UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE NANTES

THÈSE DE DOCTORAT DE

L'UNIVERSITE DE NANTES
COMUE **U**NIVERSITE **B**RETAGNE **L**OIRE

Ecole Doctorale N°601
Mathèmatique et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

« Jiajun PAN »

« Metric learning for structured data »

««»

Thèse présentée et soutenue à NANTES , le 20 décembre
Unité de recherche : LS2N, UMR CNRS 6004
Thèse N° :

Rapporteurs avant soutenance :

LESOT Marie-Jeanne,Maitre de Conférences, Univ. Paris 6, Paris
HABRARD Amaury,Professeur des universités, Université de Saint-Etienne, Saint Etienne

Composition du jury :

Président : DE LA HIGUERA Colin,Professeur des universités,Université de Nantes, Nantes
Examinateurs : CAPPONI Cécile,Maitre de Conférences, Université d'Aix-Marseille, Marseille

Dir. de thèse : Philippe Leray,Professeur des universités, Université de Nantes, Nantes
Co-dir. de thèse : Hoel Le Capitaine, Maitre de Conférences, Université de Nantes, Nantes

Invité(s)

ACKNOWLEDGEMENT

First of all, I want to thank my supervisor Hoel LA CAPITAINE and Philippe LERAY. Thank them for contributing to this work and the cultivation of me over the years. And I also want to thank Université de Nantes and MathSTIC. Thanks for the courses and help provided by the school.

Secondly, I want to thank my colleagues, friends and family. Thanks to Vincent RAVENEAU and other colleagues for helping me with problems from work to life. Thanks to Yao MA and my friends in Nantes or Paris for spending the days with me in France. Thanks to my mother and my family in my hometown for supporting and caring.

Finally, I would also like to thank those who have contributed to the establishment of friendship between China and France. They gave me the opportunity to come to France to study. Today, on China's 70th birthday, I sincerely hope that the motherland will flourish and develop harmoniously with France and will vow to contribute to China's construction and alternative friendship in the future.

TABLE OF CONTENTS

TABLE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

In contemporary life, machine learning, as part of the field of artificial intelligence, has been integrated into all aspects of our lives. When you are a Chinese and travelling in a foreign country, picking up your smart-phone and asking for voice assistants, where is the nearest delicious restaurant, to meet your needs and solve your problems, much artificial intelligence and machine learning methods are applied. This process includes dialogue recognition, natural language processing, machine translation, data mining, pattern recognition, recommendation systems, and other related algorithms. Besides, machine learning is difficult to enumerate complete theories and algorithms, but all related machine learning algorithms have the same core, that is, machine learning learn new knowledge from the data. This new knowledge arises from relevant databases, past related tasks and selected learning models, and can meet future expectations, automatically resolve target tasks, and in most cases improve itself from feedback from you and other users [M$^+$97].

In most cases, machine learning [M$^+$97] can be classified into three standard machine learning task classifications, regressions, clustering according to the tasks solved; or classified into supervised learning, unsupervised learning, and semi-supervised learning according to the presence or absence of supervised information. Supervise learning. In supervised learning, the data analyzed is accompanied by label information. The label information comes from data collection, expert experience or feedback from the last study. The label information is generally a category (such as a nearby restaurant is a good, medium or lousy rating) or a value (such as the average price of a nearby restaurant dish), depending on that the problem from related sample database needs to solve. The goal of supervised learning is generally to predict the label information for future new samples. In general, classification tasks correspond to supervised learning of discrete categories, and regression tasks correspond to supervised learning of continuous values. In unsupervised learning, the analyzed data has no label information, so it is difficult to evaluate and present the results of learning as supervised learning, but still learn meaningful patterns from the data. For example, the last classic machine learning task clustering corresponds to unsupervised learning. The purpose of clustering is to divide the data into several relatively similar internally clusters, and these clusters can be treated as new categories to give labels. Finally, in semi-supervised

learning, some data have label information while other data has no label information. In this case, it is common to use data with label information to help complete an unsupervised learning task, or to perform clustering and other processing on unlabeled information before performing supervised learning algorithms.

Whatever machine learning method they use, what they have in common is to compare and evaluate sample entities. When learning a model, the construction of the model depends on the correlation between known samples, and the performance of the algorithm will be reflected in the difference between prediction and reality when making predictions. Therefore, we can easily see that the quality of machine learning algorithms strongly depends on the quality of the metric distance, that is, the concept of describing the difference of entities. The usual physical distance is one of the metrics. How do we know which restaurant is the nearest one ? You can find the answer by calculating the physical distance. However, when applied to other machine learning samples, for example, how do we know that your pronounce "can tin" are talking about "can ting" (means restaurant) rather than "chang ding" (means spike) ? How do we know which restaurant is more delicious than other restaurants ? When dealing with these situations, metric distance is a broader concept, which is a measure of the difference in the feature space of these entities, such as Euclidean distance. When solving string recognition, for example, we calculate the metric distance between "can ting" and "chang ding" according to the "can tin" identified by your pronunciation. The metric distance between "can tin" and "can ting" in the language database is the smallest, to determine what you are talking about is "restaurant". This process is the use of metric distances in machine learning.

However, for different machine learning methods, no metric fits perfectly in all situations, just as we are unable to apply all the speech recognition tasks by calculating the distance of the character storage location in the speech database. First, there are now a lot of different metric distances. For example, to distinguish between "Can Ting" (Restaurant) and "Chang Tu" (Far away). The Hamming distance [Ham50] only cares whether the characters in the corresponding position are the same, there is only the initial letter "C" is the same at exact position, so the distance could be counted as "01111111" ; The Levenshtein distance [Lev66] is concerned the least operational change of characters, there are because the insertion of "h" and "g", the deletion of "ng", and the replacement of "i" to "u", the distance is 5 operations. Choosing Hamming distance or Levenshtein distance is different for dealing with character databases. Secondly, even if you choose the same metric distance, in the face of different databases, different results occur due to the relationship bet-

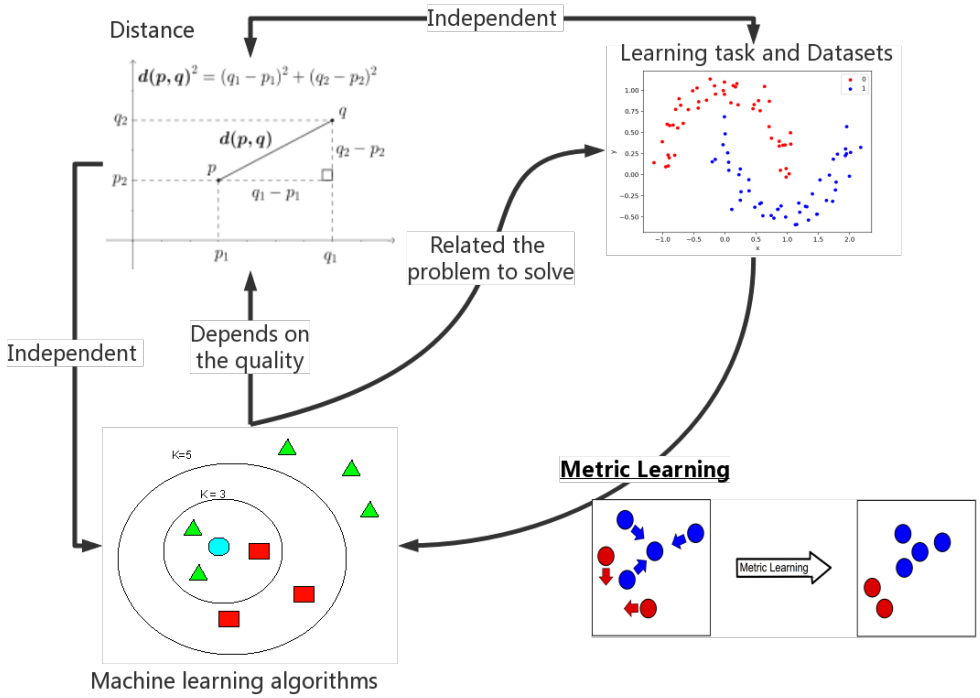


FIGURE 1 – Metric learning create the adaptive metric for dataset and target task.

ween data structure and features. For example, when processing a database of commodity classification information, each of the data is composed of independent indicators of the commodity, and the selection of the standard metric Euclidean distance can achieve better results. However, if the processed database is video evaluation information, and each piece of information is compressed information of each frame picture of the sample video, then the standard metric Euclidean distance is contrary to process the integrated information of the picture or the relevant information of each dimension in the feature space. Finally, in the face of different learning tasks, the use of adaptive metric distances can better face different target outcomes. For example, for the same user information database, when the target task is to classify the user's social credit, it was evident that the user's age, work, credit history should be paid more attention to and give higher weight to these feature dimensions ; while if the target task is friend recommendations for users, they should treat the feature dimensions such as geographic location and hobbies beyond the common. Therefore, when solving machine learning tasks, if we can find or construct a better metric, which can solve this task faster or more accurately, we can improve the effect of improving machine learning. As the Figure 1 shows, the predefined metric distances are independent of the database and learning tasks, while the quality of machine learning algorithms depends on the choice of metric distances. So an algorithm that automatically learns a better metric distance from the known database or experience is necessary, and it is called metric distance learning [Kul12, BHS13].

Metric distance learning is a branch of machine learning—representation learning [Kul12]. Feature learning or representation learning [BCV13] is a collection of techniques for learning to convert raw data into features that can be used by machine learning algorithms or that convert data into more efficiently learned features. It avoids the hassle of manually extracting features, allowing the computer to learn to use features while also learning how to extract features : learning how to learn. The output of metric distance learning is not a model that is directly used for prediction but a new metric that adapts to the task. This metric can be seen as a representation of the data in a new feature space or as a self-adaptive feature space that is adaptively extracted. This new metric is then applied to other machine learning algorithms, and improvements relative to the old metrics evaluate the performance of the new metric.

0.1 Problems and Propositions

In the past few years, there have been many branches and developments in metric distance learning, and there are various interactions and combinations with transfer learning, deep learning, and so on [Kul12, BHS13]. In our work, we mainly focus on supervised metric learning applied to classification and study the content of two aspects of the flat database and non-flat database. We proposed new methods on two questions : using high-latitude interactive information in the flat database; metric particularly learning for the relational non-flat database.

For flat databases, most of the metric distance learning base on a model of Mahalanobis distance. The Mahalanobis distance d_M can also be called the covariance distance because its principle is to multiply the Euclidean distance with the covariance matrix M_c of the samples, which is denoted as $d_M(\mathbf{x}_i, \mathbf{x}_j)) = \left((\mathbf{x}_i - \mathbf{x}_j)^T M_c^{-1}(\mathbf{x}_i - \mathbf{x}_j)\right)^{1/2}$. The Mahalanobis distance is equal to the linearity of each dimension of the sample to reflect the association between each pair of dimensions, and the scale-independent (independent of the unit of measurement scale) is achieved by regularization of the covariance matrix. For example, for user health data, height is related to weight and the unit is different, and the Markov distance can use for better classification. For the metric distance learning based on Mahalanobis distance, the weight matrix M is used instead of the covariance matrix M_c, and the samples are divided into similar sets and dissimilar sets by supervised information or unsupervised information, and the loss function L is used to learn. The weight matrix M is such that the loss function is small enough for the new metric, that is, for the samples in the similar set, the distance between the two is smaller, and for the samples in the dissimilar set, the distance between the two is larger. However, precisely because these algorithms base on Mahalanobis distance, they only consider each dimension or the association within each pair of dimensions, while ignoring the information contained in the intersection of three or more dimensions.

To address this limitation, we propose a metric learning algorithm based on the submodular function. Firstly, we propose and prove the metric of the Lovasz extension definition based on the submodular function, and try to prove the metric of the multi-linear extension definition based on the submodular function (although it fails, but given the difference, it is given as a difference) The probability of the degree is defined). Secondly, we construct a loss function learning algorithm for the modulo correlation metrics. Finally, we designed and experimented with real-world datasets and compared them with other metric learning

algorithms.

For non-flat databases, the current metric learning algorithms have evolved algorithms for string databases, time-series databases, tree and graph-like databases. Most of the models are based on the corresponding non-structural data type of metric distance (such as the Hamming distance [Ham50] for the string, the dynamic time warping distance [Kru83] for the time series, the edit tree distance [BHS06] for the tree), and then apply conventional distance metric learning algorithms. The model of the algorithm selects similar sets and dissimilar sets and constructs a loss function for learning. At present, there are not many metric learning algorithms for the relational database, which is a particular non-flat database.

For this blank, we propose a relational metric distance learning algorithm that extends graph metric learning algorithms. We propose three algorithms, which are related to the unsupervised information based on the relationship. The first algorithm is an algorithm that selects the similarity set and the dissimilar set with the association strength. The second algorithm combines with the tensor decomposition RESCAL [NTK11]. Moreover, the third algorithm pays attention to both the supervision information and non-supervised information. Then a multi-relational metric learning algorithm that aggregates various relationships is proposed.

0.2 Outline

In the first chapter, some preliminary concepts and information are introduced. We briefly introduce the related concepts of machine learning and summarize the classic classification algorithm KNN, regression algorithm SVM, clustering algorithm K-means, and these algorithms are commonly used to evaluate the effect of the distance learning algorithm. We introduced the context of relational learning and the algorithms of regular relational learning. Finally, we give the accurate definition and significance of the metrics, and some concepts and symbols related to metric learning commonly used in our work. The application and significance of the metric distance learning algorithm are introduced, and the reason why we study the metric distance algorithm is expressed.

In the second chapter, the development and current status of the metric distance learning algorithm for the flat database are summarized. We follow the classic flat database to measure the model structure of the distance learning algorithm to segment the entire chapter. Firstly, different metrics and their related metric learning algorithms are introdu-

ced. Secondly, we survey different settings of the selection and application of similar and dissimilar sets, the selection of regular terms for loss function and the selection of optimization algorithms. Then, we introduced some metric distance learning algorithms based on particular learning tasks in addition to the classic metric learning tasks. Finally, the recent popular deep neural network algorithms are highlighted with their associated depth metric learning algorithms. At the end of this chapter, we point out the limitations of a series of algorithms based on Mahalanobis distance for flat databases and express our motivation for studying the submodular metric for high dimension intersection.

In the third chapter, we present our first part of contribution : propose the submodular metric distance algorithm and design experiment for it. We first introduce set-function. Then we introduce the submodular function, which is a particular case of set-function commonly used in many fields, with the property and extension of the submodular function. Secondly, we have defined and proved the metric according to the Lovasz extension. We also try it on the multi-linear extension of the submodular function. The former has proved that the latter proves that it is not a metric without additional conditions, and can be considered as a similarity under the finite conditions we propose. Finally, we design a learning algorithm for the submodular function, and design and compare the proposed algorithm, the standard metric distance learning algorithm and the recent nonlinear metric learning algorithm on the real data set, and compare the data in multiple data. The set shows good results.

In the fourth chapter, the development and current status of metric distance learning algorithms for non-flat databases are summarized. We introduce the metric distance learning algorithm based on different non-flat databases. The first is the metric distance learning of string data. It mainly introduces string editing distance and related algorithms. Secondly, it is the metric distance learning of time series data. It mainly introduces the DTW(Dynamic Time Warping) [Kru83] and the metrics generated by DTW. There are metric learning algorithms for improving DTW and metric learning algorithms based on the metrics generated by DTW. Finally, the metric distance learning algorithm for tree or graph data is introduced. The tree editing distance and graph editing distance are mainly introduced. In the end, the current GCN(graph convolution neural network [KW16]) metric learning algorithm combined with deep learning GCN neural network is introduced. At the end of this chapter, we point out the lack of metric distance learning algorithms particularly for the relational databases and express our motivation to study metric distance learning.

In the fifth chapter, we present our second part of contribution : propose the relational

metric learning algorithms and design experiment on them. We divided this chapter into two parts. In the first part, we concern with the case of multiple relational tables between multiple entity tables. In response to this situation, we consider starting with the selection of similar, dissimilar sets. The relationship side information is used to construct the correlation strength function to evaluate the degree of similarity between entities and entities. Then, according to the evaluation results, the samples are selected as a similar set or dissimilar set and finally used in the classical metric distance learning algorithm. In the second part, we focus on a large entity table with multiple relational tables between samples. In view of this situation, we propose two schemes : one scheme is to combine multiple relational adjacency matrices into relational tensors and then perform RESCAL tensor decomposition [NTK11], and treat the decomposed matrix as a new feature space, apply the metric distance learning on this new feature space ; the other scheme is to directly accumulate the loss functions of multiple relational adjacency metric, construct a comprehensive loss function that considers both supervised information and unsupervised information, and then optimize the function. It is worth mentioning that the proposed algorithm in the first part can also be easily extended for the second case. Finally, we conducted an experimental evaluation of these three relational metric learning algorithms and compared them with other metric distance learning algorithms on real data sets, and achieved excellent results on some data sets.

At the end of this article, in the conclusion section, we re-synthesized and collated the main contributions of this paper, and made some perspectives for related work in the future. We also listed some of the problems encountered in the study, describing the studies and experiments that we tried but did not go deep because of time, equipment or other constraints. In the perspectives of the future, we have proposed some possible solutions to the problems encountered, and some new ideas presented without in-depth study.

x	unknown variable	$\mathbf{x}$	entity represented as vector
x_i	i-th value of vector $\mathbf{x}$	$\mathbf{x}_i$	i-th entity
i, j, k, l	index parameter	y	label
w	weight parameter	$\mathbf{r}$	relationship information
m	number of dimension of $\mathbf{x}$	$\mathbb{X}$	entity set
n	number of instance	$\mathbb{S}$	similar constraints set
a, b	user-given parameter	$\mathbb{D}$	dissimilar constraints set
u, v	threshold parameters	$\mathbb{C}$	constraints set
t	time parameter	$\mathcal{S}$	index set
h	depth parameter	$\mathbb{V}$	vector set
p	power parameter	$\mathbb{P}_{ij}$	common parents set
λ	balance parameter	$f()$	defined function
γ	margin	$d(\mathbf{x}_i, \mathbf{x}_j)$	distance function
α	learning rate parameter	$L()$	loss function
κ	number of constraints	$r()$	regularization function
M	parameter matrix	$\ell(\mathbf{x}_i, \mathbf{x}_j)$	encoding loss of selected constraints
M_c	co-variance matrix	H(x)	squared hinge function
L	linearly mapping transfer matrix	$d^s(\mathbf{x}_i, \mathbf{x}_j)$	similarity functions
I	identity matrix	N( )	normalization function
σ	sequences of symbols	$\phi()$	nonlinear function
χ	time series	$LS(\mathbf{x}_i, \mathbf{x}_j \vert \mathbf{r}_k)$	link-strength function
ϖ	tree	var^n	numerical variables of reference link
T	tensor	num^n	number of numerical variable
R	core tensor of RESCAL factorization	var^c	categorical variables of reference link
A_l	latent space matrix of RESCAL factorization	C	sparse matrix
$\mathbb{C}_\mathbb{L}$	constraints set selected by label	num^c	number of categorical variable
$\mathbb{C}_\mathbb{R}$	constraints set selected by relation	n_r	number of relationship

TABLE 1 – Notations

CHAPITRE 1

BASIC MATERIALS

In this chapter, we mainly introduce the background of the knowledge of machine learning and relational learning and define the concepts and symbols commonly used in metric distance learning in the full text. We introduce machine learning not only because metric distance learning still follows the macro framework of machine learning, but also because the result of metric distance learning is an adaptive metric, which strongly reflecting the measurement effect by a machine learning algorithm that relies on metric selection. In other chapters of this paper, the test experiments for metric distance learning are mostly based on the KNN algorithm, k-means algorithm or related algorithms. Relationship learning is the main problem to be solved in the second part of this paper. Relationship learning involves many different machine learning aspects. In this chapter, we briefly introduce commonly related algorithms. In our work, we mainly focus on the learning of non-planar structural data with weighted networks. The key to measuring distance learning is metrics. We denoted the mathematical definitions and properties of metrics and general concepts and symbols in standard metric distance learning models in this part. Finally, we introduce the application of metric distance learning and the reasons we pay attention to metric learning.

1.1 Introduction of Machine Learning and The Algorithms Strongly Depends on Metric

Machine learning is a field of computer science related to the study of pattern recognition, probability theory, computational statistics theory, convex analysis and many theories in the domain of artificial intelligence. Machine learning extract knowledge from data. Generally, the knowledge is a model building by explored pattern or data-driven algorithm from the input samples. Moreover, the model could be used to predict the target attribute or making decisions.

Today, machine learning is not only widely used in many real-world areas such as finan-

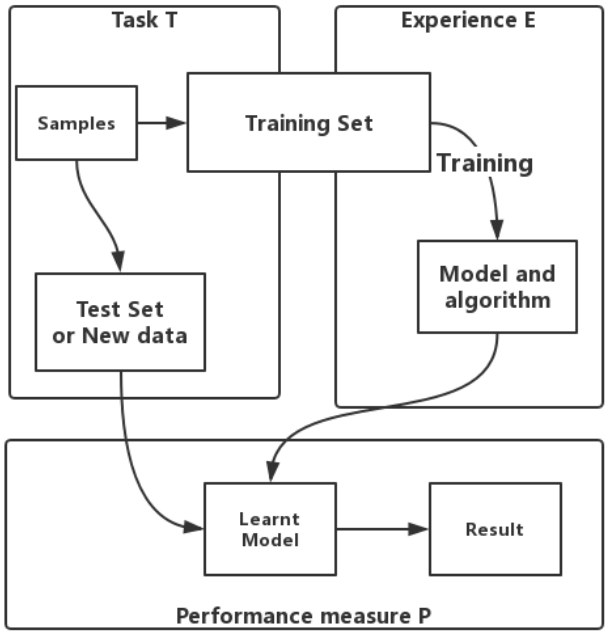


FIGURE 1.1 – Basic model of maching learning.

cial services, marketing forecasting, health-care, government analysis, but also in many related areas of artificial intelligence such as natural language understanding, non-monotonic reasoning, machine vision, pattern recognition. Standard machine learning methods include decision trees, random forests, k nearest neighbour algorithms, Bayesian learning, support vector machines, artificial neural networks, and many methods that are difficult to enumerate. Machine learning can be generalized as computer algorithms that are automatically improved through data or experience. [M$^+$97] gives a formal definition of machine learning : A machine learning is said to learn from experience E for some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

As shown in 1.1, it is a basic model of machine learning. A sample space is a set of all

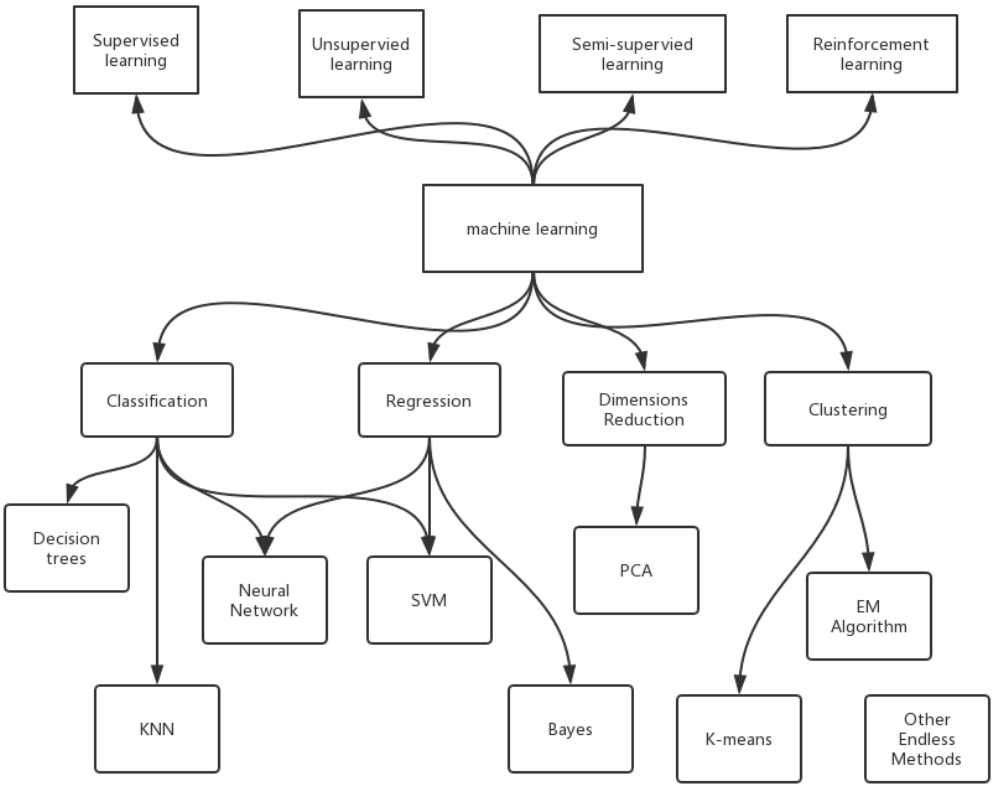


FIGURE 1.2 – Different branches of machine learning method.

the possibilities for a problem. The training sample is a piece of data for training, including some feature vectors for input and a label for output, where the feature vector is a collection of feature attributes of a sample and the label is a target tag to be output by a model.

A training set is a collection of multiple training samples. A test set is a collection of multiple test samples. Test samples are similar to training samples, but for testing. Training is the process of learning through training data and models. Testing is the process used to judge whether a model is good or bad. Prediction is the process of using a model to classify or regression future data.

As shown in Figure 1.2, there are many branches of machine learning algorithms. According to the presence or absence of label information in the training set, machine learning

can be classified into the following categories :

— Supervised learning : The sample data in the training set contains both the input features and the desired output label. Such a training set is learned, and its learning purpose is to correctly predict the label information of the sample data without labels. Supervised learning is often used to predict regression of data, classification of labels, and ordering of sequences.
— Unsupervised learning : Sample data in the training set of unsupervised learning does not carry label information. The purpose of unsupervised learning is different from supervised learning, not responding to feedback or predicting the labels, but learning patterns in the data, identifying commonalities in the data, and predicting whether such commonality exists in the new data. Unsupervised learning is often used for clustering and anomaly detection.
— Semi-supervised learning : Some sample data parts in the training set have the label information, and some have no label information. In this case, it may be the task of supervised learning after unsupervised learning, or the use of supervised learning to assist in unsupervised learning.
— Reinforcement learning : Reinforcement learning is rather special in that its feedback is not a sample label, but environmental feedback (Such as a reward and punishment signal). Reinforcement learning is to achieve the goal of the task, gradually adjust its behaviour as the environment changes, and evaluate the feedback that each action is positive or negative. The previously mentioned machine learning general assumptions are consistent with the Markov decision process (MDP), while the reinforcement learning algorithm does not have this requirement and can be used when the exact model is not feasible. Reinforcement learning is often used for automatic control of robots, the artificial intelligence of games, and optimization of market strategies.

Today's machine learning algorithms are endless. This article does not focus on the various branches of machine learning algorithms. Therefore, only a few classic machine learning algorithms related to metric distance learning algorithms are listed here. The effects of these algorithms are mostly dependent on the choice of metric distance, so they are often used as an experimental test algorithm to measure the effect of the metric learning algorithm. Notice that some models of metric distance algorithms are also associated with these algorithms.

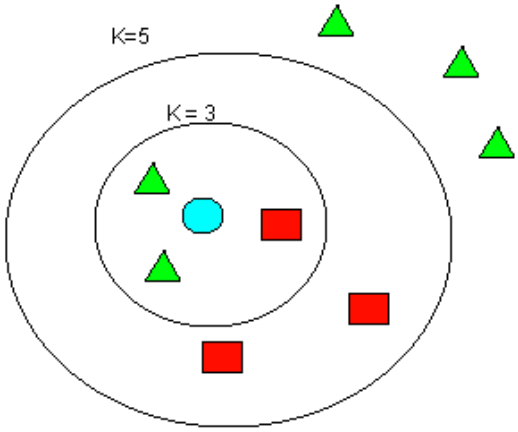


FIGURE 1.3 – An example of K-Nearest Neighbor classification.

1.1.1 K-nearest-neighbor

The KNN(K-Nearest Neighbor) classification algorithm is one of the simplest methods in machine learning classification technology[Alt92]. The so-called K Nearest Neighbor means that each sample is represented by its nearest K neighbours. As shown in Figure 1.3, the green circle is the new sample to be decided which class to give, is it a red triangle or a blue square ? If $K = 3$, since the proportion of the red triangle of all 3 nearest neighbours is $2/3$, the green circle will be given the class of the red triangle. If $K = 5$, the blue circle is given the blue square class because the ratio of the blue square all 5 nearest neighbours is $3/5$.

The core idea of the kNN algorithm is that if the majority of the K most neighbouring samples in a feature space belong to a specific category, the sample also belongs to this category and has the characteristics of the samples on this category. The method determines the category to which the sample to be classified belongs based on only the category of the nearest one or several samples in determining the classification decision. In the KNN algorithm, the selected neighbours are all label supervised samples. In the categorization decision, the method determines the category to which the new sample to be classified belongs according to only the category of the nearest one or several samples. Although the KNN method also relies on the limit theorem in principle, it is only related to a tiny number of adjacent samples.

The KNN algorithm can be used not only for classification but also for regression. By identifying the K nearest neighbours of a sample and assigning the average of the properties

of those neighbours to the new sample, the properties of the new sample can be obtained as $\rho(\mathbf{x}) = 1/K \sum^{K} \rho(\mathbf{x}_{knn})$,where the $\mathbf{x}_{knn}$ is the K nearest neighbours of the target sample. A more useful method is to give different weights to the influence of the neighbours on the sample, such as the weight is inversely proportional to the distance.

The main advantage of KNN is that it is simple and easy to implement. Although the KNN method also relies on the limit theorem in principle, it is only related to a minimal number of adjacent samples. Since the KNN method mainly relies on the surrounding limited samples, rather than relying on the discriminant domain method to determine the category, the KNN method is more suitable than the other methods for the crossover or overlapping sample set of the domain. KNN is suitable for classifying rare label entities and multiple label entities. In these cases, KNN performs better than other machine learning algorithms such as Support Vector Machine.

The main disadvantage of KNN is that when the sample is unbalanced. Such as the sample size of one class is large, and the sample size of other classes is tiny, when a new sample is entered, it may cause the bulk class to account for the majority of the K neighbours of the sample. Secondly, another disadvantage of KNN is that it is computationally intensive because each of the samples to be classified must be calculated from the distance of all known samples to obtain its K nearest neighbours. Finally, KNN is poorly understandable and interpretable and cannot give rules like decision trees.

1.1.2 Support Vector Machine

SVM(Support Vector Machine)is a generalized linear supervised learning model for binary classification of data and regression analysis[CV95]. Given a set of training examples, each of which is labelled as belonging to one or the other of the two categories, the SVM determines a decision boundary by finding the hyperplane of the largest margin, constructs a non-probabilistic binary linear classifier, and can assign a new example as one category or another. The principle of segmentation is to maximize the margin and finally transform it into a convex shape quadratic programming problem.

The SVM model can be divided into linear separable SVM, linear SVM and nonlinear SVM according to whether the training samples are linearly separable. The linear separable SVM maximizes the distance from the vector samples to the decision boundary, while the linear SVM ignores some unique support vector samples for the approximately linearly separable training set to find the decision boundary. Finally, the nonlinear SVM uses a nonlinear method such as kernel function to convert the low-dimensional nonlinear case into

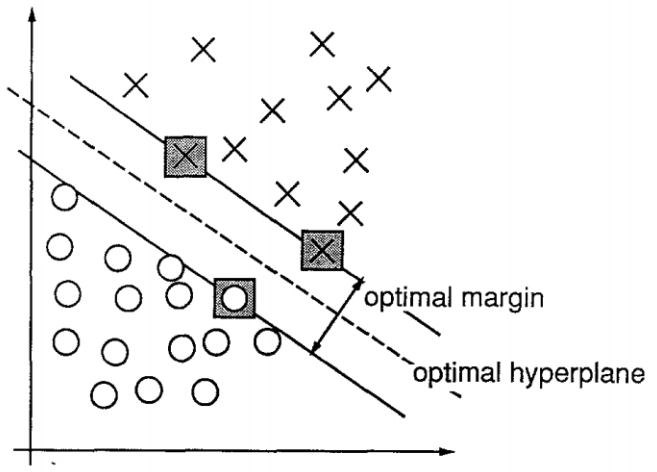


FIGURE 1.4 – The maximum-margin hyperplane as the decision boundary with support vectors.[CV95]

a high-dimensional linear case and then performs linear segmentation in high-dimensional space to find the hyperplane as the decision boundary. We will not discuss the complexity of SVM here but introduce linear separable SVM.

As shown in Figure 1.4, for the training samples $x \in \mathbb{X}$, where the $\mathbb{X}$ is the all training samples set, the learning target label is represented as a binary variable y, with $y = +1$ as the positive class and $y = -1$ as the negative class. The hyperplane of the decision boundary $wx - b = 0$ separates the learning objectives by positive and negative classes and makes the point-to-plane distance of any sample greater than or equal to 1. The parameters w is a normal vector to the hyperplane and b is the offset intercept of hyperplane from the origin. All samples above the upper interval boundary $wx - b = +1$ belong to the positive class, and samples below the lower interval boundary $wx - b = -1$ belong to the negative class. The distance between two spaced boundaries $\frac{2}{\|w\|}$ is defined as the margin, and the positive and negative samples at the interval boundary are support vectors. For finding the maximum-margin hyperplane, the problem is transfer to optimize the problem as follows :

$$\max \frac{2}{\|w\|} \tag{1.1}$$

subject to $y_i(wx_i - b) \leq 1 \; \forall x_i \in \mathbb{X}$. This example is a simple 2-dimension situation but could be easily extended to the general case.

When a classification problem does not have linear separability, using hyperplane as the decision boundary bring classification loss, that is, part of the support vector is no longer

located on the interval boundary, but enters the inside of the interval boundary or falls into one side of the decision boundary.

It is worth mentioning that the kernel method should also pay special attention in support vector machines, because it corresponds to the use of support vector machines in the case of non-linear separability. When it is difficult to find linear separable decision boundary in the feature space of the database, the feature space is mapped to the higher-dimensional Hilbert space through the kernel method, and then it is easier to find the separable boundary in the new space. In the future section in this work, we will introduce several metric learning algorithms based on the same idea and combing the kernel method.

The SVM quantifies the empirical risk of classification loss using a hinge loss function and adds a regularization term to the solution system to optimize structural risk. It leads SVM to a classifier with sparsity and robustness. Compared with KNN, SVM has better performance on the two-class problem, and the decision boundary learned by SVM is more understandable than KNN. Because SVM is more focused on support vectors with a smaller total number of samples, it is less computationally intensive than KNN when learning to train SVM, and SVM is more efficient than KNN when extended to online learning algorithms.

1.1.3 Neural Network

Artificial Neural Network is a research hotspot in the field of artificial intelligence since the 1980s. In the field of machine learning and cognitive science, the neural network is a structure and function that imitates biological neural networks. It builds a simple model similar to neurons and forms mathematical models or computational models of different networks according to different connection methods. The network performs an estimate or approximation. There are many nonlinear metric learning algorithms based on Neural Networks.

A neural network is a nonlinear computational model consisting of a large number of neurons connected. As shown in the Figure 1.5, for the j-th neuron, the input signal of a plurality of other neurons are accepted. x_i represents the signal from i-th neuron. The connection between every two neurons represents a weighting value for the signal passing through the connection. w_{ij} represents the weighting value of the i-th neuron on the j-th neuron. The neuron combines the effects of the input signal with some operation lead to the output of the neuron y. Each neuron represents a specific output function called an activation function. The output varies depending on the connection method of the network,

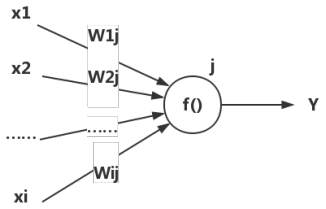


FIGURE 1.5 – A neuron in the neural network.

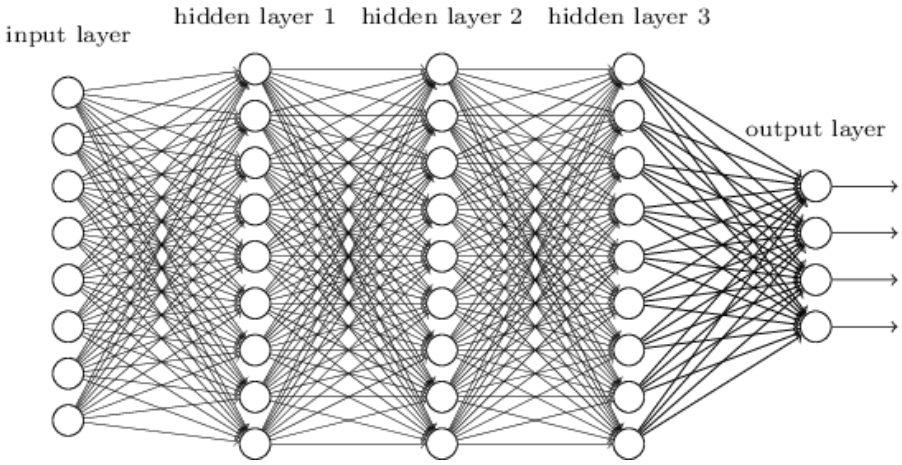


FIGURE 1.6 – The forward multi-layer network.

the weight value and the excitation function.

A neural network is a complex interconnected system, and the interconnection mode between neurons have an important impact on the property and function of the network. There are many types of interconnecting modes, including the two most basic connections and their extensions, forward networks and feedback networks.

As shown in the Figure 1.6, the forward network can be divided into several layers. The layers are arranged in the order of signal transmission. The neurons in the i-th layer only receive the signals given by the $(i-1)$-layer neurons. There is no feedback between the neurons. The first layer and the output layer are collectively referred to as the "visible layer", while the other intermediate layers are referred to as the hidden layer, and these neurons are called hidden neurons. The BP (Back-propagation) network[RHW$^+$88] is a typical forward

network. The BP neural network performs weight adjustment by propagating the sample signal in the forward direction and calculates the backpropagation error after the error by monitoring the information. BP neural networks have some limitations : for example, some areas on the error surface are flat, and the error is not sensitive to changes in weights, and the gradient changes are small. However, it is the basis of many neural network models and has laid a foundation for the development of deep learning multi-layer neural networks.

Feedback neural network is structurally different from the forward network. Each neuron of the feedback neural network represents a computing unit and accepts input signals and feedback inputs from other neurons, and each neuron is also directly output to the outside. The Hopfield network [Hop82] is of this type. The neurons in the Hopfield network are mutually constrained. Under the excitation of the external supervision information, the dynamic evolution state is entered to learn the weight adjustment until the equilibrium state is reached. Hopfield neural network is a nonlinear dynamic system with rich, dynamic characteristics, such as stability, finite-loop state and chaos state. It has various applications in associative memory and optimization calculation.

In most cases, artificial neural networks can change weights through learning or training processes based on external information. It is an adaptive system. The same network can have different functions depending on the learning method and content. The neural network as a whole can be seen as a simulation or approximation of an algorithm, function, or logic rule.

Nowadays deep learning models are utilized for solving many machine learning tasks, and particularly performance exciting results on the image processing, computer vision, natural language processing, social network filtering, machine translation, game programs and many related areas. Deep learning and multi-layer neural networks are often mentioned side by side, but it is worth pointing out that the two are not precisely equivalent. Deep learning, like metric distance learning, is a branch of representation learning which is extract features from the original sample data and learning to learn. Deep learning is called "deep" because the core idea of deep learning is to learn from shallow to deep, learn local features first to learn the full features or learn surface features first to learn depth features. The multi-layer neural network is the framework to realize the deep learning idea. In the multi-layer neural network, the pre-hidden layer is mainly to learn the preliminary local features and provide the following hidden layer for further deeper learning. The result of the last layer before the output layer is the final feature of learning. Therefore, the multi-layer hidden layer structure in the multi-layer neural network represents the deep feature

learning to learn the features, and the final output layer generally represents the normal machine learning process.

Deep learning architectures include deep neural networks, deep belief networks, recurrent neural networks, convolutional neural networks and many different extension of them. In our work, we will not introduce all of them, but recommend more details in [Den14] and [LKB+17].

