bi and Kwok, 2013] is a collection of 16105 textual data from webpages along with their associated 983 tags. Each instance is caracterized by a 500 binary features vector. Such web data is collected from a social bookmarking website namely, *del.icio.us*, data collected until April,2007.

5.1.7 Enron

Enron [Klimt and Yang, 2004] is a sparse dataset where 1702 emails are defined by 1001 binary features. They can be tagged with up to 53 genres.

5.1.8 Genbase

Genbase [Diplaris et al., 2005] is another microbiological dataset where the genes are described with 1186 binary features. They can be associated with 27 biological functions.

5.1.9 Medical

The biomedical dataset [Read et al., 2009] is taken from the Computational Medical Center [Crammer et al., 2007] which organizes Medical NLP Challenge with a rich set if medical text corpus. This dataset consists in a collection of patient symptom histories, diagnosis and prognoses reports to the insurance companies. There is 978 instances described by 1449 binary vectors which can be tagged up to 45 labels.

5.1.10 Bibtex

Bibtex dataset [Katakis et al., 2008] contains metadata for the bibtex items like the title of the paper, the authors, the keywords,etc. It consists in 7395 examples catacterized by 1836-dimensional binary features vectors which can be tagged up to 159 different labels.

5.1.11 Bookmarks

The bookmarks [Levatić et al., 2015] dataset contains metadata for bookmark items such as the URL of the web page, a description of the web page. The dataset consists in 87856 examples described by 2150 binary features vectors which are linked to 208 labels.

5.1.12 Reuters (RCV1V2S1)

Reuters Corpus Volume 1 [Ghamrawi and McCallum, 2005] consists in a large volume text corpus which exceeds 800000 newswise stories, collected and manualy organized by Reuters Ltd. for their research purposes [Lewis et al., 2004]. Significant efforts have been made in order to clean this text data with all sorts of text processing techniques, including the most famous ones: removing stop

words, stemming, transformation to TF-IDF [Al-Talib and Hassan,] and also normalization to get the new version of the Reuters data set (RCV1V2). Then, the dataset is decomposed into 5 subsets. In this paper, we are only dealing with the first subset (RCV1V2S1), consisting in 6000 real valued features vectors of dimension 47236 which are associated with 101 different tags.

Data sets description summary

We recall all the experimental data set descriptions [Luaces et al., 2012] in Table 5.1.

	Domain	# instances	# training s.	# testing s.	# features	# labels	labels density
Yeast	genetic	2417	2173	244	103	14	0.3
Emotions	audio	593	533	60	72	6	0.31
Mediamill	video	43907	39516	4391	120	101	0.043
Scene	images	2407	2166	241	294	6	0.18
Corel5k	images	5000	4500	500	499	374	0.0094
Delicious	text(tags)	16105	14495	1610	500	983	0.019
Enron	text	1702	1531	171	1001	53	0.064
Genbase	biology	662	595	67	1186	27	0.05
Medical	health	978	880	98	1449	45	0.0027
Bibtex	text	7395	6656	739	1836	159	0.015
Bookmarks	text	87856	79070	8786	2150	208	0.0098
Reuters	text	6000	5400	600	47229	101	0.026

Table 5.1: Datasets description summary

5.2 Performance measures

In order to quantify the classification quality of any multi-label classifier, we define a set of measures calculated between:

- the set of true labels vectors $\mathcal{S}_y = \{Y_i \in \mathcal{Y} \mid i \in \{1, ..., n\}\}$.
- the set of predicted labels vectors $\mathcal{S}_z = \{Z_i \in \mathcal{Y} \mid i \in \{1, ..., n\}\}$.
- the set of labels' confidence vectors $\mathcal{S}_f = \{f(x_i, \cdot) \in \mathbb{R}^{d_y} \mid i \in \{1, ..., n\}\}$.

5.2.1 Hamming Loss ↓

Hamming loss measure evaluates how many time on average a label not belonging to the instance is predicted and vice versa.

$$HammingLoss\,(\mathcal{S}_y, \mathcal{S}_z) = \frac{1}{n}\sum_{i=1}^{i=n}\frac{1}{d_y}|Y_i \Delta Z_i| \tag{5.1}$$

where Δ stands for the symmetric difference between two sets, which is, in Boolean logic, the exclusive disjunction set-theoretic equivalent.
The lesser the hamming loss, the better the classification performance.

5.2.2 Jaccard Loss ↓

Jaccard loss is the average across all instances of the jaccard distance between the predicted labels vector Z and the correct labels vector Y of a given instance.

$$JaccardLoss\left(\mathcal{S}_y, \mathcal{S}_z\right) = \frac{1}{n}\sum_{i=1}^{i=n}\frac{|\overline{Y}_i \cap Z_i| + |Y_i \cap \overline{Z}_i|}{|\overline{Y}_i \cap Z_i| + |Y_i \cap \overline{Z}_i| + |Y_i \cap Z_i|} \tag{5.2}$$

The lesser the jaccard loss, the better the classification performance.

5.2.3 Ranking Loss ↓

Ranking loss is the average across all instances of the average fraction of label pairs that are misordered for the handled instances.

$$RankingLoss\left(\mathcal{S}_y, \mathcal{S}_f\right) = \frac{1}{n}\sum_{i=1}^{i=n}\frac{1}{|Y_i||\overline{Y}_i|}|\left\{\left(y_1, y_2\right)|f\left(x_i, y_1\right) \leq f\left(x_i, y_2\right), \left(y_1, y_2\right) \in Y_i \times \overline{Y}_i\right\}| \tag{5.3}$$

The lesser the ranking loss, the better the classification performance.

5.2.4 One Error ↓

One-error describes how many times the best predicted label in term of ranking is misclassified.

$$OneError\left(\mathcal{S}_y, \mathcal{S}_f\right) = \frac{1}{n}\sum_{i=1}^{i=n}\lfloor[argmax_{y \in Y}\, f\left(x_i, y\right)] \notin Y_i\rfloor \tag{5.4}$$

5.2.5 Coverage ↓

Coverage describes on the average how far we need to go down the ordered list of labels in term of relevancy to cover all non-zero labels of the instance.

$$Coverage\left(\mathcal{S}_y, \mathcal{S}_f\right) = \frac{1}{n}\sum_{i=1}^{i=n} max_{y \in Y_i} rank\left(x_i, y\right) - 1 \tag{5.5}$$

The lesser the coverage, the better the classification performance.

5.2.6 Root Mean Square Error (RMSE) ↓

We define the *root mean square error* as the average over all instances of the Euclidean distance between the correct labels vector and the predicted real values vector of an instance.

$$RMSE\left(\mathcal{S}_y, \mathcal{S}_f\right) = \frac{1}{n}\sum_{i=1}^{i=n}\sqrt{\sum_{y\in Y_i}\left(y - f\left(x_i, y\right)\right)^2} \tag{5.6}$$

5.2.7 Accuracy ↑

We define the *accuracy* as the average across all instances of the proportion of the predicted correct labels for an instance to the total number of labels (predicted and correct) for the handled instance.

$$Accuracy\left(\mathcal{S}_y, \mathcal{S}_z\right) = \frac{1}{n}\sum_{i=1}^{i=n}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|} \tag{5.7}$$

The higher the accuracy, the better the classification performance.

5.2.8 Recall ↑

We define the *recall* as the average over all instances of the proportion of predicted correct labels to the total number of correct labels.

$$Recall\left(\mathcal{S}_y, \mathcal{S}_z\right) = \frac{1}{n}\sum_{i=1}^{i=n}\frac{|Y_i \cap Z_i|}{|Y_i|} \tag{5.8}$$

The higher the recall, the better the classification performance.

5.2.9 Precision ↑

We define the *precision* as the average over all instances of the proportion of predicted correct labels to the total number of predicted labels.

$$Precision\left(\mathcal{S}_y, \mathcal{S}_z\right) = \frac{1}{n}\sum_{i=1}^{i=n}\frac{|Y_i \cap Z_i|}{|Z_i|} \tag{5.9}$$

The higher the precision, the better the classification performance.

5.2.10 Subset Accuracy ↑

We define the *subset accuracy* as the average over all instances of the number of totally correct predicted labels vector.

$$SubsetAccuracy\left(\mathcal{S}_y, \mathcal{S}_z\right) = \frac{1}{n}\sum_{i=1}^{i=n}\lfloor Y_i = Z_i \rfloor \tag{5.10}$$

The higher the subset accuracy, the better the classification performance.

5.2.11 Average Precision ↑

Average precision evaluates the average over all instances of the average fraction of labels ranked above a given label $y \in Y$ which actually belong to Y.

$$AveragePrecision\left(\mathcal{S}_y, \mathcal{S}_f\right) = \frac{1}{n}\sum_{i=1}^{i=n}\frac{1}{|Y_i|}\sum_{y\in Y_i}\frac{\left|\left\{y'|rank\left(x_i,y'\right) \leq rank\left(x_i,y\right), y' \in Y_i\right\}\right|}{rank\left(x_i,y\right)} \tag{5.11}$$

The higher the average precision, the better the classification performance.

5.2.12 F_1-Measure ↑

We define the F_1-*Measure* as the harmonic mean of *precision* and *recall* measures.

$$F_1\left(\mathcal{S}_y, \mathcal{S}_z\right) = \frac{1}{n}\sum_{i=1}^{i=n}\frac{2\cdot precision\cdot recall}{precision + recall} = \frac{1}{n}\sum_{i=1}^{i=n}\frac{2|Y_i \cap Z_i|}{|Z_i| + |Y_i|} \tag{5.12}$$

which is a particular case of a more generalized score: the F_β-score

$$F_\beta = \frac{1}{n}\sum_{i=1}^{i=n}(1+\beta^2)\cdot\frac{precision\cdot recall}{(\beta^2\cdot precision) + recall} \tag{5.13}$$

The higher the F_1-Measure, the better the classification performance.

5.3 Experimental protocol

In order to:

- evaluate the performances of the learned model from an available dataset and gauge the generalizability of the model.
- make a comparison study of different algorithms and find out the best one.

we perform a 10-fold cross validation paradigm which is traditionally done by following the following steps:

- Partition the dataset $\mathcal{D}$ into 10 equally (or nearly equally) sized folds $\mathcal{F}_i$ ($\mathcal{D} = \bigcup_{i=1}^{10}\mathcal{F}_i$)
- Repeat 10 times the following instructions:
  - Choose a fold $\mathcal{F}_i$ to be the test set $\mathcal{T}$ containing:
    - the testing instances set $\mathcal{S}_x^{\mathcal{T}}$.
    - the labels set $\mathcal{S}_y^{\mathcal{T}}$,
  - merge the 9 remaining folds to build the learning dataset $\mathcal{L}$.

  - Learn the model by using $\mathcal{L}$.
  - Apply the learned model on the testing instances set $\mathcal{S}_x^{\mathcal{T}}$ to recover the set of predicted labels $\mathcal{S}_z^{\mathcal{T}}$
  - Compute the performance measures $Perfs$ (see 5.2) on the true labels set $\mathcal{S}_y^{\mathcal{T}}$ and the predicted one $\mathcal{S}_z^{\mathcal{T}}$.

- Compute the means and the standard deviations of the computed performances $Perfs$.

Chapter 6

Results and Discussions

6.1 Results on data sets

In what follows, a 10-fold cross validation has been performed on the 12 choosen datasets according to the previous experimental protocol 5.3.

Notations meaning

We briefly recall the meaning of the initials used from table 6.1 to table 6.12.

- *Baseline:* computes the frequency of each label in the training set. When an unseen instance comes, it returns the most frequent label.
  There is no metric learning or dimensionality reduction preprocessing to do before applying the baseline.
- *ML-kNN:* Direct application of the ML-kNN algorithm according to fig.4.1 and fig.4.2.
- *RP:* Draw a zero-mean, unit-variance gaussian random projection matrix P_{RP} of rank r. Then, application of the ML-kNN algorithm on the projected version of the training set $\langle \mathcal{L} \rangle_{P_{RP}}$ and the testing set $\langle \mathcal{T} \rangle_{P_{RP}}$ with regards to the projection matrix P_{RP} (see Eq.4.17).
- *ML-ARP:* Learning the projection matrix P_{ML-ARP} according to fig.4. Then, application of the ML-kNN algorithm on the projected version of the training set $\langle \mathcal{L} \rangle_{P_{ML-ARP}}$ and the testing set $\langle \mathcal{T} \rangle_{P_{ML-ARP}}$ with regards to the projection matrix P_{ML-ARP} (see Eq.4.17).
- *MDDM:* Learning the projection matrix P_{MDDM} according to A.1. Then, application of the ML-kNN algorithm on the projected version of the training set $\langle \mathcal{L} \rangle_{P_{MDDM}}$ and the testing set $\langle \mathcal{T} \rangle_{P_{MDDM}}$ with regards to the projection matrix P_{MDDM} (see Eq.4.17).

- *PCA:* Learning the projection matrix P_{PCA} according to A.2. Then, application of the ML-kNN algorithm on the projected version of the training set $\langle \mathcal{L} \rangle_{P_{PCA}}$ and the testing set $\langle \mathcal{T} \rangle_{P_{PCA}}$ with regards to the projection matrix P_{PCA} (see Eq.4.17).
- *OPLS:* Learning the projection matrix P_{OPLS} according to A.4. Then, application of the ML-kNN algorithm on the projected version of the training set $\langle \mathcal{L} \rangle_{P_{OPLS}}$ and the testing set $\langle \mathcal{T} \rangle_{P_{OPLS}}$ with regards to the projection matrix P_{OPLS} (see Eq.4.17).
- *CCA:* Learning the projection matrix P_{CCA} according to A.3. Then, application of the ML-kNN algorithm on the projected version of the training set $\langle \mathcal{L} \rangle_{P_{CCA}}$ and the testing set $\langle \mathcal{T} \rangle_{P_{CCA}}$ with regards to the projection matrix P_{CCA} (see Eq.4.17).
- N\A: means that the algorithm cannot be applied on the given dataset because of one of the following reason:
  - Singularity of the $Y^\tau Y$ matrix in the generalized eigenvalue problem A.5.
  - Non-positiveness of the second term of the generalized eigenvalue problems (OPLS and CCA) A.5 and A.6.
  - Maximum CPU time or RAM memory allowed reached when resolving the eigenvalue problems A.2, A.4, A.5 and A.6.

6.1.1 Yeast

- $k = 5$: Neighborhood size.
- $r = 64$: Number of reduced dimension.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.335 ± 0.01	0.512 ± 0.0169	0.484 ± 0.013	**0.53** ± 0.014	0.376 ± 0.014	0.514 ± 0.0144	0.47 ± 0.011	0.471 ± 0.018
Average Precision	0.703 ± 0.01	0.759 ± 0.0165	0.750 ± 0.016	**0.761** ± 0.013	0.714 ± 0.01	0.759 ± 0.014	0.746 ± 0.015	0.746 ± 0.013
Coverage	6.79 ± 0.089	**6.34** ± 0.15	6.44 ± 0.116	6.37 ± 0.14	6.727 ± 0.075	6.35 ± 0.142	6.494 ± 0.134	6.496 ± 0.145
F1	0.228 ± 0.005	0.308 ± 0.008	0.295 ± 0.008	0.303 ± 0.007	0.246 ± 0.008	**0.309** ± 0.008	0.29 ± 0.006	0.29 ± 0.010
Hamming Loss	0.232 ± 0.003	0.195 ± 0.007	0.202 ± 0.006	**0.191** ± 0.005	0.227 ± 0.003	0.194 ± 0.007	0.203 ± 0.006	0.204 ± 0.005
Jaccard Loss	0.664 ± 0.01	0.487 ± 0.016	0.515 ± 0.013	0.49 ± 0.014	0.623 ± 0.014	**0.485** ± 0.014	0.529 ± 0.011	0.528 ± 0.018
One Error	0.247 ± 0.014	**0.238** ± 0.028	0.24 ± 0.023	0.24 ± 0.019	0.25 ± 0.016	**0.238** ± 0.02	0.24 ± 0.021	0.243 ± 0.018
Precision	**0.749** ± 0.015	0.716 ± 0.016	0.717 ± 0.019	0.717 ± 0.013	0.72 ± 0.019	0.721 ± 0.017	0.723 ± 0.012	0.725 ± 0.014
RMSE	0.327 ± 0.004	**0.309** ± 0.004	0.311 ± 0.001	0.310 ± 0.002	0.324 ± 0.004	**0.309** ± 0.004	0.313 ± 0.002	0.314 ± 0.001
Ranking Loss	0.211 ± 0.009	**0.172** ± 0.012	0.178 ± 0.011	0.174 ± 0.01	0.204 ± 0.009	**0.172** ± 0.012	0.182 ± 0.011	0.181 ± 0.011
Recall	0.335 ± 0.009	0.589 ± 0.024	0.547 ± 0.019	**0.59** ± 0.023	0.397 ± 0.022	**0.59** ± 0.022	0.528 ± 0.017	0.531 ± 0.025
Subset Accuracy	0.014 ± 0.008	0.185 ± 0.019	0.152 ± 0.016	**0.198** ± 0.017	0.056 ± 0.014	0.183 ± 0.0169	0.142 ± 0.011	0.139 ± 0.015

Table 6.1: Experimental Results on the Yeast data set

6.1.2 Emotions

- $k = 5$: Neighborhood size.
- $r = 64$: Number of reduced dimension.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.0 ± 0.0	0.374 ± 0.042	0.374 ± 0.04	**0.477** ± 0.063	0.109 ± 0.042	0.374 ± 0.042	0.359 ± 0.058	0.384 ± 0.065
Average Precision	0.578 ± 0.014	0.715 ± 0.023	0.718 ± 0.0225	**0.761** ± 0.037	0.592 ± 0.034	0.719 ± 0.023	0.716 ± 0.0437	0.725 ± 0.034
Coverage	3.15 ± 0.165	2.27 ± 0.128	2.27 ± 0.123	**2.002** ± 0.20	2.98 ± 0.19	2.278 ± 0.128	2.315 ± 0.218	2.272 ± 0.149
F1	0.0 ± 0.0	0.226 ± 0.025	0.22 ± 0.023	**0.277** ± 0.029	0.069 ± 0.027	0.226 ± 0.025	0.215 ± 0.031	0.229 ± 0.031
Hamming Loss	0.313 ± 0.002	0.262 ± 0.013	0.261 ± 0.02	**0.226** ± 0.022	0.31 ± 0.0189	0.262 ± 0.013	0.256 ± 0.0259	0.252 ± 0.029
Jaccard Loss	1.0 ± 0.0	0.625 ± 0.04	0.625 ± 0.04	**0.522** ± 0.063	0.89 ± 0.042	0.625 ± 0.042	0.64 ± 0.058	0.615 ± 0.065
One Error	0.542 ± 0.0327	0.374 ± 0.041	0.376 ± 0.047	**0.321** ± 0.059	0.553 ± 0.073	0.374 ± 0.041	0.374 ± 0.08	0.357 ± 0.06
Precision	0.0 ± 0.0	0.550 ± 0.058	0.554 ± 0.062	**0.627** ± 0.052	0.2 ± 0.085	0.55 ± 0.058	0.521 ± 0.06	0.544 ± 0.063
RMSE	0.431 ± 0.00049	0.41 ± 0.003	0.409 ± 0.004	**0.393** ± 0.01	0.43 ± 0.003	0.41 ± 0.003	0.409 ± 0.008	0.406 ± 0.008
Ranking Loss	0.412 ± 0.016	0.259 ± 0.025	0.256 ± 0.022	**0.207** ± 0.036	0.401 ± 0.034	0.259 ± 0.025	0.263 ± 0.046	0.249 ± 0.036
Recall	0.0 ± 0.0	0.421 ± 0.054	0.423 ± 0.05	**0.545** ± 0.064	0.111 ± 0.044	0.421 ± 0.054	0.402 ± 0.066	0.437 ± 0.06
Subset Accuracy	0.0 ± 0.0	0.154 ± 0.038	0.156 ± 0.026	**0.241** ± 0.083	0.033 ± 0.021	0.154 ± 0.038	0.154 ± 0.049	0.172 ± 0.07

Table 6.2: Experimental Results on the Emotions data set

6.1.3 Mediamill

- $k = 5$: Neighborhood size.
- $r = 64$: Number of reduced dimension.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.339 ± 0.004	0.4767 ± 0.003	0.469 ± 0.004	0.4825 ± 0.002	N\A	N\A	**0.518** ± 0.003	N\A
Average Precision	0.614 ± 0.002	0.7298 ± 0.002	0.726 ± 0.0035	0.7474 ± 0.0023	N\A	N\A	**0.758** ± 0.0029	N\A
Coverage	20.86 ± 0.156	14.18 ± 0.237	14.36 ± 0.30	14.09 ± 0.26	N\A	N\A	**12.55** ± 0.227	N\A
F1	0.223 ± 0.004	0.2910 ± 0.001	0.288 ± 0.002	0.299 ± 0.001	N\A	N\A	**0.307** ± 0.001	N\A
Hamming Loss	0.035 ± 0.003	0.027 ± 0.00023	0.0279 ± 0.00021	0.027 ± 0.00021	N\A	N\A	**0.025** ± 0.00014	N\A
Jaccard Loss	0.66 ± 0.009	0.5105 ± 0.003	0.517 ± 0.004	0.505 ± 0.003	N\A	N\A	**0.466** ± 0.003	N\A
One Error	0.23 ± 0.004	0.154 ± 0.005	0.157 ± 0.005	**0.1408** ± 0.006	N\A	N\A	0.145 ± 0.003	N\A
Precision	0.7 ± 0.008	0.7498 ± 0.005	0748 ± 0.005	0.761 ± 0.004	N\A	N\A	**0.764** ± 0.004	N\A
RMSE	0.127 ± 0.007	0.116 ± 0.00023	0.116 ± 0.00027	**0.104** ± 0.00022	N\A	N\A	0.112 ± 0.00018	N\A
Ranking Loss	0.064 ± 0.0089	0.037 ± 0.00085	0.0386 ± 0.001	0.037 ± 0.00087	N\A	N\A	**0.032** ± 0.000815	N\A
Recall	0.344 ± 0.04	0.526 ± 0.0028	0.519 ± 0.004	0.539 ± 0.003	N\A	N\A	**0.566** ± 0.003	N\A
Subset Accuracy	0.048 ± 0.0012	0.167 ± 0.0038	0.160 ± 0.004	0.171 ± 0.004	N\A	N\A	**0.217**± 0.005	N\A