1.2 Introduction of Relational Learning and the Relational Database We Follow

Relational learning is one of the branches of artificial intelligence and machine learning. Its goal is to learn the relationship between target samples or internal associations in the complex structure of samples, which are uncertain and statistical[SB17]. The relationship can refer to an external association or internal association at the same time. More theories can be distinguished according to these two settings, but in essence, there is no difference between the two relationship learning. Relationship learning differs from other machine learning in that, and in addition to processing the feature information of the sample itself, the relationship should be treated as an additional source of information or the relationship need be treated as an additional feature representation. Therefore, the learning tasks of relational learning based on relationship information or predictive relationship information, including collective classification, logical interpretations, link-based clustering, link prediction.

Although when referring to relational learning, it generally refers to statistical relational learning, but can not generalize all relational learning into a kind of statistical relational learning. Next, we will introduce several kinds of relational learning theories from different angles.

1.2.1 Statistical Relational Learning

Statistical Relational Learning (SRL or probabilistic logic learning) is the most fundamental theory in relational learning. SRL focuses on the uncertainty of relationships and tries to learn the probability distribution of this uncertainty [KFD+07]. The most famous and classic Probabilistic Relational Models (PRM) in SRL [FGKP99] is a kind of Bayesian-based statistical relationship learning method. It is an extension of the standard Bayesian

network model to relational expression. PRM uses the Entity-Relationship Model, which represents the relationship between entities, as the primary representation framework. The structure of the model describes the relational model and the dependencies between the attributes. The parameters of the model describe the probability distribution of the dependencies between the objects and the attributes. Another well-known classical probability model is the Markov Logical Network (MLN) [RD06], which is defined as a collection of weighted first-order logic formulas. A formula is a constraint on a logical interpretation, and weight is a contribution to a given formula. MLN upgrades Markov networks to first-order logic and allows networks with cycles.

1.2.2 Inductive Logic Programming

Logic programming is a programming paradigm that sets the rules that the answers must conform to solve the problem, rather than setting steps to solve the problem. The process is fact + rule = result. The Inductive Logic Programming (ILP) is extended from the LP, using the first-order predicate logic to describe the relationship [DR08]. ILP differs from SRL in that it focuses more on the transformation and structure between relationships in relational learning. ILP and MLN have some similarities. The ILP language Prolog [Bra01] is a common method of representing objects and the relationships between them.

1.2.3 Graph Mining and Multi-Relational Mining

When it comes to graph-based relational learning or multi-relational data mining, it is mainly for the type of relational database. Relational databases typically consist of entity tables and relational tables. When we represent the entity as a node of the graph and the relationship is represented by the edge of the graph, the relationship learning can be described as learning the graph. The relational learning task can be described as the prediction or regression of the node attribute or the edge attribute. This type of setup is common in specific relational learning tasks and is often used to describe real-world databases, such as web mining, social network analysis. Multi-relational data mining can be described as a weighted graph or a set of different graphs of the same batch of nodes. This setting is more common in the particular multi-relational database with only one entity table and multiple relational tables. Unlike SRL and ILP, graph-based data mining tends to focus more on the structure of graphs than on other relational learning methods, rather than the properties of individual nodes or the expression of a single relational rule. For example,

[HRW10] focuses on the data mining method of subgraph representation information or [GF18] uses the graph embedding method to transform the structure of the partial graph into the characteristics of the sample. Also, when faced with multi-relational data mining with multiple relational tables or arguably multiple graphs representing different relationships, a standard method is propositionalization. Propositionalization is a learning method that automatically converts relational information into sample features. Graph embedding methods can be classified as a kind of propositionalization, while metric distance learning is a kind of representation learning, and it is naturally related to it. We will introduce more details in the following chapters. Notice that, if multiple relationships can be represented as multiple different predicates, most of the ILP relational learning algorithms can also be directly applied to multi-relational mining[Dže10].

In addition, relational learning also includes related methods such as relational reinforcement learning and deep-relationship learning, which will not be repeated in this article. This paper mainly studies the application of metric distance learning to relational learning for multi-relational mining. In the following chapters, we will introduce the metric distance learning algorithm for non-planar relational databases, including relational databases and our proposed relationship metrics learning algorithm.

1.3 Basis of Metric Learning

In this section, we introduce some of the basic concepts of metric distance learning throughout the whole article.

As the basis and goal of metric distance learning, the definition of the metric itself is crucial. Metric is a concept describing the similarity of entities in general. Moreover, in mathematics metric is a function that defines a distance between each pair of elements of a set.

A metric is a function $d : \mathbb{V} \times \mathbb{V} \to \mathbb{R}^+$ on a set $\mathbb{V}$, $\forall \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k \in \mathbb{V}$ satisfying the following conditions :

1. non-negativity : $d(\mathbf{x}_i, \mathbf{x}_j) \geq 0$
2. identity of indiscernibles : $d(\mathbf{x}_i, \mathbf{x}_j) = 0 \Leftrightarrow \mathbf{x}_i = \mathbf{x}_j$
3. symmetry : $d(\mathbf{x}_i, \mathbf{x}_j) = d(\mathbf{x}_j, \mathbf{x}_i)$
4. triangular inequality : $d(\mathbf{x}_i, \mathbf{x}_j) + d(\mathbf{x}_j, \mathbf{x}_k) \geq d(\mathbf{x}_i, \mathbf{x}_k)$

The first two conditions define a positive definite function.

There are many different metrics for different situations, such as Euclidean distances for flat databases, Hamming distances for strings, edit distances for trees or graphs, all of which meet the above definition. However, some "metric"s in the metric learning algorithm only meet the above definitions under finite conditions, or do not meet the third triangular inequality condition. These "metric"s can be called similarity, and the similarity is used for special learning tasks. It can also achieve the desired learning objectives. While for the metric learning algorithms which are not based on the current metric, they need to prove the defined metric satisfying above conditions, such as the metric learning with neural networks[CHL05].

One of our proposed algorithms is proved by a proposed norm to define a metric, because it is well known that if N is a norm, then $d(\mathbf{x}_i, \mathbf{x}_j) = N(\mathbf{x}_i - \mathbf{x}_j)$ is a metric.

A norm is a function that assigns a strictly non-negative length or size to each vector in a vector space, which has a direct link with a metric.

A norm is a function $N : \mathbb{V} \to \mathbb{R}^+$ on a vector space $\mathbb{V}$ satisfying the following conditions :

1. separates points : $N(\mathbf{x}) = 0 \Leftrightarrow \mathbf{x} = 0$
2. absolute homogeneity : $N(a\mathbf{x}) = |a|N(\mathbf{x}) \forall \mathbf{x} \in \mathbb{V}\ \forall a \in \mathbb{R}$
3. triangular inequality : $N(\mathbf{x}_i) + N(\mathbf{x}_j) \geq N(\mathbf{x}_i + \mathbf{x}_j)\ \forall \mathbf{x}_i, \mathbf{x}_j \in \mathbb{V}$

In the linear case, all norms are exceptional cases of the Minkowski gauge with a bounded convex set.

Another principal basis for metric distance learning is the selection constraints from similar and dissimilar sets. Different from other machine learning methods which direct input the sample training set as the learning model, the metric distance learning algorithm also requires classification of the sample set according to the task's target and the presence or absence of supervised information, which are generally divided into similar sets $\mathbb{S}$ and dissimilar sets$\mathbb{D}$. This segmentation process is also the reprocessing of supervised information such as labelling, sorting, or unsupervised information such as timing, structure. The techniques for selecting constraints vary according to the metric learning algorithm, and we will introduce more details in subsequent chapters.

1.4 Metric Learning Applications

Because metric distance learning is a branch of learning, it has applications that apply to any task that cares about metrics or similarities between entities. Metric distance learning can help most machine learning algorithms to improve performance and has been applied to image retrieval, face recognition, text retrieval, tracking recognition, music recommendation, web indexing in the field of machine learning or data mining. Here, we won't list a lot of applications, only the applications that metric distance learning to show the best results : computer vision, information retrieval. We will introduce more application details when describing the unique metric distance learning algorithms designed for specific tasks or specific database in subsequent chapters.

1.4.1 Computer Vision

Metric distance learning has achieved many successes in various fields of computer vision, mainly in image retrieval, face recognition, tracking recognition.

Image retrieval and classification is a fundamental task in machine vision. Image retrieval is divided into two categories : text-based retrieval and content-based retrieval[DJLW08]. Here we are referring to content-based image retrieval. The task objective is to detect other images with similar characteristics from the image database based on the content semantics of the image. Therefore, it is a native application to use the metric distance learning algorithm to find the nearest neighbour image. It is worth mentioning that most of the algorithms demonstrate their ability in image retrieval more or less when visualizing the effects of the learning algorithm. For example, [JKG08] and [CSSB10] both show good results on different image databases. [HLLM06] and [HLC10] are metric learning algorithms designed for image retrieval, and produced the most superior results at that time, and guided the further development of metric learning algorithms in the field of machine vision. Recently, an article [WWY18] has attempted to improve and develop the metric learning algorithm for image retrieval in terms of deep learning.

The task of face recognition is to identify whether two face images belong to the same person. Similar to image retrieval, face recognition can be thought of as retrieving other similar images of a person in the database. So the image retrieval algorithm mentioned earlier can also be used for face recognition. [GVS09] is one of the most popular metric learning articles specifically for face recognition. The author proposes two metric learning algorithms for the databases in [HMBLM08] and compares it with other contemporary

techniques in [HMBLM08], and has achieved better results. [CVS11] proposed an unsupervised version of the first method in [GVS09] and applied it to TV video.[CHL05] is the first article to use CNN(Convolutional Neural Network) algorithm combined with metric distance learning theory for face recognition. Based on it, new CNN metric learning algorithms are proposed and extended to the field of deep learning.

Recently, the newer deep metric learning methods for face recognition have algorithms such as [WZLQ16] and [TWR$^+$16]. Another more challenging task associated with face recognition is kinship verification [FTSC10]. The purpose of the kinship verification task is to identify whether the person in the two face images has a kinship or a macro similarity. Based on the classical metric distance learning algorithm of LMNN(Large-Margin Nearest Neighbors) [WBS06], both [HLYT14] and [LZT$^+$13] propose metric distance algorithms for finding the nearest neighbour of pairs of the face image. In the recent [LHT17], the author Lu et al. extended the algorithm to a version of deep learning.

1.4.2 Information Retrieval

Information retrieval, or called information search, refer to the search process that uses the precise algorithm and the retrieval model or tool to find the required information from the information sample set according to the needs of the task[JR10]. Information retrieval is a process of matching a search target with a sample of information. Corresponding to the metric learning algorithm, it is the process of sorting the samples according to the nearest neighbour principle after learning the metrics that meet the task requirements according to the algorithm model. Image retrieval is one kind of information retrieval. Therefore, the metric learning algorithm of image retrieval we mentioned earlier can also be classified as an information retrieval algorithm. Besides, the metric learning algorithm has also achieved remarkable results in the field of text retrieval.

Unlike other similar metrics learning algorithms, the metric learning algorithm that handles text retrieval or text analysis faces a challenge : the curse of dimensionality. The standard model for text document data is the TF-IDF model, which is usually represented as a sparse vector of the keyword frequency. For most conventional metric learning algorithms with $O(m^3)$, complexity with the m is the number of features of the sample, the dimensions of this vector far exceed the range they can handle. [DD08] and other algorithms use a nucleation method to process high-dimensional data. The nucleation model of the classical metric learning algorithm ITML [DKJ$^+$07] is also used. Another idea is to use a similarity function without PSD constraints as a metric model for the metric learning

algorithm. [QGCL08] and [QG09] are both text metric learning algorithms based on cosine similarity. [YTPM11] combines the Siamese architecture of the neural network proposed in [CHL05] to learn the projection matrix, which projects the conceptual space of the text document to the term vector. Recently, an article [XHL+19] has improved and developed the metric learning algorithm for cross-retrieving of images and texts from the aspect of deep learning.

1.5 Conclusion

In this chapter, we give a brief introduction to machine learning and explain the importance of metrics for machine learning. We highlight the two algorithms, KNN and SVM, which are applied later to most metric learning algorithms. The background related to deep learning and multi-layer neural networks is also briefly introduced. It provides a background for the introduction of the depth metric learning algorithm. For the second part of this article," Proposing relational metric learning algorithms", we introduce the similarities and differences between several different relational learning algorithms in this chapter. We emphasize that this paper focuses on relational learning and multi-relational data mining algorithms for graph-based relational databases. Then, this chapter introduces the probabilities and symbols that are common in metric learning algorithms. The mathematical definition of this metric is primarily explained. Finally, this chapter introduces the application of metric learning algorithms and embodies the importance and necessity of developing metric learning algorithms.

CHAPITRE 2

METRIC LEARNING FOR FLAT DATASET

Machine learning algorithms can handle a wide variety of tasks, but when researching and discussing algorithms, they usually start with the most straightforward database with only one table. For this table, each row of the table represents an entity, each column of the table represents different features or attributes of the entity (sometimes including the target feature as a label). For this situation or a database that can be easily converted into this case, we call it a flat database. Algorithms developed on a flat database can be extended or transformed to non-flat databases as needed. In this chapter, we will introduce the development and evolution of metric distance learning algorithms on flat databases for many years. The purpose is not only to summarize the common models of metric distance learning algorithms, but also to find the worthy development and attention from the similarities and differences of algorithms. After investigation, we found that the metric distance learning algorithm based on Mahalanobis distance plays a major role, but it is limited to learning the weights of a single-dimensional or two-dimensional union. This discovery prompted us to propose a metric distance learning algorithm for learning the weights of the coalitions of three or more dimensions in the next chapter.

There are many different types of metric distance learning algorithms, and we follow the basic model of metric learning mentioned in several surveys[YJ06][K$^+$13][BHS15]. We mentioned the simple principle of metric distance learning in the Chapter and the general model that extends from it can be divided into the following steps.

— Metric Construction : Depending on the idea or the peculiarity of of the dataset, construct a new distance function $d'(\mathbf{x}_i, \mathbf{x}_j)$. Normally, $d'(\mathbf{x}_i, \mathbf{x}_j) = d(f(\mathbf{x}_i), f(\mathbf{x}_j))$ and $d(\mathbf{x}_i, \mathbf{x}_j)$ is a current distance (for example, Minkowski distance, Euclidean distance or anything similar) and the f is a transfer function mapping the original feature space to a latent feature space. There are several model for construct the new metric, like the Mahalanobis distance or linear similarity ;

— Constraints Selection :Depending on the learning task and the availability of the target label or other feedback, metric learning algorithm require select constraints from the information of samples to learning the new metric. The most popular constraints selection forms are threshold constraints $d_M(\mathbf{x}_i, \mathbf{x}_j) \geq u, \forall(i,j) \in \mathbb{S}$ for similarity set $\mathbb{S}$ and $d_M(\mathbf{x}_i, \mathbf{x}_k) \leq l, \forall(i,k) \in \mathbb{D}$ for dissimilarity set $\mathbb{D}$ and relative constraints $d_M(\mathbf{x}_i, \mathbf{x}_k) \geq d_M(\mathbf{x}_i, \mathbf{x}_j) + \gamma \forall(i,j,k) \in \mathbb{C}$ where $\mathbb{C} = (i,j,k)|(i,j) \in \mathbb{S}, (i,k) \in \mathbb{D}$;

— Learning Processing : Generally for learning the new metric, a loss function $L(M) = \sum_{(i,j,k)\in\mathbb{C}} \ell_M(i,j,k) + \lambda r(M)$ is proposed to measure the performance of the new metric with the parameter matrix M. The loss function contains two part, one is the sum of the encoded loss based on the new metric from every triple (i,j,k) in the selected constraints set $\mathbb{C}$, the other one is the regularization $r(M)$ with a balance parameter λ;

So in this chapter, we will follow the mode, as shown in Figure 2.1 and switch to different branches to summarize the current metric learning algorithms for flat datasets. In order to make the classification of the algorithms of different metrics more precise, we mark the algorithms of different branches as different colours in the figure and the title of the algorithms of the corresponding metric. The algorithms based on Mahalanobis distance are marked as red, the algorithms based on linear similarity are marked as green, the algorithms of the nonlinear metric are marked as blue, and the algorithms of the local metric are marked as purple.

— First, in the 2.1 section, we will start with the construction of the metrics, revisit the definition of the metrics and introduce various metrics for different situations. Most metric learning algorithms are based on the Mahalanobis distance metric and try to learn a Mahalanobis distance with weights. Also, linear similarity and nonlinear metrics are metrics worth considering. We will also discuss the differences between learning global metrics and local metrics in this section.

— Second, in the 2.2 section, we will focus on how to learn the parameters of the metric. We will start with two parts of the loss function : loss coding from the selected constraints and regularization factor, and then introduce the different methods of optimizing the loss function. We will introduce how the metric learning algorithm will select and combine constraint selection methods, regularization methods, and optimization methods. Also, compare the advantages and disadvantages of these different settings.

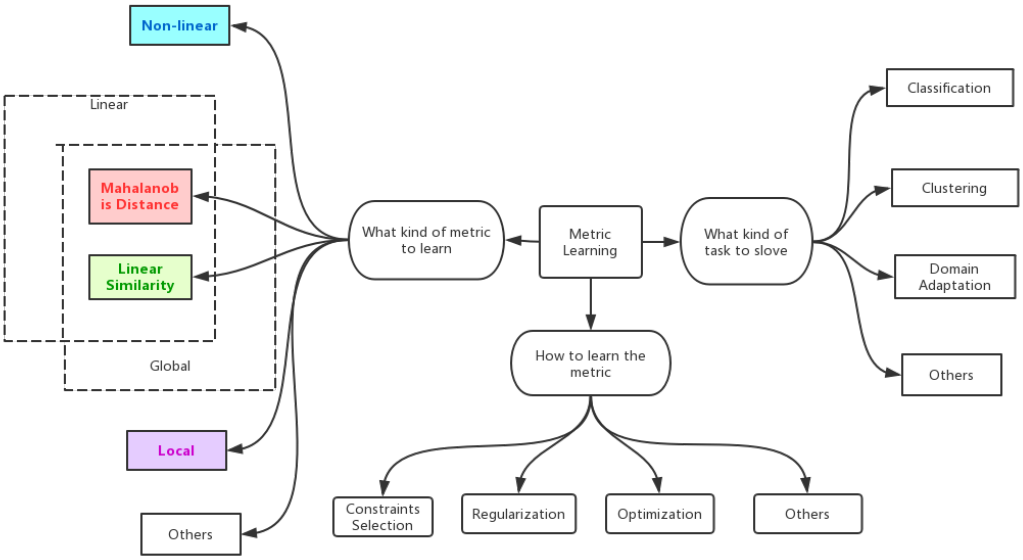


FIGURE 2.1 – Different branches of metric learning approaches.

— Metric learning can generate better metrics that can be used not only for traditional machine learning tasks but also for other special tasks. In the 2.3 section, we will cover common tasks that are solved by metric learning and specific algorithms related to specific tasks.

Nowadays, deep learning is one of the essential branches of machine learning. So in the 2.4 section, we will briefly introduce several metric learning combined with deep neural networks.

At the end of this chapter, we will summarize the metric learning algorithms we listed in the 2.5 section and briefly discuss the trends in metric learning algorithms. We illustrate the motivation again for improving the metric learning of flat datasets based on the Mahalanobis distance learning model.

2.1 Metric Constructions

As we mention in Chapter 1, a metric is a function A metric is a function $d : \mathbb{V} \times \mathbb{V} \rightarrow \mathbb{R}^+$ on a set $\mathbb{V}$, $\forall \mathbf{x} \in \mathbb{V}$ satisfying conditions : non-negativity, identity of indiscernibles, symmetry and triangular inequality.

For metric learning algorithms, the metric construction step is to define the new metric $d'(\mathbf{x}_i, \mathbf{x}_j) = d(f(\mathbf{x}_i), f(\mathbf{x}_j))$ with the transfer function f, which f need limit conditions or established form for making sure the $d'(\mathbf{x}_i, \mathbf{x}_j)$ is still a metric. There are already several models on the method of metric constructions, for example, the Mahalanobis distance model, the similarity learning model[YJ06][BHS15]. We will introduce some of them with classical metric learning algorithms.

Besides these models, the metric form also could be separated as the linear metric and nonlinear metric by the form of data, or be separated as the global metric and local metric for the particular dataset(such as heterogeneous data) or large size of the dataset. We will specially discuss the nonlinear metric and local metric learning the following sections.

2.1.1 Mahalanobis Distance Learning Model

The Mahalanobis distance [Mah36](also known as the generalized ellipsoid distance [ISF98]), is initially defined as a measure of closeness between a point and a distribution. Currently,

the Mahalanobis distance is denoted as following :

$$d_M(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T M^{-1} (\mathbf{x}_i - \mathbf{x}_j)} \tag{2.1}$$

where M has to be positive definite to obtain a proper distance. If the matrix is positive semi-definite (PSD), then it is a pseudo distance, i.e., the constraint on the identity of indiscernible is relaxed, and it is only required that $\forall x, d(x, x) = 0$. Notice that if $M = I$ the identity matrix, it corresponds to the Euclidean distance.

In its original definition [Mah36], M is the inverse variance, which is the covariance matrix of the examples. In the metric learning algorithms, the M could be the parameters to learn. The intuition behind the Mahalanobis distance is to re-weight every dimension of the feature space of the samples or could be considering mapping the samples linearly to a new latent feature space by the Cholesky decomposition that $M = L^T L$.

$$\begin{aligned} d(\mathbf{x}_i, \mathbf{x}_j) =& \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T M^{-1} (\mathbf{x}_i - \mathbf{x}_j)} && (2.2) \\ =& \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T L^T L (\mathbf{x}_i - \mathbf{x}_j)} && (2.3) \\ =& \sqrt{(L\mathbf{x}_i - L\mathbf{x}_j)^T (L\mathbf{x}_i - L\mathbf{x}_j)} && (2.4) \end{aligned}$$

So the metric learning algorithms based on the Mahalanobis distance is learning a linearly mapping transfer L matrix corresponds to a new latent space, which in this space the Euclidean distance could be better than in the original space.

Most of the classical metric learning algorithms are based on the Mahalanobis distance model. We will introduce several of them :

MMC(Mahalanobis Metric Learning) [XNJR02] [XNJR03]

In MMC [XNJR02] [XNJR03], the first Mahalanobis distance learning method is proposed for clustering. This proposed metric learning algorithm aims to maximum the distance between the dissimilar pair of data nodes, while keep the distance between the similar pair of data nodes low than the threshold(which is selected to be 1). The loss function is denoted as,

$$\max L(M) = \sum_{(i,j)\in\mathbb{D}} d_M(\mathbf{x}_i, \mathbf{x}_j) + r(M) \tag{2.5}$$

$$s.t. \sum_{(i,j)\in\mathbb{S}} d_M^2(\mathbf{x}_i, \mathbf{x}_j) \leq 1 \tag{2.6}$$

As the first metric distance learning algorithm, MMC is quickly surpassed by subsequent algorithms in terms of performance, but it clarifies the general model of metric learning. For an extended period, the new metric distance learning algorithm only differs in the choice of the algorithm's constraints selection, regularization, and optimization algorithms.

SJML(Schultz and Joachims Metric Learning) [SJ04]

In SJML [SJ04], they use the L_2 loss regularization, but with the relative distance constraints that$\ell(X^T MX) = d_M(\mathbf{x}_i, \mathbf{x}_k) - d_M(\mathbf{x}_i, \mathbf{x}_j) - \gamma, \forall(i,j,k) \in \mathbb{C}$. The additional assumption is they limit the matrix learned is a diagonal matrix $M = M'DM'^T$, where D is diagonal matrix and M' is user-given matrix. The final loss function is denoted as :

$$L(M) = \sum \ell(X^T MX) + \lambda|M|^2 \tag{2.7}$$

where $\ell(X^T MX) = [d_M(\mathbf{x}_i, \mathbf{x}_k) - d_M(\mathbf{x}_i, \mathbf{x}_j) - \gamma]_+, \forall(i,j,k) \in \mathbb{C}$ and $M = M'DM'^T \succeq 0, Ddiagonal.$

The advantage of the method is the number of parameters will be limit to only grows linearly with the number of dimensions, which more reduce the complexity of computing than the general Mahalanobis matrix learning methods. This algorithm performs better than the MMC algorithm, but it is still not as good as the most commonly used LMNN and ITML that we will introduce later.

LMNN(Large-Margin Nearest Neighbors) [WBS06]

LMNN [WBS06] is one of the most popular methods for metric learning. The authors choose the relative distance constraints and propose a regularization based on [XNJR03]. This special regularization is writed as $tr(MC)$, where $C = \sum_{(i,j)\in\mathbb{S}} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T$. So $tr(MC) = \sum_{(i,j)\in\mathbb{S}} d_M(\mathbf{x}_i, \mathbf{x}_j)$ and we can treat it as a minimizer to the new metric

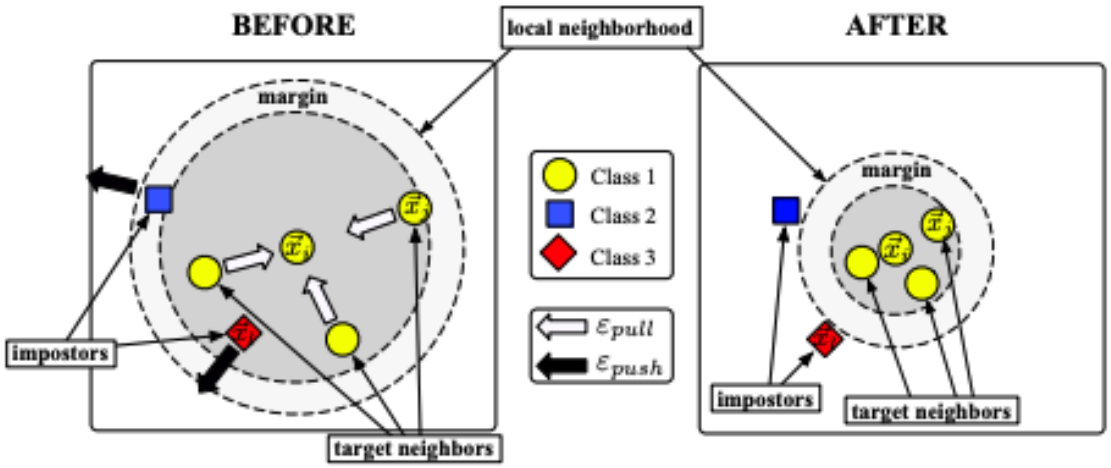


FIGURE 2.2 – Schematic illustration of one input's neighborhood before training (left) versus after training (right). [WBS06]

between the similar samples. The loss function is as follow :

$$L(M) = \sum_{(i,j,k)\in\mathbb{C}} [d_M(\mathbf{x}_i, \mathbf{x}_k) - d_M(\mathbf{x}_i, \mathbf{x}_j) - \gamma]_+ + \lambda \sum_{(i,j)\in\mathbb{S}} d_M(\mathbf{x}_i, \mathbf{x}_j) \tag{2.8}$$

where normally the γ is chosen to 1.

Figure 2.2 shows the illustration of the two aims of LMNN : differently labeled inputs lie outside this smaller radius by some finite margin, which is related the part of loss function $\sum_{(i,j,k)\in\mathbb{C}}[d_M(\mathbf{x}_i, \mathbf{x}_k) - d_M(\mathbf{x}_i, \mathbf{x}_j) - \gamma]$; while target point's k=3 target neighbors lie within a smaller radius after training, which is related the part of loss function $\sum_{(i,j)\in\mathbb{S}} d_M(\mathbf{x}_i, \mathbf{x}_j)$. The arrows in the figure indicate the gradients on distances arising from different terms in the cost function.

LMNN is the most commonly used metric distance learning algorithm, which has an excellent performance in most application scenarios, but it still has limitations as a linear algorithm, and it does not apply to some special tasks. Therefore, there are many extensions to the LMNN, such as for nonlinearization, localization, online learning. We will discuss some of them for different sets for other more particular cases.

ITML(Information-Theoretic Metric Learning) [DKJ+07]

Another most popular method for metric learning is ITML [DKJ+07]. They consider similarity and dissimilarity constraints under an information-theoretic regularization. They expect to minimize the Kullback–Leibler divergence of the PSD matrix M of learned metric and given matrix M_0 under the limit of the constraints.

Because $KL(\rho(x; M_0)||\rho(x; M)) = 1/2(tr(MM_0^{-1}) - logdet(AA_0{-}1) - d)$ and they chose M_0 with identity matrix, so the final loss function based on logdet divergence and shown as :

$$L(M) = \sum \ell(X^T MX) + \lambda(tr(M) - logdet(M)) \tag{2.9}$$

where $\ell(X^T MX) = d_M(\mathbf{x}_i, \mathbf{x}_j) - d_I(\mathbf{x}_i, \mathbf{x}_j), \forall(i, j) \in \mathbb{S}$

$\ell(X^T MX) = d_I(\mathbf{x}_i, \mathbf{x}_k) + \gamma - d_M(\mathbf{x}_i, \mathbf{x}_k), \forall(i, k) \in \mathbb{D}$

In ITML paper, they not only propose the regularization based on KL divergence but also propose the kernel metric learning, which will be discussed in the following section as a nonlinear models metric learning method. It is worth mentioning that although the computational complexity of ITML is higher than that of LMNN under many of the same conditions, it takes longer. However, the ITML algorithm has more flexibility in selecting constraints and input samples and has higher stability.

LSML(Least Squares Metric Learning) [LGZ+12]

LSML [LGZ+12] use the relative distance constraints like LMNN and the LogDet divergence regularization like ITML. Beside the combination of idea of the two papers, LSML propose a new way of encode the loss of constraints, they use squared hinge function and optimize over the sum of squared residuals in satisfying constraints. The squared hinge function $H(x)$ is defines as :

$$H(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x^2 & \text{if } x > 0 \end{cases} \tag{2.10}$$

And the relative distance constraints is be formed like $d_M(\mathbf{x}_k, \mathbf{x}_l) \geq d_M(\mathbf{x}_i, \mathbf{x}_j), \forall(i, j, k, l) \in \mathbb{C}$ where the constraints set $\mathbb{C}$ is select by the pair (i,j) is more similar than the pair (k,l).

The final loss function is as following :

$$L(M) = \sum_{l(i,j,k,l)\in\mathbb{C}} H(d_M(\mathbf{x}_i, \mathbf{x}_j) - d_M(\mathbf{x}_k, \mathbf{x}_l)) + \lambda(tr(M) - logdet(M)) \tag{2.11}$$

subject to $M \succeq 0$

The advantage of the LSML algorithm comes from the fact that the loss coding is easier to calculate. Although the author does not have LMNN and ITML in this paper, in the experiments of our paper, it can be found that LSML takes less time than LMNN and ITML under the same conditions, but the performance is relatively more mediocre.

2.1.2 Linear Similarity Learning

Although most of the linear metric learning algorithms are based on the Mahalanobis distance model, there still are several other measures attracted recent interest to learn[K$^+$13][BHS15]. These approaches are in the form of similarity functions also parametrized by a matrix M, but the M is often not required to be PSD. Most of the similarity functions are denoted as :

$$d^s_M(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i^T M \mathbf{x}_j}{f_S(\mathbf{x}_i, \mathbf{x}_j)} \tag{2.12}$$

where the $f_S(\mathbf{x}_i, \mathbf{x}_j)$ is a regularization term or other function depended on the method.

SiNo(Similarity learning for nearest neighbor classification) [QGCL08]

SiNo [QGCL08] is the simplest approach for learning similarity functions in this form. Also denoted the similarity function as :

$$d^s_M \mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i^T M \mathbf{x}_j}{N(\mathbf{x}_i, \mathbf{x}_j)} \tag{2.13}$$

where the $N(\mathbf{x}_i, \mathbf{x}_j)$ is a normalization term which depends on x_i and x_j.

The authors learned such a similarity function like the same idea in LMNN, which is that for the target point the dissimilar point should be farther than the similar one and the similar neighbour should be as near as possible. However, to optimize the similarity, they use an online learning algorithm based on voted perceptron.

SiCos(Similarity learning with cosine similarity) [QG09]

SiCos [QG09] could be considering as an special case of SiNo with the cosine as the normalization term, with the cosine similarity function as following :

$$d_M^s \mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i^T M \mathbf{x}_j}{\sqrt{x_i^T M x_i}\sqrt{x_j^T M x_j}} \tag{2.14}$$

Where the M is symmetric and PSD, which implies a projection space for the cosine similarity. For learning such a similarity function, the authors also use an online learning algorithm with two steps : firstly a projection to obtain zero hinge loss on the current positive pairs, secondly another projection on the cone of PSD matrices. This online algorithm is almost the same as POLA(Pseudo-Metric Online Learning Algorithm) [SSSN04], which will be introduced in the following sections.

OASIS(Online Algorithm for Scalable Image Similarity) [CSSB09]

OASIS [CSSB09] is a bilinear similarity learning algorithms particularly for large scale datasets. It is an online learning for image retrieval based on k-nearest-neighbor and has the simple form :

$$d_M^s \mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T M \mathbf{x}_j \tag{2.15}$$

where the M is neither required to be PSD nor symmetric. For learning, the authors use online learning belongs to the closed-form update based on Passive-Aggressive algorithms [CDK$^+$06]. The initialization is M equal to the identity matrix, F_M amounts to an unnormalized cosine similarity. Then at each step t, with the hinge loss over the triplets (x_i, x_j, x_k) of relative constraints, the following convex problem is solved :

$$M^t = \arg\min 1/2 \left\| M - M^{t-1} \right\| + c\xi \tag{2.16}$$

$$s.t. 1 - d_M(\mathbf{x}_i, \mathbf{x}_k) + d_M(\mathbf{x}_i, \mathbf{x}_j) \leq \xi, \tag{2.17}$$

$$\xi \geq 0 \tag{2.18}$$

Where the $c \geq 0$ is a trade-off parameter, and ξ is a slack variable. OASIS efficiently handle sparse datasets and get better performance on medium-scale problems, allowing it to scale to millions of examples.

2.1.3 Nonlinear Metric Learning

We have introduced so many linear metric learning algorithms that we can say that metric learning is more focused on linear metrics. The reason for this is that linear metrics are easier to construct and optimize. However, in reality, many nonlinear structure databases are arduous to learn linear metrics after simple processing directly, or can not be applied by linear metric learning algorithms. Therefore, the metric learning algorithm has naturally evolved into a nonlinear field. In recent years, some scholars have also studied nonlinear metric learning methods to solve the problem in this case.

Kernel Metric Learning

In nonlinear case of metric learning algorithms, a general idea is using embedding or mapping function before the linear projection, which define the new metric as $d_{M,\phi(\mathbf{x})}(\mathbf{x}_i, \mathbf{x}_j) = \Big((\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^T M(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))\Big)^{1/2}$. The $\phi()$ is the nonlinear transformations. So kernelization can be a satisfactory proposition[YJ06][K$^+$13][BHS15].

Kernel metric learning aims to learn a linear metric in the nonlinear feature space induced by a kernel function ϕ. By the spirit from SVM, some linear metric learning method we mentioned before have been extended with kernelization, for example, the SJML [SJ04], [DKJ$^+$07] and [SSSN04].

With the nonlinear transformations $\phi()$, GB-LMNN(Gradient-boosted LMNN) [KTS$^+$12, KXW] is a nonlinear version of LMNN. The authors apply gradient-boosting to learn nonlinear mappings directly in function space and takes advantage of this approach's robustness, speed, parallelizability and insensitivity towards the single additional hyperparameter.

To generalize the LMNN objective 2 to a nonlinear transformation $\phi(\mathbf{x})$, the Euclidean distance with transformation is denoted as

$$d_\phi(\mathbf{x}_i, \mathbf{x}_j) = (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^\top (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))$$

Extend the loss function of LMNN as follow :

$$L(\phi()) = \sum_{(i,j,k)\in\mathbb{C}} [d_\phi(\mathbf{x}_i, \mathbf{x}_k) - d_\phi(\mathbf{x}_i, \mathbf{x}_j) - \gamma]_+ + \lambda \sum_{(i,j)\in\mathbb{S}} d_\phi(\mathbf{x}_i, \mathbf{x}_j)$$

The transformation $\phi(\mathbf{x})$ is defined with gradient boosted method as an additive func-

tion :

$$\phi_t(\mathbf{x}) = \phi_{t-1}(\mathbf{x}) + \alpha h_t(\mathbf{x})$$

where $h_t(\mathbf{x}) \approx argmin_{h \in t_h} L(\phi_{t-1}(\mathbf{x}) + \alpha h_t(\mathbf{x}))$ initialize with the linear transformation ϕ_0 learned by LMNN. The α is the learning rate and t_h denotes the set of all regression trees of limited depth h.

Beside apply the kernel method with the existed linear metric learning, it is worth to mention that several kernel metric learning is based on KPCA(Kernel Principal Component Analysis [SSM98]) which is a nonlinear extension of PCA. KPCA can be considered as using a kernel projecting the data into the nonlinear feature space induced and performs dimensionality reduction in this feature space. In [CKTK10], the authors using linear metric learning algorithms based on Mahalanobis distance in the KPCA feature space.

Multiple Metric Learning

Another possible approach to learning metric for nonlinear datasets is to one metric per region of the space or even one metric per examples. Learning multiple linear metrics has the capacity to capture the heterogeneities of complex tasks[K$^+$13].

LMNN has many different extensions, and Mul-LMNN(Multiple LMNN metric learning) [WS08a] is the multiple metrics learning version of LMNN. Mul-LMNN separated the training datasets in several clusters with supervised ways(labels) or with unsupervised ways(k-means). Then for each cluster, a metric is learned in a generalization from LMNN. Notice that, this could be considered as local metric learning for each cluster, the local distance is depended on the pair of data nodes are in which cluster and the global distance could be not symmetric if the pair of data nodes are in different clusters. Beside this one, Mix-LMNN(Mixture of LMNN) [SA13] and Local-LMNN(Large Margin Local Metric Learning) [BYGP14] are also multiple metrics learning version of LMNN with several local metrics learned from the partition of feature space.

SCML(Sparse compositional metric learning) [SBS14] also learns Mahalanobis distance, but not for each cluster. The authors propose rank-one matrices by not only clustering but also Fisher's linear discriminant analysis. Then the global distance is generated as the form of a sparse linear combination of the set of rank-one matrices.

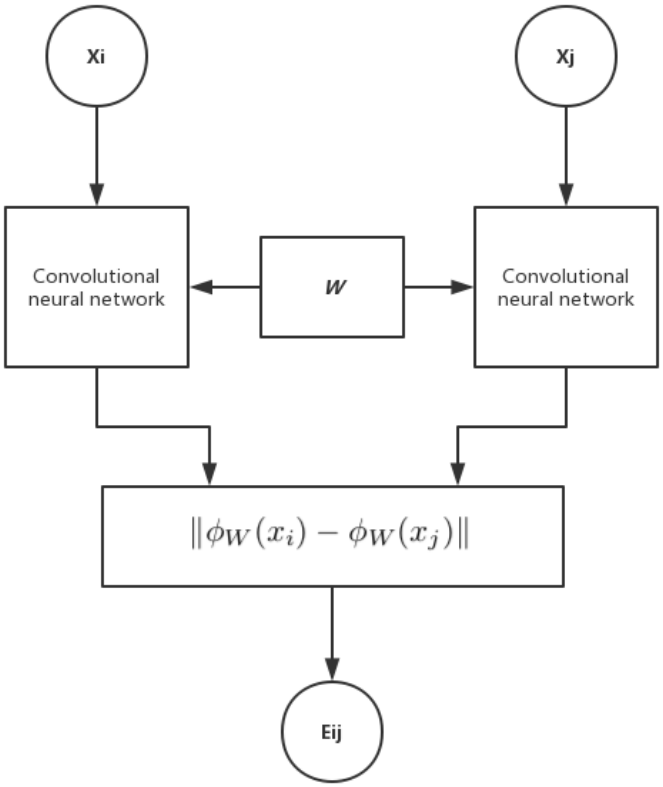


FIGURE 2.3 – Siamese Architecture of Convolutional Neural Networks.

Nonlinear Form Metric

The nonlinear metric learning algorithms we mentioned before are still learning linear metric with different nonlinear transformations. However, there are still several approaches are based on learning the direct optimization of a nonlinear form metric[BHS15].

Neural networks is widely used in nonlinear machine learning algorithms, therefore it is also a convenient choose for nonlinear metric learning. With the neural network model, LSMD(Learning Similarity Metric Discriminatively) [CHL05] is the firstly nonlinear form metric learning literature. The authors learn the nonlinear projection $\phi_W(\mathbf{x})$ parameterized by a vector W, with the relative constraints that in the low-dimensional space $\|\phi_W(\mathbf{x}_i) - \phi_W(\mathbf{x}_j)\| + \gamma \leq \|\phi_W(\mathbf{x}_i) - \phi_W(\mathbf{x}_j)\|, \forall(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) \in \mathbb{R}$ which the distance for positive pairs is smaller than negative pairs, the γ is a margin parameter. The parameter W corresponds to the weights in a convolutional neural networks.