Table 6.3: Experimental Results on the Mediamill data set

6.1.4 Scene

- $k = 5$: Neighborhood size.
- $r = 128$: Number of reduced dimension.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.0 ± 0.0	0.669 ± 0.03	0.425 ± 0.02	0.658 ± 0.024	**0.683** ± 0.025	0.64 ± 0.033	0.191 ± 0.0379	0.197 ± 0.027
Average Precision	0.426 ± 0.001	**0.859** ± 0.018	0.625 ± 0.014	0.851 ± 0.016	0.857 ± 0.016	0.845 ± 0.02	0.635 ± 0.021	0.643 ± 0.02
Coverage	2.366 ± 0.08	0.521 ± 0.07	0.621 ± 0.1	0.546 ± 0.06	**0.5** ± 0.054	0.526 ± 0.07	1.418 ± 0.094	1.390 ± 0.084
F1	0.0 ± 0.0	0.341 ± 0.016	0.269 ± 0.018	0.336 ± 0.012	**0.347** ± 0.013	0.326 ± 0.01	0.097 ± 0.019	0.100 ± 0.014
Hamming Loss	0.179 ± 0.004	**0.088** ± 0.008	0.108 ± 0.005	0.091 ± 0.007	0.089 ± 0.0077	0.097 ± 0.01	0.166 ± 0.006	0.162 ±0.004
Jaccard Loss	1.0 ± 0.0	0.330 ± 0.03	0.56 ± 0.054	0.341 ± 0.024	**0.316** ± 0.025	0.359 ± 0.03	0.808 ± 0.037	0.802 ± 0.027
One Error	0.781 ± 0.0012	**0.228** ± 0.027	0.29 ± 0.035	0.241 ± 0.025	0.239 ± 0.026	0.262 ± 0.035	0.545 ± 0.03	0.537 ±0.029
Precision	0.0 ± 0.0	0.697 ± 0.033	0.598 ± 0.032	0.684 ± 0.026	**0.710** ± 0.025	0.665 ± 0.034	0.199 ± 0.037	0.205 ± 0.029
RMSE	0.394 ± 0.002	**0.289** ± 0.007	0.315 ± 0.0057	0.292 ± 0.005	**0.289** ± 0.0057	0.298 ± 0.007	0.372 ± 0.0036	0.371 ± 0.003
Ranking Loss	0.485 ± 0.0085	0.086 ± 0.014	0.251 ± 0.009	0.091 ± 0.013	**0.083** ± 0.012	0.088 ± 0.014	0.264 ± 0.018	0.259 ± 0.016
Recall	0.0 ± 0.0	0.684 ± 0.0325	0.184 ± 0.014	0.673 ± 0.023	**0.691** ± 0.027	0.653 ± 0.035	0.194 ± 0.038	0.199 ±0.028
Subset Accuracy	0.0 ± 0.0	0.627 ± 0.0288	0.516 ± 0.063	0.618 ± 0.025	**0.648** ± 0.025	0.602 ± 0.032	0.181 ± 0.037	0.186 ± 0.024

Table 6.4: Experimental Results on the Scene data set

6.1.5 Corel5k

- $k = 5$: Neighborhood size.
- $r = 128$: Number of reduced dimension.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.0 ± 0.0	0.016 ± 0.0036	0.01 ± 0.0017	0.01 ± 0.0038	**0.03** ± 0.0059	0.01 ± 0.002	0.029 ± 0.007	N\A
Average Precision	**0.290** ± 0.005	0.251 ± 0.006	0.241 ± 0.01	0.245 ± 0.0057	0.273 ± 0.008	0.254 ± 0.007	0.260 ± 0.007	N\A
Coverage	**48.13** ± 2.1	105.6 ± 3.24	109.61 ± 3.12	108.76 ± 3.1	107.09 ± 2.96	110.097 ± 4.17	106.16 ± 2.43	N\A
F1	0.0 ± 0.0	0.010 ± 0.0022	0.007 ± 0.0011	0.0069 ± 0.002	**0.0194** ± 0.0035	0.007 ± 0.0014	0.019 ± 0.004	N\A
Hamming Loss	**0.009** ± 0.0	**0.009** ± 0.0	0.009 ± 0.0	**0.009** ± 0.0	**0.009** ± 0.0	**0.009** ± 0.0	**0.009** ± 0.0	N\A
Jaccard Loss	1.0 ± 0.0	0.983 ± 0.0036	0.988 ± 0.0017	0.989 ± 0.0038	**0.969** ± 0.0059	0.989 ± 0.002	0.970 ± 0.007	N\A
One Error	0.776 ± 0.008	0.737 ± 0.009	0.749 ± 0.02	0.738 ± 0.012	**0.707** ± 0.017	0.727 ± 0.011	0.724 ± 0.014	N\A
Precision	0.0 ± 0.0	0.031 ± 0.007	0.0265 ± 0.004	0.024 ± 0.0065	0.059 ± 0.01	0.023 ± 0.005	**0.060** ± 0.013	N\A
RMSE	**0.067** ± 0.0	0.069 ± 0.0	0.069 ± 0.0	0.069 ± 0.0	0.069 ± 0.0	0.069 ± 0.0	0.069 ± 0.0	N\A
Ranking Loss	0.147 ± 0.0021	0.139 ± 0.0036	0.141 ± 0.004	0.140 ± 0.003	**0.134** ± 0.003	0.136 ± 0.003	0.137 ± 0.003	N\A
Recall	0.0 ± 0.0	0.017 ± 0.0037	0.012 ± 0.0018	0.01 ± 0.003	**0.031** ± 0.006	0.011 ± 0.002	**0.031** ± 0.007	N\A
Subset Accuracy	0.0 ± 0.0	0.005 ± 0.0024	0.0006 ± 0.0009	0.001 ± 0.0009	**0.007** ± 0.0049	0.0008 ± 0.0009	0.005 ± 0.003	N\A

Table 6.5: Experimental Results on the Corel5k data set

6.1.6 Delicious

- $k = 5$: Neighborhood size.
- $r = 128$: Number of reduced dimension.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.0 ± 0.0	0.104 ± 0.002	0.292 ± 0.01	**0.119** ± 0.0037	0.115 ± 0.002	0.101 ± 0.002	N/A	N/A
Average Precision	**0.419** ± 0.011	0.326 ± 0.0037	0.567 ± 0.01	0.319 ± 0.0038	0.337 ± 0.002	0.324 ± 0.002	N/A	N/A
Coverage	**393.19** ± 4.3	626.73 ± 8.19	20.31 ± 0.75	623.65 ± 8.35	618.01 ± 6.61	627.92 ± 5.13	N/A	N/A
F1	0.0 ± 0.0	0.081 ± 0.0019	0.183 ± 0.0059	0.076 ± 0.003	**0.089** ± 0.001	0.079 ± 0.002	N/A	N/A
Hamming Loss	0.0193 ± 0.0	**0.014** ± 0.0	0.02 ± 0.0	**0.014** ± 0.0	0.018 ± 0.0	0.018 ± 0.0	N/A	N/A
Jaccard Loss	0.999 ± 0.0005	0.894 ± 0.0029	0.625 ± 0.009	0.902 ± 0.0038	**0.884** ± 0.002	0.897 ± 0.0029	N/A	N/A
One Error	0.596 ± 0.009	0.402 ± 0.01	0.379 ± 0.012	0.414 ± 0.013	**0.389** ± 0.004	0.406 ± 0.0069	N/A	N/A
Precision	0.0 ± 0.0	0.457 ± 0.006	0.472 ± 0.012	0.443 ± 0.015	**0.477** ± 0.011	0.440 ± 0.01	N/A	N/A
RMSE	0.064 ± 0.000046	0.063 ± 0.00006	0.102 ± 0.0003	**0.062** ± 0.00004	**0.062** ± 0.0	0.063 ± 0.0	N/A	N/A
Ranking Loss	0.173 ± 0.0029	0.135 ± 0.002	0.0596 ± 0.003	0.137 ± 0.002	**0.131** ± 0.002	0.135 ± 0.002	N/A	N/A
Recall	0.0 ± 0.0	0.114 ± 0.003	0.33 ± 0.011	**0.105** ± 0.004	0.126 ± 0.002	0.111 ± 0.003	N/A	N/A
Subset Accuracy	0.0008 ± 0.00049	**0.002** ± 0.001	0.137 ± 0.006	**0.002** ± 0.0006	**0.002** ± 0.0007	**0.002** ± 0.00099	N/A	N/A

Table 6.6: Experimental Results on the Delicious data set

6.1.7 Enron

- $k = 5$: Neighborhood size.
- $r = 128$: Number of reduced dimension.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.222 ± 0.038	0.328 ± 0.018	0.314 ± 0.015	0.359 ± 0.024	**0.397** ± 0.039	0.387 ± 0.038	0.3108 ± 0.034	0.318 ± 0.016
Average Precision	0.539 ± 0.015	0.627 ± 0.017	0.619 ± 0.017	0.637 ± 0.016	**0.666** ± 0.018	0.661 ± 0.018	0.531 ± 0.022	0.542 ± 0.014
Coverage	13.55 ± 0.62	13.68 ± 0.547	14.02 ± 0.657	14.01 ± 0.657	**12.77** ± 0.76	13.28 ± 0.62	15.024 ± 0.67	14.86 ± 0.67
F1	0.165 ± 0.026	0.213 ± 0.009	0.207 ± 0.008	0.231 ± 0.013	**0.251** ± 0.019	0.247 ± 0.019	0.209 ± 0.018	0.213 ± 0.009
Hamming Loss	0.062 ± 0.001	0.051 ± 0.001	0.053 ± 0.001	0.05 ± 0.001	**0.049** ± 0.001	**0.049** ± 0.001	0.067 ± 0.003	0.064 ± 0.002
Jaccard Loss	0.777 ± 0.038	0.671 ± 0.018	0.685 ± 0.015	0.64 ± 0.02	**0.602** ± 0.039	0.612 ± 0.038	0.689 ± 0.034	0.681 ± 0.016
One Error	0.463 ± 0.02	0.309 ± 0.028	0.33± 0.028	0.288 ± 0.03	**0.260** ± 0.028	0.253 ± 0.023	0.500 ± 0.037	0.475 ± 0.036
Precision	0.519 ± 0.015	0.579 ± 0.025	0.564 ± 0.024	0.6117 ± 0.02	0.629 ± 0.03	**0.632** ± 0.038	0.477 ± 0.04	0.488 ± 0.026
RMSE	0.186 ± 0.004	0.163 ± 0.0	0.164 ± 0.0	**0.162** ± 0.0009	0.174 ± 0.004	0.175 ± 0.023	0.176 ± 0.0016	0.175 ± 0.001
Ranking Loss	0.118 ± 0.009	0.096 ± 0.007	0.1 ± 0.007	0.099 ± 0.007	**0.088** ± 0.008	0.091 ± 0.007	0.117 ± 0.009	0.115 ± 0.009
Recall	0.257 ± 0.055	0.371 ± 0.02	0.359 ± 0.016	0.409 ± 0.03	**0.457** ± 0.04	0.442 ± 0.04	0.418 ± 0.04	0.422 ± 0.025
Subset Accuracy	0.001 ± 0.002	0.072 ± 0.018	0.06 ± 0.017	0.078 ± 0.022	**0.104** ± 0.032	0.088 ± 0.027	0.048 ± 0.025	0.054 ± 0.016

Table 6.7: Experimental Results on the Enron data set

6.1.8 Genbase

- $k = 5$: Neighborhood size.
- $r = 128$: Number of reduced dimension.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ±	0.0 ±	0.0 ±	N\A
Average Precision	0.423 ± 0.013	**0.427** ± 0.012	0.427± 0.012	**0.427** ± 0.012	**0.427** ±	**0.427** ±	**0.427** ±	N\A
Coverage	**4.241** ± 0.26	4.365 ± 0.27	4.365 ± 0.27	4.365 ± 0.27	4.365 ±	4.365 ±	4.365 ±	N\A
F1	0.0 ± 0.0	0.0 ± 0.0	0.0± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	N\A
Hamming Loss	**0.047** ± 0.001	**0.047** ± 0.001	0.047± 0.001	**0.047** ± 0.001	**0.047** ± 0.001	**0.047** ± 0.001	**0.047** ± 0.001	N\A
Jaccard Loss	1.0 ± 0.0	1.0 ± 0.0	1.0± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	N\A
One Error	**0.752** ± 0.015	**0.752** ± 0.015	0.752± 0.015	**0.752** ± 0.015	**0.752** ± 0.015	**0.752** ± 0.015	**0.752** ± 0.015	N\A
Precision	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	N\A
RMSE	**0.195** ± 0.002	**0.195** ± 0.002	0.195± 0.002	**0.195** ± 0.002	**0.195** ± 0.002	**0.195** ± 0.002	**0.195** ± 0.002	N\A
Ranking Loss	**0.156** ± 0.005	**0.156** ± 0.005	0.156± 0.005	**0.156** ± 0.005	**0.156** ± 0.005	**0.156** ± 0.005	**0.156** ± 0.005	N\A
Recall	0.0 ± 0.0	0.0 ± 0.0	0.0± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	N\A
Subset Accuracy	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	N\A

Table 6.8: Experimental Results on the Genbase data set

6.1.9 Medical

- $k = 5$: Neighborhood size.
- $r = 128$: Number of reduced dimension.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.0 ± 0.0	0.612 ± 0.052	0.535 ± 0.033	0.580 ± 0.05	**0.679** ± 0.042	0.621 ± 0.041	N\A	N\A
Average Precision	0.399 ± 0.025	0.795 ± 0.035	0.752 ± 0.022	0.777 ± 0.026	**0.829** ± 0.035	0.804 ± 0.035	N\A	N\A
Coverage	5.313 ± 0.57	3.198 ± 0.994	3.369 ± 0.521	3.144 ± 0.809	**2.462** ± 0.61	2.987 ± 0.64	N\A	N\A
F1	0.0 ± 0.0	0.321 ± 0.025	0.282 ± 0.016	0.305 ± 0.026	**0.354** ± 0.02	0.325 ± 0.02	N\A	N\A
Hamming Loss	0.028 ± 0.0	0.015 ± 0.002	0.018 ± 0.001	0.017 ± 0.001	**0.013** ± 0.001	0.015 ± 0.02	N\A	N\A
Jaccard Loss	1.0 ± 0.0	0.387 ± 0.05	0.464 ± 0.03	0.419 ± 0.053	**0.320** ± 0.042	0.378 ± 0.002	N\A	N\A
One Error	0.735 ± 0.0355	0.254 ± 0.043	0.315 ± 0.034	0.277 ± 0.0369	**0.210** ± 0.046	0.241 ± 0.04	N\A	N\A
Precision	0.0 ± 0.0	0.671 ± 0.048	0.593 ± 0.036	0.643 ± 0.0522	**0.738** ± 0.042	0.682 ± 0.04	N\A	N\A
RMSE	0.204 ± 0.007	0.159 ± 0.007	0.126 ± 0.002	**0.122** ± 0.004	0.148 ± 0.008	0.157 ± 0.04	N\A	N\A
Ranking Loss	0.145 ± 0.009	0.063 ± 0.014	0.067 ± 0.011	0.062 ± 0.011	**0.051** ± 0.011	0.057 ± 0.007	N\A	N\A
Recall	0.0 ± 0.0	0.643 ± 0.054	0.564 ± 0.033	0.605 ± 0.05	**0.709** ± 0.042	0.643 ± 0.01	N\A	N\A
Subset Accuracy	0.0 ± 0.0	0.519 ± 0.0539	0.445 ± 0.034	0.490 ± 0.057	**0.587** ± 0.046	0.537 ± 0.04	N\A	N\A

Table 6.9: Experimental Results on the Medical data set

6.1.10 Bibtex

- $k = 5$: Neighborhood size.
- $r = 128$: Number of reduced dimension.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.0 ± 0.0	0.14 ± 0.008	0.101 ± 0.008	0.122 ± 0.008	**0.305** ± 0.008	0.195 ± 0.006	N\A	N\A
Average Precision	0.203 ± 0.01	0.334 ± 0.0123	0.279 ± 0.0126	0.295 ± 0.01	**0.509** ± 0.01	0.391 ± 0.007	N\A	N\A
Coverage	42.13 ± 1.59	58.61 ± 2.02	64.46 ± 1.49	62.45 ± 1.02	**40.39** ± 1.32	50.94 ± 1.62	N\A	N\A
F1	0.0 ± 0.0	0.087 ± 0.004	0.063 ± 0.004	0.076 ± 0.0035	**0.178** ± 0.004	0.116 ± 0.03	N\A	N\A
Hamming Loss	0.015 ± 0.0001	0.0135 ± 0.0002	0.014 ± 0.0002	0.0139 ± 0.0002	**0.0123** ± 0.0002	0.013 ± 0.0002	N\A	N\A
Jaccard Loss	1.0 ± 0.0	0.859 ± 0.009	0.898 ± 0.008	0.877 ± 0.008	**0.694** ± 0.008	0.804 ± 0.006	N\A	N\A
One Error	0.853 ± 0.014	0.604 ± 0.019	0.681 ± 0.02	0.641 ± 0.018	**0.425** ± 0.017	0.552 ± 0.008	N\A	N\A
Precision	0.0 ± 0.0	0.270 ± 0.01	0.203 ± 0.01	0.241 ± 0.011	**0.456** ± 0.012	0.335 ± 0.01	N\A	N\A
RMSE	0.095 ± 0.0	0.091 ± 0.0	0.092 ± 0.0	0.092 ± 0.0	**0.086** ± 0.0004	0.089 ± 0.0003	N\A	N\A
Ranking Loss	0.330 ± 0.007	0.228 ± 0.006	0.258 ± 0.008	0.251 ± 0.008	**0.142** ± 0.006	0.194 ± 0.007	N\A	N\A
Recall	0.0 ± 0.0	0.144 ± 0.007	0.104 ± 0.008	0.124 ± 0.008	**0.327** ± 0.008	0.201 ± 0.006	N\A	N\A
Subset Accuracy	0.0 ± 0.0	0.059 ± 0.006	0.046 ± 0.006	0.055 ± 0.006	**0.172** ± 0.011	0.109 ± 0.0069	N\A	N\A

Table 6.10: Experimental Results on the Bibtex data set

6.1.11 Bookmarks

- $k = 5$: Neighborhood size.
- $r = 128$: Number of reduced dimension.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.0 ± 0.0	0.209 ± 0.0028	0.193 ± 0.0028	**0.210** ± 0.0029	N\A	N\A	N\A	N\A
Average Precision	0.158 ± 0.002	**0.376** ± 0.0029	0.346 ± 0.002	0.357 ± 0.0021	N\A	N\A	N\A	N\A
Coverage	66.61 ± 1.268	**56.75** ± 0.457	59.8 ± 0.5	59.99 ± 0.414	N\A	N\A	N\A	N\A
F1	0.0 ± 0.0	**0.107** ± 0.0014	0.098 ± 0.001	0.098 ± 0.001	N\A	N\A	N\A	N\A
Hamming Loss	0.009 ± 0.0	**0.008** ± 0.0	0.008 ± 0.0	**0.008** ± 0.0	N\A	N\A	N\A	N\A
Jaccard Loss	1.0 ± 0.0	0.790 ± 0.002	0.806 ± 0.002	**0.785** ± 0.002	N\A	N\A	N\A	N\A
One Error	0.922 ± 0.0019	**0.641** ± 0.003	0.679 ± 0.003	0.67 ± 0.003	N\A	N\A	N\A	N\A
Precision	0.0 ± 0.0	**0.228** ± 0.003	0.204 ± 0.003	0.225 ± 0.0	N\A	N\A	N\A	N\A
RMSE	0.079 ± 0.0	**0.073** ± 0.0	0.074 ± 0.0	**0.073** ± 0.001	N\A	N\A	N\A	N\A
Ranking Loss	0.263 ± 0.002	**0.187** ± 0.002	0.2 ± 0.002	0.201 ± 0.0029	N\A	N\A	N\A	N\A
Recall	0.0 ± 0.0	**0.214** ± 0.002	0.197 ± 0.003	0.197 ± 0.0032	N\A	N\A	N\A	N\A
Subset Accuracy	0.0 ± 0.0	0.191 ± 0.002	0.18 ± 0.002	**0.195** ± 0.003	N\A	N\A	N\A	N\A