As shown in the Figure 2.3, given two input data points pairs $\mathbf{x}_i$ and $\mathbf{x}_j$, both of them are passed through identical convolutional networks with common parameters W and the output of convolutional neural networks is denoted as $E_{i,j} = \phi_W(\mathbf{x}_i) - \phi_W(\mathbf{x}_j)\|$.

These convolutional neural networks are trained by the loss function :

$$L = \sum_{\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k} \alpha_1 E_{i,j} + \alpha_2 \exp(-\alpha_3 \sqrt{E_{i,k}}) \tag{2.19}$$

where the $\alpha_1, \alpha_2, \alpha_3$ are learning parameters chosen according to the networks.

The convolutional neural networks lead to a high price on computational complexity, but the authors demonstrate the advantage of face verification tasks.

Non-NCA(Nonlinear Neighbourhood Components Analysis) [SH07] is also based on neural networks and is a nonlinear extension of NCA(Neighbourhood Components Analysis) [GHRS04]. Instead of convolutional neural networks, Non-NCA learns a nonlinear, low-dimensional representation of the data using a deep belief network. And for training, the Non-NCA optimizing NCA objective for parameters of the last layer. Non-NCA also has a high computational complexity limit but performs well when enough data is available. The experiment on digit recognition dataset shows the Non-NCA performs significantly better than compared algorithms.

2.1.4 Local Metric Learning

Most of the metric learning algorithms are learning a global metric which is suitable for all data points. However, a single metric (especially a single linear metric) may not well capture the complexity of the task for heterogeneous datasets. As several metric algorithms we mentioned in nonlinear metric learning, for example, Mul-LMNN [WS08a] and SCML [SBS14], they benefit from using multiple local metrics. Besides this, local metric typically leads to better performance and less overfitting than global metric on some tasks. Of course, they require a higher price on computation and time cost.

BMML(Bregman Distances Metric Learning) [WJH$^+$09], [WHJ$^+$10]

BMML [WJH$^+$09], [WHJ$^+$10] is learning the Bregman divergences which is a metric but not satisfy the triangle inequality or symmetry. The Bregman divergences is defined as :

$$d_f(\mathbf{x}_i, \mathbf{x}_j) = f(\mathbf{x}_i) - f(\mathbf{x}_j) - (\mathbf{x}_i - \mathbf{x}_j)^T \bigtriangledown f(\mathbf{x}_j) \tag{2.20}$$

where the f could be $f(\mathbf{x}) = 1/2\mathbf{x}^T M\mathbf{x}$ for Mahalanobis distance, $f(\mathbf{x}) = \sum_{i=1}^{m} \mathbf{x}_i log\mathbf{x}_i$ with considering $\mathbf{x}$ as a discrete probability distribution for KL divergence, where m is the number of dimension of $\mathbf{x}$.

The authors consider a symmetrized version of the Bregman distance function and re-write it as follows :

$$d_f(\mathbf{x}_i, \mathbf{x}_j) = (\bigtriangledown f(\mathbf{x}_i) - \bigtriangledown f(\mathbf{x}_j))^T(\mathbf{x}_i - \mathbf{x}_j) \tag{2.21}$$

$$= (\mathbf{x}_i - \mathbf{x}_j)^T \bigtriangledown^2 f(\mathbf{x}^a)(\mathbf{x}_i - \mathbf{x}_j) \tag{2.22}$$

where $\mathbf{x}^a$ is a point on the line segment between $\mathbf{x}_i$ and $\mathbf{x}_j$. This Bregman distance function can be viewed as a general Mahalanobis distance that introduces a local distance metric $M = \bigtriangledown f(\mathbf{x}^a)$.

Following the maximum margin framework for classification,by setting the $f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i h(x_i^T x_i)$is a strictly convex function), they cast the problem of learning a Bregman distance function from pairwise constraints set $\mathbb{R}$ into the following optimization problem :

$$\min 1/2\alpha^T G\alpha + \lambda \sum_{i,j\in\mathbb{R}} l(y_{i,j}(d_f(\mathbf{x}_i, \mathbf{x}_j) - b)) \tag{2.23}$$

where G is the Gram matrix, $l(t) = max(0, 1 - t)$ is the hinge loss, $y_{i,j} = 1$ if (i, j) are similar pairs and $y_{i,j} = -1$ if (i, j) are dissimilar pairs. The $lambda$ is the trade-off parameter.

As can be seen from the author's experiments, BMML performs similarly to ITML on low-dimensional databases, mostly from their similar core ideas. In the higher dimensional database, BMML performs well, which is better than other algorithms in the same period and can also be used for datasets that cannot be processed by ITML.

LFDA(Local Fisher Discriminant Analysis) [Sug06]

LFDA [Sug06] is not particularly proposed as a metric learning algorithm but a linear supervised dimensionality reduction method. As a localized variant of Fisher discriminant analysis, LFDA is particularly useful when dealing with multimodality, where one or more classes consist of separate clusters in input space. LFDA takes the local structure of the data into account so that the multimodal data can be embedded appropriately.

The LFDA reduce the dimensions with the following transformation matrix T_{lfda} :

$$T_{lfda} = argmax_{T \in \mathbf{R}^{d \times m}}(T^T B' T) \tag{2.24}$$

subject to $T^T W' T$

Where the B' is the local between-class scatter matrix and W' is the local within-class scatter matrix. The core optimization problem of LFDA is solved as a generalized eigenvalue problem. Because the original article used LFDA as a method of dimensionality reduction, it was not compared with other metric distance learning algorithms. However, in the subsequent experiments in our work, it can be seen that although the accuracy and stability of LFDA are not as good as classical algorithms such as LMNN and ITML, the computational efficiency is much higher than other metric distance learning algorithms of the same period.

2.2 Learning Processing

In the previous section, we introduce different metric construction and the related algorithms, also mention several approaches about how the metrics are learned. In this section, we will summarize the method about the learning processing. Besides the different way to define the metric, there are much different in constraints selection, loss function optimizing and other details. We will introduce the differences between the three main branches : constraint selection, regularization selection, and optimization methods.

2.2.1 Constraints Selection

The constraints form is selected by the task requirement and the training information. The metric learning could be supervised or unsupervised algorithms, which is dependent on the available information from datasets.

— Supervised Constraints : The metric learning algorithm has access to the dataset of labelled samples, which in the training set, it has label y for every entity of $\mathbf{x}$. In this case, we selected the constraints based on the target label. For the similar set $\mathbb{S}$, $\forall(i,j) \in \mathbb{S}$, $y_i = y_j$ or y_i and y_j are neighborhood in meaningful notion sets, while the dissimilar set $\mathbb{D}$ is opposite.

— Semi-supervised Constraints : However, in most of the real dataset, there are few labels easy to get. The access to all labels of the individual training set is not com-

plete, and the side-information (for example, the order or rank of instances, citations of the link of entities) is cheaper to get. In this case, a meaningful setting is required to select the constraints, for example, the similar set $\mathbb{S}$ could be $\forall(i,j) \in \mathbb{S}$, $\exists r(i,j) > 0$ where $r(i,j)$ is a link between two entities and the dissimilar set $\mathbb{D}$ is $\forall(i,j) \in \mathbb{D}$,$\forall r(i,j) = 0$ which means there is no link between the pairs.

— Unsupervised Constraints : Several metric learning algorithms have no access to supervised information. In this case, they focus on dimension reduction or clustering, which will select the constraints subject to the task requirement.

There are several ways to use the $\mathbb{S}$ and $\mathbb{D}$ as constraints as in the following subsections :

Threshold Constraints

The most popular form of metric learning constraints is threshold constraints : For all the sample pairs in selected constraints set $\mathbb{C}$, they will be separated in $\ell(X^T M X) = d_M(\mathbf{x}_i, \mathbf{x}_j) - u, \forall(i,j) \in \mathbb{S}$ for similarity set $\mathbb{S}$ and $\ell(X^T M X) = v - d_M(\mathbf{x}_i, \mathbf{x}_k), \forall(i,k) \in \mathbb{D}$ for dissimilarity set $\mathbb{D}$, where the parameters u and v are threshold parameters. The u and v are user given or depended on the learning tasks.

Threshold constraints aim to limit the learned distance between similar pair of data point near enough and to be sure the distance between dissimilar ones far enough.

In MMC [XNJR02] [XNJR03] we mentioned before, the algorithm is the first metric learning using the threshold constraints. Notice that in the first version [XNJR02], the optimization problem is as following :

$$\arg\min \sum_{(i,j)\in\mathbb{S}} d_M^2(\mathbf{x}_i, \mathbf{x}_j) \tag{2.25}$$

$$s.t. \sum_{(i,k)\in\mathbb{D}} d_M^2(\mathbf{x}_i, \mathbf{x}_k) \geq 1 \tag{2.26}$$

Here the authors aim to minimize the margin between similar examples to 0 while the margin between dissimilar examples is set to 1.

In IKML(Idealized Kernels Metric Learning) [KT03], the authors also use the threshold constraints but in a special way : the threshold parameters u and v in threshold constraints $d_M(\mathbf{x}_i, \mathbf{x}_j) \leq u, (i,j) \in \mathbb{S}$ and $d_M(\mathbf{x}_i, \mathbf{x}_k) \geq v, (i,k) \in \mathbb{D}$ are not user-chosen parameters, while they use the original distance as the threshold. For the similarity constraints, they want the learned metric will be smaller than the original one, $d_M(\mathbf{x}_i, \mathbf{x}_j) \leq d_I(\mathbf{x}_i, \mathbf{x}_j)$ where the matrix I is an identity matrix. And for the dissimilarity constraints, they add

a margin as $d_M(\mathbf{x}_i, \mathbf{x}_k) \geq d_I(\mathbf{x}_i, \mathbf{x}_k) + \gamma$ to protect the learned metric is bigger than the original one. The final loss function is denoted as :

$$L(M) = \sum \ell(X^T M X) + \lambda |M|^2 \tag{2.27}$$

where $\ell(X^T M X) = d_M(\mathbf{x}_i, \mathbf{x}_j) - d_I(\mathbf{x}_i, \mathbf{x}_j), \forall (i,j) \in \mathbb{S}$
$\ell(X^T M X) = d_I(\mathbf{x}_i, \mathbf{x}_k) + \gamma - d_M(\mathbf{x}_i, \mathbf{x}_k), \forall (i,k) \in \mathbb{D}$

The ITML [DKJ+07] we mentioned before, and LDRML(Log-determinant regularized Distance Metric Learning) [ZMW+09] are not only the metric algorithms with threshold constraints but also with the information-theoretic forms. The ITML is based on the logdet divergence and learns the metric close to a known prior metric which the trace of the learned metric subject to the threshold constraints. The LDRML extent the ITML with using a set of prior matrices with the same constraints.

Relative Constraints

Another popular form of metric learning constraints is relative constraints : For all the sample pairs in selected constraints set $\mathbb{C}$, they will be selected as triples (i, j, k) which the (i, j) is from similarity set $\mathbb{S}$ and (i, k) is from dissimilarity set $\mathbb{D}$. The relative distance constraints is in the following form : $\ell(X^T M X) = d_M(\mathbf{x}_i, \mathbf{x}_k) - d_M(\mathbf{x}_i, \mathbf{x}_j) - \gamma \forall (i, j, k) \in \mathbb{C}$, where the γ is a margin parameter.

Relative constraints aim to be sure the learned distance between similar pair of data points nearer than the distance between the dissimilar ones. In this case, the relative constraints could be denoted as $d_M(\mathbf{x}_i, \mathbf{x}_k) \geq d_M(\mathbf{x}_i, \mathbf{x}_j)$, which do not require one to specify any parameters(unlike the threshold constraints). However, typically relative constraints add the desired margin to improve the performance. In many cases the γ is chosen equal to 1 for normalized dataset but also could be other value or even related functions as the requirement of task and algorithms.

In most cases, it is typically easier to provide relative distance constraints than threshold constraints. Firstly, for many subjects, it is possible for comparing which one is more similar to target object while it may be challenging to determine an arbitrary pair of the object should be considered similar, for example, the image, music or other entities with sophisticated features or structures. Secondly, the margin γ is more comfortable to choose than the threshold parameters of u and v, which are hard to define the boundaries of similar and dissimilar. Thirdly, the threshold parameters u and v could lead to a problem when set-

ting the similar threshold u is greater than the dissimilar threshold v (Not possible in most algorithms but legal in rare cases).

On the other hand, the threshold constraints are more straightforward and standard to create, when the fully supervised information (like target labels) are available, with setting all pairs of objects of the same class are similar constraints and pairs of objects of different classes are dissimilarity constraints.

LMNN(Large-Margin Nearest Neighbors) [WBS06] is the famous metric learning algorithm which chooses the relative distance constraints and with a margin, $\gamma = 1$ for normalized data. That is the reason this algorithm is called as large-margin. There are many extended version of LMNN but most of keeping the relative distance constraints selection.

SCML(Sparse compositional metric learning) [SBS14] has a similar strategy as LMNN, which is not only also learns Mahalanobis distance, but also using relative distance constraints. Thought the algorithm is based on the idea of learning a sparse linear combination of the set of rank-one matrices. The constraints selection is still the same. And the authors also chose the margin equal to 1.

Other Constraints Approaches

Besides the two most popular constraints, there are several other constraints forms.

Some of the constraints are different because of the ways for using the available information. For example, in CibML(Choquet-integral-based Metrics Learning)[BJL11] they use the Shapley indices and interaction indices as additional threshold constraints. In addition, the algorithms for non-flat data sets in the next chapter also construct constraints based on structure or side information.

Some of the constraints are different because the ways for manage the relation of similar and dissimilar sample sets. For example the QWML(Quadruplet-wise Metric Learning) [LTC13] which propose quadruplet based constraints of the form $\ell(X^T MX) = d_M(x_p, x_q) - d_M(\mathbf{x}_i, \mathbf{x}_j) - \gamma \forall (i, j, p, q) \in \mathbb{C}$. The constraints set is denoted as $\mathbb{C} = (i, j, p, q)|(i, j) \in \mathbb{S}, (p, q) \in \mathbb{D}$, where the 4 samples ordered as $p \prec i \sim j \prec q$, that the pair (i,j) is more similar than the pair (p,q). The γ is also an added margin. This constraints is in case that pair or triplet based constraints cannot completely capture the relations between the examples since they are not fully determined.

Some of the constraints are different because of the ways to generate the constraint pairs from the sample set. In CWSML(Constraint Weighted Selection Metric learning)[LC18], instead of selecting constraints samples random form the similar and dissimilar set, an on-

line constraints selection method is proposed for metric learning algorithms. The algorithm assigns a weight to each sample, each time selecting a limit based on the weight $w_{\mathbf{x}_i}$. For each selected constraints, the loss $L(M)$ produced by the metric defined under the current parameter matrix M is calculated according to the metric learning algorithm. Also, according to the loss L, update the weight w of each sample. The more the loss, the higher the contribution that the selected sample can provide to the metric learning, so the updated weight will help the probability that the sample-related constraint is selected increases at the next iteration.

The different constraints form depend on the availability of information and the deterministic of the relationship between samples. It is necessary to choose different constraint solutions according to the different learning tasks and data sets.

2.2.2 Regularization

Typically the metric learning loss function consists of two parts. One part is the encode loss from the selected constraints, and the other part is regularization. As most of the machine learning algorithms, regularization limits the complexity of the model to avoid over-fitting and obtain better generalization.

Most of the linear models metric learning algorithms are similar on the constraints selection or could be adapted to multiple constraints form, however, they have particular regularization lead to different performance for metric learning and suited to different datasets.

Regularization	Definition	Properties
L_1 norm	$r(M) = \sum \|M_{i,j}\|$	Convex, sparsity, non-smooth
Frobenius Norm or L_2 norm	$r(M) = \sum M_{i,j}^2$	Strongly convex, smooth
Linear (trace) norm	$r(M) = tr(MC)$	Convex, low rank, (non-smooth)
Information-Theoretic	$r(M) = tr(M) - logdet(M)$	Convex, low rank, LogDet divergence

TABLE 2.1 – The difference between different Regularization.(Where C is a chosen matrix, for example an identity matrix for nuclear-norm regularizer)

In the general loss function $L(M) = \sum_{(i,j,k)\in\mathbb{C}} \ell_M(i,j,k) + \lambda r(M)$, the regularization $r(M)$ is also selected by the task requirement, while many methods improve the metric learning performance by selecting better regularization. As shown in Table 2.1, the different regularizers lead to different properties, and we will introduce the related algorithms in following subsections.

Norm Regularization

Frobenius Norm regularization is one of the most common forms of regularization. L_2 Frobenius norm loss is also known as Tikhonov regularization or ridge regression. This regularization is widely used and also apply for metric learning algorithms. It is easy to optimize, and strong convexity, which provides control over the values of the data feature vector and parameter learned matrix.

In paper IKML [KT03] and SJML [SJ04], they both use the Tikhonov regularization in loss function. The difference is the chosen constraints form. In POLA(Pseudo-Metric Online Learning Algorithm) [SSSN04] the regularizer is the squared Frobenius norm, which follows from the fact that the algorithm employs Euclidean projections to define the updates.

Most of the metric learning algorithms with the norm regularization are easier on optimization. The drawback of this regularization is that norm regularization is harder to avoid the over-fitting and lead to more computation complexity on a higher dimension comparing the other regularization.

Linear Regularization

Linear regularization means the regularizer is a linear function with respect to the Mahalanobis matrix M, $r(M) = tr(MC)$, where the matrix C is a chosen matrix by the requirement of the learning task.

When the C is the identity matrix, the resulting regularizer is the trace-norm regularizer, which is known to prefer low-rank solutions. The trace-norm regularization or also called nuclear-norm is used in IRML(Inductive Regularized Metric Learning) [JKD10] with general linear threshold constraints. The authors of this algorithm also describe the trace-norm formulation in conjunction with kernelization in order to describe a supervised nonlinear dimensionality reduction scheme.

When the C is a generated matrix related the selected constraints, the resulting regularizer is a single limit to the learned metric, which are more sensitive to over-fitting. For example, in MMC [XNJR02] [XNJR03], the regularizer could be written as $r(M) = tr(MC)$ where the $C = \sum_{(i,j)\in\mathbb{S}}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T$. LMNN [WBS06] also use the same regularization, but need to point out they use the different constraints and the aim for $C = \sum_{(i,j)\in\mathbb{S}}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T$ is related the nearest neighbors selection for the target data point.

Another metric learning algorithms related to nearest neighbours selection is NCA(Neighbour Components Analysis) [GHRS04], which is special without regularizer or could be treated like the $C = 0$. NCA has many extensions, some of which enforce different regularization.

Information-Theoretic Regularization

The authors proposed the information-theoretic regularization ITML(Information-Theoretic Metric Learning) [DKJ$^+$07], in the form as $r(M) = tr(M) - logdet(M)$. This regularizer can be treated as LogDet divergence, which be rewritten as :

$$Div(M, M_0) = tr(MM_0^{-1}) - logdet(MM_0^{-1}) - m \tag{2.28}$$

with the m dimension dataset.

The LogDet divergence brings various useful properties for metric learning contexts, such as scale invariance, translation invariance, range space preservation and connections to multivariate Gaussian. The authors of ITML discuss those above properties, prove that the LogDet divergence is over PSD (positive semi-definite) space and make the LogDet regularization as a natural choice for metric learning model.

The LogDet regularization is also applicable for several algorithms base on a similar idea, like LDRML (Log-determinant regularized Distance Metric Learning) [ZMW$^+$09] and FSOL(Fast Similarity Online Metric Learning)[JKDG09].

2.2.3 Optimization

For all the machine learning algorithms, the optimization is not only one of the most important part but also one of the unique part of the methods. The authors always design their own optimization techniques specific to the individual algorithms, improve or develop the related baseline optimization method and extent or combine with optimization techniques from other areas. Therefore we would not fully discuss all metric learning optimization. Instead, we will summarize some of the main techniques which are most popular and successfully utilized.

Gradient Descent

Standard gradient descent is the most popular and simplest optimization technique. However, since it is designed for unconstrained optimization problems, it cannot be applied

naively.

So a possible approach for metric learning algorithms with PSD (positive semi-definite) matrices M is to factorize the matrix $M = L^T L$, and then apply gradient descent on the linear projection L directly. The PSD properties imply that we can always factorize in this manner. Gradient descent then proceeds by iteratively computing the gradient of L and moving in the direction of the gradient : $L_{t+1} = L_t - \eta_t \bigtriangledown \mathbb{L}(L_t)$, where η_t is the step size for t-th iteration. NCA(Neighbourhood Components Analysis) [GHRS04], CCML(Metric learning by collapsing classes) [GR06], and similar algorithms utilize this approach. The advantage is this method is simple and fast while the problem is it leads to non-convex on L and could deal with the non-PSD matrix model or nonlinear model.

Another related gradient descent optimization is generalized gradient descent, or mostly the special case of it called as projected gradient method. The idea of this approach is for each step of gradient descent $L_{t+1/2} = L_t - \eta_t \bigtriangledown \mathbb{L}(L_t)$, add a step that projection back to the cone of PSD matrices $L'_{t+1} = \arg\min_{L'^*} \|L'^* - L'_{t+1}\|^2$. This method add the positive constraints for each learn matrix and the advantage is stabler than the previous approach. The MMC [XNJR02] [XNJR03], LMNN [WBS06] and similar algorithms utilize this approach.

Bregman Projections

For the above gradient descent, there is a problem that when are a large number of constraints this method lead to a high price to compute the entire gradient of the loss function. And Bregman projections is one of the possible situations in this case, which is based on making simpler updates based on a single constraint at a time.

The method of Bregman projections is a simple first-order technique propose in [Bre67]. It originally solve optimization problems, which is to minimize a convex function with linear inequality constraints as following :

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) \\ &\text{s.t. } C\mathbf{x} \leq \mathbf{b} \end{aligned} \tag{2.29}$$

where C is a parameter matrix and $\mathbf{b}$, is a constant vector.

However, unlike the classical linear programming or quadratic programming computing with all constraints, Bregman projections choose one constraint at each iteration. So unlike

projected gradient descent, the Bregman projection is not an orthogonal projection after each update, but perform a projection satisfied the chosen constraint that optimizes the particular function.

The Bregman projections optimize technique is utilized in ITML(Information-Theoretic Metric Learning) [DKJ$^+$07] and similar algorithms based on minimizing the regularizer term of the loss function.

Online Learning

Another way to learn metrics through a single constraint at a time is online learning. Online learning is a natural choice when the examples arrive in a stream, or all the examples are not available at the same time.

Stochastic gradient descent is one of online learning approach and able to update and learn the metric when constraints are given one after the other. With the sequence of constraints $(\mathbb{C}_1, \mathbb{C}_2, \cdots, \mathbb{C}_n)$,the aim of stochastic gradient descent is learning the distance d_{M_t} with the parameter matrix M_t iteratively updated,

$$d_{M_t} = f(d_{M_{t-1}}, \mathbb{C}_t) \tag{2.30}$$

where $f()$ is a function able to update M_t with the current metric and the new single constraint.

There are many metric learning algorithms use the stochastic gradient descent and online learning approach, because the computation complexity is not related to the size of the constraints.

POLA(Pseudo-Metric Online Learning Algorithm) [SSSN04] is the first metric learning with stochastic gradient descent and online learning approach. It based on Mahalanobis distance model and use the threshold constraints with one parameter b to separate the similar and dissimilar pair of samples, $d_M(\mathbf{x}_i, \mathbf{x}_j) \leq b \leq d_M(\mathbf{x}_i, \mathbf{x}_k) \forall (i,j) \in \mathbb{S}, \forall (i,k) \in \mathbb{D}$ where $b \geq 1$. POLA optimize the problem with two orthogonal projections step for each iteration. At iteration t with the constraint $(\mathbf{x}_i, \mathbf{x}_j, y_{ij})$, where $y_{ij} = 1$ if $(\mathbf{x}_i, \mathbf{x}_j) \in \mathbb{S}$ and $y_{ij} = -1$ if $(\mathbf{x}_i, \mathbf{x}_j) \in \mathbb{D}$,

— step 1 : Projecting of the current solution with the new constraint,

$$\begin{aligned} M_{t+1/2}, b_{t+1/2} &= \arg\min \|M_t - M\|^2 + (b_t - b)^2 \\ &\text{s.t. } [y_{ij}(d^2_M(\mathbf{x}_i, \mathbf{x}_j) - b) + 1]_+ = 0 \end{aligned} \tag{2.31}$$

This step search the solution for the (M,b) achieving a loss of 0 on the new constraint while staying as close as possible to the current solution (M_t, b_t).

— step 2 : Projecting the new solution onto the set of admissible solutions which the M is PSD and $b \geq 1$,

$$M_{t+1}, b_{t+1} = \arg\min \|M_{t+1/2} - M\|^2 + (b_{t+1/2} - b)^2 \quad \text{s.t. } M \in \mathbb{S}_+, b \geq 1 \tag{2.32}$$

This step search the acceptable solution for the (M,b).

POLA solves the problem of being able to handle a large number of sample databases in machine learning. It also shows the benefits of online multiple selection constraints set $\mathbb{C}$. However, it's worth noting that the way to select constraints set still need improving. In the subsequent online learning metric distance algorithm, several algorithms, such as CWSML(Constraint Weighted Selection Metric learning)[LC18], have proposed new ideas for this defect.

FSOL(Fast Similarity Online Metric Learning) [JKDG09] is a similar online metric learning approach as POLA but it is an improved version based on LogDet divergence regularization which is same as ITML. The regret bounds of FSOL is tighter than POLA. Therefore the FSOL is more efficient and faster than POLA. MDML(Mirror Descent Metric Learning) [KS12] is based on composite mirror descent [DSSST10], which allows online optimization of many regularized problems. The authors proposed a general framework for online learning with the Mahalanobis distance learning model.

2.3 Learning Tasks

Thought most of the metric learning algorithms only considering learning a metric for classical machine learning tasks, like clustering or classification task, there are several approaches specifically designed for other kinds of tasks such as transfer learning or domain adaptation. We already introduced many metric learning algorithms for machine learning tasks, so more attention will be paid on particular tasks and applications.

Notice that in this chapter, we mainly focus on the metric learning for the flat dataset, and there are several special learning tasks related to the complex structure of the non-flat dataset. For these learning task related to the non-flat dataset, we will discuss them in Chapter 4.

2.3.1 Classification and Clustering

Almost all of the metric learning algorithms we mentioned earlier can be used for classification. Notice that while most metric learning algorithms measure their performance with k-nearest neighbour, there is a group of metric learning methods specifically designed for a nearest neighbour classifier, like NCA(Neighbourhood Components Analysis) [GHRS04], SiNo(Similarity learning for nearest neighbour classification) [QGCL08], LMNN(Large-Margin Nearest Neighbors) [WBS06] and many extension based on them. Another group of metric learning methods pays more attention to learning low-rank matrices for other subsequent classification like SVM, for example, SDML(Sparse Metric Learning via Smooth Optimization) [YHC09] and SCML(Sparse compositional metric learning) [SBS14].

Those metric learning algorithms could be adopted for clustering, and many constraints selection method of metric learning also could solve the clustering problem. For example, the threshold constraints selection in MMC [XNJR02] [XNJR03] and LLMA(Locally Linear metric Adaptation) [CY04].

2.3.2 Transfer Learning, Multi-task learning and Domain Adaptation

Transfer learning is another important branch of machine learning. Unlike traditional machine learning assuming the training and test data are drawn from the same feature space and the same distribution, transfer learning transfer knowledge between the tasks and domains. It researches the difference between the tasks and domains, transforms the feature represents or parameters of the model to reduce the difference and applies the knowledge from source task to the related target problem.

There are numbers of approaches or algorithms combine the transfer learning and metric learning, particularly in the multi-task learning and domain adaptation.

For multi-task learning, there are two tasks, the source task T_S we already solved and the target task T_T need to solve. We want to adapt the knowledge learned from the T_S to the T_T.

mt-LMNN(multi-task Large margin metric learning) [PW10] is a multi-task learning verison of LMNN, follows the idea of RMTL(Regularized Multi–Task Learning) [EP04]. RMTL borrowed the idea of hierarchical Bayesian framework to SVMs for multi-task learning. They assume the $f(\mathbf{x}) = W\mathbf{x}$ to be a hyper-plane for task T and set $W_S = W_0 + \mathbf{w}_S$ for T_S and $W_T = W_0 + \mathbf{w}_T$ for T_T, then learn the parameters $W_0, \mathbf{w}_S$ and $\mathbf{w}_T$ simultaneously by optimize SVM model, for details, we refer the reader to the original paper. The

author in mt-LMNN instead the SVMs with the LMNN metric learning algorithms. They define the metric for task t as $d_t(x, x') = (x - x')^T(M + M_t)(x - x')$ and like the RMTL, the model will be denoted as follow :

$$\begin{aligned}
&\min_{M_0, M_t, c_{t_i}} Loss(M_0, M_t, d_t(\mathbf{x}_i, \mathbf{x}_j)) \\
&\quad = \sum_{t \in S,T} \sum_{i=1}^{n_i} d_t(\mathbf{x}_i, \mathbf{x}_j) + \frac{\lambda_1}{2} \sum_{t \in S,T} \|M_t\|^2 + \lambda_2 \|M_0 - I\|^2 \\
&\quad \text{s.t.} d_t(\mathbf{x}_i, \mathbf{x}_k) - d_t(\mathbf{x}_i, \mathbf{x}_j) \geq 1 - m, \forall (\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) \in \mathbf{R}_t^{lmnn}
\end{aligned}$$

The mt-LMNN performs better than the LMNN and RMTL.

Similar to the multi-task learning, in domain adaptation, we have the source one and the target one but not for the task but the data domains. The source domain D_S and the target domain D_T. We want to adapt the knowledge learned from the D_S to the D_T.

An impotant method for domain adaptation is MMD (Maximum Mean Discrepancy) [BGR$^+$06], which is used to compare distributions based on RKHS(Reproducing Kernel Hilbert Space) [Aro50]. The distance of two distribution P_S andP_T could be defined as :

$$\begin{aligned}
Dist(P_S, P_T) &= sup(\frac{1}{n_1} \sum_{i=1}^{n_1} f(\mathbf{x}_{S_i}) - \frac{1}{n_2} \sum_{i=1}^{n_2} f(\mathbf{x}_{T_i})) \\
&= \|\frac{1}{n_1} \sum_{i=1}^{n_1} \phi(\mathbf{x}_{S_i}) - \frac{1}{n_2} \sum_{i=1}^{n_2} \phi(\mathbf{x}_{T_i})\|
\end{aligned}$$

Where $\phi(\mathbf{x})$ is function in a RKHS. This method proposes the distance between the distribution of two domain is equal to the distance between the means of them mapped into an RKHS.

DAML(Domain Adaptation Metric Learning) [GTX11] is a metric learning algorithm based on the MMD method. In this paper, the authors use the MMD as follow :

$$Dist_\phi(P_S, P_T) = \|\frac{1}{m} \sum_{i=1}^{m} \phi(\mathbf{x}_i) - \frac{1}{n} \sum_{i=1+m}^{m+n} \phi(\mathbf{x}_i)\| \tag{2.33}$$

This will be a regularization in the metric learning loss function. It will take the minimum to match the mean of D_S and D_T in RKHS. Therefore the authors learn the function $\phi(\mathbf{x})$ to achieve the desired RKHS for learning the metric knowledge between D_S and D_T.

With the MMD regularization, DAML uses the threshold constraints to encode the cost

and compose the loss function. They also propose a KPCA(kernel principal component analysis) to improve the calculation.

MLHD(Metric Learning across Heterogeneous Domains)[QC12] is another similar algorithm, but unlike the DAML focus on the different domain with same dimensions(after KPCA), MLHD aims at achieving a common RKHS for a different domain with different dimensions.

The mode of MLHD is combined with three-part : 1) The priors information part with the MMD ; 2)The posteriors information part with threshold constraints ; 3)The LogDet regularization for a PSD (positive semi-definite) matrix which reparameterize the priors and posteriors parts.

GDA(Graph-based Domain Adaptation)[DTC12] is totally different from the algorithm mentioned before. It did not extend the metric learning with the transfer learning approach, but use supervised metric learning to change or twist the feature space for semi-supervised domain adaptation learning.

The main idea of this algorithm is an iteration of two-part : 1) use a graph construction to link the data between all the data from the D_S and D_T to transfer the knowledge and use the semi-supervised algorithm to inference the unlabeled data ; 2) use the labelled data, and the new inferred labelled data for supervised metric learning, after learning the new feature representations will reduce the difference between D_S and D_T.

In this paper, the authors use the ITML [DKJ$^+$07] as a supervised metric learner and GRF (Gaussian Random Fields [RH05]) as the semi-supervised learner, but these could be replaced to any other learning algorithm. It offers a new way to combine the metric learning and domain adaptation, and maybe improved by other algorithms.

2.3.3 Other Tasks

A group of metric learning algorithms could be considering as dimension reduction algorithms, which are based on the Mahalanobis distance model and try to learn a lower rank of the metric. The Mahalanobis distance learning $d_M(\mathbf{x}_i, \mathbf{x}_j) = \Big((\mathbf{x}_i - \mathbf{x}_j)^T M(\mathbf{x}_i - \mathbf{x}_j)\Big)^{1/2}$ can be viewed as using the Euclidean distance on the linearly projected data with L, which $M = L \times L^\top$. So learning a lower rank M provides a possible way to perform a form of supervised dimension reduction. NCA(Neighbourhood Components Analysis) [GHRS04] and its extensions could be utilized in this task. Also, the metric learning algorithm LFDA (Local Fisher Discriminant Analysis) [Sug06] is originally devised as a dimension reduction algorithm.

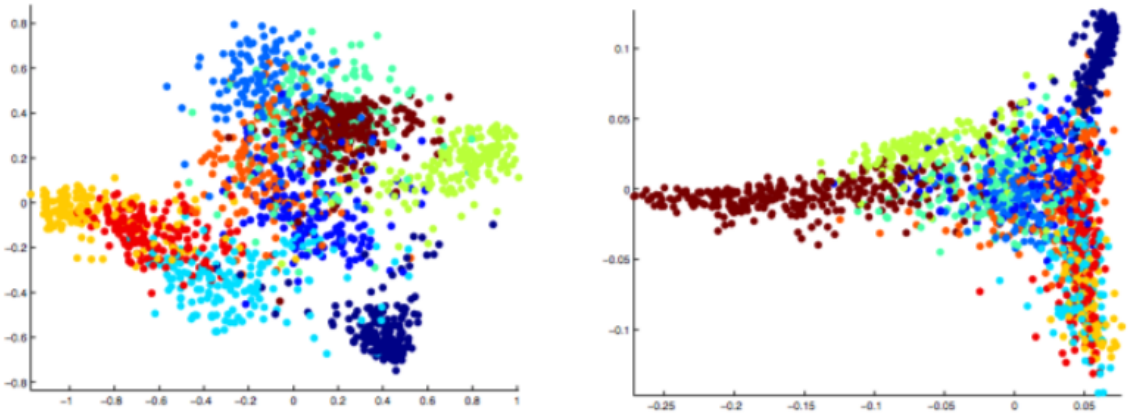


FIGURE 2.4 – Data visualization before and after metric learning. [JKD10]

The problem of ranking is another extension task beyond the simple nearest-neighbour applications of most metric learning approaches, which is presented in RML(Metric learning to rank)[ML10] with the Mahalanobis distance learning model.

Some metric learning algorithms also be used to improve the performance of other supervised learning methods. The authors in KRML(Metric learning for Kernel Regression)[WT07] replaces the squared Euclidean distance within the Gaussian kernel with a Mahalanobis distance utilizing for supporting the kernel regression problem.

Finally some metric learning algorithms preset performance of data visualization, which shows the data point before and after training, for example, the Figure 2.4 from IRML [JKD10]

2.4 Deep Metric Learning

As we mentioned, deep learning is also a kind of representational learning, which is similar to the core thinking of metric learning. Today, with the development of deep learning becoming more and more popular, deep learning and metric learning naturally have a combination and intersection. Deep metric learning algorithms have emerged and applied to big data mining, image processing, tracking recognition and many areas. Notice that because the tasks like the image classification, ranking or network relational learning are the interaction of the deep learning and metric learning, most of the deep metric learning is based on the CNN(convolutional neural networks). In this section, we will introduce several deep metric learning based on CNN.

2.4.1 Deep Metric Learning based on Siamese Network

As we mentioned before, in the nonlinear metric learning section, LSMD [CHL05] is the first metric learning algorithm with the neural network model, and it always extends to CNN. In this article, the authors propose the "Siamese network" system for metric learning. That as shown in Figure 2.3, Siamese network is identical convolutional networks with shared parameters W. The two input of the networks are the data points pairs $\mathbf{x}_i$ and $\mathbf{x}_j$ from the selected constraints ; the output of convolutional neural networks is the similarity between data points. The idea of this Siamese network is mapping the data (in this case are the images) though CNN to a latent feature space where the similar pairs are near, and the dissimilar pairs are far.

The Siamese network is the basis of many other deep learning metric algorithms. For example, in the DIML(Deep Image-classification Metric Learning) [CYY$^+$18], the authors add an extra metric learning layer at the end of the CNN and could be considered the same as the Siamese network.

Inspired by Siamese network, the authors in PRDML(Deep Metric Learning for Person Re-identification) [YLLL14] and TNDML(Triplet Network Deep Metric Learning) [HA15] propose the Triplet network as shown in Figure 2.5. As we introduced before the two most popular constraints forms are threshold constraints and relative constraints, the Siamese network is related to the threshold constraints while the Triplet network corresponds to the relative constraints. The input of the Triplet network are the data points triple x_i x_j and x_k from the selected relative constraints which the (i, j) is from similarity set $\mathbb{S}$ and (i, k) is from dissimilarity set $\mathbb{D}$. The output of convolutional neural networks is the comparator of the similarities between data points ; in words, this encodes the pair of distances between the select constraints. The authors in TNDML [HA15] trained the Triplet network with simple random gradient descent. The authors in TNDML [HA15] train the Triplet network with simple stochastic gradient descent. The authors in [YLLL14] not only test the Triplet network with extent stochastic gradient descent but also test the Siamese network with additional cosine layer before the output layer.

The result of these algorithms shows that for person re-identification and image scene classification task, the deep metric learning algorithms get the generalized favourable performance.

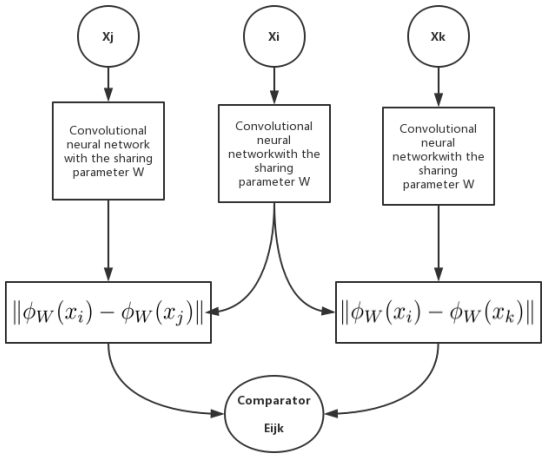


FIGURE 2.5 – Triplet network with Convolutional Neural Networks.

2.4.2 Deep Metric Learning based on Feature Embedding

Another group of deep metric learning methods are based on learning lifted structured feature embedding.

Notice that the feature embedding methods are highly related to metric learning algorithms and several of the proposed papers consider they are same. The authors in FEDML(Deep metric learning via lifted structured feature embedding) [OSXJS16] get inspired from the constraints feature embedding [HCL06](which could be treated as threshold constraints metric learning) and triple features embedding [SKP15] (which could be treated as relative constraints metric learning) and proposed the lifted structured feature embedding which takes into account all pairwise edges within the mini-batches as shown in Figure 2.6

The authors in FEDML [OSXJS16] utilize the lifted structured feature embedding for each mini-batch of the training set for CNN, instead of randomly sampling pairs or triplets to construct the training batches. This app approach combines the metric learning with CNN and takes full advantage of the training batches by lifting the vector of pairwise distances (between samples $O(m)$) within the batch to the matrix of pairwise distances (with the metric parameter $O(m^2)$).