Table 6.11: Experimental Results on the Bookmarks data set

6.1.12 Reuters subset 1

- $k = 5$: Neighborhood size.
- $r = 4000$: Number of reduced dimension.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.0 ± 0.0	**0.193** ± 0.0139	0.193 ± 0.014	0.189 ± 0.01	N\A	N\A	N\A	N\A
Average Precision	0.272 ± 0.012	**0.581** ± 0.01	0.568 ± 0.008	0.559 ± 0.012	N\A	N\A	N\A	N\A
Coverage	28.76 ± 1.529	**16.52** ± 0.825	17.34 ± 0.69	17.93 ± 0.83	N\A	N\A	N\A	N\A
F1	0.0 ± 0.0	**0.123** ± 0.008	0.121 ± 0.008	0.1192 ± 0.006	N\A	N\A	N\A	N\A
Hamming Loss	0.028 ± 0.0001	**0.026** ± 0.0004	0.026 ± 0.0004	**0.026** ± 0.0002	N\A	N\A	N\A	N\A
Jaccard Loss	1.0 ± 0.0	**0.806** ± 0.013	0.806 ± 0.014	0.810 ± 0.01	N\A	N\A	N\A	N\A
One Error	0.769 ± 0.017	**0.435** ± 0.014	0.459 ± 0.015	0.454 ± 0.019	N\A	N\A	N\A	N\A
Precision	0.0 ± 0.0	**0.359** ± 0.023	0.355 ± 0.019	0.355 ± 0.02	N\A	N\A	N\A	N\A
RMSE	0.129 ± 0.015	0.119 ± 0.001	0.116 ± 0.0005	**0.115** ± 0.001	N\A	N\A	N\A	N\A
Ranking Loss	0.167 0.003±	**0.071** ± 0.004	0.076 ± 0.003	0.080 ± 0.005	N\A	N\A	N\A	N\A
Recall	0.0 ± 0.0	**0.205** ± 0.016	0.203 ± 0.016	0.194 ± 0.01	N\A	N\A	N\A	N\A
Subset Accuracy	0.0 ± 0.0	0.062 ± 0.012	0.066 ± 0.007	**0.063** ± 0.007	N\A	N\A	N\A	N\A

Table 6.12: Experimental Results on the Reuters data set

6.2 Statistical Tests

In this section, we propose to run the Friedman statistical test on the previous results in order to test the two following hypothesis:

- H_0: all the algorithms are equivalent.
- H_1: at least one algorithm is different from the others.
  In this case, we aim at knowing the best and the worst algorithm by attributing to each algorithm a unique rank.
  The lesser the rank, the better the algorithm.

6.2.1 Friedman Test

The statistical Friedman test [Demšar, 2006] is a non-parametric test used in order to detect differences in several different treatments across multiple attempts. In our case, the treatments are the different compared algorithms. The attempts are their performance on the datasets (we only use RMSE and Hamming Loss which are most general reconstruction errors).
We briefly recall the main steps of the Friedman test.
First of all, let us introduce the notations for this purpose.
We consider N different algorithms caracterized by M different attempts (data sets).
Let $R_{i,j}$ be the rank attributed to the j^{th} of N algorithms applied on the i^{th} of M attempts. The main idea of the Friedman test revolves around comparing the average rank of each algorithm R_j:

$$R_j = \frac{1}{M} \cdot \sum_{i=1}^{M} R_{i,j} \tag{6.1}$$

We define the general average rank $\overline{R}$:

$$\overline{R} = \frac{1}{N \cdot M} \sum_{i=1}^{M} \sum_{j=1}^{N} R_{i,j} \tag{6.2}$$

Then, we define the sum of squares between traitments SS_t and the errors SS_e as:

$$\begin{aligned} SS_t &= M \cdot \sum_{j=1}^{N} (R_j - \overline{R})^2 \\ SS_e &= \frac{1}{M(N-1)} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(R_{i,j} - \overline{R}\right)^2 \end{aligned} \tag{6.3}$$

Finally, we compute the statistic test Q:

$$Q = \frac{SS_t}{SS_e} \tag{6.4}$$

Based on the previous quantities, one may compute the Friedman statistic:

$$\chi_f^2 = \frac{12M}{N\,(N+1)} \left(\sum_{j=1}^{N} R_j^2 - \frac{N(N+1)^2}{4} \right) \tag{6.5}$$

Under the null hypothesis H_0, χ_f^2 is distributed according to the χ_{N-1}^2 distribution [Plackett, 1983] with $N-1$ degrees of freedom, with the condition that both N and M are big enough (traditionnaly, $M > 10$ and $N > 5$).
When the previous condition is not satisfied, exact critical values have been proposed by [Sheskin, 2003]. Because of the undesirably conservative property of χ_f^2, a better statistic can be derived from Eq.(6.5) [Iman and Davenport, 1980]:

$$F_f = \frac{(M-1)\chi_f^2}{M(N-1) - \chi_f^2} \tag{6.6}$$

which is distributed according to the $F-$distribution [Ramirez et al., 2000] with $(N-1)$ and $(N-1)(M-1)$ as input parameters. To decide whether the null-hypothesis H_0 is rejected or not, we define the p-value p associated with the test as follow:

$$p = \mathbb{P}(F_f \geq Q) \tag{6.7}$$

Given a level of significance $\alpha \in [0\ 1]$ (as a rule of a thumb $\alpha = 0.01$ or $\alpha = 0.05$), we apply the following rule (Eq. 6.8) to reject or accept H_0:

$$\begin{cases} if\ p \leq \alpha & reject\ H_0 \\ if\ p \geq 1-\alpha & accept\ H_0 \end{cases} \tag{6.8}$$

Finally, when H_0 is rejected, one can proceed with a post-hoc test. The Nemenyi test [Šubelj et al., 2015] allows us to decide whether the mean rank difference between two algorithms is significant or not. For this purpose, we compute the critical distance $CritDist$ as follows [Demšar, 2006]:

$$CritDist = q_\alpha \sqrt{\frac{N(N+1)}{6N}} \tag{6.9}$$

where q_α represents the critical values for the two-tailed Nemenyi test.
Table 6.13 recalls the most important values of q_α for two cases of $\alpha \in \{0.01, 0.05\}$ [Demšar, 2006].

Number of algorithms	2	3	4	5	6	7	8
$q_{0.05}$	1.96	2.343	2.569	2.728	2.85	2.949	3.031
$q_{0.01}$	1.645	2.052	2.291	2.459	2.589	2.693	2.78

Table 6.13: Critical values for the two tailed Nemenyi test

Finally, we apply the following rule (Eq.6.10) to decide whether two algorithms are significanlty different or not:

$$\begin{cases} if\ |R_i - R_j| \geq CritDist & Then\ the\ i^{th}\ and\ j^{th}\ algorithms\ are\ different \\ if\ |R_i - R_j| \leq CritDist & Then\ the\ i^{th}\ and\ j^{th}\ algorithms\ are\ equivalent \end{cases} \tag{6.10}$$

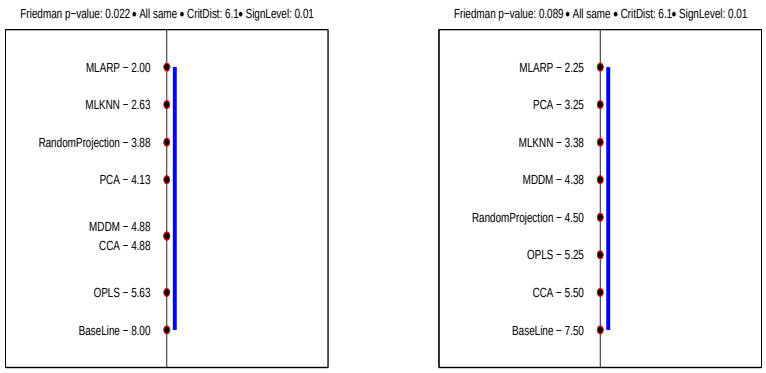
6.2.2 Statistical Results

Because of the non appliable properties of some algorithms (MDDM, CCA, OPLS, PCA), we choose to follow an ascendent order where, at first we only use the friedman teston the datasets where all algorithms are computable, then we add little by little datasets and remove little by little the non appliable algorithm from the process. Doing this, we have identified five different statistical tests to run.

Warning
There is no evidence of significant differences for algorithms joined by the vertical lines. Depending on the model we are dealing with, different groups can be formed. These groups are identified using the mean rank of an algorithm $\pm$ the critical distance ($CritDist$). One should notice that the algorithms are ranked by putting the best one in the top and naturally the worst one in the bottom.

Test 1

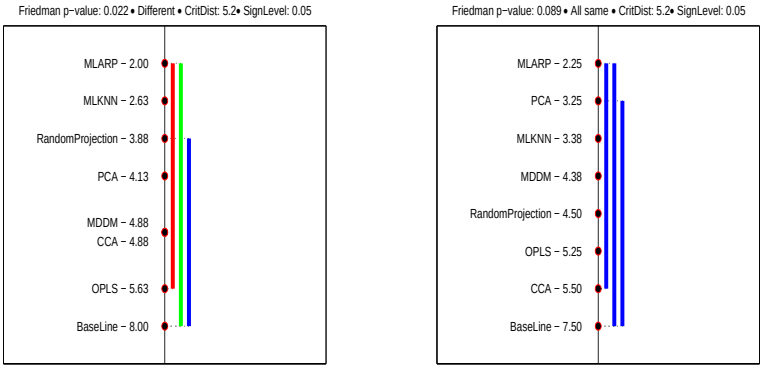
In this test, all the eight algorithms are choosen. They are appliable on only four datasets (Yeast, Emotions, Scene, Enron).



(a) *RMSE* (b) *Hamming loss*

Figure 6.1: Nemenyi test for $\alpha = 0.01$

Based on figure 6.1, we notice that for a significance level $\alpha = 0.01$, all the algorithms are equivalent. There is no reason to choose a particular approach than another. Nonetheless, ML-ARP comes first in term of mean rank (2).



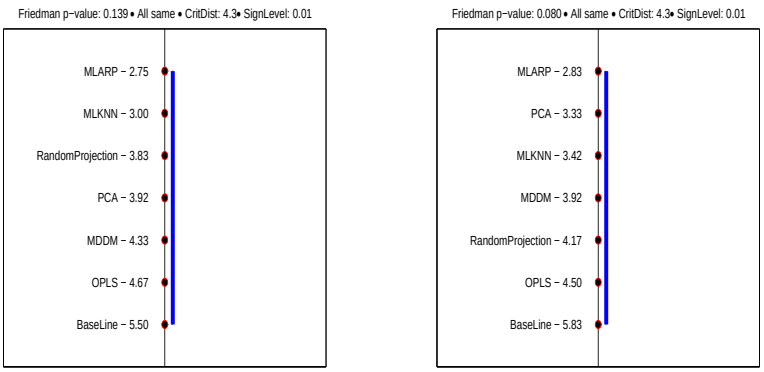
(a) *RMSE* (b) *Hamming loss*

Figure 6.2: Nemenyi test for $\alpha = 0.05$

From figure 6.2(a), we have $p < \alpha$, so the null hypothesis H_0 is totally rejected. Therefore, we can form at least two different groups of algorithms. A group consisting in the baseline and the second group consisting in the remaining algorithms. Because of its bad mean rank, the baseline is the worst classifier. It would never be choosen. Then, we notice that there is an equivalence between the remaining seven algorithms, but ML-ARP comes first again. So, one will tend to choose ML-ARP.