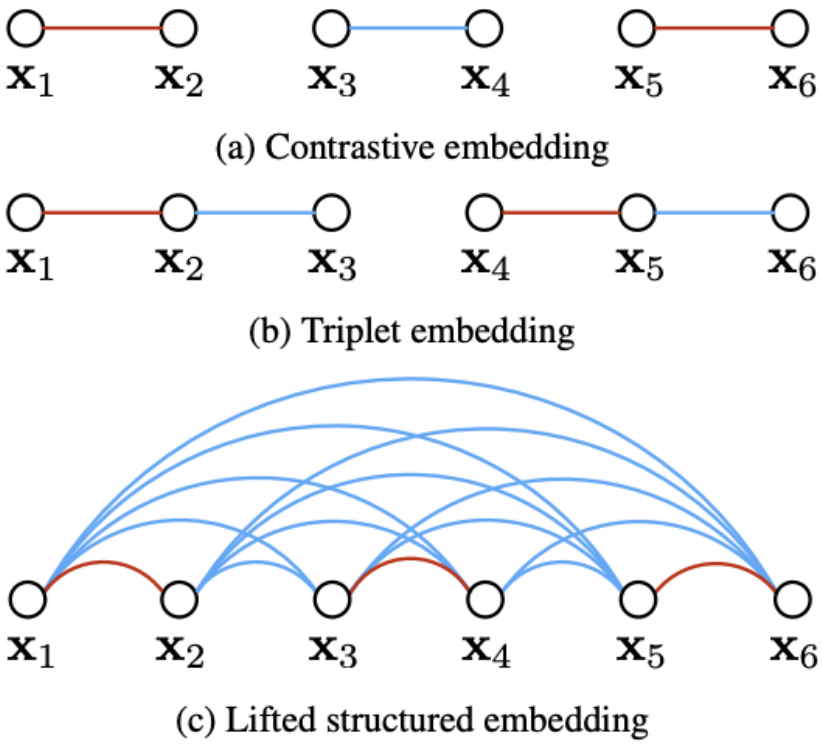


FIGURE 2.6 – For a mini batch of training set with 3 samples : (a) constraints feature embedding ; (b) triple fearture embedding ; (c) lifted structured feature embedding.[OSXJS16]

In BIEDML(Boosting Independent Embeddings Robustly Deep Metric learning) [OWPB18], the authors of this paper follow the model in FEDML [OSXJS16], and extend it with boosting based metric learning methods [BWL+11] [NLJ16] [KTS+12] for the feature embedding part. According to the boosting for CNN, this approach gains more robust and better performance.

2.5 Conclusion

Name	Year	Metric	Contraints	Regularizer	Additional Information
MMC [XNJR02, XNJR03]	2002	Mahalanobis distance	Threshold	None	First Metric Learning
SJML [SJ04]	2004	Mahalanobis distance	Relative	Diagonal	
NCA [GHRS04]	2004	Mahalanobis distance	Relative	None	For k-nn
LMNN [WBS06]	2005	Mahalanobis distance	Relative	Linear Regularizer	Most Popular; For k-nn
ITML [DKJ+07]	2007	Mahalanobis distance	Relative	LogDet	Most Popular; Also proposed online version
LSML [LGZ+12]	2012	Mahalanobis distance	Relative	Linear Regularizer-LogDet	Dimensionality reduction
SiNo [QGCL08]	2008	Linear similarity	Relative	Linear Regularizer	For k-nn
SiCos [QG09]	2009	Cosine similarity	Relative	L_2 Norm	Online
OASIS [CSSB09]	2009	Bilinear similarity	Relative	Frobenius Norm	Online
POLA [SSSN04]	2004	Mahalanobis distance	Relative	Frobenius Norm	Online
GB-LMNN [KTS+12, KXW]	2012	Kernel Mahalanobis distance	Relative	Linear Regularizer	Nonlinear version of LMNN
K-PCA [CKTK10]	2010	Kernel Mahalanobis distance	Threshold	None	KPCA trick
Mul-LMNN [WS08a]	2008	Multiple Local Metric	Relative	Linear Regularizer	Multiple Local Metric version of LMNN
Mix-LMNN [SA13]	2013	Multiple Local Metric	Relative	Linear Regularizer	Multiple Local Metric version of LMNN
Local-LMNN [BYGP14]	2014	Multiple Local Metric	Relative	Linear Regularizer	Multiple Local Metric version of LMNN
SCML [SBS14]	2014	Multiple Metric	Relative	L_1 Norm	Also proposed Multi-Task verion; Support for SVM
LSMD [CHL05]	2005	Nonlinear Local Metric	Relative	None	First Nonlinear Metric Learning, CNN
Non-NCA [SH07]	2007	Nonlinear	Relative	None	Nonlinear version of NCA
BMML [WJH+09, WHJ+10]	2009	Bregman distances	Threshold	Frobenius Norm	
LFDA [Sug06]	2006	Local Mahalanobis distance	Threshold	None	Dimensionality reduction
IKML [KT03]	2003	Mahalanobis distance	Threshold	Frobenius Norm	Also proposed kernel verion
CibML [BJL11]	2011	Mahalanobis distance	Threshold	None	Shapley and interaction indices as addtional contraints
QWML [LTC13]	2013	Mahalanobis distance	Quadruplet	L_2 Norm	Propose quadruplet based constraints
IRML [JKD10]	2010	Kernel Mahalanobis distance	Threshold	Nuclear Norm	Nonlinear dimensionality reduction
LDRML [ZMW+09]	2009	Mahalanobis distance	Threshold	LogDet	
FSOL[JKDG09]	2009	Mahalanobis distance	Relative	LogDet+L_1	Online
MDML [KS12]	2012	Mahalanobis distance	Threshold	Compared different version	Composite mirror descent method
SDML [YHC09]	2009	Mahalanobis distance	Relative	Linear Regularizer	Support for SVM
LLMA [CY04]	2004	Mahalanobis distance	Threshold	None	Support for Clustering
mt-LMNN [PW10]	2010	Mahalanobis distance	Relative	Linear Regularizer+L_2 Norm	multi-task version of LMNN;
DAML [GTX11]	2011	Nonlinear	Relative	MMD	Domain Adaptation
MLHD[QC12]	2012	Nonlinear	Threshold	LogDet	Domain Adaptation
GDA[DTC12]	2012	Mahalanobis distance	Relative	LogDet	Domain Adaptation; Semi-supervised
DIML [CYY+18]	2018	Nonlinear CNN Metric	Threshold	L_2 Norm	CNN
PRDML [YLLL14]	2014	Nonlinear CNN Metric	Relative	L_1 Norm+Batch Normalization	CNN
TNDML [HA15]	2015	Nonlinear CNN Metric	Relative	L_2 Norm	CNN
FEDML [OSXJS16]	2016	Nonlinear CNN Metric	Lifted Structured	L_1 Norm+Batch Normalization	Lifted structured feature embedding for CNN
BIEDML [OWPB18]	2018	Nonlinear CNN Metric	Lifted Structured	L_1 Norm+Batch Normalization	CNN extension for boosting based metric learning
CWSML[LC18]	2018	Mahalanobis distance	Dynamic Weighted	Depends on Selected Algorithm	Dynamic weighted online constraint selection

TABLE 2.2 – Survey on Metric Learning Algorithms

In the Table 2.2, we collated all the metric distance learning algorithms introduced in this chapter and the metric models they learned, the regularization factors used, the constraint selection and the uniqueness, then sorted by time. These algorithms are only a small part of the many algorithms that measure distance learning algorithms from the time they are proposed to the present. Even so, we can still find some context in the development of metric distances learning theory. The metric learning algorithm develops metric distances from the linear distance(such as Mahalanobis distance, linear similarity) to nonlinear metric (such as nucleation metrics, neural networks), from global metrics to targeted local metrics. For learning, according to the needs of the task, metrics distance algorithms evolved from simple regularization factors to sophisticated regularization methods, from classical threshold constraints or relative constraints to multiple comparisons or dynamic constraint selection, from quadratic programming optimization to stochastic gradient descent online learning. The metric distance learning algorithm also communicates and integrates with many fields, such as transfer learning and deep learning. With these developments, the various learning tasks that metric distance learning can solve are becoming more and more widespread.

After investigating the current metric learning of the metric learning algorithm, it shows that the metric learning method based on Mahalanobis is the core and foundation of the metric learning algorithm for the flat database. The use of the inverse of the covariance matrix for M, i.e. the actual Mahalanobis distance, implies that the weight of a feature pair is proportional to the co-factor of the features. Although the co-factor of a pair of features depends on all other pairwise covariances, the actual distance definition only considers the pairwise combination of features, whereas a-tuple-wise(where $a \geq 2$) combinations bring a lot more information. We are investigating the possibility of learning weights to coalitions of features whose cardinality can be greater than two. In the following chapter 3, we consider submodular set-functions and definition based on the extension of it. This definition allows associating weights to subsets of features. We will show how to learn the defined metric and shows the effectiveness of this approach on a real-world dataset.

CHAPITRE 3

SUBMODULAR METRIC LEARNING

As introduced in Chapter 2, most of the linear model metric learning algorithms are based on Mahalanobis metric and try to learn a matrix M as distance transformation. Mahalanobis metric with M gives each used feature dimensions and their pairs a weight to map the entities with a new feature space which should be better to classify or cluster for machine learning tasks.

However, this distance definition method implies that the weight of a feature pair is proportional to the co-factor of the features, which leads to being limited to weight the pairwise combination of features. For example, the Mahalanobis distance learning model $d_M(\mathbf{x}_i, \mathbf{x}_j) = \Big((\mathbf{x}_i - \mathbf{x}_j)^T M(\mathbf{x}_i - \mathbf{x}_j)\Big)^{1/2}$, using the d_M is equivalent to using the Euclidean distance on the linearly projected data with L, which $M = L \times L^\top$. The triangular matrix L contains the weights for every single dimension and pairwise of dimensions, while the combinations of a-tuple-wise where $a \geq 2$ still could offer more information.

For mining the potentiality of more complex coalitions of features, we investigate the possibility of giving (and learning) weights to coalitions of features whose cardinal can be higher than two. To this aim, we propose to consider the set function. A set function is a function with the input is a set, and the output is a value measuring the quality of the set. In our case, set-functions maps a collection of subsets of features to values is considering as associating weights to subsets.

This definition clearly allows high order interactions between features, at the price of higher complexity than conventional Mahalanobis based approaches. In order to deal with this complexity, we propose the use of the submodular function, which is a special kind of set function with the submodular property. Therefore we propose the SML(Submodular Metric Learning) algorithm, while submodular function is used to define the metric and reduce the complexity of learning processing. Further, we decrease the complexity of computation of SML, thanks to limit the order of interactions that are taken into account as constraints by using the k-additive fuzzy measure.

In this Chapter 3, firstly, we will introduce the set function and the submodular function,

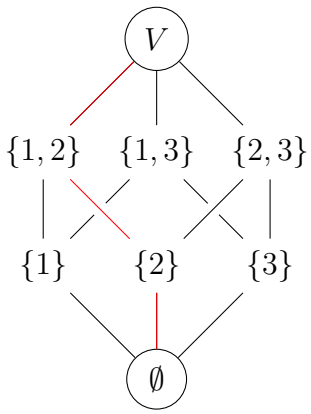


FIGURE 3.1 – Hasse diagram using set-functions on a 3-dimensional problem. $V = \{1, 2, 3\}$

the property and extension of it in Section 3.1. Secondly, in Section 3.2, we will recall the definition of norm and metric, with discussion the performance of different norms, then we will propose to define a norm, and therefore a distance metric. In Section 3.3, the learning processing of this metric is presented. The experiment design and result of approaches will be given in Section 3.4.

This proposed method was published on 27th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning[PLC19b].

3.1 Submodular Function

3.1.1 Set-function and Submodular

In mathematics, a set function is a function whose input is a set. The output is usually a number which judges the quality (or value, or cost) of the set.

In this work, we probe the possibility of giving weights to coalitions of features whose cardinal can be greater than two. For this aim, we consider a class of set-functions $f(\mathcal{S}) : 2^(V) \rightarrow [0, 1]$, that maps subsets $\mathcal{S}$ of a ground set $\mathcal{S}$ to unit interval values. This definition allows associating weights to subsets, in our case, subsets of all feature dimensions of samples.

As shown in Figure 3.1, it is a Hasse diagram of 3-dimensional entity. If we learn a Mahalanobis metric on this entity, for each single dimension {1},{2},{3} and the pairs of dimension a,b,a,c,b,c, the transformation matrix will give weights while the more complex

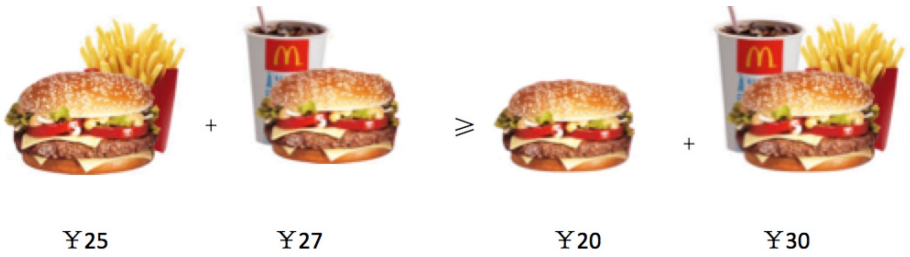


FIGURE 3.2 – The price of McDonald menus. [mcd]

coalition {1,2,3} get ignored. We consider a set-function $f(\mathcal{S})$, in this case $\mathcal{S} \subseteq \{1,2,3\}$, so for another subsets of coalitions of dimensions could be given a weight.

For assuring the condition of metric or norm, we select a special set function, that is submodular function. Such a set function $f(\mathcal{S})$ is said to be submodular if $\forall \mathcal{S}_1, \mathcal{S}_2 \subseteq V$,

$$f(\mathcal{S}_1) + f(\mathcal{S}_2) \geq f(\mathcal{S}_1 \cup \mathcal{S}_2) + f(\mathcal{S}_1 \cap \mathcal{S}_2) \tag{3.1}$$

In terms of optimization, submodular functions can be seen as the discrete equivalent of continuous convex functions. An alternative definition of submodularity is given as $f(\mathcal{S}_1 \cup \{j\}) - f(\mathcal{S}_1) \geq f(\mathcal{S}_2 \cup \{j\}) - f(\mathcal{S}_2)$, where $\mathcal{S}_1 \subseteq \mathcal{S}_2 \subseteq \mathcal{V}$ and elements $j \in \mathcal{V}$, $j \notin \mathcal{S}_2$. Defined this way, adding an element i to a larger set $\mathcal{S}_2$ does not increase $f(\mathcal{S})$ as much as adding the same element j to a smaller set $\mathcal{S}_1$. This property is known as the diminishing return or diminishing marginal utility. It performs like the concavity, and in other ways it resembles convexity. Consequently, problems which concern optimizing a convex or concave function can also be described as the problem of maximizing or minimizing a submodular function subject to some constraints. Thanks to the diminishing returns property, submodular functions have been the topic of research in economics and operation research for quite a long time. Figure 3.2 shows diminishing returns in the price of McDonald menus and Figure 3.3 shows example for submodularity as $f(\mathcal{S}_{a,b} \cup \{i\}) - f(\mathcal{S}_{a,b}) \geq f(\mathcal{S}_{a,b,c,d} \cup \{i\}) - f(\mathcal{S}_{a,b})$.

More recently, submodular functions have attracted interest in the machine learning community (see, e.g., [Bac13]), because of their potential use (clustering, covering, feature selection, social networks) and their similarity to convex functions. In this work, we propose to use set-functions, and in particular submodular set-functions, to weight coalition of features.

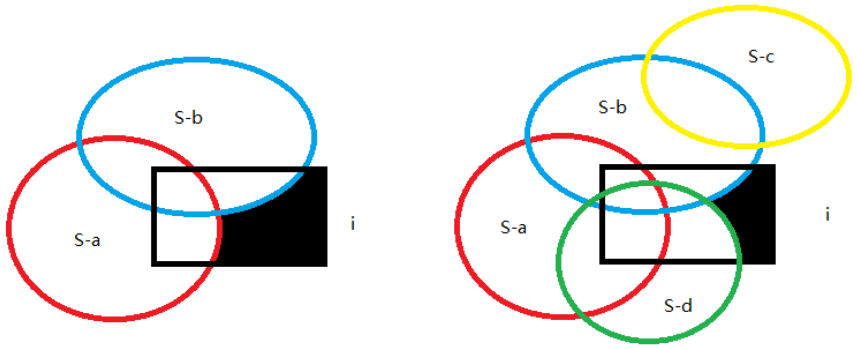


FIGURE 3.3 – $f(\mathcal{S}_{a,b} \cup \{i\}) - f(\mathcal{S}_{a,b}) \geq f(\mathcal{S}_{a,b,c,d} \cup \{i\}) - f(\mathcal{S}_{a,b})$

3.1.2 Lovasz Extension

For optimizing a submodular minimum or maximum problem, we generally apply for the set-function extension. Note that, most of the extensions of set functions try to touch the concave closure of the convex closure of the set function, while they did not require the set function is submodular or not.

One of the most popular extensions of set function is the Lovasz extension. Lovasz extension [Lov83] (also known as the Choquet integral), allows extending a set-function defined on the vertices of the unit hypercube to the full unit hypercube $[0,1]^{|\mathcal{V}|}$. Another appealing property of the Lovasz extension is its ability to draw a link between set-functions and convex functions.

The Lovasz extension $L_f()$ of $\mathbf{x} \in [0,1]^m$ with respect to a set-function $f()$ is defined as :

$$L_f(\mathbf{x}) = \int_0^{+\infty} f(\{x \geq z\})dz + \int_{-\infty}^{0} [f(\{x \geq z\}) - f(V)]dz \tag{3.2}$$

or in the discrete case,

$$L_f(\mathbf{x}) = \sum_{i=1}^{m} x_{(i)} \left[f(\{j | x_j \geq x_{(i)}\}) - f(\{j | x_j \geq x_{(i+1)}\}) \right] \tag{3.3}$$

where $(\cdot)$ denotes a nondecreasing permutation of the input vector $\mathbf{x}$ such that $x_{(m)} \geq \cdots \geq x_{(1)}$ and $x_{(m+1)} = \infty$ by convention where m is the number of dimension of $\mathbf{x}$

Example : Let us consider a two-dimensional observation $\mathbf{x} = [0.87, 0.34]$, so
$x_1 = 0.87$, $x_2 = 0.34$,
$x_{(1)} = 0.34$,$x_{(2)} = 0.87$,$x_{(3)} = \infty$

and the set-function f_1 showed in Figure 3.4 with
$f(\{\emptyset\}) = 0$, $f(\{1\}) = 0.5$,
$f(\{2\}) = 0.5$, $f(\{1,2\}) = 1$.
The Lovasz extension of $\mathbf{x}$ with respect to $f(S)$ is equal to

$$\begin{aligned} L_f(\mathbf{x}) &= x_{(1)} \times \Big[f(\{j|x_j \geq x_{(1)}\}) - f(\{j|x_j \geq x_{(2)}\})\Big] \\ &+ x_{(2)} \times \Big[f(\{j|x_j \geq x_{(2)}\}) - f(\{j|x_j \geq x_{(3)}\})\Big] \\ &= 0.34 \times (f(\{1,2\}) - f(\{1\})) \\ &+ 0.87 \times (f(\{1\} - f(\{\emptyset\})) \\ &= 0.605 \end{aligned}$$

For a set function $f : 2^{\mathcal{V}} \to \mathbb{R}$, the convex closure $f^- : [0,1]^{\|V\|} \to \mathbb{R}$ is the point-wise highest convex function from $[0,1]^{\|\mathcal{V}\|}$ to $\mathbb{R}$ that always lowerbounds $f(\mathcal{S})$. The minimum values of $f(\mathcal{S})$ and f^- are equal.

If S is a minimizer of $f(\mathcal{S})$, $1_{\mathcal{S}}$ is a minimizer of f^-. Moreover, if $\mathbf{x}$ is a minimizer of f^-, then every set in the support of $P_f^-(\mathbf{x})$ is a minimizer of $f(\mathcal{S})$.

For a submodular set-function $f(\mathcal{S})$, the Lovasz extension $L_f(\mathbf{x})$ is non-decreasing, and the convex closure are one and the same [Bac13]. So the minimizer of the submodular function is equal to the minimizer of the Lovasz extension.

3.1.3 Multi-linear Extension

The Lovasz extension is related to the minimizer of submodular function, while the maximizer of submodular function is mainly related to multi-linear extension [VCZ11] [CVZ14].

Given a set-function $f(\mathcal{S})$, the multi-linear extension $Mu_f(\mathbf{x})$ for $\mathbf{x} \in [0,1]^m$ is defined as :

$$Mu_f(\mathbf{x}) = \sum_{\mathcal{S} \subseteq \mathcal{V}}^{2^{|\mathcal{V}|}} f(\mathcal{S}) \prod_{i \in \mathcal{S}} x_i \prod_{j \in \mathcal{V} \setminus \mathcal{S}} (1 - x_j) \tag{3.4}$$

or in a expectation form as :

$$Mu_f(\mathbf{x}) = E_{\mathcal{S} \sim \mathbf{x}}[f(\mathbf{x})] \tag{3.5}$$

where the expectation $E[]$ is over random draws of sets $\mathcal{S} \subseteq \mathcal{V}$, with each element $i \in \mathcal{V}$ drawn in $\mathcal{S}$ with probability x_i. The multilinear extension forms an upper bound on

the Lovasz extension, which $Mu_f(\mathbf{x}) \geq L_f(\mathbf{x})$

Same as the lovasz extension, mullti-linear extension Mu_f is non-decreasing in all direction if the set-function $f(\mathcal{S})$ is submodular. And if the $\mathbf{x}$ is binary vector, multilinear extension agrees with submodular function, $Mu_f(1_\mathcal{S}) = f(\mathcal{S})$.

Unlike the Lovasz extension which is always convex if the set-function is submodular, the multi-linear extension is concave on specific directions and cross-convex on $1_i - 1_j$ directions.

3.1.4 Related Machine Learning Approaches based on Submodular Function

Through the years, there have been several propositions for both submodular function minimization [FHI06], and the generally NP-hard submodular maximization problem requiring approximation [KG12]. In the machine learning area, there are already several approaches using the submodular function [Bac13].

In [PC11], the authors present an algorithm optimizing the F-score to learn a multi-label classifier. For the multi-label task, the submodular function is used for the intersection of pairwise of all the labels. Then they minimum the submodular function via the graph-cuts. This article focuses the multi-label classification and proves the submodular function can process well with the intersection of pairwise information, which in our case is the feature space dimensions and in the case is the labels.

With the help of Lovasz extension, the related optimization problem of the submodular function becomes simpler. In [YB15], the authors developed the tractable convex surrogates submodular losses with Lovasz hinge. They analyzed the conventional methods of set prediction, namely margin rescaling and slack rescaling. However, these two methods lead to tight convex substitution and increase incorrect prediction. Instead of them, the Lovasz extension is applied to the access loss function to calculate the gradient or cut plane. Experiments with real image datasets demonstrate that Lovasz hinges perform better than other algorithms.

Besides using the submodular function for a machine learning task, there are also several algorithms are considering define a metric for learning with the Lovasz extension of a submodular function.

In [IB13], they extend the recently Bregman divergences to a specific class Lovasz Bregman divergences. The recently Bregman divergences is a measure of the distance bet-

	$f(\{1\})$	$f(\{2\})$	$f(\{1,2\})$
f_1	0.5	0.5	1
f_2	0.25	0.25	0.25
f_3	0.2	0.5	0.7
f_4	0.5	0.5	0.7

TABLE 3.1 – Values of the set-functions used in Figure 3.4.

ween two points, defined from convex function and extended to between functions or between sets. The authors proposed the Lovasz Bregman divergences, which the parameters are the Lovasz extension of a submodular function, and learning the proposed divergences to rank based clustering. The authors use the Lovasz Bregman divergences as a measure and give several properties(such as non-negativity and convexity, equivalence classes, linearity and linear separation), however, they did not prove it is metric and try metric learning method on it. Note that the properties of Lovasz Bregman divergences is interesting as a measure and the authors use them for the "learning to rank" problem in web ranking.

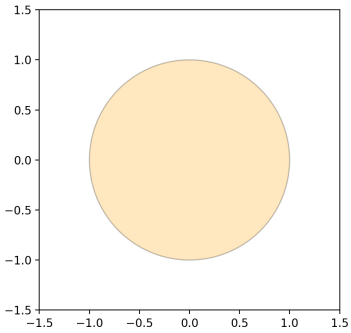
The authors investigate the possibility of learning a distance for higher-order based on a submodular function in [GIL+15], which is similar to our aim, but they restrict to binary vectors. In this paper, they learn a hamming distance due to the use of the symmetric difference between binary sets. They proved the possibility to define a metric by the submodular function and show the performance of the submodular Hamming metric on metric minimization task (clustering) and metric maximization task (diverse k-best).

In [Bac13], they give another proof of the links between submodularity and convexity. Unlike we use submodular constraints to find a metric, they use a support function of submodular set function as a regularizer for optimize the loss and mostly focus the supervised learning task like the form of variable or feature selection problems. We got inspirations from the methods they used to optimize the submodular functions.

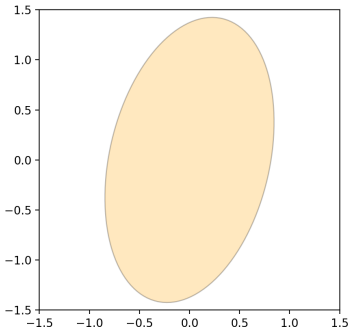
3.2 Definition and Proof of Submodular Extension Metric

For metric learning algorithms we mentioned in Chapter 2, most of the metric they learn is a distance transformation with Mahalanobis metric, while several of them are not.

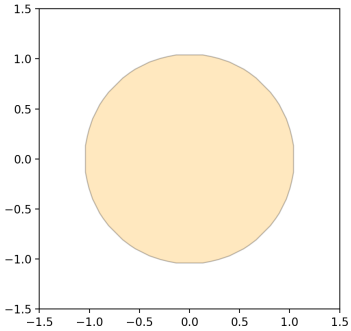
In Figure 3.4, the unit balls for various distances, including the one provided ($L_f(\mathbf{x})$) in this work are given. As can be seen, it allows obtaining a wide range of convex shapes, generalizing both Euclidean distance and Mahalanobis distance.



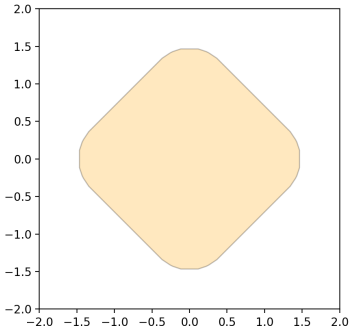
(a) Euclidean distance



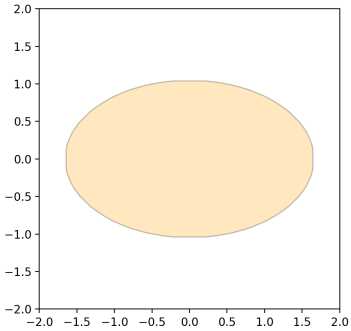
(b) Mahalanobis distance



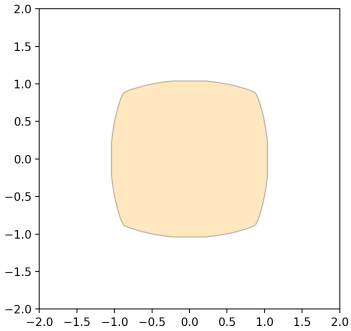
(c) Lovasz distance with f_1



(d) Lovasz distance with f_2



(e) Lovasz distance with f_3



(f) Lovasz distance with f_4

FIGURE 3.4 – Unit balls ($d^2(\mathbf{x}, 0) \leq 1$) for different metrics. The set-functions $f_1()$, $f_2()$, $f_3()$ and $f_4()$ given in Table 3.1, respectively.

In this work, we propose to define a norm and there-fore a distance metric which could break the limit of the Mahalanobis metric. We introduce the submodular function and explain how we extend the Mahalanobis transformation matrix to a submodular function giving weight to subsets of features.

3.2.1 Lovasz Extension Norm

The Lovasz extension allows setting weights to subsets, and now turn to its use for defining a norm, given some conditions on $f(\mathcal{S})$. In particular, we mentioned that for a submodular function $f(\mathcal{S})$, its Lovasz extension and the convex closure are one and the same, so we are able to define a norm with the Lovasz extension of a submodular function. In the sequel, we consider the vector space $\mathbb{V}$ as $\mathbb{R}^{|\mathcal{V}|}$.

Proposition 1. *The function $L_f(|\mathbf{x}|) : \mathbb{R}^{|\mathcal{V}|} \rightarrow \mathbb{R}^+$ is a norm if and only if $f()$ is a submodular set-function.*

Démonstration. It is straightforward to show that $L_f(|\mathbf{x}|)$ satisfies the separate points and absolute homogeneity conditions, whatever $f(\mathcal{S})$. We now turn to the triangular inequality condition. Let us assume that $f(\mathcal{S})$ is a submodular set-function. By definition, the function is positively homogeneous. A set-function $f(\mathcal{S})$ is submodular, if and only if $L_f(\mathbf{x})$ is convex (see [Bac13, Lov83]). The convexity of $L_f(\mathbf{x})$ implies convexity of $L_f(|\mathbf{x}|)$ (by composition of convex non-decreasing functions). By convexity of $L_f(|\mathbf{x}|)$, we have $\frac{1}{2}L_f(|\mathbf{x}_i|) + \frac{1}{2}L_f(|\mathbf{x}_j|) \geq L_f(|\frac{1}{2}\mathbf{x}_i| + \frac{1}{2}|\mathbf{x}_j|) = \frac{1}{2}L_f(|\mathbf{x}_i| + |\mathbf{x}_j|)$, by homogeneity of $L_f(\mathbf{x})$. On the other hand, if $L_f(|\mathbf{x}|$ is a norm, then it is convex, implying convexity of $L_f(\mathbf{x})$, and therefore submodularity of $f(\mathcal{S})$, concluding the proof. □

Corollary 1. *The function $(L_f(|\mathbf{x}|^p))^{\frac{1}{p}} : \mathbb{R}^d \rightarrow \mathbb{R}^+$ is a norm for any $p \geq 1$.*

Finally, we define the Lovasz extension metric $d_f(\mathbf{x}_i, \mathbf{x}_j)$ using the squared Lovasz extension norm as follow :

$$d_f^2(\mathbf{x}_i, \mathbf{x}_j) = L_f\left((\mathbf{x}_i - \mathbf{x}_j)^2\right) \tag{3.6}$$

Note that Equation 3.6 can be easily generalized on p-power Lovasz extension norms, exactly the same way as L_p norms of Euclidean spaces.

The red path in Figure 3.1 corresponds to one possible use of learned weights for computing one distance $L_f(\mathbf{x})$. In practice, we will see that computing a distance between two

objects using the proposed $d_f(\mathbf{x}_i, \mathbf{x}_j)$ (see Eq. 3.6) reduces to a weighted path from $\emptyset$ to V in the lattice. In Figure 3.4, the unit balls for various distances, including the one provided ($L_f(\mathbf{x})$) in this work are given. As can be seen, it allows obtaining a wide range of convex shapes, generalizing both Euclidean distance (first on row 1) and Mahalanobis distance (second on row 1).

3.2.2 Multi-linear Extension Dissimilarity

Unfortunately we could not define a norm with the multi-linear extension, because of two reasons :

— Reason one : Because the structure of the multi-linear basis function $\prod_{i\in\mathcal{S}} x_i \prod_{j\in V\setminus\mathcal{S}}(1-x_j)$, it is hard to find a transform function for make the multi-linear extension subjected the homogeneity which is the second condition to the norm.

— Reason two : When the $f(\mathcal{S})$ is a submodular function, the multi-linear extension $Mu_f(\mathbf{x})$ will be a concave function on certain direction which is against to the triangle inequality which is the third condition to the norm.

However, we still try to define the multi-linear extension as a dissimilarity. The metric definition we mentioned before is a function satisfying non-negativity, identity of indiscernibles, symmetry and triangular inequality. We try to define the multi-linear dissimilarity with satisfying these condition with an absolute value of difference under some limits. The multi-linear dissimilarity is denoted as :

$$d_{M_f}(\mathbf{x}_i, \mathbf{x}_j) = Mu_f(v(\mathbf{x}_i, \mathbf{x}_j)) \tag{3.7}$$

where $v(\mathbf{x}_i, \mathbf{x}_j) = (\|x_{i,1} - x_{j,1}\|, \|x_{i,2} - x_{i,2}\|, \cdots, \|x_{i,m} - x_{j,m}\|)^T$ is the absolute value of difference of two entities.

Proposition 2. *The function $Mu_f(v(\mathbf{x}_i, \mathbf{x}_j)) : \mathbb{R}^{|\mathcal{V}|} \to \mathbb{R}^+$ is a metric if $f(\mathcal{S})$ is a submodular set-function, and under the limit that there is not only one positive dimension of $v(\mathbf{x}_i, \mathbf{x}_j)$.*

A metric is a function $d : \mathbb{V} \times \mathbb{V} \to \mathbb{R}^+$ on a set $\mathbb{V}$, $\forall \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k \in \mathbb{V}$ satisfying the following conditions :

1. non-negativity : We normalized every feature $\mathbf{x} \in [0, 1]^m$, so the absolute value of difference $v(\mathbf{x}_i, \mathbf{x}_j)$ will be non-negativity. If the $f(\mathcal{S})$ is submodular, the multi-

linear extension will be non-decreasing, so $d_{M_f}(\mathbf{x}_i, \mathbf{x}_j) = Mu_f(v(\mathbf{x}_i, \mathbf{x}_j))$ is non-negativity.

2. identity of indiscernibles : When $\mathbf{x}_i = \mathbf{x}_j$, the $v(\mathbf{x}_i, \mathbf{x}_j) = (0, 0, \cdots, 0)^T$, and $d_{M_f}(\mathbf{x}_i, \mathbf{x}_j) = 0$.

3. symmetry : The absolute value of difference $v(\mathbf{x}_i, \mathbf{x}_j)$ is certainly symmetry, therefore the multi-linear dissimilarity $d_{M_f}(\mathbf{x}_i, \mathbf{x}_j)$ is symmetry.

4. triangular inequality : When $f(\mathcal{S})$ is submodular, Mu_f is is concave on certain directions and cross-convex on $1_i - 1_j$ directions. When there is only one dimension i of the $v(\mathbf{x}_i, \mathbf{x}_j)$ is positive and others are zero, the Mu_f is concave and against triangular inequality. When there are two dimensions i and j of the $v(\mathbf{x}_i, \mathbf{x}_j)$ is positive and others are zero, the $\frac{\delta^2 Mu_f(v_i)}{\delta v_i \delta v_j} \leq 0$, which means it is convex and satisfies triangular inequality. So under the limit that there is not only one positive dimension of $v(\mathbf{x}_i, \mathbf{x}_j)$ and $f(\mathcal{S})$ is submodular, $d_{M_f}(\mathbf{x}_i, \mathbf{x}_j)$ satisfies triangular inequality.

We could not be sure the select constraints from the sample only have difference under the condition that there is not only one positive dimension of $v(\mathbf{x}_i, \mathbf{x}_j)$, so the denoted multi-linear dissimilarity is a pseudo metric. Thought that we still try to learn this dissimilarity and observe its performance through the experiments.

3.3 Proposed Submodular Metrics Learning Algorithm

In this section, we proposed SML(Submodular Metric Learning) algorithm, which learned the proposed Lovasz extension metric in the last section. Following usual metric learning formulation, we are now able to write the following optimization problem using relative constraints $\mathbb{C}$ where $d_M(\mathbf{x}_i, \mathbf{x}_k) \geq d_M(\mathbf{x}_i, \mathbf{x}_j) + \gamma \forall (i, j, k) \in \mathbb{C}$, given a submodular set-function $f(\mathcal{S})$,

$$min_f \sum_{(i,j,k)\in\mathbb{C}} \ell(i, j, k) + \lambda r(f) \tag{3.8}$$

where the r is the regularizer on $f(\mathcal{S})$, and ℓ is the hinge loss defined as $\ell(i, j, k) = [\gamma + d_f^2(x_i, x_j) - d_f^2(x_i, x_k)]_+$. In the sequel, and following earlier works, the margin γ is set to 1. Written as a constrained optimization problem, it gives

submodular constraint	$f(\{1\})$	$f(\{2\})$	$f(\{3\})$	$f(\{1,2\})$	$f(\{1,3\})$	$f(\{2,3\})$	$f(\{1,2,3\})$
$f(\{1\}) + f(\{2\}) \geq f(\{1,2\})$	-1	-1	0	1	0	0	0
$f(\{1,2\}) + f(\{2,3\}) \geq f(\{2\}) + f(\{1,2,3\})$	0	1	0	-1	0	-1	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

TABLE 3.2 – Submodular constraints, as a ternary matrix S, with linear inequalities on a small subsample for which $|\mathcal{V}| = 3$.

$$\begin{aligned} &\min r(f) \\ &\text{s.t. } \ell(i,j,k) \leq 0, \forall (i,j,k) \in \mathbb{C} \\ &\quad f(\mathcal{S}) \text{ is submodular} \end{aligned} \tag{3.9}$$

Although we are aware that one can consider sparse LP solutions [YZH+15] to tackle this problem, we do not consider this family of approaches in this work. Naturally, it can be used to improve our proposition further.

For the Multi-linear extension dissimilarity, we use the same solution and only need to change the way to compute the distance for the selected relative constraints part.

3.3.1 Set-function Vector and Constraints Matrix

In order to adapt to compute the Lovasz extension metric $L_f(\mathbf{x})$ and multi-linear dissimilarity d_{M_f} of the set-function $f(\mathcal{S})$, we use the following vector notation for the set-function :

$\mathbf{f} = (f(\{1\}), f(\{2\}), \cdots, f(\{1,2\}), \cdots, f(\{1, \cdots, d\}))^T$.

Therefore the submodularity of $f(\mathcal{S})$ can be written as an inequality. In particular, by using a matrix of $\{-1, 0, 1\}$ values, one can write each of the $\frac{1}{2}2^{|\mathcal{V}|}(2^{|\mathcal{V}|}+1)$ submodular constraints, see Table 3.2 for a simple illustration with $|\mathcal{V}| = 3$. Let C_s be such a matrix. Consequently, the submodularity constraint of Equation 3.10 can be written as $C_s\mathbf{f} \leq 0$.

Same as the submodularity constraint, the relative constraints $\ell(i,j,k) = d(i,k) - d(i,j) - \gamma \leq 0 \forall (i,j,k) \in \mathbb{C}$ could also be rewritten as $C_r\mathbf{f} \leq \mathbf{b}$ where the $\mathbf{b}$ is the constant margin vector with all value are equal to the margin γ and the matrix C_r is computed from the learned metric and selected samples in set $\mathbb{C}$.

For computing the Lovasz extension metric, we rewrite the metric $d_f^2(\mathbf{x}_i, \mathbf{x}_i)$ with m-dimensions as following :

$$
\begin{aligned}
d_f^2(\mathbf{x}_i, \mathbf{x}_j) =& L_f\left((\mathbf{x}_i - \mathbf{x}_i)^2\right) && (3.10)\\
=& \sum_{k=1}^{m} (x_i - x_j)_{(k)}^2 [f(\{l|(x_i - x_j)_l^2 \geq (x_i - x_j)_{(k)}^2\}) && (3.11)\\
& - f(\{l|(x_i - x_j)_l^2 \geq (x_i - x_j)_{(k+1)}^2\})]\\
=& f(\oslash)\left[0 - (x_i - x_j)_{(m)}^2\right]\\
& + f((x_i - x_j)_{(1)}^2)\left[(x_i - x_j)_{(m)}^2 - (x_i - x_j)_{(m-1)}^2\right]\\
& \cdots\\
& + f(\mathcal{V} \setminus (x_i - x_j)_{(m)}^2)[(x_i - x_j)_{(2)}^2 - (x_i - x_j)_{(1)}^2]\\
& + f(\mathcal{V})[(x_i - x_j)_{(1)}^2 - 0]
\end{aligned}
$$

where $(.)$ is the permutation defined within Equation 3.3,$\mathcal{V}$ is the full set of all m-dimensions of the features.

Straightforward manipulation of the Lovasz extension w.r.t the set-function $\mathbf{f}$ leads to the following expression, $d_f^2(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{a}_{ij}\mathbf{f}$ as a calculated vector $\mathbf{a}_{ij}$ for i-th and j-th sample multiply with the submodular set function vector, where :

$$
\mathbf{a}_{ij} = \begin{pmatrix}
0 - (x_i - x_j)_{(m)}^2 \\
0 \\
\cdots \\
(x_i - x_j)_{(m)}^2 - (x_i - x_j)_{m-1}^2 \\
0 \\
\cdots \\
0 \\
\cdots \\
(x_i - x_j)_{(2)}^2 - (x_i - x_j)_{(1)}^2 \\
0 \\
\cdots \\
(x_i - x_j)_{(1)}^2 - 0
\end{pmatrix}, \qquad (3.12)
$$

For computing the multi-linear dissimilarity, it will be easier because the form of d_{M_f}

with m-dimensions can be written as following :

$$\begin{aligned} d_{M_f}(\mathbf{x}_i, \mathbf{x}_j) =& Mu_f(v(\mathbf{x}_i, \mathbf{x}_j)) && (3.13)\\ =& \sum_{S\subseteq\mathcal{V}}^{2^{|\mathcal{V}|}} f(S) \prod_{k\in S}(v(\mathbf{x}_i, \mathbf{x}_j)_k) \prod_{l\in\mathcal{V}\setminus S}(1 - v(\mathbf{x}_i, \mathbf{x}_j)_l) && (3.14)\\ =& \mathbf{f} \prod_{k\in S}(\|x_{i_k} - x_{j_k}\|) \prod_{l\in V\setminus S}(1 - \|x_{i_l} - x_{j_l}\|) && (3.15) \end{aligned}$$

So the expression of multi-linear dissimilarity d_{M_f} could be also written as a calculated vector $\mathbf{a}'_{ij}$ for i-th and j-th sample multiply with the submodular set function vector, $d_{M_f} = \mathbf{a}'_{ij}\mathbf{f}$, where :

$$\mathbf{a}'_{ij} = \begin{pmatrix} (1 - \|x_{i_1} - x_{j_1}\|) * (1 - \|x_{i_2} - x_{j_2}\|) * \cdots * (1 - \|x_{i_m} - x_{j_m}\|) \\ \|x_{i_1} - x_{j_1}\| * (1 - \|x_{i_2} - x_{j_2}\|) * \cdots * (1 - \|x_{i_m} - x_{j_m}\|) \\ (1 - \|x_{i_1} - x_{j_1}\|) * \|x_{i_2} - x_{j_2}\| * \cdots * (1 - \|x_{i_m} - x_{j_m}\|) \\ \cdots \\ \|x_{i_1} - x_{j_1}\| * \|x_{i_2} - x_{j_2}\| * \cdots * \|x_{i_m} - x_{j_m}\| \end{pmatrix}, \quad (3.16)$$

x_{i_k} is the k-th dimension feature of the sample x_i.

Therefore, the inequality $C_r^T\mathbf{f} + \mathbf{b} \leq 0$ could be finally written as

$$C_r^T = \left(\mathbf{a}_{ij}^1 - \mathbf{a}_{ik}^1, \cdots, \mathbf{a}_{ij}^\kappa - \mathbf{a}_{ik}^\kappa\right), \quad (3.17)$$

corresponding to the κ constraints of $\mathbb{C}$, and the $\mathbf{a}_{ij}^l - \mathbf{a}_{ik}^l$ is the calculated vectors for l-th selected constraints triple of (i, j, k)-th sample.

Because the Lovasz extension metric $L_f(\mathbf{x})$ and multi-linear dissimilarity d_{M_f} are linear in $f(\mathcal{S})$, with such a form of set-function, the Equation 3.10 can be written as a linear inequality program :

$$\begin{aligned} &\min \mathbf{f}^T\mathbf{r} && (3.18)\\ &\text{s.t. } C^T\mathbf{f} + \mathbf{t} \leq 0 \\ &0 \leq \mathbf{f} \leq 1 \end{aligned}$$

where

$$C = \begin{pmatrix} C_r^T \\ C_s^T \end{pmatrix}, \mathbf{t} = \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix}, \tag{3.19}$$

and $\mathbf{r}$ is the unit 2^m dimensional vector as the regularization for set-function vector $\mathbf{f}$.

In practice, all constraints cannot be satisfied with real data, so that we introduce non-negative slack variables ξ_i for each of these constraints. We subsequently add the penalty term $a\sum_{i=1}^{2^m}\xi_i$ to $\mathbf{f}^T\mathbf{r}$, where a is a trade-off parameter (set to 1 in our experiments).

3.3.2 Submodular Constraints Matrix Reduction

The computational complexity of the SML algorithm comes from two aspects, one is the size of the sample size that can be solved from the optimization method (in the future we will improve the SML algorithm to an online learning method or other computational complexity that does not depend on the sample size Method), on the other hand is derived from submodular constraints matrix. The number of values to be learned in the submodular constraints matrix, for a m-dimensional dataset is $2^m - 2$. Furthermore, as indicated earlier, the number of constraints for verifying submodularity is $\frac{1}{2}2^m(2^m + 1)$. That makes the problem intractable for size-able dimensional data sets. Somewhat naive way of tackling this inability is to reduce the dimension of the data. Use dimension reduction methods and then learn the metric in the new feature space. However, the dimension reduction and metric learning would not be jointly learned, and so the resulting metric would be sub-optimal.

To deal with this problem, we propose in this work to consider the extension of the concept of k-additive fuzzy measure, see [Gra16], to set-functions to simplify the optimization problem. To do so, we consider pseudo-Boolean functions, that can express set-functions as a polynomial of degree m. Formally, we define a k-additive set-function as an additive set-function whose corresponding pseudo-Boolean function has a polynomial development of degree at most k. Interestingly, if a set-function is k-additive, it means that there are no interactions between subsets of more than k elements.

Therefore, k-additive set-functions restrict their values to sets $\mathcal{S}$ for which we have $|\mathcal{S}| \leq k$. This drastically reduces the number of variables required to fully define the set-function $f(\mathcal{S})$, going from 2^m to 2^k, with $k << m$. Note that this definition differs from the proposition of [BFNS14], where the objective is to find the subset verifying this cardinality constraint.

Additionally, it also corresponds to the fact that the inverse of the function $f(\mathcal{S})$ (also

known as Mobius Transform, see [Gra16]), defined as

$$f^{-1}(\mathcal{S}) = \sum_{\mathcal{T} \subset \mathcal{S}} (-1)^{|\mathcal{S} \setminus \mathcal{T}|} f(\mathcal{T}), \text{ for all } \mathcal{S} \subseteq \mathcal{V} \tag{3.20}$$

vanishes for subsets whose cardinal is greater than k.

The Lovasz extension can be written using the inverse function as

$$L_f(\mathbf{x}) = \sum_{\mathcal{T} \subseteq \mathcal{V}} f^{-1}(\mathcal{T}) \min_{i \in \mathcal{T}} x_i, \tag{3.21}$$

therefore simplifying the problem. Note that there is a one-to-one correspondence between $f(\mathcal{S})$ and $f^{-1}()$, since we have

$$f(\mathcal{S}) = \sum_{\mathcal{T} \subset \mathcal{S}} f^{-1}(\mathcal{T}) \tag{3.22}$$

In order to obtain a metric, the set-function $f(\mathcal{S})$ must remain submodular (see Proposition 1). One can write the submodular constraint imposed on $f(\mathcal{S})$ with its inverse $f^{-1}(\mathcal{S})$. Furthermore, writing the submodular constraint on the set-function with the help of its inverse function allows to decrease the number of constraints of the optimization problem.

Proposition 3. *Let $f : 2^{|}\mathcal{V}| \rightarrow [0,1]$ be a set-function and $f^{-1}(\mathcal{S}) : 2^{|}\mathcal{V}| \rightarrow [0,1]$ its inverse function defined by Equation 3.20, then a) and b) are equivalent.*

a) $f(\mathcal{S})$ is a submodular set-function,
b) $\sum_{\mathcal{T} \subset \mathcal{S}_1 \cup \mathcal{S}_2, \mathcal{T} \not\subset \mathcal{S}_1, \mathcal{T} \not\subset \mathcal{S}_2} f^{-1}(\mathcal{T}) \leq 0$.

Démonstration. Let us introduce, for every $\mathcal{T}$ in $\mathcal{V}$, $I(\mathcal{T}) = \{k | 1 \leq k \leq 2, \mathcal{T} \subset \mathcal{S}_k\}$. Submodularity of $f(\mathcal{S})$ can be written as

$$f(\mathcal{S}_1 \cup \mathcal{S}_2) + f(\mathcal{S}_1 \cap \mathcal{S}_2) \leq f(\mathcal{S}_1) + f(\mathcal{S}_2) \tag{3.23}$$

for $\mathcal{S}_k$ in $\mathcal{V}$. Using Equation 3.20, we write $f(\mathcal{S}_1 \cap \mathcal{S}_2) = \sum_{\mathcal{T} \subset \mathcal{S}_1 \cap \mathcal{S}_2} f^{-1}(\mathcal{T})$, and developing $f(\mathcal{S}_1 \cup \mathcal{S}_2)$ gives $\sum_{I(\mathcal{T}) \neq \emptyset} f^{-1}(\mathcal{T}) + \sum_{\mathcal{T} \subset \mathcal{S}_1 \cup \mathcal{S}_2, \mathcal{T} \not\subset \mathcal{S}_1, \mathcal{T} \not\subset \mathcal{S}_2} f^{-1}(\mathcal{T})$.

Consequently, Equation 3.23 is satisfied if and only if

$$\begin{aligned} &\sum_{\mathcal{T}\subset\mathcal{S}_1\cap\mathcal{S}_2} f^{-1}(\mathcal{T}) + \sum_{I(\mathcal{T})\neq\emptyset} f^{-1}(\mathcal{T}) \\ &+ \sum_{\mathcal{T}\subset\mathcal{S}_1\cup\mathcal{S}_2, \mathcal{T}\not\subset\mathcal{S}_1, \mathcal{T}\not\subset\mathcal{S}_2} f^{-1}(\mathcal{T}) \\ &\leq \sum_{\mathcal{T}\subset\mathcal{S}_1} f^{-1}(\mathcal{T}) + \sum_{\mathcal{T}\subset\mathcal{S}_2} f^{-1}(\mathcal{T}) \end{aligned} \tag{3.24}$$

holds. Finally, it is straightforward to obtain $\sum_{T\subset\mathcal{S}_1\cup\mathcal{S}_2, T\not\subset\mathcal{S}_1, T\not\subset\mathcal{S}_2} f^{-1}(T) \leq 0$ from Equation 3.24, concluding the proof. □

Using this formulation, the number of constraints for preserving submodularity decreases to $\frac{1}{8}2^m(m^2-m)$, which is much more reasonable for practical problems. More precisely, incorporating the constraint b) of Proposition 2 allows decreasing the size of the ternary matrix C_s, whose size was initially $\frac{1}{2}2^m(2^m+1)$. Moreover, we use the proposition relying on k-additive set-functions, i.e. we restrict the computation of the inverse set-function f^{-1}, given by Equation 3.20, on sets $\mathcal{S}$ for which $|\mathcal{S}| \leq k$ holds. In that case, this is even reduced to $\frac{1}{8}2^k(k^2-k)$, where $k << m$, typically lower than 10. Combining this reduction and a k-additive set function leads to a tractable problem.

The solution obtained with this k-additive approach, and implementing submodular constraints following Proposition 2., is denoted as $L_f^k(\mathcal{S})$ hereafter.

3.4 Experiments and Result

Now, we conduct experiments which demonstrate the performance, and in particular the classification generalization performance, of the proposed method of metric learning on some real-world datasets.

3.4.1 Datasets and Experiment Design

For the test datasets, 8 data sets from the UC Irvine Machine Learning Repository [Lic13] are chosen, and their main characteristics are given in Table 3.3.

From most of the metric learning algorithms, we could see cross-validation on the task of K-nearest neighbours classification is used for comparing the accuracy. The K is selected between $[3, 5]$, and the folds of cross-validation are set between $[3, 15]$. Different values

Dataset	m	c	n
seeds	7	3	210
sonar	60	2	208
iono	34	2	351
balance	4	3	625
glass	10	7	214
digits	64	10	1797
liver	6	2	347
segment	19	7	2310

TABLE 3.3 – UCI datasets used in the experiments. c indicates the number of classes.

are tested, without significant modification on the comments that can be drawn from the results. So the shown results are set with $K = 5$ with 10 folds of cross-validation.

As mentioned earlier, due to the complexity of the model, our first proposition Lovasz extension metric learning algorithm $L_f(\mathbf{x})$ is not able to process datasets for which the dimension is (even moderately) large. Consequently, a PCA(Principal component analysis) is used on the data whose dimension is greater than 10 : sonar, ionosphere, digits and segment, for which the lost variance is 12.02, 21.97, 26.26 and 0.008, respectively. The other datasets remained unchanged.

For a fair comparison, and avoid the potential benefit obtained from PCA, the result of all other metric learning algorithms are also obtained after PCA projection. Results without PCA for other metric learning approaches were worse than those obtained with PCA.

Several state-of-the-art linear and nonlinear metric learning algorithms are compared against the results obtained with our proposed method : LMNN [WS09], ITML [DKJ$^+$07], LSML [LGZ$^+$12], LFDA [Sug06], GMML [ZHS16] and GB-LMNN [KTS$^+$12, KXW]. We also give the results obtained without metric learning, i.e. the Euclidean distance for which $M = Id$ and the kernel principal component analysis algorithm [SSM99].

In order to build the set $\mathbb{C}$ of relative constraints, the triples of objects $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ are randomly selected based on labels, for which $(\mathbf{x}_i, \mathbf{x}_j)$ have the same labels and $(\mathbf{x}_i, \mathbf{x}_k)$ have different labels. For the number of constraints, in LMNN [WS09] they use all supervised information and limit with maximum number of iteration ; in ITML [DKJ$^+$07] it could be a user-select number of constraints and limit with maximum number of iteration ; in LSML [LGZ$^+$12] they choose 100 for all dataset and compared algorithms ; in GMML [ZHS16] and GB-LMNN [KTS$^+$12, KXW] the number of constraints depends on the size of datasets. To be fair, for all learning algorithms(which could select the number of

constraints with the original code), the same maximum number of constraints is set, and as shown in Figure 3.5, after testing several different sets the ranks are similar.

3.4.2 Result for the Lovasz Extension Norm

In the first part of the experiments, we are using SML, that is using all possible orders of feature interactions. In particular, the only constraints are related to submodularity and relative distance constraints.

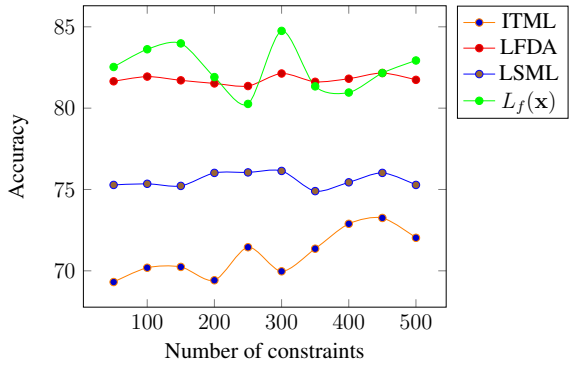
Accuracy score obtained of the cross-validation of KNN algorithms on the 8 datasets for each method are given in Table 3.4. In this table, the number of constraints κ is 200, and the accuracy score is the avenge of 10 times running of the accuracy rate, than multiply 100. As can be seen, the proposed $L_f(\mathbf{x})$ generally performs better than all the other metric learning algorithms (with the notable exception of Ionosphere and Segment datasets). More precisely, given the average rank of each method, we obtain the following ranking $SML \succ GB-LMNN \succ GMML \succ LFDA \sim LMNN \succ LSML \succ KPCA \succ ITML \succ Id.$ Statistical significance of the results are assessed using a Friedman test [Fri40] as suggested by [Dem06]. The value of the Friedman test, $F_F = 7.10 > F_{0.05}(8, 56)$ shows the significance of the difference between the ranks.

Dataset	Euc.	KPCA	LMNN	ITML	LSML	LFDA	GMML	GB-LMNN	SML
balance	72.66	73.74	78.86	77.17	73.82	80.22	80.34	68.90	**81.03**
digits	93.77	94.20	94.33	90.94	92.83	94.10	94.73	**94.78**	93.88
glass	61.02	65.51	65.21	57.24	64.27	58.17	64.86	66.79	**68.72**
iono	85.75	88.72	87.17	85.18	86.04	77.18	87.56	**94.32**	86.60
liver	66.48	66.61	63.87	62.26	65.70	66.14	64.72	66.48	**66.49**
seeds	82.57	88.95	88.52	87.62	88.10	89.48	88.67	88.10	**90.88**
sonar	50.87	56.16	55.69	48.92	51.53	52.86	55.83	**66.76**	56.95
segment	78.10	78.19	82.52	80.29	**85.67**	83.61	83.34	83.42	84.23

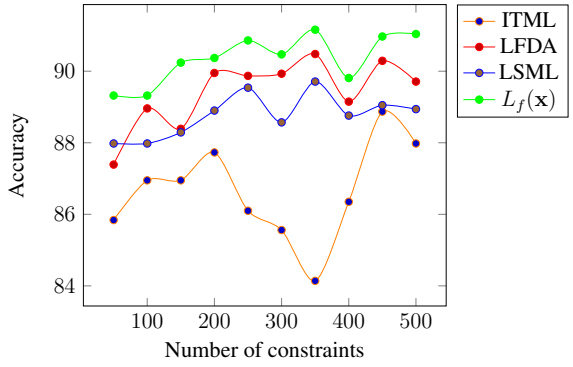
TABLE 3.4 – Accuracy score of KNN with SML and different comparing metrics learning algorithms.

3.4.3 Result for the K-additive Complexity Reduction

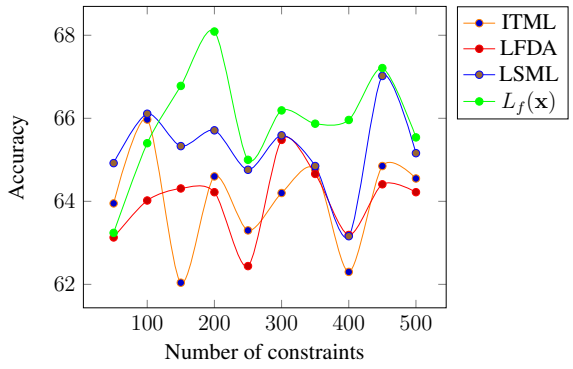
This part of the experiments uses the SML-K , which is the SML with the modified k value for K-additive complexity reduction on the submodular constraints matrix. In particular, it uses the constraint obtained by using the result of Proposition 2 (in which submodularity



(a) Balance



(b) Seeds



(c) Glass

FIGURE 3.5 – Different numbers of constraints

constraints can be written using the inversion f^{-1}). In this experiment, we use the same datasets as in the previous experiment, but we use SML-K and make this k varies from 1 to $\min(10, m)$. A value of $k = 1$ means that there is no interaction between features, and only singletons are considered. Increasing k adds orders of interaction, and finally reaches the order of interaction tackled by the first $L_f(\mathbf{x})$ approach. It can be noted that each time we decrease k, the number of free parameters of $f(\mathcal{S})$ is divided by 2, so that running time of the method is now very reasonable, even for quite large dimensional data. The results for varying k are given in Figure 3.6.

Dataset	LMNN	ITML	LSML	LFDA	GMML	GB-LMNN	SML	SML-K
balance	78.86	77.17	73.82	80.22	80.34	68.90	81.03	**81.14**
digits	94.33	90.94	92.83	94.10	94.73	**94.78**	93.88	94.03
glass	65.21	57.24	64.27	58.17	64.86	66.79	**68.72**	68.17
iono	87.17	85.18	86.04	77.18	87.56	**94.32**	86.60	88.61
liver	63.87	62.26	65.70	66.14	64.72	66.48	66.49	**66.60**
seeds	88.52	87.62	88.10	89.48	88.67	88.10	**90.88**	90.23
sonar	55.69	48.92	51.53	52.86	55.83	**66.76**	56.95	58.75
segment	82.52	80.29	**85.67**	83.61	83.34	83.42	84.23	83.72

TABLE 3.5 – Accuracy of KNN with SML, SML-K and different comparing metrics learning algorithms.

Table 3.5 gives the results obtained through a grid search of k (last column) and Table 3.6 gives the running time in seconds. Interestingly, one can see that SML-K often gives better results than SML, showing that using all the m-tuple-wise combinations are not always

Dataset	LMNN	ITML	LSML	LFDA	GMML	GB-LMNN	SML	SML-K
balance	16.95s	7.35s	0.01s	0.01s	0.42s	0.23s	0.01s	0.01s
digits	254.0s	0.71s	0.01s	0.07s	1.76s	4.60s	126.0s	2.60s
glass	4.19s	7.53s	3.95s	0.01s	0.32s	0.39s	0.94s	0.09s
iono	6.58s	6.12s	0.09s	0.09s	0.16s	2.32s	150.9s	2.54s
liver	36.11s	7.55s	5.43s	0.02s	0.85s	3.52s	0.01s	0.01s
seeds	2.45s	14.30s	2.71s	0.01s	0.61s	2.42s	0.02s	0.01s
sonar	2.80s	3.11s	0.02s	0.01s	1.04s	2.07s	124.9s	2.17s
segment	76.75s	1.26s	0.05s	0.01s	1.03s	2.12s	168.8s	2.76s

TABLE 3.6 – Running time in seconds of SML, SML-K and different comparing metrics learning algorithms.

Dataset	Euc.	LMNN	ITML	M_f
balance	72.66	78.86	77.17	62.22
glass	61.02	65.21	57.24	65.23
iono	85.75	87.17	85.18	78.45
seeds	82.57	88.52	87.62	83.42
sonar	50.87	55.69	48.92	42.53

TABLE 3.7 – Accuracy of KNN with multi-linear extension dissimilarity and different metrics learning algorithm

useful, and may even penalize the performances (e.g. balance, ionosphere, liver and sonar). Note that for low dimensional datasets, the running time of the SML is low, and quickly increases with the dimension of the data. According to these results, one can draw the following comments. As can be expected, increasing k allows obtaining a better classification accuracy on almost all datasets. One interesting point is that going from order 1 (weighted feature) to order 2 (e.g. Mahalanobis) is generally sufficient to obtain better results. Increasing to high-order can be worth the computational effort (e.g. Balance), but sometimes the difference is not significant (e.g. Digits). One possible future work could be finding the optimal k with respect to a given loss function, or optimal k that trades-off accuracy for computation. Nonetheless, experimental results show that the SML-K allows to successfully (i.e. outperforms other metric learning algorithms, both in accuracy and running time) consider huge scale problem, by choosing a sufficiently low value of k.

3.4.4 Result for Multi-linear Extension Dissimilarity

Several tests are also done with the multi-linear extension dissimilarity. As shown in Table 3.7, the performance of the multi-linear extension dissimilarity learning algorithm is far from good and almost worse than all the metric learning and ever not better than Euclidean distance. The problem of this dissimilarity may come from the extra transfer function, the no strong constraints, the no stable dissimilarity definition or other reasons.

Virtual Reality Result

We also tried to find the difference between the metric based on the aggregate function and the metric based on the Mahalanobis distance. With virtual reality technology, we have successfully constructed the possibility of directly observing two distances in three-

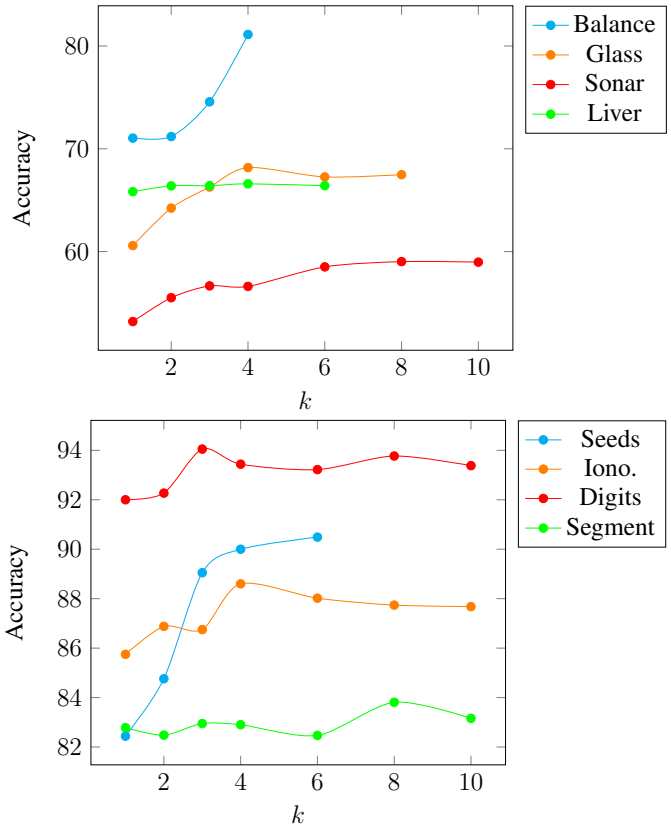


FIGURE 3.6 – Evolution of the classification performance using L_f^k as a function of k-additive constraints, where k is varying from 1 (single feature weighting) to $\min(10, d)$.

dimensional space. After several invited testers observed the embedded 3D projections obtained on the dataset using various metric learning algorithms, most of them pointed out that the scenes with metrics based on the aggregate function were based on the Mahalano ratio. Scenes of various metrics (including Euclidean distances) are notably different. In several cases, Lovasz extended metric scenarios make it easier to classify data points visually. Unfortunately, we don't have enough time to go further and do not expand our research with human vision.

Figure 3.7 and Figure 3.8 give the 3D projection of the embedding obtained with various metric learning algorithms on two 3-classes datasets, Seeds and Balance. As can be observed, our proposition allows better visual discrimination of the classes than existing

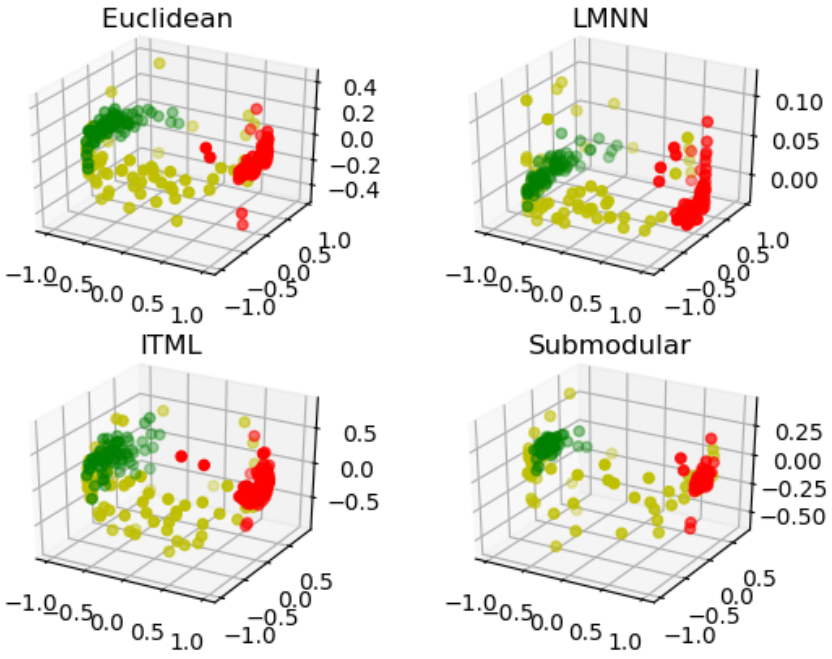


FIGURE 3.7 – 3-dimensional embedding of Seeds dataset using different metrics.

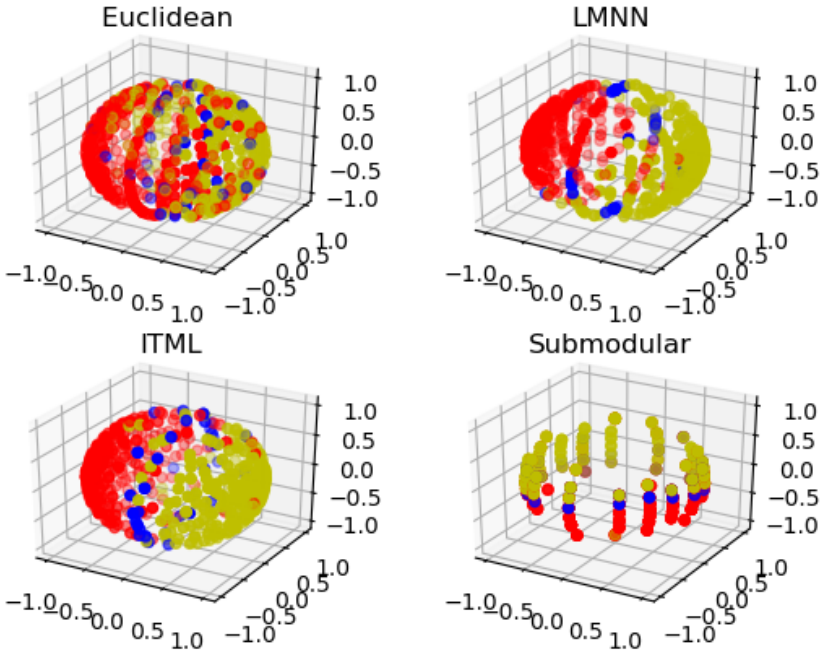


FIGURE 3.8 – 3-dimensional embedding of Balance dataset using different metrics.

approaches.

3.5 Conclusion

In this chapter, we present the first major contribution of this paper : In order to solve the defect that the general Mahalanobis distance-based metric distance algorithm cannot consider high-order dimension information, we introduce the set function to weight all possible combinations of dimensions. Also, a new metric distance is proposed based on the Lovasz extension of the submodular set function. We have also tried other extensions such as multi-linear extensions and defined a similarity that is more compromised under finite conditions. Then we propose a linear programming learning method that learns the Lovasz extended metric and multi-linear similarity, and reduces the computational complexity of the submodular constraints by the k-additive fuzzy measure technique.

In the subsequent design experiments, we tested on multiple real-world datasets that the Lovasz extended metric is superior to the current classical metric learning algorithm and related nonlinear metric learning algorithms in low-dimensional data sets. Unfortunately, the performance of the multi-linear similarity is poor. Moreover, the Lovasz extension metric on the high-latitude database also has limited performance due to high computational complexity.

Although the proposed algorithm has both advantages and disadvantages, it is worthwhile to note that this paper demonstrates the potential for metric learning of high high-order dimension intersection information. In the future, we will further explore this aspect of development, such as improving the learning of metrics through different regularization, and will also consider improving the shortcomings of current algorithms reducing the complexity of algorithms through online learning and other methods.

CHAPITRE 4

METRIC LEARNING FOR NON-FLAT DATASET

The vast majority of metric learning approaches are meant to be applied to data described by feature vectors, and constraints are generated by the target labels or other supervised information. Such a distance is perfectly adapted for flat or iid data but obviously fails to take into account complex and/or (semi-)structured, non-iid data without considering the structured information. In real-world datasets, data can be in many different non-iid forms : for example, string sequences such as text documents or conversation records, time series such as speech or object tracking records, trees such as XML documents or spanning the tree of RNA, graphs such as social networks or web page jump network. These data sets are so-called non-flat datasets.

At the beginning of this work, we hoped that the goal was to propose a general metric suitable for heterogeneous structures for non- flat data. But soon we found many difficulties and challenges. In this chapter, we list a number of different non-planar data and related metric distance learning algorithms that we have investigated. From the similarities and differences of various algorithms and development over time, it can be seen that the metric learning of non-flat data has developed from shallow to deep. So we start with the most complex non-flat datasets, the relational datasets. After investigation showed that there is not much research on relational datasets, we decided a new goal, which is to fill the gap of metric distance learning in relational databases. This is also the motivation for our proposed distance learning learning algorithm in the next chapter.

For so many different structure datasets, several metric learning algorithms have been especially proposed for them in the current years. It is interesting and necessary to apply metric learning theory to non-flat datasets because metric learning has many advantages when dealing with these data sets. One of the advantages of applying metric learning methods to non-iid data is that they can be used as a proxy for any metric-based algorithm that accesses data as if the data consisted of feature vectors without having to manipulate

these complex objects. Another reason for suggesting metric learning for non-flat data sets is that there are already many structural metrics associated with representing structured objects, such as edit distances and alignment indices. Those existing metrics can be learned by metric learning strategies, just like learning metrics from feature vectors.

In this chapter, we will introduce metric learning algorithms that are specific to non-flat data sets, in the order as follows :

— String sequence dataset is one of the most comprehensive non-flat datasets. For measuring the similarity between the samples of the string sequence dataset, there are several metrics such as alignment-based score, character-level metric, edit string distance and vector-space cosine similarity. Most of them are metrics which could be learned with supervised information (Notice that there are also unsupervised or semi-supervised metric learning algorithms with these metrics). In Section 4.1, we will review the definition of these string sequence metrics and summarize the related metric learning algorithms.

— Time Series 4.2

— Graph Network 4.3

We would not discuss the metric learning method for relation dataset, because most of them are similar or based on the metric learning for the network we already mentioned. The algorithms specific to the relation dataset are discussed in more detail in the next chapter.

At the end of this chapter, it is a summary of metric learning methods for the non-flat dataset in Section 4.4, and we again illustrate the motivation to fill the gap in applying metric distance learning algorithms to relational datasets.

4.1 Metric Learning method for String Sequence Dataset

In this section, we will focus on metric learning algorithms which are specific to the string sequence dataset. The string sequence means every sample $\mathbf{x}$ of the dataset is a finite sequence of symbols σ with symbols in the alphabet set $\mathcal{A}$. The task of string sequence metric learning is learning a similarity or distance $d(\mathbf{x}_i, \mathbf{x}_j) = d(\sigma_i, \sigma_j)$.

4.1.1 Metric for String Sequence

For describe the similarity of two string, there are already several metrics. The basic ones are quite simple, such as alignment-based measures (e.g., the Needleman-Wunsch

score [NW70] and the Smith-Waterman score [SW$^+$81]), character-level analysis measures (e.g., the Jaro-Winkler metric [Win90]). Most of them are based on encoding the cost $\ell(\sigma_i, \sigma_j)$ of converting σ_i to σ_j with a linear gap penalty function. Besides these, vector-space cosine similarity [BYRN$^+$99] is a standard measure of a string sequence, which has a similar idea to the previous measures, but it is an extension of the cosine similarity which is between character-level space.

Another standard string sequence metric is the string edit distance. In fact, edit distance is one of the most versatile tools for metric learning algorithms for non-flat datasets. The first edit distance learning algorithm was proposed for string sequences and quickly extended to other structured data sets, such as tree or graph data sets. A standard edit distance [Lev66] $d(\sigma_i, \sigma_j)$, or called Levenshtein edit distance, is defined as the cost $\ell(\sigma_i, \sigma_j)$ of the cheapest edit script which is a sequence of operations that turns σ_i to σ_j. It replaces the encoding penalty function part of the other metrics we mentioned earlier to the insert/delete cost.

4.1.2 String Metric Learning Algorithms

EDL(Edit Distance Learning) [RY98] is the first proposed edit distance learning method. The authors model the edit distance as a memoryless stochastic transducer $\phi =< \delta, \sigma_i, \sigma_j >$ that turns σ_i to σ_j with the joint probability $\rho(\sigma_i, \sigma_j|\phi) = \sum_{\sigma_k:<\sigma_i,\sigma_j>} \rho(\sigma_k|\phi)$. The evaluate and the estimation of the joint probability is performed with a EM processing. The supervised information of training set is encoding in the M-step for maximizing the likelihood probability.

Based on this model, there are many different extension algorithms. For example, the model could be extended with alignment-based measures, such as in ADDSSM(Adaptive duplicate detection using string similarity measures) [BM03] which combine the Needleman-Wunsch score [NW70] and applied it to duplicate detection, or in OAASM(Optimizing amino acid substitution matrices) [SVA06] which combine the Smith-Waterman score [SW$^+$81] and plug it in their local alignment kernel(Notice that the kernel Smith-Waterman score can be optimized by the gradient descent process, which not only brings the advantages of avoiding the need for an iterative process but also brings the disadvantages of non-convex object functions). In BSME(Bayesian similarity model estimation) [Tak09], the author extended the model using Bayesian expectation-maximization techniques to provide better estimation performance. The authors in RFSED(A conditional random field for string edit distance) [MBP12] have similar ideas, but they apply conditional random fields model for

edit sequences instead of Bayesian expectations, and unlike the generic model, positive and negative instances of string pairs are trained in their models. In SED(stochastic edit distance) [OS06], the authors propose to use a conditional sensor as a discriminant model to directly simulate the posterior probability $\rho(\ sigma_j|\ sigma_i)$ using the editing operation to change the input string $\ sigma_i$ to the output. The string $\ sigma_j$. The parameter estimates in this paper are also done using the EM iterative model, where the maximization step is improved on [RY98].

Another group of string metric learning algorithms are based on the vector-space cosine similarity [BYRN+99], which is defined as following :

$$s(\sigma_i, \sigma_j) = \frac{< \sigma_i, \sigma_j >}{\|\sigma_i\|\|\sigma_j\|} \tag{4.1}$$

Notice that the σ_i, σ_j is represented as vectors. There are different variants of the cosine metric. For example the SoftTFIDF [CM06] which define the similarity as

$$s(\sigma_i, \sigma_j) = \frac{sim(\sigma_i, \sigma_j) < w(\sigma_i), w(\sigma_j) >}{\|w(\sigma_i)\|\|w(\sigma_j)\|} \tag{4.2}$$

where the w is a weight value $w(t, \sigma)$ for each token t in string σ based on TF-IDF method and $sim(\sigma_i, \sigma_j)$ is Jaro-Winkler score. In SfNEM(Robust similarity measures for named entities matching) [MYC08], the authors review different extension of SoftTFIDF and proposed a generic model combining the Soft-TFIDF with Levenshtein alignment.

In addition to string metric learning algorithms based on standard sequence string metrics, several new approaches attempt to combine deep learning techniques. For example, in LSSRN(Learning text similarity with siamese recurrent networks) [NVR16], the authors proposed a text similarity learning algorithm based on the Siamese deep recurrent networks. In the Section 2.4, we already introduce the deep metric learning based on the Siamese network frame. This paper is also based on the same framework but replaces CNN with LSTM (long-short-term memory network) [HS97], which has been successful in natural language processing-related tasks and is suitable for sequence learning.

4.2 Metric Learning method for Time Series Dataset

Time series data and its classification and prediction are widely used in machine learning and data mining fields such as bioinformatics, speech recognition, object recognition

and tracking, and patient care.

The field of time data mining involves ordered data streams with temporal interdependence. In the past decade, much interesting time data mining and machine learning algorithm techniques have been proposed and proved to be useful in many applications. These methods include various methods for describing differences in time series dataset objects and can be viewed as similarities of time series entities. In this section, we will introduce those similarities and related metric learning algorithms.

4.2.1 Metric for Time Series

Euclidean distance can still naturally play a role in the measurement of time series data, but it is also the simplest method. For a pair of time series $\mathbf{x}_i = \chi_i$ and $\mathbf{x}_j = \chi_j$ with the length t, the Euclidean metric for them could treat the moments as different feature dimensions as following :

$$d(\mathbf{x}_i, \mathbf{x}_j) = (\sum^{t}(\chi_{i_k} - \chi_{j_k})^2)^{1/2} \tag{4.3}$$

Also, several time-series metrics are the extension of this model, such as STS(Short time-series distance) [MLKCW03] which proposed method for unevenly sampled time series, SVD(singular value decomposition) [WS08b] which decomposes the co-variances matrix of the data and use the eigenvectors as features, CID(Complexity-invariant distance) [BKTDS14] which uses a complexity estimate for information as a correction factor for Euclidean distance and CRD(Compression rate distance) [VA15] which is based on the same idea but replace the complexity estimate with compression rate.

However, for the above time series indicators, most of the hypothetical processing time series are the same length, and even some require uniform sampling. In real-world data sets, data is typically sampled at different lengths by different sensors at different frequencies in different scenarios. Therefore, in order to measure the similarity of these data, it is necessary to align the time series.

Time series alignment

In applications ranging from bioinformatics to audio processing, the problem of aligning time series is ubiquitous. The goal of time series alignment is to align two similar time series with the same global structure but local time differences.