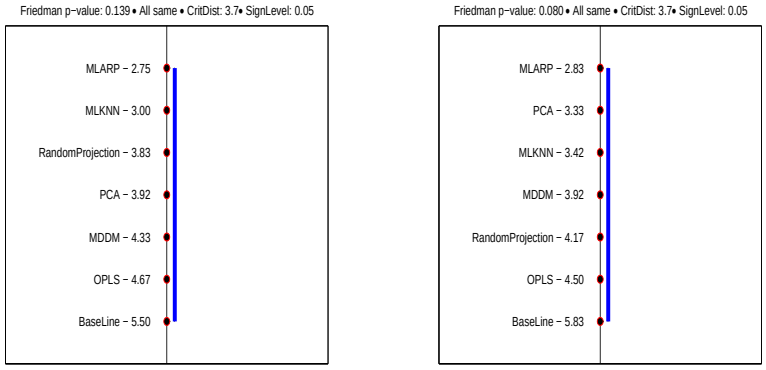
Test 2

In this test, we cancel CCA algorithm, so we add two datasets to the test (corel5k, genbase).



(a) *RMSE* (b) *Hamming loss*

Figure 6.3: Nemenyi test for $\alpha = 0.01$



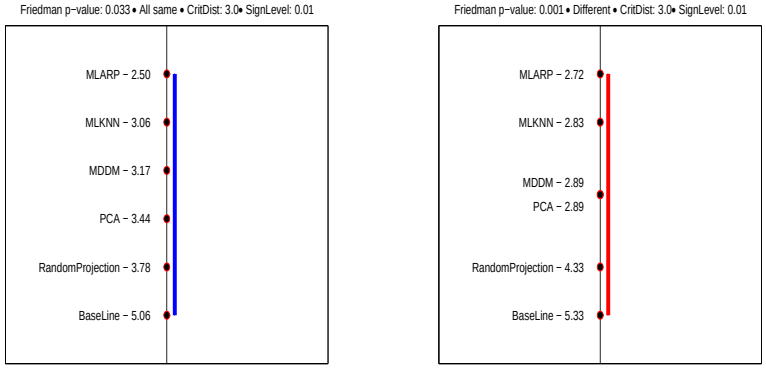
(a) *RMSE* (b) *Hamming loss*

Figure 6.4: Nemenyi test for $\alpha = 0.05$

From figures 6.3 & 6.4, the tested algorithms are equivalent with some preference for the ML-ARP because of its well mean rank (it comes first once again).

Test 3

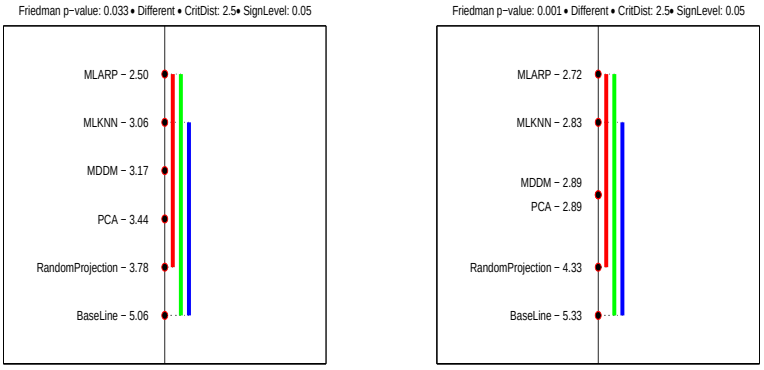
In this test, we cancel the OPLS algorithm. So we add the following datasets to the test: delicious, bibtex, medical.



(a) *RMSE* (b) *Hamming loss*

Figure 6.5: Nemenyi test for $\alpha = 0.01$

Figure 6.5(b) shows that the null hypothesis H_0 is rejected for a significance level of 0.01. Because of its well mean rank, ML-ARP is statistically the best one.



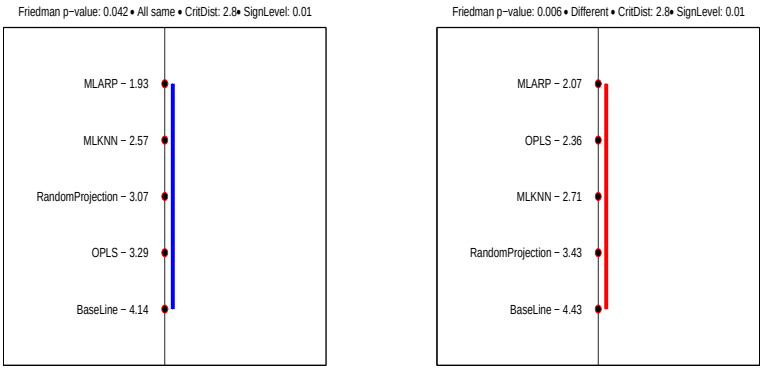
(a) *RMSE* (b) *Hamming loss*

Figure 6.6: Nemenyi test for $\alpha = 0.05$

Both figures 6.6(a) and (b) shows that the null hypothesis H_0 is rejected for a significance level of 0.05. We should notice that ML-ARP stands out from the other algorithms and comes first.

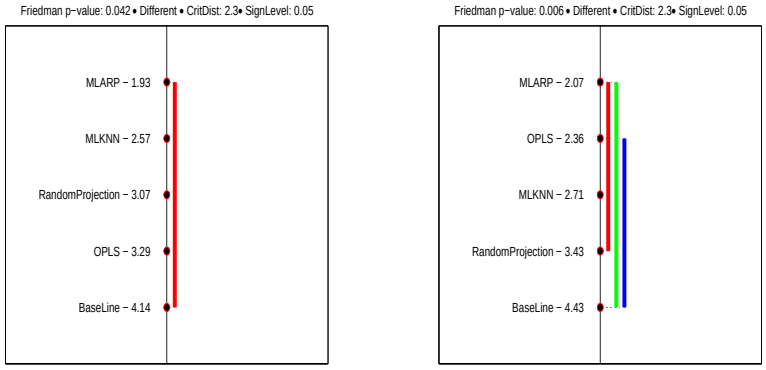
Test 4

In this test, we cancel MDDM and PCA, we recover OPLS. So we add the mediamill database to the second test.



(a) *RMSE* (b) *Hamming loss*

Figure 6.7: Nemenyi test for $\alpha = 0.01$



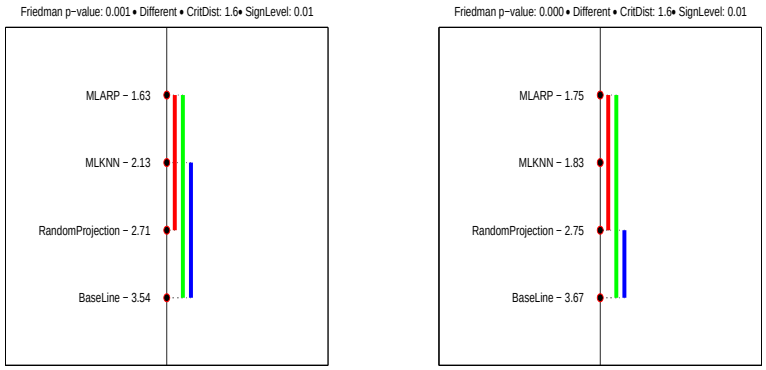
(a) *RMSE* (b) *Hamming loss*

Figure 6.8: Nemenyi test for $\alpha = 0.05$

From figures 6.7 & 6.8, the null hypothesis H_0 is rejected. The ML-ARP is statistically the best algorithm.

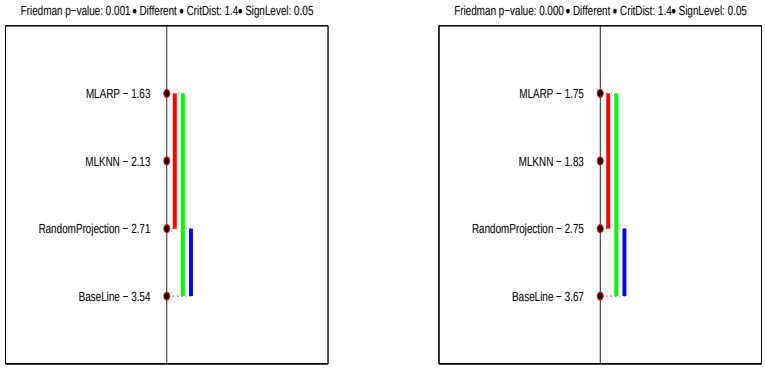
Test 5

In this test, we cancel all the algorithms based on resolving eigenvalues problem. So, we add the bookmarks and the reuters databases to the test.



(a) *RMSE* (b) *Hamming loss*

Figure 6.9: Nemenyi test for $\alpha = 0.01$



(a) *RMSE* (b) *Hamming loss*

Figure 6.10: Nemenyi test for $\alpha = 0.05$

Based on figures 6.9 & 6.10, the null hypothesis H_0 is rejected. By this way, the baseline is considered as very bad classifier. We notice that ML-ARP comes first one again with an equivalence with ML-kNN.

6.2.3 Statistical tests summary

			BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA	$\sim \setminus \neq$
Test1	$\alpha = 0.01$	RMSE	8.00	2.63	3.88	**2.00**	4.88	4.13	5.63	4.88	$\sim$
		HL	7.50	3.38	4.50	**2.25**	4.38	3.25	5.25	5.5	$\sim$
	$\alpha = 0.05$	RMSE	8.00	2.63	3.88	**2.00**	4.88	4.13	5.63	4.88	$\neq$
		HL	7.50	3.38	4.50	**2.25**	4.38	3.25	5.25	5.50	$\sim$
Test2	$\alpha = 0.01$	RMSE	5.50	3.00	3.83	**2.75**	4.33	3.92	4.67	X	$\sim$
		HL	5.83	3.42	4.17	**2.83**	3.92	3.33	4.50	X	$\sim$
	$\alpha = 0.05$	RMSE	5.50	3.00	3.83	**2.75**	4.33	3.92	4.67	X	$\sim$
		HL	5.83	3.42	4.17	**2.83**	3.92	3.33	4.50	X	$\sim$
Test3	$\alpha = 0.01$	RMSE	5.06	3.06	3.78	**2.50**	3.17	3.44	X	X	$\sim$
		HL	5.33	2.83	4.33	**2.72**	2.89	2.89	X	X	$\neq$
	$\alpha = 0.05$	RMSE	5.06	3.06	3.78	**2.50**	3.17	3.44	X	X	$\neq$
		HL	5.33	2.83	4.33	**2.72**	2.89	2.89	X	X	$\neq$
Test4	$\alpha = 0.01$	RMSE	4.14	2.57	3.07	**1.93**	X	X	3.29	X	$\sim$
		HL	4.43	2.71	3.43	**2.07**	X	X	2.36	X	$\neq$
	$\alpha = 0.05$	RMSE	4.14	2.57	3.07	**1.93**	X	X	3.29	X	$\neq$
		HL	4.43	2.71	3.43	**2.07**	X	X	2.36	X	$\neq$
Test5	$\alpha = 0.01$	RMSE	3.54	2.13	2.71	**1.63**	X	X	X	X	$\neq$
		HL	3.67	1.83	2.75	**1.75**	X	X	X	X	$\neq$
	$\alpha = 0.05$	RMSE	3.54	2.13	2.71	**1.63**	X	X	X	X	$\neq$
		HL	3.67	1.83	2.75	**1.75**	X	X	X	X	$\neq$

$\sim$: H_0 is accepted
$\neq$: H_0 is rejected

In all cases, ML-ARP comes first whether the null-hypothesis H_0 is rejected or not.
So, statistically, we conclude that the ML-ARP combined with the ML-kNN algorithm is the best multi-label classifier.

Table 6.14: Mean ranks obtained by launching five Friedman statistical tests

6.3 ML-ARP Fast Version

The ML-ARP algorithm is mainly based on iteratively optimizing a multi-label criterion measure which can typically be the Hamming Loss or the RMSE. In each iteration a ML-kNN process is learned and tested on the projected version of the training set. This process can take long time. In order to accelerate the optimization process, we propose to replace the ML-kNN process by a simple kNN classifier. This new version of the ML-ARP is called Fast ML-ARP.
Unlike the ML-kNN which consists in two steps: a training and a testing one, kNN classifier is performed by only following a testing phase.

Remark
We only replace ML-kNN with kNN in the "projection learning phase". After the projection is learnt, it is still combined to ML-kNN.

kNN Classifier

The idea around the kNN classifier is as simple as possible.
Let x be an unseen instance taken from $\mathcal{X}$.
In order to predict its associated label vector $Y \in \mathcal{Y}$, we seek the neighborhood $\mathfrak{N}(x)$ of the unseen instance x based on the training data set $\mathcal{L}$. Then, a label y_l is considered as a proper one to x if the majority of its neighbors is associated with this label. Finally, the probability of this event $f(x, y_l)$ is computed by averaging the number of neighbors which are associated with the label y_l across the neighborhood size k.
Formally, we have:

$$\begin{cases} f(x,\cdot) = \left[f(x,y_1),...,f(x,y_{d_y})\right] \in [0\ 1]^{d_y} \\ f(x,y_l) = \frac{\sum_{(x^+,Y^+)\in\mathfrak{N}(x)} \lfloor y_l \in Y^+ \rfloor}{k} \quad \forall l \in \{1,...,d_y\} \end{cases} \tag{6.11}$$

then:

$$\begin{cases} Y = \left[y_1,...,y_{d_y}\right] \in \mathcal{Y} \\ y_l = \lfloor f(x,y_l) \geq \frac{1}{2} \rfloor \quad \forall l \in \{1,...,d_y\} \end{cases} \tag{6.12}$$

Chapter 7

Conclusions and Future Works

7.1 Discussions

Based on the statistical tests (See 6.2), we deduce the following facts:

- *Case of low and medium dimensions:*
  This case coincides with the four first statistical tests (see 6.2.2, 6.2.2, 6.2.2, 6.2.2). The ML-ARP performances are absolutely competitive with the remaining state of art dimensionality reduction methods. The majority of the statistical tests (for the two values of significance levels $\alpha = 0.01$ *or* 0.05) show that we cannot statistically differentiate between different algorithms with regards to their performances. We should notice that the ML-ARP algorithm always comes first in term of mean rank. This tends to highlight the robustness of the ML-ARP approach.

- *Case of high dimensions:*
  When we are dealing with high dimensions data (see test 6.2.2), all the algorithms based on resolving eigenvalues problem (MDDM, CCA, OPLS, PCA) came to nothing because of the high complexity of the analysis ($\mathcal{O}(n^3)$).
  We should notice that ML-ARP performances for high dimensions data are similar to those obtained by the ML-kNN algorithm without dimension reduction preprocessing whereas the dimensions have been greatly reduced (from 47229 to 4000 features with the prior that a Tf-IDF preprocessing have been performed on the reuters dataset). The statistical test results show that there is no significant differences between the two methods. On the other hand, the dimensionality reduction learned by the ML-ARP allows a significanty faster nearest neighbors identification (the accelaration rate is of the order of the dimensionality reduction). Finally, one may notice that ML-ARP performs better than a simple random projection. Thus, the adaptative projection tends to greatly extract the emboddied pattern linking the input space $\mathcal{X}$ to the output one $\mathcal{Y}$. Therefore, we show that the ML-ARP algorithm is an operational and very competitive multi-label dimensionality reduction method.

7.2 Conclusions

This work proposes a novel multi-label dimensionality reduction approach. The ML-ARP algorithm uses random projections according to a novel paradigm. In our framework, random projections coupled with the most famous multi-label classifier (ML-kNN), are iteratively adapted by optimizing one of the ML-kNN performances criterion. Thus, we obtain a multi-label supervised dimensionality reduction based on the ML-kNN classifier.
An expertimental framework has been done to highlight the validity of our proposal: The ML-ARP performs as well as the remaining state of the art multi-label dimensionality reduction approaches (MDDM, OPLS, CCA, PCA), but in addition it allows us to deal with high dimensional data. Moreover, additionnal experiments show that ML-ARP is able to greatly reduce the input dimensions without harming the ML-kNN predictive performances. These results are not due to a simple random projection, but they are due to the iterative optimization process. Therefore, ML-ARP is a promising novel multi-label dimensionality reduction approach.

7.3 Future works

Some directions of ongoing research can be sketched as follow:

- The optimization problem resolution can be accelerated by improving the RVNS heuristic, or seeking another optimization heuristic like [Kirkpatrick et al., 1983, Das and Chakrabarti, 2005, Oblow, 2001, Rudlof and Köppen, 1997, Hansen and Mladenović, 2001].
- In our optimization framework, we use a sparse speed matrix to build the neighborhood of the current solution, we can use another way to build this neighborhood.
- The neighborhood identification task (according to the instances) can be fastly performed by inserting clustering approaches, like Voronoi cells [Ghosh et al., 1997] and product quantization [Jegou et al., 2011].
- The reducing dimension number r which coincides with the projection matrix rank, is currently given. So, to improve performances, we propose to perform a smarter choice of r by using a cross-validation principle.
- The neighborhood size k is currently given. So, for a better learned model, we propose to well choose the number of neighbors used as second input of the ML-kNN algorithm by applying a cross-validation way.

Chapter 8

Acknowledgments

This work done with *Orange Labs* was a gorgeous chance for my learning and my professional improvement. Therefore, I am very happy to be providing with an opportunity to be a part of it. I am also grateful for having the chance to meet so many wonderful professionals who led me though this internship period.

First of all, I would like to express my hearthfelt gratitude to my internship advisor Frank MEYER for his moral and intellectual support, his listening and his continuous encouragement. I will never forget his joy expressed for my acceptance to pursue a PhD position in his team☺.
I would also acknowledge Wissam SIBLINI, my second internship advisor for his overflowing motivation and his critical sense. This work would not be as easy to do without his cooperation and his advices.

Last but not the least, my sincere thanks also go to Fabrice CLEROT, the "Profiling and datamining" (PROF) team manager, for his immense knowledge, his availability and his kindness. It was a great honnor to be part of your team.

I perceive as this internship as a big milestone opportunity in my career improvement. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their betterment, in order to reach desired research career objectives. Hope to continue contribution with all of you in the future.

Bibliography

[Abdi and Williams, 2010] Abdi, H. and Williams, L. J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459.

[Ajmera et al., 2013] Ajmera, J., Ahn, H.-i., Nagarajan, M., Verma, A., Contractor, D., Dill, S., and Denesuk, M. (2013). A crm system for social media: challenges and experiences. In *Proceedings of the 22nd international conference on World Wide Web*, pages 49–58. ACM.

[Al-Talib and Hassan,] Al-Talib, G. A. and Hassan, H. S. A study on analysis of sms classification using tf-idf weighting.

[Bi and Kwok, 2013] Bi, W. and Kwok, J. T.-Y. (2013). Efficient multi-label classification with many labels. In *ICML (3)*, pages 405–413.

[Bingham and Mannila, 2001] Bingham, E. and Mannila, H. (2001). Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250. ACM.

[Bishop, 2006] Bishop, C. M. (2006). Pattern recognition. *Machine Learning*.

[Boutell et al., 2004] Boutell, M. R., Luo, J., Shen, X., and Brown, C. M. (2004). Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771.

[Chen et al., 2009] Chen, G., Zhang, J., Wang, F., Zhang, C., and Gao, Y. (2009). Efficient multi-label classification with hypergraph regularization. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1658–1665. IEEE.

[Crammer et al., 2007] Crammer, K., Dredze, M., Ganchev, K., Talukdar, P. P., and Carroll, S. (2007). Automatic code assignment to medical text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, pages 129–136. Association for Computational Linguistics.

[Das and Chakrabarti, 2005] Das, A. and Chakrabarti, B. K. (2005). *Quantum annealing and related optimization methods*, volume 679. Springer Science & Business Media.

[Dasgupta and Gupta, 2003] Dasgupta, S. and Gupta, A. (2003). An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65.

[Demšar, 2006] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30.

[Diplaris et al., 2005] Diplaris, S., Tsoumakas, G., Mitkas, P. A., and Vlahavas, I. (2005). Protein classification with multiple algorithms. In *Panhellenic Conference on Informatics*, pages 448–456. Springer.

[Elisseeff and Weston, 2001] Elisseeff, A. and Weston, J. (2001). A kernel method for multi-labelled classification. In *Advances in neural information processing systems*, pages 681–687.

[Ghamrawi and McCallum, 2005] Ghamrawi, N. and McCallum, A. (2005). Collective multi-label classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 195–200. ACM.

[Ghosh et al., 1997] Ghosh, S., Nowak, Z., and Lee, K. (1997). Quantitative characterization and modeling of composite microstructures by voronoi cells. *Acta Materialia*, 45(6):2215–2234.

[Godbole and Sarawagi, 2004] Godbole, S. and Sarawagi, S. (2004). Discriminative methods for multi-labeled classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 22–30. Springer.

[Gretton et al., 2007] Gretton, A., Fukumizu, K., Teo, C. H., Song, L., Schölkopf, B., and Smola, A. J. (2007). A kernel statistical test of independence. In *Advances in neural information processing systems*, pages 585–592.

[Guo and Schuurmans, 2012] Guo, Y. and Schuurmans, D. (2012). Semi-supervised multi-label classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 355–370. Springer.

[Hansen and Mladenović, 2001] Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3):449–467.

[Hotelling, 1936] Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

[Iman and Davenport, 1980] Iman, R. L. and Davenport, J. M. (1980). Approximations of the critical region of the fbietkan statistic. *Communications in Statistics-Theory and Methods*, 9(6):571–595.

[Jegou et al., 2011] Jegou, H., Douze, M., and Schmid, C. (2011). Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128.