For a pair of time series χ_i and χ_j of length t_i and t_j, their time series alignment can be expressed as a series of indexes :

$$\wp_{ij} = \begin{pmatrix} \wp_i(k) \\ \wp_j(k) \end{pmatrix}, k = 1, 2, \cdots, t_{ij} \tag{4.4}$$

where $\wp_i(k)$ is an index for realigning time series χ_i, $\wp_j(k)$ is the one for χ_j, while the $\wp_i(k)$ of χ_i corresponds to time moment $\wp_j(k)$ of χ_j. The t_{ij} is the length of the time series alignment. Under such an alignment the time series χ_i and χ_j can be realigned to new ones X'_i and X'_j, which have the same length and can be compared directly by the Euclidean distance based on metrics we mentioned before. To ensure that the time series alignment is at least longer than the longest alignment of the two and shorter than the cumulative length of the two, time series alignment needs to follow three constraints as follows :

— Boundary constraint : $\wp(1) = (1, 1)$,$\wp(t_{i,j}) = (t_i, t_j)$;

— Monotonicity constraint :if $\wp(K) = (i, j)$ and $\wp(k + 1) = (i', j')$, then $i' \geq i$ and $j' \geq j$;

— Continuity constraint :if $\wp(K) = (i, j)$ and $\wp(k + 1) = (i', j')$, then $i' \leq i + 1$ and $j' \leq j + 1$;

These constraints are also treated as time series alignment properties. In the following section, several metric learning algorithms treat these constraints as part of the metric learning constraint construction.

Dynamic time warping

One of the most popular time series distances based on time series alignment is DTW (Dynamic Time Warping) [Kru83], which is intended to provide optimal alignment between two temporal sequences and measure their similarities. DTW is widely applied on automatic speech recognition, speaker recognition, online signature recognition or partial shape matching application. DTW is based on the time series alignment and uses a dynamic programming technique to search for an optimal match that has minimal cost while satisfies all the constraints of time series alignment.

DTW warps sequences by nonlinearly stretching or contracting in the time dimension to determine a measure of their similarity, regardless of some nonlinear changes in the time dimension, as shown in Figure 4.1. For calculate the cost of warping, DTW is generated by a cumulative cost matrix C^{DTW}. Given a pair of time series χ_i and χ_j, the cost matrix C^{cost} is constructed according to the time series alignment, where the entry at the index

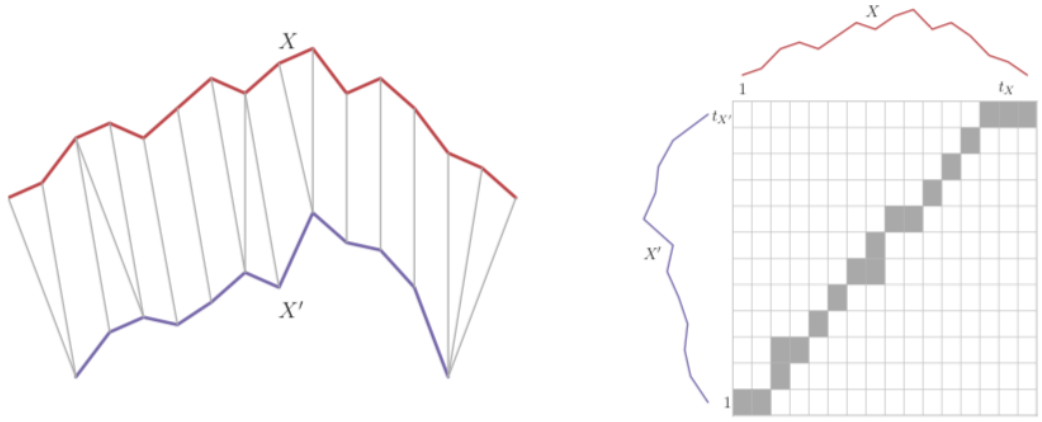


FIGURE 4.1 – Example of dynamic time warping alignment.

(i, j) represents the cost computed by distance of alignment of the time moment i from the series χ_i with j in time χ_j. Than the C^{DTW} will be computed from C^{cost} where each element $C^{DTW}_{i,j}$ represents the minimum cost as :

$$C^{DTW}_{i,j} = C^{cost}_{i,j} + \min(C^{DTW}_{i-1,j-1}, C^{DTW}_{i-1,j}, C^{DTW}_{i,j-1}) \tag{4.5}$$

Where the $C^{DTW}_{1,1} = CC^{cost}_{1,1}$ and $C^{DTW}_{t_i,t_j}$ is the best alignment. The pseudo-code of DTW is detailed in Algorithm 1.

Although DTW can be thought of as a metric of the similar distance between two given time series, it does not guarantee triangular inequalities.

4.2.2 Metric Learning for Temporal Sequence Alignment

As we mentioned before, after time series alignment, the distance between the realigned temporal sequence could be learned as a feature vector. There are several metric learning algorithms for time series are based on temporal sequence alignment.

In LSMLTS(Localized supervised metric learning) [SSHE10], the author specifically applies ANMM (average neighborhood boundary maximization) [WZ07] to the time physiology data. With a statistical time-domain method and a wavelet domain method, the temporal sequence is aligned and represented into a 50-dimensional feature. More details are in the original paper. For learning the metric, the author defines that :

— the local within-class compactness : $\ell(\mathbb{S}) = \sum_{\mathbb{S}} d^2_M(\mathbf{x}_i, \mathbf{x}_j)$ should be low, where

Algorithm 1 Dynamic time warping

Require: Time seriesχ_i with length t_i,time seriesχ_j with length t_j,d()
Ensure: $DTW(\chi_i, \chi_j)$

$DTW(\chi_i, \chi_j) := array(a, b)$
for k from 1 to t_i **do**
  $DTW[k, 0] := \infty$
end for
for l from 1 to t_j **do**
  $DTW[0, l] := \infty$
end for
$DTW[0, 0] := 0$
for k from 1 to t_i **do**
  for l from 1 to t_j **do**
    $cost = d(X_i[k], X_j[l])$
    $DTW[k, l] = cost + mininum($
    $DTW[k-1, l]$, // insertion
    $DTW[k, l-1]$, // deletion
    $DTW[k-1, l-1])$ // match
  end for
end for
return $DTW[a, b]$

for the aligned sequence, the patients of the same label in set $\mathbb{S}$ are close together;

— the between-class scatterness as $\mathbb{D} = \sum_{\mathbb{D}} d_M^2(\mathbf{x}_i, \mathbf{x}_k)$ should be high,where for the aligned sequence, the patients of different labels in set $\mathbb{D}$ are far away from each other.

Therefore, the metric is learned by mininum the ratio between the sum of the distance :

$$\min_M \frac{\sum_{\mathbf{x}_i \in \mathbb{N}} \sum_{\mathbf{x}_j \mathbb{S}} d_M^2(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{\mathbf{x}_i \in \mathbb{N}} \sum_{\mathbf{x}_k \mathbb{D}} d_M^2(\mathbf{x}_i, \mathbf{x}_k)} \tag{4.6}$$

where the set $\mathbb{N}$ is the set for all instance.

In MLTSA(Metric learning for temporal sequence alignment) [GLAB14], the author proposed a similarity measure from annotated examples and presented the approach to learn it and improve the relevance of the alignments.

For a pair $(\mathbf{x}_i, \mathbf{x}_j)$ of training instances (A, B), the author define the binary matrix $Y \in \{0,1\}^{T_A \times T_B}$, $Y_{(\mathbf{a}_i, \mathbf{b}_j)=1}$ for every $i \in 1, ..., u$ and 0 otherwise, u is the size of the match of (A, B). The metric of pairs is based on Mahalanobis metric, denoted as :

$$d_M(\mathbf{x}_i, \mathbf{x}_j) = -((\mathbf{a}_i - \mathbf{b}_j))^T M((\mathbf{a}_i - \mathbf{b}_j)) \tag{4.7}$$

They aims finding a matrix M , which makes the predicted alignments close to the ground-truth on the training instance.

The loss function they used is the same idea of LMNN, while the loss is not only the hamming loss but also area loss.

4.2.3 Metric Learning for Dynamic Time Warping

DTW can be thought of as a measure of time series (Notice that it is not an accurate measure because it does not follow the triangular inequality property) and it is one of the most popular ways to measure the similarity of time series data. Therefore, several metric learning algorithms learn DTW-based metrics or apply DTW as regularization.

ITML and LMNN are the most classic metric learning algorithms, so in the past few years, many scholars have proposed their DTW extensions. These algorithms are based on an extension of DTW, which is called MDDTW(Mahalanobis distance Dynamic Time Warping) [MLWG15]. For traditional DTW we mentioned before, the local distance function $d(.,.)$ for generate the cost matrix C^{cost} is Euclidean distance. The MDDTW replace the Euclidean distance function to the Mahalanobis distance function with the parameter

matrix M. There are several advantages when applying MDDTW instead of the traditional DTW to measure the similarity of time series. First is with this form, it will be easy to study the corresponding metric learning algorithm based on the Mahalanobis distance. Secondly, the stretching or contraction of the time series is along the time axis as a whole rather than independent. Finally, a good Mahalanobis distance will rebuild a relationship among variables accurately which could be treated the learned parameter matrix M give weight to dimensions of features.

LG-DTW(LogDet-Dynamic Time Warping Learning) [MLWG15] proposed the MDDTW and was designed to learn a Mahalanobis distance under DTW with the same LogDet regularizer as ITML. It could be veiw as a DTW version of ITML. The author choose the relative constraints and the loss function is :

$$L(M) = \sum_{(i,j,k)\in\mathcal{C}} (DTW_M(\mathbf{x}_i, \mathbf{x}_j) - DTW_M(\mathbf{x}_i, \mathbf{x}_k) + \gamma) + \lambda(tr(M) - logdet(M)) \quad (4.8)$$

which is similar with ITML. The problem for optimizing the loss function is similar with the MDDTW, the computation of DTW is also based on the learned Mahalanobis distance and will be changed when the metric is changed, which lead to NP-hard for optimizing. The author propose an iterative closed-form approach for solving the problem, with the object function as :

$$M_{t+1} = \arg\min_{(i,j,k)\in\mathcal{C}} (DTW_M(\mathbf{x}_i, \mathbf{x}_j) - DTW_M(\mathbf{x}_i, \mathbf{x}_k) + \gamma) + \lambda_t D_{logdet}(M, M_t)) \quad (4.9)$$

with the trade-off parameter λ_t and LogDet regularizer $D_{logdet}(M, M_t)$ keeping the metric matrix M close to the one in the previous iteration.

metricDTW(local distance metric learning in Dynamic Time Warping) [ZXI16] proposed point-wise DTW, which is actually the same as MDDTW and an adaptation of LMNN for univariate time series k-NN classification. The authors say they learned the local metrics for $M_{i,j}$ and $M_{i,k}$ in different clusters of different time series and tried to keep $M_{i,j} \equiv M_{i,k}$ for global metric learning. However, they simplify the matrix by constraining M with only a single weight value on the diagonal, which results in eventually they learn only a scalar rather than a metric.

DTW-LMNN [SHZL17] is another DTW verion of LMNN for the multivariate time

series case. The loss function is more similar to the LMNN than the previous one and as following :

$$L(M) = \sum_{(i,j,k)\in\mathcal{C}} (DTW_M(\mathbf{x}_i, \mathbf{x}_j) - DTW_M(\mathbf{x}_i, \mathbf{x}_k) + \gamma) + \lambda \sum_{(i,j)\in\mathcal{S}} DTW_M(\mathbf{x}_i, \mathbf{x}_j) \quad (4.10)$$

It has the same NP-hard problem like the LG-DTW and also solves the problem with the iterative approach, which first computes the DTW with a fixed metric then computes the metric with fixed DTW. The computation complexity of DTW-LMNN is higher than LG-DTW, which lead to slower and lower performance.

In addition to these MDDTW-based metric learning algorithms for learning DTW distances, other scholars combine metric learning with DTW in other ways. For example, in DTWCL(Dynamic time warping constraint learning) [YYH$^+$11], the author applies LMNN to adapt to global path constraints, which improve the performance of alignment constraints construction for DTW.

4.3 Metric Learning method for Tree and Graph Dataset

Many non-iid data sets are structurally like trees or graphs. The proliferation of social networks on the Internet has spurred the creation of more tree or graph structures datasets. If we survey the machine learning method in the past few years, there are two types of algorithms worth paying attention to : one is the graph learning using the edit distance method[GXTL10] we mentioned in Section 4.1, and the other is the graph learning using the embedded method[GF18]. Logically, for tree and graph datasets, there are two important sets of metric learning methods. One is based on edit distance. Most of them are similar to the proposed string edit distance metric learning. Another set of metric learning methods focuses on embedding the structural information of trees and groups. Most related metric learning methods are to use the existing embedding method to represent structural information as the similarity of trees and groups, or to combine existing embedding methods with the prepossessing. The related embedding methods included the graph factorization[BN02], random walk[PARS14], deep learning [KW16] and miscellaneous[TQW$^+$15]. In this work, we will not introduce all related algorithms, but will introduce several most important ones.

4.3.1 Metric Learning with Edit Distance Method

As we mentioned before, edit distance is widely used in the string or tree-structured data. After many scholars proposed the metric learning algorithm for string editing distance, they expanded it into a tree or graph version of distant learning algorithm based on themselves or others.

Extending the work of EDL [RY98] and SED [OS06] on string edit similarity learning, Learning STED(stochastic tree edit distance) [BHS06]and PMTED(Learning probabilistic models of tree edit distance)[BBHS08] propose both a generative and a discriminative model for learning tree edit costs. The author points out the problem of classical tree edit distance, which hard to tune the fixed priory edit costs. To overcome this drawback, they focus on the automatic learning of a non-parametric stochastic tree ED. They update the EM algorithm of classical tree edit distance for learning the primitive edit costs with the conditional adaptation in [Sel77] which it is cheaper to compute.

The work of MLbTSD(Learning Metrics Between Tree Structured Data) [BHS07] is also based on the tree edit distance in EDL [RY98], but it has a more complicated variant, which not only allows the operation of a single node but also allows the subtree. The estimation of parameters is also based on the EM algorithm. However, this method has a theoretical limitation, that is, factoring in some cases may not be correct, as [Emm12] point out. In OSTD(On stochastic tree distances) [Emm12], the authors show that a correct factorization is achieved only when the most probable editing script is considered instead of all possible scripts. According to this theory, corresponding EM updates are derived.

In GESL(Good edit similarity learning by loss minimization) [BHS12], the authors propose good edit similarity learning by loss minimization, which is deal with strings, and extend it to trees and lead to robust and competitive similarities in LDTDS(Learning discriminative tree edit similarities for linear classification) [BBHS16].

The tree edit distance (TED) $d_{TED}(\varpi_i, \varpi_j)$ between two trees ϖ_i and ϖ_j is the minimum number of edit operations to change ϖ_i into ϖ_j. Based on TED, the authors define the edit similarity function as :

$$d_{C^{cost}}(\varpi_i, \varpi_j) = 2e^{-d_{TED}(\varpi_i, \varpi_j)} - 1 \tag{4.11}$$

where C^{cost} is the positive cost matrix defining the possible edit operations over nodes of trees. The algorithm GESL could optimize this similarity while satisfying several conditions.

4.3.2 Metric Learning with Embedding Structure Information Method

In SPML(Structure Preserving Metric Learning) [SHJ11], the authors propose a metric learning algorithm based on the adjacency matrix A of a network. They learn a Mahalanobis distance metric defined by matrix M transformation with structure preserving embedding, which is more effective on the inherent connectivity structure of the network and show as :

$$d_M(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T M \mathbf{x}_i + \mathbf{x}_j^T M \mathbf{x}_j - \mathbf{x}_i^T M \mathbf{x}_j - \mathbf{x}_j^T M \mathbf{x}_i \tag{4.12}$$

The authors use as supervised constraints the fact that the distances to all disconnected nodes $\mathbf{x}_j$ must be larger than the distance to the farthest connected neighbour of all neighbour nodes x_l. However, not every connected neighbour is considered. As an alternative solution, they propose to set an additional input parameter K and only visited K connected neighbour nodes x_l. The associated constraint on the distance is thus given by

$$d_M^2(\mathbf{x}_i, \mathbf{x}_j) > (1 - A_{ij}) \max_l (A_{il} d_M^2(\mathbf{x}_i, x_l)), \forall i, j \tag{4.13}$$

The objective function of this approach, denoted as SPML, is :

$$\begin{aligned} &L(M) = \tfrac{\lambda}{2}\|M\|^2 + \\ &\tfrac{1}{|\mathbb{C}|} \textstyle\sum_{(i,j,k)\in\mathbb{C}} max(d_M^2(\mathbf{x}_i, \mathbf{x}_j) - d_M^2(\mathbf{x}_i, \mathbf{x}_k) + 1, 0), \end{aligned} \tag{4.14}$$

where $\mathbb{C}$ is a set of triplets (i, j, k), for which $A_{ij} = 1$ and $A_{ik} = 0$. Using a convenient notation, one can rewrite the difference between the two distances in the loss function with a sparse matrix C as follows

$$d_M^2(\mathbf{x}_i, \mathbf{x}_j) - d_M^2(\mathbf{x}_i, \mathbf{x}_k) = C^{(i,j,k)} X^T M X,$$

where C is a sparse matrix storing the parameters $C_{jj}^{(i,j,k)} = 1$, $C_{ik}^{(i,j,k)} = 1$, $C_{ki}^{(i,j,k)} = 1$, $C_{kk}^{(i,j,k)} = -1$, $C_{ij}^{(i,j,k)} = -1$ and $C_{ji}^{(i,j,k)} = -1$. Otherwise $C^{(i,j,k)} = 0$.

Finally, they use a projected stochastic sub-gradient descent to find a solution of the minimization of loss function $L(M)$.

With the development of deep learning today, there are several algorithms combining

deep learning method to deal with the graph datasets.

In SRLR(Demystifying relational latent representations)[DB17], the authors point out that although the latent space generated from deep learning is not well interpretable, the clustering performed in this space can show the re-presentation of features. They construct and benefit from local metrics in latent space.

In BCNML(Brain Connectivity Networks Metric Learning) [KPF+18], the authors propose a new graphical re-presentation method for solving functional brain connections problem. The idea is similar to the [CHL05] mentioned in Chapter 2 which learn a metric with "Siamese network". The progress of this paper is that the authors use a siamese graph convolutional neural network to learn a graph similarity metric. This paper is the first algorithm to apply GCN (graph convolution neural network) [KW16] to metric learning method.

GCN is a natural generalization of convolution neural network model in the graph structure. The convolutional neural network is based on local perception region, shared weights and downsampling in the spatial domain because it has stable and invariable characteristics and can extract spatial features of images well. Graph structure does not have the translation invariance of the image, and the traditional convolution method is not suitable for graph structure. The essence of GCN is to extract the structural features of graphs by defining the receptive field, for example, spectral approach processes graph structure by eigenvalues and eigenvectors of the Laplacian matrix of the graph.

The Figure 4.2 shows the structure of graph convolution neural network. For each layer of GCN, the forward function is as following :

$$H_{i+1} = f(H_i, A) = f_{act}(AH_iW_i) \tag{4.15}$$

where $f_{act}()$ is an active function(generally select the ReLU nonlinear active function), A is the processing adjective matrix, W_i is the parameter of the layer H_i.

In BCNML [KPF+18], the authors transfer the training set to the bipartite graph as the training pairs. The inputs of siamese GCN are the training pairs, and the outputs are combined by an inner product layer followed by a single fully connected as the similarity estimate. The model is driven by the hinge loss of similar pairs (matching graphs) and dissimilar pairs (non-matching graphs).

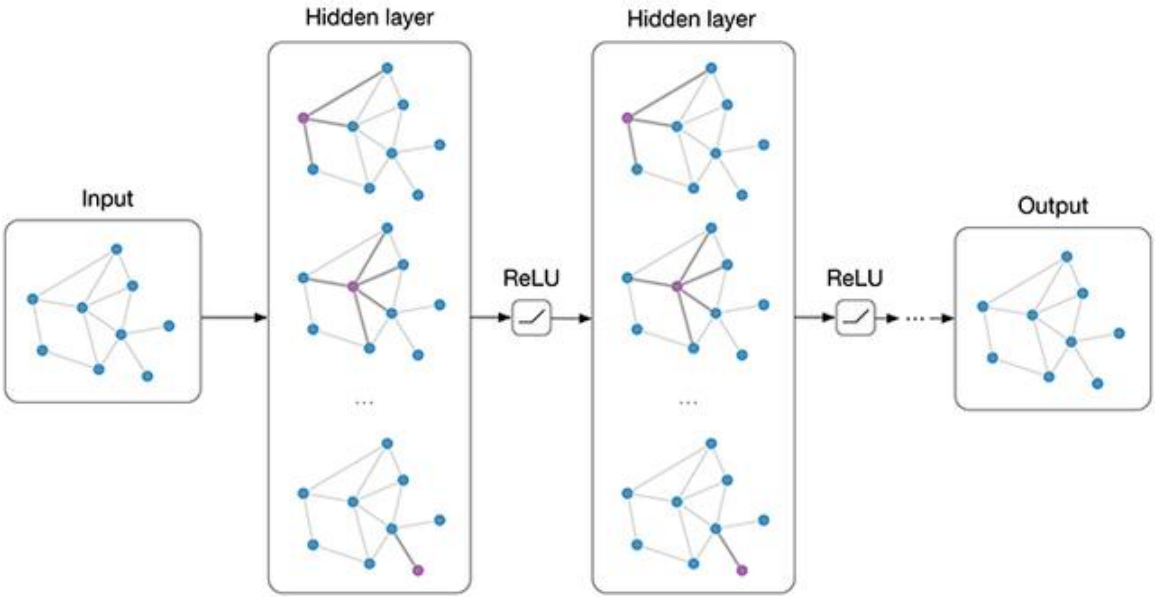


FIGURE 4.2 – The structure of graph convolution neural network.[KW16]

4.4 Conclusion

In this chapter, we mainly introduce the development and current status of metric distance learning in non-flat databases from three aspects : string database, time-series database and graph database. As can be seen from the summary, most non-flat metric distance learning algorithms are based on specific non-flat metrics, such as edit distance and dynamic timing alignment. With the development of deep learning today, several algorithms like SRLR and BCNML use the feature learning nature of deep learning to deal with non-flat databases directly.

Name	Year	Structure type	Structure method	Additional Information
EDL[RY98]	1998	String	Edit Distance	The first proposed edit distance learning method
ADDSSM[BM03]	2003	String	Edit Distance	Needleman-Wunsch score [NW70]
OAASM[SVA06]	2006	String	Edit Distance	Smith-Waterman score [SW+81]
SED[OS06]	2006	String	Edit Distance	Online learning
BSME[Tak09]	2009	String	Edit Distance	Bayesian expectation-maximization techniques
RFSED[MBP12]	2012	String	Edit Distance	A conditional random field for Bayesian expectatio
SfNEM[MYC08]	2008	String	Vector-space Cosine Similarity	SoftTFIDF [CM06]
LSSRN[NVR16]	2016	String	Vector-space Cosine Similarity	LSTM[HS97]
GESL[BHS12]	2012	String	Edit Similarity	Edit similarity learning but not proof to be a distance
LSMLTS[SSHE10]	2010	Time Series	Temporal Sequence Alignment	ANMM[WZ07]
MLTS[GLAB14]	2014	Time Series	Temporal Sequence Alignment	Extension of LMNN on Time Series
DTWCL[YYH+11]	2011	Time Series	Dynamic Time Warping	Select the constarints for metric learning with DTW
MDDTW[MLWG15]	2015	Time Series	Dynamic Time Warping	Extension of LMNN and ITML on Time Series
LG-DTW[MLWG15]	2015	Time Series	Dynamic Time Warping	Same LogDet regularizer as ITML
metricDTW[ZXI16]	2016	Time Series	Dynamic Time Warping	point-wise DTW
DTW-LMNN [SHZL17]	2017	Time Series	Dynamic Time Warping	Extension of LMNN on Time Series
STED[BHS06]	2006	Tree	Edit Distance	Extending the work of EDL [RY98]and SED [OS06]
MLbTSD[BHS07]	2007	Tree	Edit Distance	Extending the work of EDL [RY98] but also allows the subtree
PMTED[BBHS08]	2008	Graph	Edit Distance	Extending the work of STED[BHS06]with PRM
OSTD[Emm12]	2010	Tree	Edit Distance	Online learning
LDTDS[BBHS16]	2016	Tree	Edit Similarity	Extending the work of GESL[BHS12]
SPML[SHJ11]	2011	Graph	Embedding Structure Information	Embedding structure information in metric and constraints selection
SRLR[DB17]	2017	Graph	Embedding Structure Information	Learning the metric in the latent space generated by deep learning
BCNML[KPF+18]	2018	Graph	Embedding Structure Information	Learning the metric with GCN [KW16]

TABLE 4.1 – Survey on Non-flat Metric Learning Algorithms

In the Table 2.2, we collated all the metric distance learning algorithms introduced in this chapter. By comparing these algorithms, we can see that the metric distance learning algorithm for non-flat databases inherits various models and techniques of flat database metric distance learning algorithms on the one hand, and develops unique means of handling for many unique structures on the other hand. In many cases, prepossessing of non-flat data (such as timing alignment of time series databases) can be not only as optimization of the feature learning properties of the metric distance learning algorithm but also as a pre-preparation for subsequent metric distance learning. This duality reflects the characteristic of metric distance learning as a re-presentation of learning.

After introducing various non-flat data and related metric distance learning algorithms, we can see that with the development of time, non-flat metric distance learning algorithms are more and more focused on more complex and more realistic databases. The relational databases not only have the complex topology of the graph-like database but also have additional feature information such as the relationship between the tables and the side information. There are not many metric distance learning algorithms explicitly developed for relational databases, but they are worth exploring. In order to fill this gap and further develop based on the prior relational metric learning algorithm, in the next chapter, we propose several metric learning algorithms for relational databases.

CHAPITRE 5

RELATIONAL METRIC LEARNING

As mentioned in the previous Chapter 4, most metric learning methods are specific to applications in flat databases ; that is, feature vectors primarily describe samples. However, non-flat databases outside the column, such as strings, time series, trees or graphs, are still worth noting. Although many real-world datasets present multiple relationships between observations, such as social service networks, Wikipedia networks, molecular biology classifications, etc., most machine learning algorithms focus on only partially vectorized characterizations. Table plane database. These databases still contain a large amount of potential data for the association table between the entity table and the entity. It is also worth developing a specialized machine learning algorithm. Although some computer learning has been tried in some machine learning, such as [Get07], as far as we know, the metric distance learning algorithm that is specially processed for such databases still has little attention. The primary purpose of this chapter is to propose several metric distance learning algorithms that individually consider relational databases.

In this Chapter 5, we introduce the concept and classical approaches of relational learning in Section 5.1. Then in the following sections, there are three ways proposed to process different relational data :

The goal to propose a metric learning algorithm that can be applied to a relational database with multiple entity tables and multiple relationships between entity tables. To achieve this goal, we have tried three ideas :

— Propose a metric based on relational information for relational data : This method is inspired by edit distance, but the difficulty lies in the representation of the relationship structure. In this article, we have not proposed a proper way.

— Select the constraints based on relational information : The performance of metric learning algorithms is highly dependent on constraints selection. In Section 5.2, we proposed the LSCS(Relational Link-strength Constraints Selection) algorithm. A general model is denoted as link-strength constraints which are generated by a link-strength function. LSCS is suitable to the relational data which the structure

of the relationship graph is complex because the link-strength function transfers the side information of the relationships between entities to a weight value measuring the relational similarity between entities. This proposed method was published on Conférence sur L'Apprentissage Automatique in 2018 [PLCL] and the Eighth International Workshop on Statistical Relational AI at the 27th International Joint Conference on Artificial Intelligence in 2018 [PCL18].

— Combine the existing relational learning algorithms and metric learning algorithms : There are several relational learning algorithms that are suitable for combining with metric learning algorithms. We chose the RESCAL decomposition based on the relational tensor. The proposed RFML(RESCAL Factorization Metric Learning) is apply the exist metric learning algorithms on a union space which is the combine of RESCAL Factorization latent space and original feature space. This is a simple basic application of metric learning algorithms, so in this article, we propose this method mainly as a baseline for comparing other algorithms. This part will be presented in Section 5.3

The third proposed relational metric learning algorithm is MRML(Multi-Relational Metric Learning) in Section 5.4. MRML is based on both "Select the constraints based on relational information" and "Combine the existing relational learning algorithms and metric learning algorithms" ideas, which sums the loss from relationship constraints and label constraints. MRML uses both label constraints and relational constraints. This proposed method was published on a workshop of 5th International Conference on Learning Representations[PCL18] in 2017 and published on 27th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning[PLC19a] in 2019.

Note that, the LSLC is proposed for the datasets with one mainly entities table (or several table in one type) with multi-relation reference table from the parents entities table (one or more). The RFML and MRML are proposed for the datasets with only one entities table with multi-relation links between the entities. In experiment Section 5.5, we will discuss the difference again.

In the end, there are experiments designed for the different dataset and comparing algorithms, and results shown in the Section 5.5.

At the end of this chapter, in the conclusion Section 5.6, we re-synthesized and collated the main contributions of this chapter. We also listed some of the problems encountered in the study, describing the studies and experiments that we tried but did not go deep because of time, equipment or other constraints.

5.1 Relational Learning and Metric Learning

As we mentioned in Chapter 1, Relational learning deals with learning models for which data consists of a generally complex relational structure. As a difference with flat datasets, the primary learning tasks pay more attention to supervised information from the relations and knowledge from the topology of the relational graph, such as collective classification, or link prediction. Relationship learning can be divided into different branches depending on the model and method as we introduced in Chapter 1, including statistical relational learning, inductive logic programming, relational reinforcement learning, graph mining and multi-relational mining. In our work, we focus on relational learning based on graph-based mining and multi-relational mining. In Chapter 4, we have introduced several metric distance learning methods for graph databases, many of which can be extended to relational databases. For example, our proposed MRML is inspired by SPML(Structure-Preserving Metric Learning) [SHJ11] and extends it from a single un-weighted graph to a set of the weighted graph representing multiple relationships.

Notice that, although there are many machine learning tasks related to relational learning, we specifically focus on using relational information for collective classification. Collective classification is a classification of related entities that may share identical classes [SNB$^+$08] from the relationship information. For collective classification, there are several approaches to learn metric with relational information.

In DRLR(Demystifying Relational Latent Representations) [DB17], the authors propose an analysis of the meaning of the latent space learned by a deep learning algorithm on relational datasets, mainly because of the black box problem of deep approaches. The same analysis can be conducted for the meaning of metric learning, which can also be treated as mapping original feature space to a latent space. This work mainly focuses on the usefulness of latent space and the redundancy of the features. They show the excellent performance of using unsupervised relational information for a classifier in a latent space.

In [KLF01], they use relational graph information with propositionalization, transfer the relational representation of a learning problem into a propositional representation. Another approach uses metric learning on graphs for domain adaptation [DTC12]. To this aim, they propose an iterative learning algorithm on the graph. From the resource domain, the nodes with labels, they learn a new metric and apply it on the related target domain, the nodes without labels. Then, the graph is updated depending on the learned distance, and constraints with low entropy instances are selected for the next iteration.

Some metric learning algorithms consider relational data as heterogeneous networks for each different relationships. For instance, in [ZPX13], they propose a heterogeneous metric learning algorithm, which integrates the structure of different graphs into a joint graph regularization. They use two mapping function for the feature space of the object entities and subject entities in one relation, and then introduce a joint graph regularization for iterative optimize the loss function. In [DCS17], they start from the same principle but use meta-path-based random walks to incorporate the heterogeneous network structures into skip-gram vectors for dealing with the relational graph. Those algorithms use joint regularization for different entities in heterogeneous networks, which has an excellent performance on considering the structure information in the relational dataset but do not consider the side information in the relational links. It processes the relational variables the same way as the entities and subject to their algorithm, but it ignores the differences between entity tables and association tables. Our proposed method includes the value of different variables on the relationship in the datasets and distinguishes them with entities.

5.2 Relational Link-strength Constraints Selection

The first proposed approach LSCS(Relational Link-strength Constraints Selection) starts with constraints selection. We got inspiration from approaches in [DTC12], which is metric learning method for (hyper)graph data. Instead of using label constraints, the authors used graph-based constraints which enforce the distance between the unlinked node and target node to be bigger than the distance between the k-farthest linked node and the target node.

We propose a more general model denoted as link-strength constraints which are generated by a link-strength function measuring the similarity of nodes by the side information of the relationships between them. The proposed link-strength function gives the possibility of using similarity learning to encode the relational information into the constraints for the metric learning algorithm.

We have tried to define a metric based on the link strength function, but this will result in either sacrificing the consideration of non-numerical side information or making it difficult to prove that it is a metric and can only be counted as similarity.

5.2.1 Link-strength Function

A basic statement is two connected individuals are more similar than two unconnected individuals. We evaluate the amount of link similarity between individuals by proposing a link-strength function to measure that.

A link-strength function $LS(\mathbf{x}_i, \mathbf{x}_j|\mathbf{r})$ is a function with the input is the relational information between the entities nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ and the output is a real value to measure the "strength" or the similarity of the two nodes. The symbol $\mathbf{r}$ represents the related information of the entities nodes $\mathbf{x} \in R^{n \times m}$, where n is the number of instances and m the number of node attributes.

There are many ways to encode the relational information for this link-strength function, and we choose that the link-strength depends on the side-information of the common parents, which are the values of the references.

A relational schema $\mathbb{R} = \mathbb{R}_e \cup \mathbb{R}_r$ contains a set of relational information where $\mathbb{R}_r$ denotes the set of relational information(like groups and types) between entities in same tables and $\mathbb{R}_r$ denotes the set of relational information(like reference links) between different tables. For a relation subset $\mathbf{r}_k \subseteq \mathbb{R}_r$ including all references between two entity tables, for examples like the reference Ratings between User and Movie as shown in Figure 5.1, we consider it as a many-to-many relationship. Let $\mathbb{P}_{ij}$ be the set of common parents of $\mathbf{x}_i$ and $\mathbf{x}_j$, and $num_{\mathbb{P}_{ij}} = |\mathbb{P}_{ij}|$, the number of common parents.

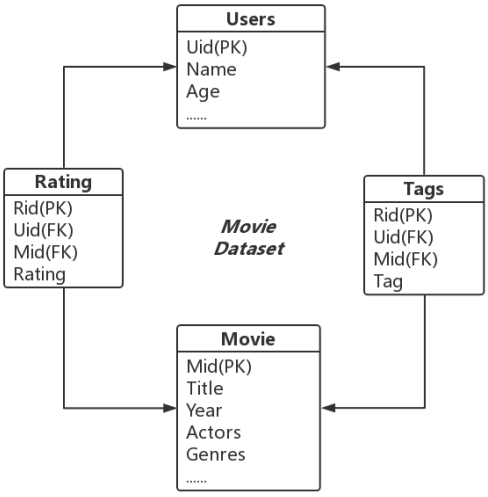


FIGURE 5.1 – Data structure of movie datasets.

In Figure 5.2, we give a sub-sample of a bipartite relational graph, along with an example of common parents.

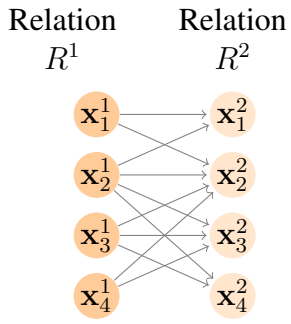


FIGURE 5.2 – Bipartite relational graph for a *many-to-many* relationship table. The common parents of $\{\mathbf{x}_2^2, \mathbf{x}_3^2\}$ is the set of entities $\mathbb{P}_{\mathbf{x}_2^2,\mathbf{x}_3^2} = \{\mathbf{x}_2^1, \mathbf{x}_3^1, \mathbf{x}_4^1\}$

Naturally, a similar node would get similar references from the same parents nodes. Consequently, we consider that the link-strength depends on the side-information of the common parents, which are the values of the references. In the Figure 5.1 is a relational dataset with two entities tables(Users and Movie) and two relational tables (Rating and Tags). The Figure 5.3 shows the common parents of two target data nodes in the Movie table.

Additionally, as shown in Figure 5.3 the references can be quantified by num^n numerical variables var^n (the red link) and num^c categorical variables var^c(the blue link).

Given a relation $\mathbf{r}_k$, we propose to defined the link-strength function $LS()$ as :

$$\begin{aligned} LS(\mathbf{x}_i, \mathbf{x}_j|\mathbf{r}_k) &= LS(\mathbf{x}_i, \mathbf{x}_j|\mathbb{P}_{ij}) \\ &= \sum_{l=1}^{num_{\mathbb{P}_{ij}}} \Big(\gamma \cdot f^a(l,i,j) + (1-\gamma) \cdot f^b(l,i,j)\Big) \end{aligned} \tag{5.1}$$

where

$$f^a(l,i,j) = \sum_{m=1}^{num^n} \exp\left(-\left|var_m^n(\mathbf{p}_l, \mathbf{x}_i) - var_m^n(\mathbf{p}_l, \mathbf{x}_j)\right|\right), \tag{5.2}$$

and

$$f^b(l,i,j) = \sum_{m=1}^{num^c} \left(var_m^c(\mathbf{p}_l, \mathbf{x}_i) \circ var_m^c(\mathbf{p}_l, \mathbf{x}_j)\right), \tag{5.3}$$

in which $x \circ x' = 1$ iff $x = x'$, and 0 otherwise, and $\mathbf{p}_l$ is the l-th parent node in the set $\mathbb{P}_{ij}$ for relation $\mathbf{r}_k$. Notice that numerical association attributes v are normalized in the

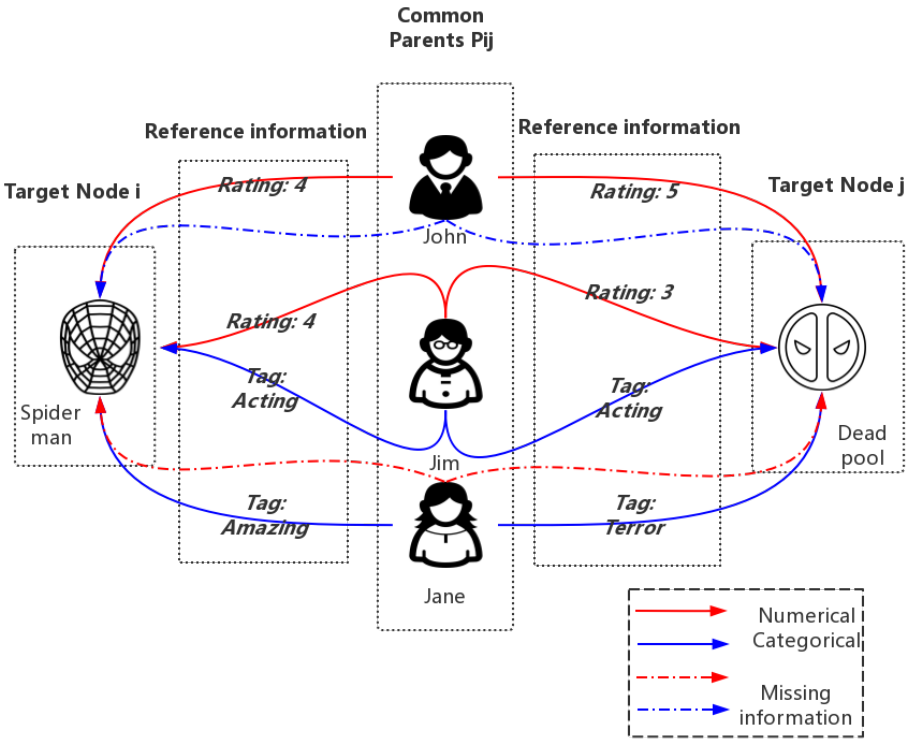


FIGURE 5.3 – The common parents of movie "Spider man" and "Dead pool".

unit interval before link strength computation and we use the exponential function rather than other kernel function because we pay more attention to the change from the difference close to 0 while the gradient of the exponential function changes more when approaching 0. Note also that we restrict to unit-length slot chains, i.e. the length of the sequence of foreign key references is equal to 1. We have not done more normalization for other factors yet but it maybe interesting in the future work.

5.2.2 Link-strength Constraints Selection

For relational constraints selection, the simplest way is the relative link constraints $d_M(\mathbf{x}_i, \mathbf{x}_j) \leq d_M(\mathbf{x}_i, \mathbf{x}_k) + m, (i,j) \in \mathbb{S}, (i,k) \in \mathbb{D}$ with the adjacent matrix A that $A_{ij} = 1, A_{ik} = 0, \forall (i,j) \in \mathbb{S}, \forall (i,k) \in \mathbb{D}$, which only check the relative distance between the connected and disconnected relational links.