[Ji and Ye, 2009] Ji, S. and Ye, J. (2009). Linear dimensionality reduction for multi-label classification. In *IJCAI*, volume 9, pages 1077–1082. Citeseer.

[Joseph et al., 2013] Joseph, E., Galeano, P., and Lillo, R. E. (2013). The mahalanobis distance for functional data with applications to classification. *arXiv preprint arXiv:1304.4786*.

[Katakis et al., 2008] Katakis, I., Tsoumakas, G., and Vlahavas, I. (2008). Multilabel text classification for automated tag suggestion. *ECML PKDD discovery challenge*, 75.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., et al. (1983). Optimization by simmulated annealing. *science*, 220(4598):671–680.

[Klimt and Yang, 2004] Klimt, B. and Yang, Y. (2004). The enron corpus: A new dataset for email classification research. In *European Conference on Machine Learning*, pages 217–226. Springer.

[Levatić et al., 2015] Levatić, J., Kocev, D., and Džeroski, S. (2015). The importance of the label hierarchy in hierarchical multi-label classification. *Journal of Intelligent Information Systems*, 45(2):247–271.

[Lewis et al., 2004] Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

[Li et al., 2013] Li, P., Li, H., and Wu, M. (2013). Multi-label ensemble based on variable pairwise constraint projection. *Information Sciences*, 222:269–281.

[Liu, 2012] Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

[Luaces et al., 2012] Luaces, O., Díez, J., Barranquero, J., del Coz, J. J., and Bahamonde, A. (2012). Binary relevance efficacy for multilabel classification. *Progress in Artificial Intelligence*, 1(4):303–313.

[Maitra and Yan, 2008] Maitra, S. and Yan, J. (2008). Principle component analysis and partial least squares: Two dimension reduction techniques for regression. *Applying Multivariate Statistical Models*, 79.

[Oblow, 2001] Oblow, E. (2001). Spt: a stochastic tunneling algorithm for global optimization. *Journal of Global Optimization*, 20(2):191–208.

[Plackett, 1983] Plackett, R. L. (1983). Karl pearson and the chi-squared test. *International Statistical Review/Revue Internationale de Statistique*, pages 59–72.

[Ponce et al., 2006] Ponce, J., Berg, T. L., Everingham, M., Forsyth, D. A., Hebert, M., Lazebnik, S., Marszalek, M., Schmid, C., Russell, B. C., Torralba, A., et al. (2006). Dataset issues in object recognition. In *Toward category-level object recognition*, pages 29–48. Springer.

[Ramirez et al., 2000] Ramirez, D. E. et al. (2000). The generalized f distribution. *Journal of Statistical Software*, 5(1):1–14.

[Ran and Oh, 2015] Ran, R. and Oh, H. (2015). Adaptive sparse random projections for wireless sensor networks with energy harvesting constraints. *EURASIP Journal on Wireless Communications and Networking*, 2015(1):1.

[Read et al., 2009] Read, J., Pfahringer, B., and Holmes, G. (2009). Generating synthetic multi-label data streams. In *ECML/PKKD 2009 Workshop on Learning from Multi-label Data (MLD'09)*, pages 69–84.

[Rosipal and Krämer, 2006] Rosipal, R. and Krämer, N. (2006). Overview and recent advances in partial least squares. In *Subspace, latent structure and feature selection*, pages 34–51. Springer.

[Rudlof and Köppen, 1997] Rudlof, S. and Köppen, M. (1997). Stochastic hill climbing with learning by vectors of normal distributions.

[Sheskin, 2003] Sheskin, D. J. (2003). *Handbook of parametric and nonparametric statistical procedures*. crc Press.

[Snoek et al., 2006] Snoek, C. G., Worring, M., Van Gemert, J. C., Geusebroek, J.-M., and Smeulders, A. W. (2006). The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proceedings of the 14th ACM international conference on Multimedia*, pages 421–430. ACM.

[Sorower, 2010] Sorower, M. S. (2010). A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis*.

[Šubelj et al., 2015] Šubelj, L., Fiala, D., and Bajec, M. (2015). Network-based statistical comparison of citation topology of bibliographic databases. *arXiv preprint arXiv:1502.05061*.

[Sun et al., 2011] Sun, L., Ji, S., and Ye, J. (2011). Canonical correlation analysis for multilabel classification: A least-squares formulation, extensions, and analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(1):194–200.

[Sun et al., 2009] Sun, L., Ji, S., Yu, S., and Ye, J. (2009). On the equivalence between canonical correlation analysis and orthonormalized partial least squares. In *IJCAI*, volume 9, pages 1230–1235.

[Takacs et al., 2007] Takacs, G., Pilaszy, I., Nemeth, B., and Tikk, D. (2007). On the gravity recommendation system. In *Proceedings of KDD cup and workshop*, volume 2007.

[Trohidis et al., 2008] Trohidis, K., Tsoumakas, G., Kalliris, G., and Vlahavas, I. P. (2008). Multi-label classification of music into emotions. In *ISMIR*, volume 8, pages 325–330.

[Wan et al., 2016] Wan, S., Mak, M.-W., and Kung, S.-Y. (2016). Sparse regressions for predicting and interpreting subcellular localization of multi-label proteins. *BMC bioinformatics*, 17(1):1.

[Xiao et al., 2011] Xiao, Y., Kaku, I., Zhao, Q., and Zhang, R. (2011). A reduced variable neighborhood search algorithm for uncapacitated multilevel lot-sizing problems. *European Journal of Operational Research*, 214(2):223–231.

[Yu et al., 2005] Yu, K., Yu, S., and Tresp, V. (2005). Multi-label informed latent semantic indexing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 258–265. ACM.

[Zhang and Zhou, 2007] Zhang, M.-L. and Zhou, Z.-H. (2007). Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048.

[Zhang and Zhou, 2010] Zhang, Y. and Zhou, Z.-H. (2010). Multilabel dimensionality reduction via dependence maximization. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(3):14.

Appendix A

State-of-the-art Multi-Label Dimensionality Reduction

This appendix is specially devoted to briefly describe the four multi-label dimensionality reduction methods: MDDM, PCA, OPLS, CCA.
First of all, we introduce the notations used in this apprendix.
Based on the training set $\mathcal{L} = \{(x_i^{\mathcal{L}}, y_i^{\mathcal{L}}) \in \mathcal{X} \times \mathcal{Y} \mid i \in \{1, ..., N_{\mathcal{L}}\}\}$, one can form the two following matrices:

- Instance matrix $X \in \mathbb{R}^{N_{\mathcal{L}} \times d_x}$, where the i^{th} row of X represents the i^{th} instance $x_i^{\mathcal{L}}$ of $\mathcal{L}$.
- Label matrix $Y \in [0\ 1]^{N_{\mathcal{L}} \times d_y}$, where the i^{th} row of Y represents the i^{th} labels vector $y_i^{\mathcal{L}}$ of $\mathcal{L}$.

A.1 MDDM: Multi-label Dimensionality reduction via Dependence Maximization

MDDM attempts to find a lower-dimensional feature space in which features and labels are strongly dependent. Thus, we seek the r-rank projection matrix $P \in \mathcal{P}$ which maximizes the dependence between the projection version $\langle x \rangle_P$ of the instance x and its labels vector y. Based on the Hilbert-Schmidt Independance Criterion (HSIC) [Gretton et al., 2007], we seek to resolve the following problem:

$$P^* = argmax_P\ tr(HXP^\tau PX^\tau HL) \tag{A.1}$$

where:

- $L = YY^\tau$: the matrix of inner product of instances in $\mathcal{Y}$
- $H = (H_{i,j})_{N_{\mathcal{L}} \times N_{\mathcal{L}}}$ such that: $H_{i,j} = \delta_{i,j} - \frac{1}{N_{\mathcal{L}}}$

If we denote P_i the i^{th} row of P, one can easily deduce that:

$$\max_{P} \; tr(HXP^{\tau}PX^{\tau}HL) = \max_{P} \; \sum_{i=1}^{r} P_i \left(X^{\tau}HLHX\right) P_i^{\tau} \tag{A.2}$$

Traditionally A.2 is resolved via an eigenvalue problem. The optimal P^* is thus obtained by seeking the eigenvectors associated to the r largest eigenvalues of $X^{\tau}HLHX$. For further details, see [Zhang and Zhou, 2010].

A.2 PCA: Principal Component Analysis

PCA attemps to find a lower-dimensional feature space in which almost all the energy of the input data is conserved. This energy is traduced by the trace of the covariance matrix $\Xi = X^{\tau}X$. Thus, we seek the r-rank projection matrix $P \in \mathcal{P}$ which conserves almost all the energy of Ξ. In other words, we aim at resolving the following problem:

$$P^* = argmax_P \; tr(PX^{\tau}XP^{\tau}) \tag{A.3}$$

By denoting P_i the i^{th} row of P, one can easily deduce that:

$$max_P \; tr(PX^{\tau}XP^{\tau}) = \max_{P} \; \sum_{i=1}^{r} P_i \left(X^{\tau}X\right) P_i^{\tau} = \max_{P} \; \sum_{i=1}^{r} P_i \Xi P_i^{\tau} \tag{A.4}$$

Traditionally A.4 is resolved via an eigenvalue problem. The optimal P^* is then obtained by seeking the eigenvectors associated to the r largest eigenvalues of Ξ. For further details, see [Maitra and Yan, 2008].

A.3 CCA: Canonical Correlation Analysis

The objective of CCA is to compute two projection matrices $P_x \in \mathcal{P}$ and $P_y \in \mathbb{R}^{r \times d_y}$ maximizing the following amount:

$$\max_{P_x, P_y} \; tr(P_x X^{\tau} Y P_y)$$

under the following conditions:

- $P_x X^{\tau} X P_x = I$
- $P_y Y^{\tau} Y P_y = I$

Assuming that YY^{τ} is nonsingular, P_x is given by seeking the r principal eigenvectors of the following generalized eigenvalue problem:

$$X^{\tau}Y \left(Y^{\tau}Y\right)^{-1} Y^{\tau}X p_x = \lambda X^{\tau}X p_x \tag{A.5}$$

where: λ represents the eigenvalue and p_x its associated eigenvector.
For further details, see [Sun et al., 2011].

A.4 OPLS: Orthogonal Partial Least Square

OPLS attempts to find a lower-dimensional feature space $P \in \mathcal{P}$ in which the covariance of data is maximized. Thus, OPLS seeks to resolve the following problem:

$$\max_{P} \; tr\left(PX^{\tau}YY^{\tau}XP^{\tau}\right) \; subject\ to\ PX^{\tau}XP^{\tau} = I \tag{A.6}$$

It can be shown that the rows of the optimal P are given by the r principal eigenvectors of the following generalized eigenvalue problem:

$$X^{\tau}YY^{\tau}Xp_x = \lambda X^{\tau}Xp_x \tag{A.7}$$

where: λ represents the eigenvalue and p_x its associated eigenvector.
For further details, see [Maitra and Yan, 2008].