We extend the relative link constraints by separating $\mathbb{S}$ and $\mathbb{D}$ with the dedicated link-

strength function as the above subsection. The proposed constraints enhance the classical metric learning algorithms using relative constraints, such as ITML [DKJ+07] and LSML [LGZ+12].

We select the strongest links as similarity constraints and the weakest links as dissimilarity constraints. Remark that if two entities $\mathbf{x}_i$ and $\mathbf{x}_j$ do not have common parents, their link strength is zero, and therefore considered as dissimilar. With the link-strength function, we select the relative constraints set $\mathbb{C} = \{(i, j, k) : LS(\mathbf{x}_i, \mathbf{x}_j) \geq LS(\mathbf{x}_i, \mathbf{x}_k)\}$ and use the constraints on two different classical metric learning algorithms, ITML and LSML.

5.3 Metric Learning with RESCAL Factorization

The link-strength constraints algorithm in Section 5.2 is good at the process the datasets with complex structure. However, in many real-world datasets, the primary information for the relational dataset is not the structure or topology information, but the multi-relational links between the entities. For example, the facebook focuses on the following link or graphing relation between users, and the parties system focus the vote and support relation between party members. Typically, this kind of relational dataset only contains one entities table with multi-relationship between them. For this particular kind of multi-relational datasets, we considered the relational tensor and tensor factorization and proposed RFML(Metric Learning with RESCAL Factorization).

5.3.1 Relational Tensor

In [Dže10], the authors present different relational frameworks, primarily first-order logic, relational database model and set theory. Relational tensor is one of them, which is a mathematical definition combining n_r-tuples and set theory. In particular, relational tensor rely on the use of n_r-array relations defined as a set of n_r-tuples. Let $\times$ denote the Cartesian product of sets. An n_r-array relation $\mathbb{R}$ is then defined as a subset of the Cartesian product of n_r sets [FGKP99], as follows :

$$\begin{aligned} \mathbb{R} &\subseteq \mathbb{V}_1 \times \cdots \times \mathbb{V}_{n_r} \\ \mathbb{V}_1 \times \cdots \times \mathbb{V}_{n_r} &= \{(v_1, \cdots, v_{n_r}) | v_1 \in \mathbb{V}_1 \wedge \cdots \wedge v_{n_r} \in \mathbb{V}_{n_r}\} \end{aligned} . \tag{5.4}$$

The domain of $\mathbb{R}$, $dom(\mathbb{R})$, which also denotes the Cartesian product $\mathbb{V}_1 \times \cdots \times \mathbb{V}_{n_r}$,

is the set of all possible relationships over the entities in their domains. For a set $\mathbb{X}$ and a subset $\mathbb{X}_{sub} \subseteq \mathbb{X}$, the characteristic function of $\mathbb{X}_{sub}$ is a boolean-valued function $f_{cha|\mathbb{X}_{sub}} : \mathbb{X} \to \{0, 1\}$ which indicates for all elements in $\mathbb{X}$, whether they are also an element of the subset $\mathbb{X}_{sub}$ [Hal98]

$$\forall x \in \mathbb{X} : f_{cha|\mathbb{X}_{sub}})(\mathbb{X}) = \begin{cases} 1 & \text{if } x \in X_{sub} \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

For a relation R, its characteristic function is a function

$$f_{cha|\mathbb{R}} : \mathbb{V}_1 \times \cdots \times \mathbb{V}_n \to \{0, 1\} = \begin{cases} 1 & \text{if a relation exists} \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

This work focuses on dyadic (binary-valued relations) relational data, where $R_k \subseteq \mathbb{V}_i \times \mathbb{V}_j$ for all relations R_k. Similarly to the RDF convention [AVH04], a relationship $R_k(a, b)$ refers to a as the subject and to b as the object of the relationship. Relational learning domain shows strong links with learning the characteristic function of relation from supervised information, in particular, to predict the existence of a relation between two individuals.

A relational tensor is a tensor which stores the relationships of relational data with the characteristic function $f_{cha|\mathbb{R}}$. For dyadic relational data modelling, we use a labelled directed graph, in which entities are nodes and relationships are labelled directed edges pointing from the subject to the object. The relational tensor then consists of the union of the characteristic function of the relations.

A relational tensor T with n entities and n_r different relations can be written as $T \in \mathbb{R}^{n \times n \times n_r}$:

$$t_{ijr} = \begin{cases} 1 & \text{if } \diagdown(i, j)exist \\ 0 & \text{otherwise.} \end{cases} \tag{5.7}$$

As shown in Figure 5.4, a third order tensor, which contains the characterizing function of relationships between entities is used. All of the proposed relational learning process in this work are based on this relational tensor. Notice that we restrict to a binary tensor (i.e. presence or absence of relation) in this work, but relations can be valued (e.g. rating of a

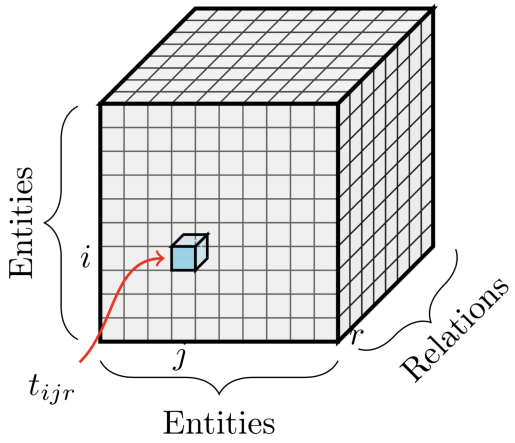


FIGURE 5.4 – Relational tensor.

movie by a user), or vector-valued (e.g. ratings of actors in a movie).

Several approaches using relational tensors have been proposed for relational learning. In [ME11], focusing on existing link prediction models, the authors extend matrix factorization to use the side information and overcome the imbalance. Tucker Decomposition (TD) model is used on user-tag-item relational tensor to provide high-quality tag recommendations. In [RST10], they improve TD to PITF(Pairwise Interaction Tensor Factorization) with an adaptation of the Bayesian Personalized Ranking (BPR) criterion. PITF factors the tensor to a fixed diagonal core tensor, user matrix, item matrix, and pairwise tag matrix. The pairwise tag matrix can be trained with pairwise constraints form supervised information.

5.3.2 Metric learning on the RESCAL latent space

In [Nic13] and [NTK11], they propose RESCAL factorization. RESCAL factorization decomposes the relational tensor $T \in \mathcal{R}^{n \times n \times n_r}$ to a core tensor $R \in \mathcal{R}^{a \times a \times n_r}$ and a matrix $A \in \mathcal{R}^{n \times a}$, as shown in Figure 5.5. where a is a user-given positive integer parameter with $0 < a < n$.

Figure 5.5 shows the RESCAL factorization decomposes the relational tensor to two-part : the first part is a core tensor R with information loss, that we need and could control its size ; the second part is a matrix A_l which we pay attention to its capacity. The matrix A_l can be regarded as an embedding of the entities from dataset into an a-dimensional latent space. For the relational tensor, this matrix L quantifies the similarity of the relationships

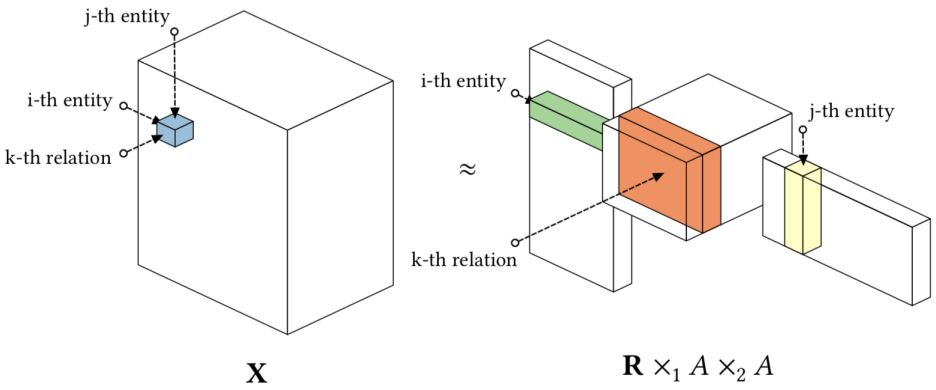


FIGURE 5.5 – RESCAL factorization decomposes relational tensor X to a matrix A which represents the relational information and a core tensor R. [Nic13]Notice that, in our work, we denoted the notation T as relational tensor X, A_l as factor matrix A

between entities and can be seen as a new latent feature space.

Notice that, the latent space A_l, only use the relational information but not the original features information. In [NTK11], [Nic13] and [NTK12], their solution is using a bigger tensor with the features as a relational between entities. In our opinion, this leads to the base and loss during the tensor factorization.

So in this work, we only use the RESCAL factorization on the relational information to get the latent space which represent the relation information and use it as a new " relational feature" for the entities. Our proposed RFML is applying the usual metric learning algorithms on the combine of "relational feature" latent space and original feature space. For example, in the next experiment Section 5.5 the RFML-ITML means applying the ITML on such union space.

5.4 Metric Learning with Multi-Relation

In the Section 5.3, we use the classical metric learning algorithms with RESCAL factorization, which is quite simple. Through we combine the metric learning method and tensor technique, but considering in [Nic13] they also solve a similar task, it is not a significantly progressive innovation.

Therefore, in this section, we consider starting with the metric learning method and proposing an algorithm for the multivariate relational data set, which is called MRML(Multi-Relation Metric Learning).

5.4.1 Relational Constraints

There are several relational constraints methods, which based on the relational side-information or relational structures. We will introduce some of them :

— The simple exist relational constraints : This constraints is the simplest way we mentioned in Chapter 4, $d_M(\mathbf{x}_i, \mathbf{x}_j) \leq d_M(\mathbf{x}_i, \mathbf{x}_k) + \gamma, (i,j) \in \mathbb{S}, (i,k) \in \mathbb{D}$ with the adjacent matrix $A_{ij} = 1, A_{ik} = 0, \forall (i,j) \in \mathbb{S}, \forall (i,k) \in \mathbb{D}$, which only check the relative distance between the connected and disconnected relational links.

— Our proposed link-strength constraints : In Section 5.2, we proposed the link-strength constraints,$\mathbb{C} = \{(i,j,k) : LS(\mathbf{x}_i, \mathbf{x}_j) \geq LS(\mathbf{x}_i, \mathbf{x}_k)\}$ where the $LS(\mathbf{x}_i, \mathbf{x}_j)$ is a link-strength function could be any function measure the similarity with the side-information from the relational structure, in our case, which is depended on the side-information on the common parents of the target pair of nodes.

— Connected neighbour constraints : As we mentioned in Chapter 4,in SPML(Structure Preserving Metric Learning) [SHJ11], the authors propose a metric learning algorithm based on the adjacency matrix A of a network for learning a Mahalanobis distance metric defined by matrix M, which is more effective on the inherent connectivity structure of the network. The associated constraint on the distance is $d_M^2(x_i, x_j) > (1 - A_{ij}) \max_l(A_{il} d_M^2(x_i, x_l)), \forall i, j$ with the adjacent matrix A_{ij}, which is depended on the fact that the distances to all disconnected nodes x_j must be larger than the distance to the farthest connected neighbour of all neighbour nodes x_l.

5.4.2 Proposed Loss Function for Multi-Relation

Our proposed approach of metric learning MRML(Multi-Relation Metric Learning) considers all of the three information : features, links and labels. We make use of the Mahalanobis distance for the metric definition, and add relational constraints into the objective function. Formally, the objective function is almost the same as the general model of metric learning algorithms, with the following modification :

$$L(M) = \sum_{(i,j,k) \in \mathbb{C}_\mathbb{S}} \ell_M(i,j,k) + \lambda r(M) \tag{5.8}$$

where $\mathbb{C}_\mathbb{S} = \{\mathbb{C}_\mathbb{R} \cup \mathbb{C}_\mathbb{L}\}$, i.e. the union of constraints obtained from relational informa-

tion, $\mathbb{C}_{\mathbb{R}}$, and constraints obtained from labels, $\mathbb{C}_{\mathbb{L}}$.

For the label information constraints, two popular approaches are mainly used for taking into account target labels : similar/dissimilar constraints and relative constraints. Without loss of generality, we take the relative one, given by $d^2_M(x_i, x_j)+\gamma \leq d^2_M(x_i, x_k),\ \forall(i, j, k) \in \mathbb{C}_{\mathbb{L}}$. The set $\mathbb{C}_{\mathbb{L}}$ contains (i, j, k) triples of data, where the (x_i, x_j) share the same label and (x_i, x_k) have different labels, and γ is a margin. The relative constraints make sure the entities in different classes be farther with the margin than the entities with the same labels. Based on common usage [WBS06], we choose $\gamma = 1$.

We can decompose the loss $\ell_M(i, j, k)$ of Eq. (5.8) into two distinct losses ℓ_L and ℓ_R that take into account label constraints and relational constraints. Using a hinge-loss, we obtain

$$\ell_L = \frac{1}{|\mathbb{C}_{\mathbb{L}}|} \sum_{(i,j,k)\in\mathbb{C}_{\mathbb{L}}} \max(d^2_M(x_i, x_j) - d^2_M(x_i, x_k) + 1, 0) \tag{5.9}$$

for label constraints.

For the relational constraints, we propose to use a multi-relationship tensor in place of the adjacency matrix. More precisely, every slice of the relational tensor is seen as an adjacency matrix. Consequently, every slice gives a particular sum of loss functions. and summing over all slices allows to obtain constraints from all relational links. Therefore, we consider the connected neighbour constraints,

$$d^2_M(x_i, x_j) > (1 - \mathbb{R}_r(i, j)) \max_l(\mathbb{R}_r(i, l)d^2_M(x_i, x_l)), \forall(i, j) \tag{5.10}$$

for each slice r of T.

Summing up over all slices, and using a hinge loss, gives

$$\ell_R = \frac{1}{n_r} \sum_{z=1}^{n_r} \frac{1}{|C_z|} \sum_{(i,j,k)\in C_z} \max(d^2_M(x_i, x_j) - d^2_M(x_i, x_k) + 1, 0), \tag{5.11}$$

where C_z is the set of constraints obtained through the z-th relation $\mathbf{r}$ of the tensor cube T, more precisely $C_z = \{(i, j, k)|\mathbf{r}_z(i, j) = 1, \mathbf{r}_z(i, k) = 0\}$. Note also that $\bigcup_{z=1}^{n_r} C_z = \mathbb{C}_{\mathbb{R}}$. Combining label constraints and relational constraints finally gives $L(M) = \lambda'\ell_R + (1 - \lambda')\ell_L + \lambda r(M)$, where the parameter λ' is introduced to control the trade-off between relational constraints and label constraints. Setting λ' to 0 makes the approach consider only label constraints, while setting it to 1 only uses relational constraints. Taking the squared

Frobenius norm as regularization term, the objective function is now

$$L(M) = \frac{\lambda}{2}\|M\|^2 + \lambda'\ell_R + (1-\lambda')\ell_L \tag{5.12}$$

5.4.3 Stochastic Sub-gradient Descent Learning Processing

In the sequel, we use a stochastic sub-gradient descent with mini-batches to optimize the loss function of MRML.

Using a convenient notation, one can rewrite the difference between the two distances in the loss function with a sparse matrix C as follows

$$d_M^2(x_i, x_j) - d_M^2(x_i, x_k) = C^{(i,j,k)} X^T M X,$$

where C is a sparse matrix storing the parameters $C_{jj}^{(i,j,k)} = 1$, $C_{ik}^{(i,j,k)} = 1$, $C_{ki}^{(i,j,k)} = 1$, $C_{kk}^{(i,j,k)} = -1$, $C_{ij}^{(i,j,k)} = -1$ and $C_{ji}^{(i,j,k)} = -1$. Otherwise $C^{(i,j,k)} = 0$.

The sub-gradient of the objective function is :

$$\begin{aligned} \nabla L(M) &= \lambda M + \tfrac{1-\lambda'}{|\mathbb{C}_\mathbb{L}|} \textstyle\sum_{(i,j,k)\in {\mathbb{C}_\mathbb{L}}^+} X C^{(i,j,k)} X^T \\ &+ \tfrac{\lambda'}{n_r} \textstyle\sum_{z=1}^{n_r} \tfrac{1}{|C_z|} \sum_{(i,j,k)\in C_z^+} X C^{(i,j,k),z} X^T, \end{aligned} \tag{5.13}$$

where ${\mathbb{C}_\mathbb{L}}^+$ and C_z^+ are subset of $\mathbb{C}_\mathbb{L}$ and C_z, respectively, for which $d_M^2(x_i, x_j) - d_M^2(x_i, x_k) + 1 > 0$.

In the sequel, we use a stochastic sub-gradient descent with mini-batches to optimize the loss function. As a benefit of this choice, complexity will be independent of the number of constraints we choose.

Algorithm 2 Metric learning based on relational tensor with stochastic sub-gradient descent

Require: X, T,$\mathbb{C}_{\mathbb{L}}$, $\mathbb{C}_{\mathbb{R}}$ and parameters λ, $n_c \leq |C|$, λ',t

Ensure: M

$M_0 = I_m$
for t_i from 1 to $t-1$ **do**
  $C = 0_{n,n}$, $C^z = 0_{n,n}, \forall z \in \{1, \cdots, n_r\}$
  $n_L = 0$, $n_R = 0$
  for b from 1 to n_c **do**
    sample (i, j, k) from $\mathbb{C}_{\mathbb{L}}$ with probability λ'
    if $d^2_M(x_i, x_j) - d^2_M(x_i, x_k) + 1 > 0$ **then**
      $n_L += 1$
      $C_{jj} += 1$, $C_{ik} += 1$, $C_{ki} += 1$,
      $C_{kk} += -1$, $C_{ij} += -1$, $C_{ji} += -1$.
    end if
    sample (i, j, k) from $\mathbb{C}_{\mathbb{R}}$ with probability $1 - \lambda'$
    for r from 1 to n_r **do**
      if $d^2_M(x_i, x_j) - d^2_M(x_i, x_k) + 1 > 0$ **then**
        $n_R += 1$
        $C^r_{jj} += 1$, $C^r_{ik} += 1$, $C^r_{ki} += 1$,
        $C^r_{kk} += -1$, $C^r_{ij} += -1$, $C^r_{ji} += -1$.
      end if
    end for
  end for
  $\nabla_{t_i} = \frac{1-\lambda'}{n_L} XCX^T + \frac{\lambda'}{n_R n_r} \sum_z XC^z X^T + \lambda M_{t_i}$
  $M_{t_{i+1/2}} = M_{t_i} - \frac{\nabla_{t_i}}{t_i \lambda}$
  $M_{t_{i+1}} = Proj_{\mathbb{S}^+_m}(M_{t_{i+1/2}})$ // projection on closed convex cone of $\mathbb{S}_m$, space of symmetric m-by-m matrices
end for
return M_T

The algorithm 2 shows the detail of the MRML with stochastic sub-gradient descent. As in [SHJ11], this algorithm is variation of the PEGASOS algorithm [SSSSC11] without projection, giving, with probability $1 - \delta$, a bound on the optimization error ε, given by $\frac{84R^2 \ln(t/\delta)}{\lambda t}$, if the norm of any input x is at most R, and t is the number of iterations.

5.5 Experiments and Result

The evaluation of the proposition is done by comparing the effect of learned metric with K-nearest-neighbour classification. For the set of KNN classification, we use K equal to 5 and score the performance with accuracy rate via randomly shuffled 3-fold cross-validation. Notice that we tried different values for K (in particular 3, 5, 7 and 9), and the results were consistent with the results reported here for $K = 5$ on most datasets. For each experiment, the number of constraints varies from 100 to 500, and we give the average value of each set as the final result. All the experiments were run on a 3.1Ghz Intel Core i5 processor, with 16 Go 1867 MHz DDR3, and the code will be published for research reproducibility. We also give results obtained without learning a metric, i.e. using a Euclidean distance for the KNN algorithm (`Euc`).

5.5.1 Experiments on One Relation Dataset For LSCS

In this section, we mainly focus on testing the proposed LSCS(Relational Link-strength Constraints Selection) algorithm on one $\mathbb{R}_r$ relation dataset. Notice that, LSCS could be easily extended to multi-relational datasets by summing the link-strength for each relation. It could also be extended from the reference relation $\mathbb{R}_r$ to $\mathbb{R}_e$, that consider the group structures as every edge between nodes is the side-information of common parents but as a binary value. We will test this approach with multiple relation dataset and compare with another proposed algorithm and other metric learning algorithms in the next section.

We conduct experiments to compare the performance of the constraints generated by link-strength function and the constraints generated by the label information. To compare fairly, we set the number of constraints generated by different ways is the same, and the formula is both in the relative distance constraint. Basically, any relational data for which classification is needed can be tackled by our proposition.

Datasets and Tasks

We consider several real-world relational datasets which contains feature information for mapping with the learned metric, the target label information, and the relational information (existing links or valued links). We learn the metric of one entity table for predicting target label, so for the same dataset, we can learn different metrics for different tasks. Here are the descriptions of the chosen datasets and tasks :

— Movie : MovieLens dataset [HK16] is a classical relational dataset which is widely used in many related papers. It consists of a relational table which has 100,000 ratings (1-5) from 943 users on 1682 movies ; a movie entity table with feature information about the movies ; and a user entity table with id, age, gender, occupation, and other feature information on users. Each user has rated at least 20 movies, so the supervised relational information is quite dense. We define two tasks on this dataset :
  — Movie-item : We select the movies table as the entity table to learn the metric on. We choose the most popular genre as the target label and use the release date and other genres as the attributes.
  — Movie-year : We select the users table as the entity table to learn the metric on. The age of users is discretized into 5 bins as the target label, and the other feature information is the attributes.
— BookCX : We also consider the book-crossing database [ZMKL05]. We select a randomly sampled subset BookCX from the data. This subset contains 2,400 users giving 5,000 ratings (1-10) on 10,000 books. For this dataset, we use the bag-of-words model to encode the text information from the titles, the authors and the publishers into binary attributes.
  — BookCX-year : We consider the public year segmented into 5 bins as the target label and the bag-of-word of text information as the attributes.
  — BookCX-word : We apply PCA (Principal Component Analysis) on the bag-of-word of text information and limit the number of dimensions to 12. Then we randomly choose one of the processed dimension and segment it into 5 bins as the target label. The other features are considered as attributes.
— Citelike : There are two versions of citelike dataset, Citelike-t, and Citelike-a, both used in the paper [WCL13]. They were collected from CiteULike and Google Scholar. CiteULike allows users to create their own collections of articles. There are abstracts, titles, and tags for each article. They manually select hundreds of seed tags and collect all the articles with at least one of these tags. They also crawl the citations between the articles from Google Scholar. Notice that the final number of tags associated with all the collected articles is far more than the number of seed tags. To reduce the computation complexity, we apply PCA on the large and sparse tag feature space and limit the number of dimensions to 12. We randomly choose one of the processed features and segment it into 5 bins as the target label. Notice that the

ITML	Movie-item	Movie-user	BookCX-year	BookCX-word	Citelike-t	Citelike-a
Euc	98.58 ±0.46	68.28 ±4.00	36.16 ±1.25	90.36 ±0.91	88.03 ±0.45	88.94 ±0.55
Lab	98.62 ±0.52	67.72 ±5.38	36.19 ±1.09	89.57 ±0.87	85.91 ±0.62	89.89 ±0.23
Rel	97.48 ±0.66	68.66 ±3.40	36.12 ±1.17	**91.29** ±0.74	90.76 ±0.56	89.94 ±0.41
Pro	97.54 ±0.42	69.04 ±4.02	36.38 ±1.62	90.33 ±0.74	92.06 ±0.46	**90.35** ±0.32
Both	**98.67** ±0.50	**69.48** ±3.08	**36.97** ±1.69	90.43 ±0.72	**92.65** ±0.40	**90.35** ±0.32

TABLE 5.1 – The accuracy score of KNN with ITML

LSML	Movie-item	Movie-user	BookCX-year	BookCX-word	Citelike-t	Citelike-a
Euc	98.58 ±0.46	68.28 ±4.00	36.16 ±1.25	90.36 ±0.91	88.03 ±0.45	88.94 ±0.55
Lab	99.06 ±0.58	65.92 ±5.38	36.36 ±1.06	94.46 ±1.06	85.53 ±0.67	94.62 ±0.33
Rel	98.63 ±0.52	66.67 ±4.03	36.21 ±1.21	94.91 ±0.41	85.65 ±0.63	94.62 ±0.65
Pro	98.63 ±0.40	**66.98** ±4.63	**36.42** ±1.52	**94.92** ±0.61	**85.69** ±0.57	**94.63** ±0.42
Both	**99.12** ±0.52	**66.98** ±4.63	**36.42 ±1.15**	**94.92** ±0.61	**85.69** ±0.57	**94.63 ± 0.30**

TABLE 5.2 – The accuracy score of KNN with LSML

sampling of Citelike-t and Citelike-a are independent, and the density of the links is different.

For all the used datasets, the balance parameter between association attributes in the link-strength function is set to $\gamma = \frac{num^n}{num^n+num^c}$ to adapt to different situations of the datasets, where num^n is the number of numerical variables and num^c is the number of categorical variables as mentioned before.

Result of Comparing Different Constraints

In Tables 5.1, 5.2 and 5.3, `Lab` indicates the result obtained using only the constraints generated from label, `Rel` shows the result obtained by the constraints generated from the relative link constraints, i.e. using the adjacency matrix A of the graph. `Pro` gives the performance of our proposition based on link-strength constraints and `Both` shows the best result with both label constraints and the link-strength constraints while the proportion of them are appropriated.

Result of Comparing Different Proportion of label constraints and the link-strength constraints

The Table 5.4 and Table 5.5 show the results with different set of proportion of label constraints and the link-strength constraints. Proportion equal to 1 corresponds to the situation of using only labels, and a proportion of 0 corresponds to the fact of using only

LSML	Movie-item	Movie-user	BookCX-year	BookCX-word	Citelike-t	Citelike-a
Euc	98.58 ±0.46	68.28 ±4.00	36.16 ±1.25	90.36 ±0.91	88.03 ±0.45	88.94 ±0.55
Lab	99.06 ±0.58	65.92 ±5.38	36.36 ±1.06	94.46 ±1.06	85.53 ±0.67	94.62 ±0.33
Rel	98.63 ±0.52	66.67 ±4.03	36.21 ±1.21	94.91 ±0.41	85.65 ±0.63	94.62 ±0.65
Pro	98.63 ±0.40	**66.98** ±4.63	**36.42** ±1.52	**94.92** ±0.61	**85.69** ±0.57	**94.63** ±0.42
Both	**99.12** ±0.52	**66.98** ±4.63	**36.42 ±1.15**	**94.92** ±0.61	**85.69** ±0.57	**94.63 ± 0.30**

TABLE 5.3 – The accuracy score of KNN with MMC

Proportion	Movie-item	Movie-user	BookCX-year	BookCX-word	Citelike-t	Citelike-a	Mondial
1.0	98.62 ±0.52	67.72 ±5.38	36.19 ±1.09	89.57 ±0.87	85.91 ±0.62	89.89 ±0.23	70.39 ±8.29
0.8	98.52 ±0.23	66.21 ±3.71	36.21 ±1.06	89.52 ±0.55	86.71 ±0.67	89.71 ±0.42	**72.00** ±7.16
0.6	98.42 ±0.54	66.87 ±4.09	36.66 ±1.67	89.41 ±0.82	**92.65** ±0.40	89.61 ±0.31	71.03 ±6.22
0.4	**98.67** ±0.50	67.65 ±4.82	**36.97** ±1.69	88.87 ±0.78	90.87 ±0.38	89.91 ±0.35	70.82 ±7.66
0.2	97.32 ±1.52	**69.48** ±3.08	36.28 ±1.16	**90.43** ±0.72	91.24 ±0.41	90.21 ±0.37	69.21 ±7.98
0.0	97.54 ±0.42	69.04 ±4.02	36.38 ±1.62	90.33 ±0.74	92.06 ±0.46	**90.35** ±0.32	71.24 ±7.77

TABLE 5.4 – The accuracy score of KNN with ITML while the proportion of label constraints and the link-strength constraints gradient change from full label constraints to full link-strength constraints.

link-strength based constraints. As can be seen in the Table, results tend to be better when using mostly link-strength constraints.

As can be seen, except on Movie-item task and BookCX-year task, comparing with constraints generated only from labels, the link-strength constraints lead to a significant improvement of accuracy. On most datasets, the link-strength constraints show better performance than the relative link constraints, except the BookCX-word task with ITML. For both labels and relational information, it provides better accuracy than the constraints obtained from labels and similar to the constraints generated with link-strength function. Considering the different number of references in these datasets, for example, 100, 000 references for 943 entities for Movie-user and 5,000 references for 10,000 entities for BookCX-year, we speculate that density or sparsity of the references leads to the deviation of results.

5.5.2 Experiments on Multi-relation Dataset

Datasets and Tasks

To conduct this study, we use 5 benchmark real-world databases. In Table 5.6, properties of the datasets we used are given, where n is the number of instances, n_r is the number of types of relations and m is the number of features.

— Elite : DutchElite dataset [DvR08], this is a dataset containing the relational in-

Proportion	Movie-item	Movie-user	BookCX-year	BookCX-word	Citelike-t	Citelike-a	Mondial
1.0	99.06 ±0.58	65.92 ±5.38	36.36 ±1.06	94.46 ±1.06	85.53 ±0.67	94.62 ±0.33	68.45 ±8.47
0.8	99.04 ±0.54	65.67 ±4.32	36.22 ±1.14	94.89 ±0.60	85.46 ±0.61	94.59 ±0.38	**71.03** ±7.35
0.6	**99.12** ±0.52	65.78 ±3.80	36.13 ±1.18	94.78 ±0.58	85.53 ±0.79	94.61 ±0.41	70.22 ±8.42
0.4	99.08 ±0.67	66.21 ±4.52	36.3 ±1.50	94.85 ±0.68	85.45 ±0.70	**94.63 ±0.30**	69.79 ±6.54
0.2	98.87 ±0.62	66.14 ±5.39	**36.42 ±1.15**	94.83 ±0.86	85.45 ±0.67	**94.63** ±0.52	70.67 ±7.22
0.0	98.63 ±0.40	**66.98** ±4.63	**36.42** ±1.52	**94.92** ±0.61	**85.69** ±0.57	**94.63** ±0.42	70.62 ±7.04

TABLE 5.5 – The accuracy score of KNN with LSML while the proportion of label constraints and the link-strength constraints gradient change from full label constraints to full link-strength constraints.

Dataset	n	n_r	m	Classes
Elite	4747	41	7	2
Mondial	185	23	4	2
Movie	1804	26	5	18
UW	278	4	3	2
MG	4893	6	2	3

TABLE 5.6 – Dataset characteristics.

formation of administrative elite in The Netherlands. The label distinguishes if the elite is top200 or not.

— Mondial : Mondial dataset [May99], it is a dataset containing the relational version of the geographical Web data sources. The labels are the classes of entities.

— Movie : Movie-Remark dataset [Lic13], which is a dataset containing the relational form across several files of movie information, labels are the movie types.

— UW : UW-std version of UW-CSE dataset [KSHG12], it is the relationships of students and professors of the Department of Computer Science and Engineering at the University of Washington. The labels are defined by the phase they are in.

— MG : Mutagenesis dataset [DLD$^+$91], comprises of molecules trialled for mutagenicity on Salmonella typhimurium. We just use the atom and the bond between them as the relationship. The labels are the types of atom.

As mentioned in the introduction, usual metric (learning) approaches solely rely on the use of features, and do not make use of the relations between observations. In order to fairly compare metrics, we propose to first embed the data into a space that reflects the relations within the data.

Experiments Configuration for LSCS

The link-strength function could be extended to multi-relational datasets by summing the link-strength for each relation, and also be extended from the reference relation $\mathbb{R}_r$ to $\mathbb{R}_e$ considering the group structures as every edge between nodes is the side-information of common parents but as a binary value. In that case, the link-strength function would consider the additional term

$$\sum_{k}^{\ell_{ij}^k} v_m P_k(i,j), \tag{5.14}$$

where $P_k(i,j)$ is the parent adjacency matrix of the k-relation in the group structure defined as

$$P_k(i,j) = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ have} \\ & \text{common parents in relation } \mathbf{r}_k \\ 0 & \text{otherwise,} \end{cases} \tag{5.15}$$

and ℓ_{ij}^k is the number of common parents of $\mathbf{x}_i$ and $\mathbf{x}_j$ in the relation $\mathbf{r}_k$.

Experiments Configuration for Metric Learning with RESCAL Factorization

As mentioned before, for metric based on rescal factorization, the latent feature space is based on matrix A_l, which is generated by $T \approx R \times_1 A_l \times_2 A_l$ where the factor matrix is $A_l \in \mathbb{R}^{n \times a}$ and the core tensor is $R \in \mathbb{R}^{a \times a \times m}$. A_l and R depend on the user-given positive integer parameter a.

If we choose a large value for a, the approximation error decreases, at the cost of higher complexity. In the experiments, there will be a better factorization A_l with more dimensions while the larger rank leads to a bigger core tensor R and tedious calculation. So we test the performance of Euclidean distance on the latent feature space of dataset Mondial with gradual change rank.

In the sequel, we choose to find an optimal trade-off between accuracy and complexity as follows. We set the value of a as a proportion of the total number of observations n. For each value of a, accuracy and running times (in seconds) are computed. The result is given in Figure 5.6, for the Mondial dataset, where the value $a = \lfloor 0.3 \times n \rfloor$ is selected. As can be expected, both accuracy and running times increase along with a.

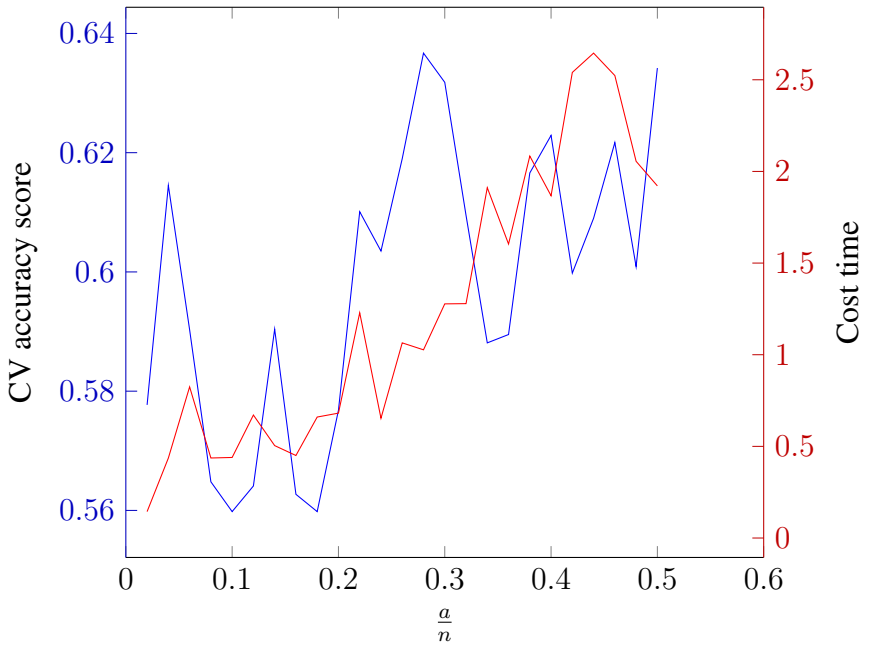


FIGURE 5.6 – Evolution of performance and complexity metrics of RESCAL for dimension space of varying size.

Dataset	Euc.	Res	Res+Fea	LSCS-ITML no-label	LSCS-LSML no-label	MRML no-label
Elite	87.60±12.08	**91.14±0.80**	89.59±0.85	84.56±12.46	88.25±1.28	87.80±5.67
Mondial	68.66±7.99	61.31±7.83	58.59±5.30	64.66±8.81	59.57±5.97	**69.40±1.33**
Movie	38.56±1.86	33.31±7.84	39.56±2.25	38.21±1.21	39.42±1.52	**40.07±2.07**
UW	86.03±5.17	54.03±8.65	87.51±7.87	85.55±9.08	84.73±6.19	**87.63±4.60**
MG	86.11±0.79	61.72±1.04	75.39±2.19	79.06±19.96	82.94±9.33	**86.16±1.17**

TABLE 5.7 – Cross-validation accuracy of KNN with different metrics related on different combination of data information.

In this chart, the x-axis is $\frac{a}{n}$ and $0 < \frac{a}{n} < 1$, which means we flexibly chose the a, n is the number of instance. When chosen rank is bigger, the cost time obviously linearly increases, while the performance do not be better after a threshold $a = 0.3n$.

Result of Comparing Different Algorithms Without Label Information

In this section, we compare the performance of different metrics related on different combination of features information and relational information without using labels, but only relational constraints.

In Table 5.7 and Table 5.8, cross-validation accuracy and F1 score (and their standard deviations) of K-nearest-neighbour algorithm using different metrics are given. Euc. stands

Dataset	Euc.	Res	Res+Fea	LSCS-ITML no-label	LSCS-LSML no-label	MRML no-label
Elite	86.39±2.60	**89.93±11.08**	87.25±0.78	87.86±1.00	86.63±1.02	86.41±2.67
Mondial	68.27±6.56	60.04±5.41	57.13±7.31	67.86±7.00	68.47±8.26	**69.42±6.58**
Movie	32.44±1.88	24.04±2.24	34.10±1.36	33.21±1.29	34.31±2.89	**34.25±0.89**
UW	87.84±5.09	45.45±6.56	87.26±5.39	84.10±8.36	84.04±6.42	**87.34±6.43**
MG	83.45±7.23	60.31±2.87	74.05±2.04	83.67±7.72	83.63±7.48	**85.84±2.24**

TABLE 5.8 – Cross-validation f1 score of KNN with different metrics related on different combination of data information.

for the usual Euclidean distance in the feature space (i.e. $M = Id$). Inv. stands for the usual Mahalanobis distance, which is using the simple covariance matrix Σ ($M = \Sigma^{-1}$). Res stands for the usual Euclidean distance in the latent space obtained using RESCAL factorization, as described before. Finally, Res+Fea corresponds to the joint space of RESCAL embedding A_l and the original feature space X, given by (X, A_l). Each observation i is described by $(x_{i1}, \cdots, x_{in}, al_{i1}, \cdots, al_{ir})$. MRML no-label is MRML in a restricted unsupervised setting where it uses only relational constraints and no target labels. LSCS-ITML no-label are LSCS-LSML no-label are classical metric learning algorithm ITML and LSML with LSCS but only selecting constraints with relational side-information. In the tables, best values are indicated in bold font. From Table 5.7 and Table 5.8, one can see that MRML no-label consistently performs better than the other approaches, except for the Elite dataset, for both accuracy and F1 metrics.

Result of Comparing Different Algorithms With Label Information

For the MRML, the impact of λ' in the loss function $L(M) = \frac{\lambda}{2}\|M\|^2 + \lambda' \ell_R + (1-\lambda')\ell_L$ need be studied, which is controlling the importance of labels and relations on the learned metric, on the performances. Consequently, for each dataset, we evaluate the learned metric for λ' varying from 0 (no relations) to 1 (no labels). Results are shown in Figure 5.7. As can be seen, for all datasets, the best accuracy is obtained in-between, showing that taking both information into account is useful, as expected.

For each dataset, the optimum λ' is chosen according to this preliminary analysis.

We finally compare the accuracy and F1 metrics of ITML, LSML and LFDA with the proposed RFML and MRML algorithm. RFML-ITML is using ITML with RFML, while LSCS-ITML is using ITML with the LSCS, the rest follow the same naming principles. We use the MLN(Markov Logic network) as the baseline, which is learned by the pracmln package[NPB$^+$] with default discrimination parameters.

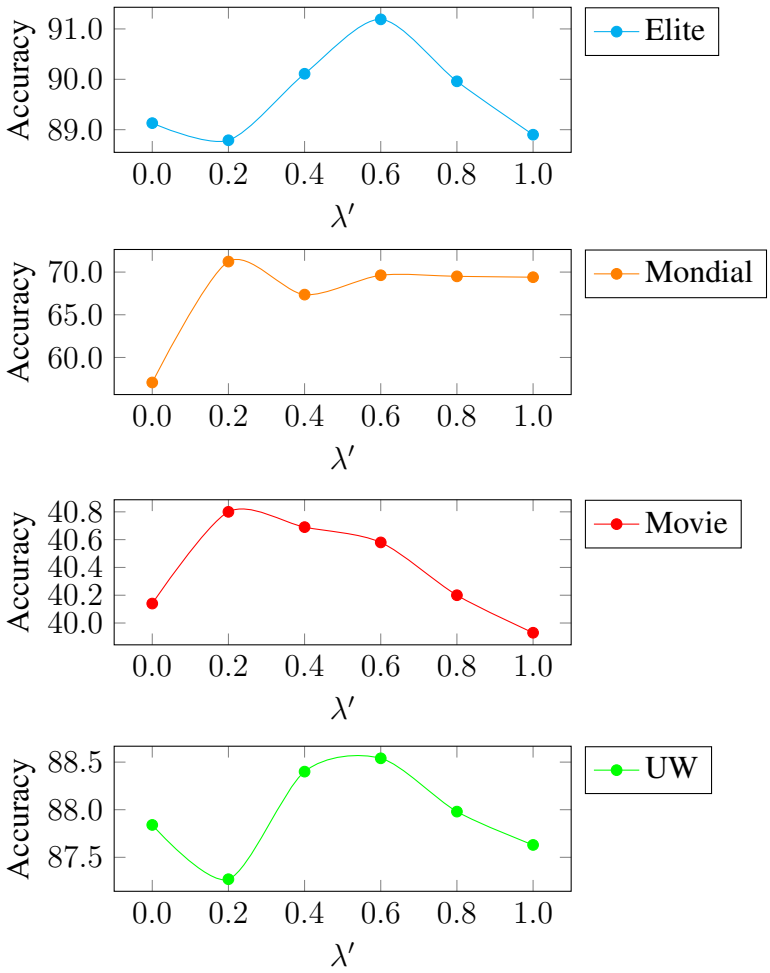


FIGURE 5.7 – Performance of MRML with respect to different values of λ'.

Dataset	MLN	RFML-ITML	RFML-LSML	RFML-LFDA	LSCS-ITML	LSCS-LSML	MRML
Elite	NT	89.28±0.33	90.76±0.87	90.20±.54	89.18±0.64	88.66±00.80	**91.19±1.33**
Mondial	67.71±7.82	61.29±6.64	56.54±11.28	59.46±7.11	64.66±5.90	59.35±6.90	**71.23±6.27**
Movie	**40.84±0.84**	39.16±2.76	38.54±1.81	39.86±1.53	38.16±1.25	39.38±1.62	40.80±1.26
UW	77.12±6.12	70.21±1.48	55.88±7.03	**90.63±5.28**	85.98±8.99	85.32±3.60	88.54±2.67
MG	81.45±9.71	79.74±1.48	70.61±1.58	72.03±1.27	82.98±10.10	84.51±5.17	**86.16±1.26**

TABLE 5.9 – Cross-validation accuracy of KNN with different metric learning methods.

Results are shown in Tables 5.9 for accuracy rating and 5.10 for accuracy and F1 metrics, respectively. NT means the running of the algorithm is out of the time limit. As can be observed in Table 5.9, MRML performs better than other approaches, except on the UW dataset which RFML-LFDA is better and the Movie dataset which MLN is better(although both of them are closely followed by MRML). From Table 5.10, it shows LSCS, and

Dataset	MLN	RFML-ITML	RFML-LSML	RFML-LFDA	LSCS-ITML	LSCS-LSML	MRML
Elite	NT	87.43±1.02	88.89±1.37	88.15±1.22	87.15±2.04	82.64±10.96	**89.31±1.05**
Mondial	**75.82±8.45**	58.99±9.74	53.60±6.75	57.28±7.40	70.34±6.85	67.56±10.00	69.67±8.63
Movie	35.48±6.25	33.48±1.78	32.29±2.72	33.92±1.72	34.91±0.41	**36.16±1.25**	35.06±0.36
UW	78.61±3.58	68.72±2.64	57.23±2.68	**90.41±3.05**	87.20±3.11	85.40±7.18	88.02±1.07
MG	68.61±8.78	79.25±0.89	68.70±1.74	70.13±1.90	**83.57±7.45**	83.48±7.20	82.40±1.63

TABLE 5.10 – Cross-validation F1 score of KNN with different metric learning methods.

MRML performs better than other approaches, except on the UW dataset and Mondial, and the best one is LSCS-ITML on MG, LSCS-LSML on Movie and MRML on Elite. The f1 score shows LSCS sometimes is steadier. Statistical significance of the results are assessed using a Friedman test [Fri40] as suggested by [Dem06]. The value of the Friedman test, $F_F = 7.63 > F_{0.05}(6, 24)$ shows the significance of the difference between the ranks.

Result of Comparing Cost Time of Different Algorithms

In this section, the cost time of different algorithms in our experiments will be comparing. The cost time of the link-strength algorithm is depended on the number of relational constraints selection which is user-given, so the time of it will not be compared.

Dataset	RESCAL	ITML	LSML	LFDA	MRML
Elite	2864	206.6	4.13	3.02	**25.30**
Mondial	1.17	1.89	0.09	**0.01**	21.52
Movie	832	434.1	0.96	2.19	**22.15**
UW	0.96	44.4	0.06	**0.01**	16.80
MG	96.14	92.61	3.19	1.37	**21.12**

TABLE 5.11 – Running times, in seconds, of different metric learning methods.

Running times for the different metric learning methods are given in Table 5.11, where best values are indicated in bold font. The columns Res corresponds to the running time of RESCAL algorithm, that is required for ITML, LSML and LFDA algorithms. Consequently, the total running time of each of these algorithms is given by adding the RESCAL projection and their running times. According to this table, MRML is much faster on Elite, Movie, and MG, while being reasonably slower on Mondial and UW (except for UW and ITML). As described before, the complexity of our method does not depend on the number of observations, so that it scales well to extensive scale data, both in volume n or dimension m.

5.6 Conclusion

In this chapter, we present the second major contribution of this paper : the current metric distance learning algorithm is very scarce in the specialized field of the relational database. To fill this gap, we propose three graph-based and side information-based metric distance learning algorithm to solve the relational learning task. The experimental results on the real database clearly show that the proposed algorithms outperform other metric learning algorithms in terms of accuracy or cost time.

For a variety of relational learning algorithms, we chose to start with graph mining and multi-relational learning. The relational database is mainly regarded as an entity table as a node on the graph, and the relational table is a pattern of the weighted edge of the graph. And we are targeting two different situations :

The first case is a relational data set with complex structure and side information on the relational link. For this case, we propose a LSCS(Relational Link-strength Constraints Selection) algorithm. We propose a method for calculating link strength based on graph analysis techniques. A constraint set of metric distance learning algorithms is then selected according to the method. After selecting the set, the experiment is conducted by combining the traditional metric distance learning algorithms of ITML and LSML.

The second case is that for a relational data set with a large entity table and multiple relationships between entities, we propose two new methods for metric distance learning :

— The first method is based on RESCAL. First, we establish the relational table as the relationship tensor and use the RESCAL factorization technique to obtain a resolved latent space on the tensor. Then we use the latent space as the feature space that re-presentation the relationship information and combine it with the original feature space. Finally, we use various metric distance learning algorithms in the newly acquired embedded space. This method is more often seen as a baseline for relational metric learning in this chapter.

— The second method is MRML (Multi-Relation Metric Learning). The method is based on the features and relationships of entities in the multi-relational data, proposes a relationship constraints for each relationship and accumulates it, and then combines with the constraints according to the label selection. Finally, we propose a stochastic sub-gradient descent algorithm to learn the metric.

According to the experimental results, we can propose that the three relationship metric distance learning algorithms have their own advantages and disadvantages, which are sui-

table for different situations. And from the experiment process, we also found that there is still room for improvement. For example, in the LSCS there are still many possibilities for the calculation of the link strength, and in MRML more optimized results can be obtained by performing a given weight analysis for each relationship.

CONCLUSION AND PERSPECTIVES

At the end of this article, we organize full-text context and summarize contributions and make some predictions for the future.

5.7 Summary of Contributions

Metric distance learning as a branch of machine learning has been developed and applied in various fields. The main contribution of this paper is to propose new algorithms to supplement the gaps in today's metric distance learning algorithms for both flat and non-flat databases. As the structure of this paper, in Chapter 1, after briefly introducing the relevant background, the Chapter 2 and the Chapter 3 correspond to the first major contribution of this paper, summarizing the development and current situation of the metric learning algorithm of the flat database, and proposing submodular extension metric learning algorithm, the Chapter 4 and the Chapter 5 correspond to the second major contribution of this paper, summarizes the development and current situation of metric learning algorithm for non-flat database, and proposes three kinds of metric learning specifically for relational database.

5.7.1 Contributions for Flat Datasets

We summarize the development of metric distance learning in Chapter 2. In order to distinguish it from the second contribution, we only focus on the simple flat database, which is a single table that represents the characteristics of the entity as a vector. After the research summary, it can be seen that the basis of most metric distance learning algorithms is the Mahalanobis distance. The drawback of the Mahalanobis distance is that only the wight a single dimension or the intersection of two dimensions is considered. It is worth mentioning that other partial metrics, such as linear correlation also have this defect. The submodular metric learning algorithm [PLC19b] we proposed in Chapter 3 starts from the set function. We take the set of arbitrary dimensions as the input of the set function and use the weight of this intersection as the output. This model gives the possibility to learn a metric considering the intersection of any number of dimensions. To learn this complex

set function, we apply the special properties of the submodular function. Starting from the extension of the submodular function, the Lovasz metric and Multi-linear similarity are proposed and proved. Finally, we give the learning algorithms of these two kinds of metrics and carry out experiments on the real database. The experimental results show that the Lovasz metric has an absolute superiority compared with other current metrics, while the Multi-linear similarity does not perform well.

5.7.2 Contributions for Non-Flat Datasets

Although we introduced many metric distance learning algorithms for non-flat databases in Chapter 4, such as for string databases, time-series databases, tree or graph databases. However, when investigating metric distance learning algorithms for relational databases, the current results are still minimal. So in Chapter 5, we present three metric learning algorithms specifically for relational databases.

Relationship learning has many branches from different angles. We focus on relational learning based on graph mining and multi-relationship learning. We propose three methods for two types of relational databases. The first type is a sophisticated relational database of multi-entity tables and multi-relation tables. We propose LSCS(Relational Link-strength Constraints Selection) [PCL18], which is a method based on graph analysis technology to calculate the link strength between entities, and then select the constraints set for the metric distance learning algorithm according to this method. The second type is the relational database of the single entity table and multi-relation tables. We propose to use the RESCAL factorization technique to obtain the latent space in the application of the classic metric distance learning algorithm as the baseline algorithm. Then we propose another multi-relational metric distance learning algorithm that sums the relationship constraints and label constraints. An algorithm of learning such a metric by stochastic gradient descent is proposed. The experimental results show that these three methods have their advantages and disadvantages in different situations.

5.8 Perspectives of Future

During the research period, we also tried many different methods to accomplish the expected tasks. There have been various failures and regrets that have not been further explored. In this section, we will not only discuss possible solutions and related technologies

based on the shortcomings of current algorithms but also explore some of the perspectives and conjectures we have tried but have not gone further.

5.8.1 Possible Improvements to Proposed Algorithms

— For the submodular metric distance learning algorithm, there are two shortcomings : First, the research on other extensions is not enough. We are not satisfied with the performance of the similarity based on the definition of multi-linear extension with additional restrictions. If a conversion function or other technique that allows multiple linear expansions to satisfy all the conditions of a metric could be proposed, then this situation is well worth exploring. Secondly, the computational complexity of the current algorithm is high. Two limitations limit current algorithms. The first limitation is the size of submodular constraints with the large increase in dimensions. Although we use k-additive to solve, it still limits the performance of the submodular metric distance learning algorithm on high-dimensional databases. In future work, we will try to study more optimization techniques for set expressions or measure expressions. The second limitation is that as the number of learning samples increases, the size of the label constraints also increases. In this regard, we will try to extend the algorithm to the online learning version in the future so that it does not depend on the number of samples to learn.

— For the LSCS(Relational Link-strength Constraints Selection), we want to try more graph analysis methods in the future to determine the link strength. The existing common parent method limits the link strength of only the structure of $\mathbf{x}_i$-$\mathbb{P}_{ij}$-$\mathbf{x}_j$, and can not adequately consider the link strength between nodes with more complicated links and more distant links. We hope to improve this problem by using the shortest path and other graph analysis technique. Another possible improvement is to extend the selection process to online learning version. Each time the current algorithm has a new node because the link structure is changed, it is necessary to recalculate all the link strengths and reselect all the constraints sets. We hope to improve it by combining the online selection constraints set in CWSML(Constraint Weighted Selection Metric learning)[LC18].

— For the MRML (Multi-Relation Metric Learning), its performance is still limited by choice of relationship constraints. Although simple sum all the relationship constraints can accomplish the expected task, a natural cognition is that different relationships should have different effects on the metric between entities, so we hope

to weight different relationship constraints. Limited to the time constraints and the complexity of learning more parameters, we have not completed this part yet, but in the future, this idea is still worth trying. On the other hand, inspired by the success of RESCAL factorization applications, we also hope that in the future we will find suitable factorization or other tenser techniques for a relational tensor that can improve the performance of MRML or reduce its computational complexity.

5.8.2 Perspectives to Related Research

When we are working on metric distance learning algorithm specifically for relational databases, we have noticed a class of deep learning algorithms based on the graph topology–GCN (graph convolution neural network) [KW16]. As introduced in Chapter 1, we mentioned a point of view of deep neural networks, where the hidden layer is a representation of features. As a type of multi-layer neural network, each layer of the neural network of the GCN is a re-presentation of the topology of the input network. The relationship metric distance algorithm we are looking for also needs to re-presentation the relationship information. This gives us an interest in combining GCN with relational metrics learning.

Note that in Chapter 4 we have introduced the latest graph dataset metric distance learning algorithm BCNML(Brain Connectivity Networks Metric Learning) [KPF+18] which is based on the application of GCN. In the future, we will try two aspects of research following the inspiring of this paper : one is that we will try to extend BCNML from the single processing graph to the processing of multiple relational graphs ; And the second is that we will try to extend BCNML from the "Siamese network" of GCN to the "Triplet network" of GCN, where the "Triplet network" structure of GCN is proposed in TNDML(Triplet Network Deep Metric Learning) [HA15] and in TNDML, the experimental results show that for the metric distance learning algorithm, "Triplet network" performs better than "Siamese network".

BIBLIOGRAPHIE

[Alt92] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3) :175–185, 1992.

[Aro50] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3) :337–404, 1950.

[AVH04] Grigoris Antoniou and Frank Van Harmelen. *A semantic web primer*. MIT press, 2004.

[Bac13] Francis Bach. Learning with submodular functions : A convex optimization perspective. *Foundations and Trends® in Machine Learning*, 6(2-3) :145–373, 2013.

[BBHS08] Marc Bernard, Laurent Boyer, Amaury Habrard, and Marc Sebban. Learning probabilistic models of tree edit distance. *Pattern Recognition*, 41(8) :2611–2629, 2008.

[BBHS16] Aurelien Bellet, José F Bernabeu, Amaury Habrard, and Marc Sebban. Learning discriminative tree edit similarities for linear classification — application to melody recognition. *Neurocomputing*, 214 :155–161, 2016.

[BCV13] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning : A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8) :1798–1828, 2013.

[BFNS14] Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *SODA*, pages 1433–1452, 2014.

[BGR+06] Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14) :e49–e57, 2006.

[BHS06] Marc Bernard, Amaury Habrard, and Marc Sebban. Learning stochastic tree edit distance. In *European Conference on Machine Learning*, pages 42–53. Springer, 2006.

[BHS07] Laurent Boyer, Amaury Habrard, and Marc Sebban. Learning metrics between tree structured data : Application to image recognition. In *European Conference on Machine Learning*, 2007.

[BHS12] Aurélien Bellet, Amaury Habrard, and Marc Sebban. Good edit similarity learning by loss minimization. *Machine Learning*, 89(1-2) :5–35, 2012.

[BHS13] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv :1306.6709*, 2013.

[BHS15] Aurélien Bellet, Amaury Habrard, and Marc Sebban. Metric learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 9(1) :1–151, 2015.

[BJL11] Gleb Beliakov, Simon James, and Gang Li. Learning choquet-integral-based metrics for semisupervised clustering. *Fuzzy Systems, IEEE Transactions on*, 19(3) :562–574, 2011.

[BKTDS14] Gustavo EAPA Batista, Eamonn J Keogh, Oben Moses Tataw, and Vinicius MA De Souza. Cid : an efficient complexity-invariant distance for time series. *Data Mining and Knowledge Discovery*, 28(3) :634–669, 2014.

[BM03] Mikhail Bilenko and Raymond J Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 39–48. ACM, 2003.

[BN02] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

[Bra01] Ivan Bratko. *Prolog programming for artificial intelligence*. Pearson education, 2001.

[Bre67] Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3) :200–217, 1967.

[BWL+11] Jinbo Bi, Dijia Wu, Le Lu, Meizhu Liu, Yimo Tao, and Matthias Wolf. Adaboost on low-rank psd matrices for metric learning. In *CVPR 2011*, pages 2617–2624. IEEE, 2011.

[BYGP14] Julien Bohné, Yiming Ying, Stéphane Gentric, and Massimiliano Pontil. Large margin local metric learning. In *European Conference on Computer Vision*, pages 679–694. Springer, 2014.

[BYRN+99] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[CDK+06] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar) :551–585, 2006.

[CHL05] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005.

[CKTK10] Ratthachat Chatpatanasiri, Teesid Korsrilabutr, Pasakorn Tangchanachaianan, and Boonserm Kijsirikul. A new kernelization framework for mahalanobis distance learning algorithms. *Neurocomputing*, 73(10-12) :1570–1579, 2010.

[CM06] William W Cohen and Einat Minkov. A graph-search framework for associating gene identifiers with documents. *BMC bioinformatics*, 7(1) :440, 2006.

[CSSB09] Gal Chechik, Uri Shalit, Varun Sharma, and Samy Bengio. An online algorithm for large scale image similarity learning. In *Advances in Neural Information Processing Systems*, pages 306–314, 2009.

[CSSB10] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(Mar) :1109–1135, 2010.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3) :273–297, 1995.

[CVS11] Ramazan Gokberk Cinbis, Jakob Verbeek, and Cordelia Schmid. Unsupervised metric learning for face identification in tv video. In *2011 International Conference on Computer Vision*, pages 1559–1566. IEEE, 2011.

[CVZ14] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM Journal on Computing*, 43(6) :1831–1879, 2014.

[CY04] Hong Chang and Dit-Yan Yeung. Locally linear metric adaptation for semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 20. ACM, 2004.

[CYY+18] Gong Cheng, Ceyuan Yang, Xiwen Yao, Lei Guo, and Junwei Han. When deep learning meets metric learning : Remote sensing image scene classification via learning discriminative cnns. *IEEE transactions on geoscience and remote sensing*, 56(5) :2811–2821, 2018.

[DB17] Sebastijan Dumančić and Hendrik Blockeel. Demystifying relational latent representations. In *International Conference on Inductive Logic Programming*, pages 63–77. Springer, 2017.

[DCS17] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec : Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 135–144. ACM, 2017.

[DD08] Jason V Davis and Inderjit S Dhillon. Structured metric learning for high dimensional problems. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 195–203. ACM, 2008.

[Dem06] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan) :1–30, 2006.

[Den14] Li Deng. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, 3, 2014.

[DJLW08] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z Wang. Image retrieval : Ideas, influences, and trends of the new age. *ACM Computing Surveys (Csur)*, 40(2) :5, 2008.

[DKJ+07] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM, 2007.

[DLD+91] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital ener-

gies and hydrophobicity. *Journal of medicinal chemistry*, 34(2) :786–797, 1991.

[DR08] Luc De Raedt. *Logical and relational learning*. Springer Science & Business Media, 2008.

[DSSST10] John C Duchi, Shai Shalev-Shwartz, Yoram Singer, and Ambuj Tewari. Composite objective mirror descent. In *COLT*, pages 14–26, 2010.

[DTC12] Paramveer S Dhillon, Partha Talukdar, and Koby Crammer. Metric learning for graph-based domain adaptation. 2012.

[DvR08] Wilco Dekker and Ben van Raaij. *De elite*. De Volkskrant, 2008.

[Dže10] Sašo Džeroski. Relational data mining. *Data Mining and Knowledge Discovery Handbook*, pages 887–911, 2010.

[Emm12] Martin Emms. On stochastic tree distances and their training via expectation-maximisation. 2012.

[EP04] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004.

[FGKP99] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *IJCAI*, volume 99, pages 1300–1309, 1999.

[FHI06] Satoru Fujishige, Takumi Hayashi, and Shigueo Isotani. *The minimum-norm-point algorithm applied to submodular function minimization and linear programming*. Kyoto University. Research Institute for Mathematical Sciences [RIMS], 2006.

[Fri40] Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1) :86–92, 1940.

[FTSC10] Ruogu Fang, Kevin D Tang, Noah Snavely, and Tsuhan Chen. Towards computational models of kinship verification. In *2010 IEEE International conference on image processing*, pages 1577–1580. IEEE, 2010.

[Get07] Lise Getoor. *Introduction to statistical relational learning*. MIT press, 2007.

[GF18] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance : A survey. *Knowledge-Based Systems*, 151 :78–94, 2018.

[GHRS04] Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *Advances in neural information processing systems*, pages 513–520, 2004.

[GIL+15] Jennifer A Gillenwater, Rishabh K Iyer, Bethany Lusch, Rahul Kidambi, and Jeff A Bilmes. Submodular hamming metrics. In *NIPS*, pages 3141–3149, 2015.

[GLAB14] Damien Garreau, Rémi Lajugie, Sylvain Arlot, and Francis Bach. Metric learning for temporal sequence alignment. In *Advances in Neural Information Processing Systems*, pages 1817–1825, 2014.

[GR06] Amir Globerson and Sam T Roweis. Metric learning by collapsing classes. In *Advances in neural information processing systems*, pages 451–458, 2006.

[Gra16] Michel Grabisch. *Set Functions, Games and Capacities in Decision Making*. Springer, 2016.

[GTX11] Bo Geng, Dacheng Tao, and Chao Xu. Daml : Domain adaptation metric learning. *IEEE Transactions on Image Processing*, 20(10) :2980–2989, 2011.

[GVS09] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Is that you ? metric learning approaches for face identification. In *2009 IEEE 12th international conference on computer vision*, pages 498–505. IEEE, 2009.

[GXTL10] Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13(1) :113–129, 2010.

[HA15] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015.

[Hal98] Paul Richard Halmos. *Naive set theory*. Springer Science & Business Media, 1998.

[Ham50] Richard W Hamming. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2) :147–160, 1950.

[HCL06] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[HK16] F Maxwell Harper and Joseph A Konstan. The movielens datasets : History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4) :19, 2016.

[HLC10] Steven CH Hoi, Wei Liu, and Shih-Fu Chang. Semi-supervised distance metric learning for collaborative image retrieval and clustering. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 6(3) :18, 2010.

[HLLM06] Steven CH Hoi, Wei Liu, Michael R Lyu, and Wei-Ying Ma. Learning distance metrics with contextual constraints for image retrieval. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2072–2078. IEEE, 2006.

[HLYT14] Junlin Hu, Jiwen Lu, Junsong Yuan, and Yap-Peng Tan. Large margin multi-metric learning for face and kinship verification in the wild. In *Asian Conference on Computer Vision*, pages 252–267. Springer, 2014.

[HMBLM08] Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled faces in the wild : A database forstudying face recognition in unconstrained environments. 2008.

[Hop82] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8) :2554–2558, 1982.

[HRW10] Tamás Horváth, Jan Ramon, and Stefan Wrobel. Frequent subgraph mining in outerplanar graphs. *Data Mining and Knowledge Discovery*, 21(3) :472–508, 2010.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8) :1735–1780, 1997.

[IB13] Rishabh Iyer and Jeff Bilmes. The lovász-bregman divergence and connections to rank aggregation, clustering, and web ranking. *arXiv preprint arXiv :1308.5275*, 2013.

[ISF98] Yoshiharu Ishikawa, Ravishankar Subramanya, and Christos Faloutsos. Mindreader : Querying databases through multiple examples. 1998.

[JKD10] Prateek Jain, Brian Kulis, and Inderjit S Dhillon. Inductive regularized learning of kernel functions. In *Advances in neural information processing systems*, pages 946–954, 2010.

[JKDG09] Prateek Jain, Brian Kulis, Inderjit S Dhillon, and Kristen Grauman. Online metric learning and fast similarity search. In *Advances in neural information processing systems*, pages 761–768, 2009.

[JKG08] Prateek Jain, Brian Kulis, and Kristen Grauman. Fast image search for learned metrics. In *2008 IEEE Conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[JR10] Bernard J Jansen and Soo Young Rieh. The seventeen theoretical constructs of information searching and information retrieval. *Journal of the American Society for Information Science and Technology*, 61(8) :1517–1534, 2010.

[K+13] Brian Kulis et al. Metric learning : A survey. *Foundations and Trends® in Machine Learning*, 5(4) :287–364, 2013.

[KFD+07] Daphne Koller, Nir Friedman, Sašo Džeroski, Charles Sutton, Andrew McCallum, Avi Pfeffer, Pieter Abbeel, Ming-Fai Wong, David Heckerman, Chris Meek, et al. *Introduction to statistical relational learning*. MIT press, 2007.

[KG12] Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability : Practical Approaches to Hard Problems*, 3(19) :8, 2012.

[KLF01] Stefan Kramer, Nada Lavrač, and Peter Flach. Propositionalization approaches to relational data mining. In *Relational data mining*, pages 262–291. Springer, 2001.

[KPF+18] Sofia Ira Ktena, Sarah Parisot, Enzo Ferrante, Martin Rajchl, Matthew Lee, Ben Glocker, and Daniel Rueckert. Metric learning with spectral graph convolutions on brain connectivity networks. *NeuroImage*, 169 :431–442, 2018.

[Kru83] Joseph B Kruskall. The symmetric time warping algorithm : From continuous to discrete. *Time warps, string edits and macromolecules*, 1983.

[KS12] Gautam Kunapuli and Jude Shavlik. Mirror descent for metric learning : A unified approach. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 859–874. Springer, 2012.

[KSHG12] Hassan Khosravi, Oliver Schulte, Jianfeng Hu, and Tianxiang Gao. Learning compact Markov logic networks with decision trees. *Machine Learning*, 89(3) :257–277, 2012.

[KT03] James T Kwok and Ivor W Tsang. Learning with idealized kernels. In *ICML*, pages 400–407, 2003.

[KTS+12] Dor Kedem, Stephen Tyree, Fei Sha, Gert R Lanckriet, and Kilian Q Weinberger. Non-linear metric learning. In *Advances in Neural Information Processing Systems*, pages 2573–2581, 2012.

[Kul12] Brian Kulis. Metric learning : A survey. *Foundations and Trends in Machine Learning*, 5(4) :287–364, 2012.

[KW16] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. 2016.

[KXW] Dor Kedem, Zhixiang Eddie Xu, and Kilian Q Weinberger. Gradient boosted large margin nearest neighbors.

[LC18] Hoel Le Capitaine. Constraint selection in metric learning. *Knowledge-Based Systems*, 146 :91–103, 2018.

[Lev66] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[LGZ+12] Eric Yi Liu, Zhishan Guo, Xiang Zhang, Vladimir Jojic, and Wei Wang. Metric learning from relative comparisons by minimizing squared residual. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 978–983. IEEE, 2012.

[LHT17] Jiwen Lu, Junlin Hu, and Yap-Peng Tan. Discriminative deep metric learning for face and kinship verification. *IEEE Transactions on Image Processing*, 26(9) :4269–4282, 2017.

[Lic13] M. Lichman. UCI machine learning repository, 2013.

[LKB+17] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42 :60–88, 2017.

[Lov83] László Lovász. Submodular functions and convexity. In *Mathematical Programming The State of the Art*, pages 235–257. Springer, 1983.

[LTC13] Marc T Law, Nicolas Thome, and Matthieu Cord. Quadruplet-wise image similarity learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 249–256, 2013.

[LZT+13] Jiwen Lu, Xiuzhuang Zhou, Yap-Pen Tan, Yuanyuan Shang, and Jie Zhou. Neighborhood repulsed metric learning for kinship verification. *IEEE transactions on pattern analysis and machine intelligence*, 36(2) :331–345, 2013.

[M+97] Tom M Mitchell et al. Machine learning. 1997. *Burr Ridge, IL : McGraw Hill*, 45(37) :870–877, 1997.

[Mah36] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. National Institute of Science of India, 1936.

[May99] Wolfgang May. Information extraction and integration with FLORID : The MONDIAL case study. Technical Report 131, Universität Freiburg, Institut für Informatik, 1999. Available from `http://dbis.informatik.uni-goettingen.de/Mondial`.

[MBP12] Andrew McCallum, Kedar Bellare, and Fernando Pereira. A conditional random field for discriminatively-trained finite-state string edit distance. *arXiv preprint arXiv :1207.1406*, 2012.

[mcd] Mcdonald menus/food.

[ME11] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 437–452. Springer, 2011.

[ML10] Brian McFee and Gert R Lanckriet. Metric learning to rank. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 775–782, 2010.

[MLKCW03] Carla S Möller-Levet, Frank Klawonn, Kwang-Hyun Cho, and Olaf Wolkenhauer. Fuzzy clustering of short time-series and unevenly distributed sampling points. In *International Symposium on Intelligent Data Analysis*, pages 330–340. Springer, 2003.

[MLWG15] Jiangyuan Mei, Meizhu Liu, Yuan-Fang Wang, and Huijun Gao. Learning a mahalanobis distance-based dynamic time warping measure for multivariate time series classification. *IEEE transactions on Cybernetics*, 46(6) :1363–1374, 2015.

[MYC08] Erwan Moreau, François Yvon, and Olivier Cappé. Robust similarity measures for named entities matching. In *COLING 2008*, pages 593–600. ACL, 2008.

[Nic13] Maximilian Nickel. *Tensor factorization for relational learning*. PhD thesis, lmu, 2013.

[NLJ16] Romain Negrel, Alexis Lechervy, and Frédéric Jurie. Mlboost revisited : A faster metric learning algorithm for identity-based face retrieval. In *BMVC : British Machine Vision Conference 2016*, 2016.

[NPB+] Daniel Nyga, Mareike Picklum, Michael Beetz, et al. pracmln – markov logic networks in Python, 2013–. [Online ; accessed <date>].

[NTK11] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, volume 11, pages 809–816, 2011.

[NTK12] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing yago : scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, pages 271–280. ACM, 2012.

[NVR16] Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. Learning text similarity with siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 148–157, 2016.

[NW70] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3) :443–453, 1970.

[OS06] Jose Oncina and Marc Sebban. Learning stochastic edit distance : Application in handwritten character recognition. *Pattern recognition*, 39(9) :1575–1587, 2006.

[OSXJS16] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4004–4012, 2016.

[OWPB18] Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Deep metric learning with bier : Boosting independent embeddings robustly. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[PARS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk : Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[PC11] James Petterson and Tibério S Caetano. Submodular multi-label learning. In *Advances in Neural Information Processing Systems*, pages 1512–1520, 2011.

[PCL18] Jiajun Pan, Hoel Le Capitaine, and Philippe Leray. Relational constraints for metric learning on relational data. *arXiv preprint arXiv :1807.00558*, 2018.

[PLC19a] Jiajun Pan and Hoel Le Capitaine. Metric learning with relational data. In *27th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 367–372, 2019.

[PLC19b] Jiajun Pan and Hoel Le Capitaine. Metric learning with submodular functions. In *27th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 355–360, 2019.

[PLCL] Jiajun Pan, Hoel Le Capitaine, and Philippe Leray. Multi-relational constraints in metric learning. *Conférence sur L'Apprentissage Automatique 2018*.

[PW10] Shibin Parameswaran and Kilian Q Weinberger. Large margin multi-task metric learning. In *Advances in neural information processing systems*, pages 1867–1875, 2010.

[QC12] Qiang Qian and Songcan Chen. Metric learning across heterogeneous domains by respectively aligning both priors and posteriors. *arXiv preprint arXiv :1208.1829*, 2012.

[QG09] Ali Mustafa Qamar and Eric Gaussier. Online and batch learning of generalized cosine similarities. In *2009 Ninth IEEE International Conference on Data Mining*, pages 926–931. IEEE, 2009.

[QGCL08] Ali Mustafa Qamar, Eric Gaussier, Jean-Pierre Chevallet, and Joo Hwee Lim. Similarity learning for nearest neighbor classification. In *2008 Eighth IEEE International Conference on Data Mining*, pages 983–988. IEEE, 2008.

[RD06] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2) :107–136, 2006.

[RH05] Havard Rue and Leonhard Held. *Gaussian Markov random fields : theory and applications*. Chapman and Hall/CRC, 2005.

[RHW+88] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3) :1, 1988.

[RST10] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 81–90. ACM, 2010.

[RY98] Eric Sven Ristad and Peter N Yianilos. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5) :522–532, 1998.

[SA13] Murat Semerci and Ethem Alpaydın. Mixtures of large margin nearest neighbor classifiers. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 675–688. Springer, 2013.

[SB17] Jan Struyf and Hendrik Blockeel. Relational learning. *Encyclopedia of Machine Learning and Data Mining*, pages 1090–1096, 2017.

[SBS14] Yuan Shi, Aurélien Bellet, and Fei Sha. Sparse compositional metric learning. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[Sel77] Stanley M Selkow. The tree-to-tree editing problem. *Information processing letters*, 6(6) :184–186, 1977.

[SH07] Ruslan Salakhutdinov and Geoff Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Artificial Intelligence and Statistics*, pages 412–419, 2007.

[SHJ11] Blake Shaw, Bert Huang, and Tony Jebara. Learning a distance metric from a network. In *Advances in Neural Information Processing Systems*, pages 1899–1907, 2011.

[SHZL17] Jingyi Shen, Weiping Huang, Dongyang Zhu, and Jun Liang. A novel similarity measure model for multivariate time series based on lmnn and dtw. *Neural Processing Letters*, 45(3) :925–937, 2017.

[SJ04] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. *Advances in neural information processing systems (NIPS)*, page 41, 2004.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet : A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[SNB+08] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3) :93, 2008.

[SSHE10] Jimeng Sun, Daby Sow, Jianying Hu, and Shahram Ebadollahi. Localized supervised metric learning on temporal physiological data. In *2010 20th International Conference on Pattern Recognition*, pages 4149–4152. IEEE, 2010.

[SSM98] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5) :1299–1319, 1998.

[SSM99] Bernhard Schölkopf, Alexander J Smola, and Klaus-Robert Müller. Advances in kernel methods. chapter kernel principal component analysis, 1999.

[SSSN04] Shai Shalev-Shwartz, Yoram Singer, and Andrew Y Ng. Online and batch learning of pseudo-metrics. In *Proceedings of the twenty-first international conference on Machine learning*, page 94. ACM, 2004.

[SSSSC11] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos : Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1) :3–30, 2011.

[Sug06] Masashi Sugiyama. Local fisher discriminant analysis for supervised dimensionality reduction. In *Proceedings of the 23rd international conference on Machine learning*, pages 905–912. ACM, 2006.

[SVA06] Hiroto Saigo, Jean-Philippe Vert, and Tatsuya Akutsu. Optimizing amino acid substitution matrices with a local alignment kernel. *BMC bioinformatics*, 7(1) :246, 2006.

[SW+81] Temple F Smith, Michael S Waterman, et al. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1) :195–197, 1981.

[Tak09] Atsuhiro Takasu. Bayesian similarity model estimation for approximate recognized text search. In *2009 10th International Conference on Document Analysis and Recognition*, pages 611–615. IEEE, 2009.

[TQW+15] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line : Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.

[TWR+16] Oren Tadmor, Yonatan Wexler, Tal Rosenwein, Shai Shalev-Shwartz, and Amnon Shashua. Learning a metric embedding for face recognition using the multibatch method. *arXiv preprint arXiv :1605.07270*, 2016.

[VA15] Vo Thanh Vinh and Duong Tuan Anh. Compression rate distance measure for time series. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10. IEEE, 2015.

[VCZ11] Jan Vondrák, Chandra Chekuri, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 783–792. ACM, 2011.

[WBS06] Kilian Q Weinberger, John Blitzer, and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1473–1480. MIT Press, 2006.

[WCL13] Hao Wang, Binyi Chen, and Wu-Jun Li. Collaborative topic regression with social regularization for tag recommendation. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[WHJ+10] Lei Wu, Steven CH Hoi, Rong Jin, Jianke Zhu, and Nenghai Yu. Learning bregman distance functions for semi-supervised clustering. *IEEE Transactions on Knowledge and Data Engineering*, 24(3) :478–491, 2010.

[Win90] William E Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. 1990.

[WJH+09] Lei Wu, Rong Jin, Steven C Hoi, Jianke Zhu, and Nenghai Yu. Learning bregman distance functions and its application for semi-supervised cluste-

ring. In *Advances in neural information processing systems*, pages 2089–2097, 2009.

[WS08a] Kilian Q Weinberger and Lawrence K Saul. Fast solvers and efficient implementations for distance metric learning. In *Proceedings of the 25th international conference on Machine learning*, pages 1160–1167. ACM, 2008.

[WS08b] Xiaoqing Weng and Junyi Shen. Classification of multivariate time series using locality preserving projections. *Knowledge-Based Systems*, 21(7) :581–587, 2008.

[WS09] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10(Feb) :207–244, 2009.

[WT07] Kilian Q Weinberger and Gerald Tesauro. Metric learning for kernel regression. In *Artificial Intelligence and Statistics*, pages 612–619, 2007.

[WWY18] Qi Wang, Jia Wan, and Yuan Yuan. Locality constraint distance metric learning for traffic congestion detection. *Pattern Recognition*, 75 :272–281, 2018.

[WZ07] Fei Wang and Changshui Zhang. Feature extraction by maximizing the average neighborhood margin. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[WZLQ16] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European conference on computer vision*, pages 499–515. Springer, 2016.

[XHL+19] Xing Xu, Li He, Huimin Lu, Lianli Gao, and Yanli Ji. Deep adversarial metric learning for cross-modal retrieval. *World Wide Web*, 22(2) :657–672, 2019.

[XNJR02] Eric P Xing, Andrew Y Ng, Michael I Jordan, and Stuart Russell. Distance metric learning with application to clustering with side-information. In *NIPS*, volume 15, page 12, 2002.

[XNJR03] Eric P Xing, Andrew Y Ng, Michael I Jordan, and Stuart Russell. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, 15 :505–512, 2003.

[YB15] Jiaqian Yu and Matthew Blaschko. Learning submodular losses with the lovasz hinge. In *ICML*, pages 1623–1631, 2015.

[YHC09] Yiming Ying, Kaizhu Huang, and Colin Campbell. Sparse metric learning via smooth optimization. In *Advances in neural information processing systems*, pages 2214–2222, 2009.

[YJ06] Liu Yang and Rong Jin. Distance metric learning : A comprehensive survey. *Michigan State Universiy*, 2(2) :4, 2006.

[YLLL14] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Deep metric learning for person re-identification. In *2014 22nd International Conference on Pattern Recognition*, pages 34–39. IEEE, 2014.

[YTPM11] Wen-tau Yih, Kristina Toutanova, John C Platt, and Christopher Meek. Learning discriminative projections for text similarity measures. In *Proceedings of the fifteenth conference on computational natural language learning*, pages 247–256. Association for Computational Linguistics, 2011.

[YYH+11] Daren Yu, Xiao Yu, Qinghua Hu, Jinfu Liu, and Anqi Wu. Dynamic time warping constraint learning for large margin nearest neighbor classification. *Information Sciences*, 181(13) :2787–2796, 2011.

[YZH+15] Ian En-Hsu Yen, Kai Zhong, Cho-Jui Hsieh, Pradeep K Ravikumar, and Inderjit S Dhillon. Sparse linear programming via primal and dual augmented coordinate descent. In *NIPS*, pages 2368–2376, 2015.

[ZHS16] Pourya Zadeh, Reshad Hosseini, and Suvrit Sra. Geometric mean metric learning. In *International Conference on Machine Learning*, pages 2464–2471, 2016.

[ZMKL05] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32. ACM, 2005.

[ZMW+09] Zheng-Jun Zha, Tao Mei, Meng Wang, Zengfu Wang, and Xian-Sheng Hua. Robust distance metric learning with auxiliary knowledge. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[ZPX13] Xiaohua Zhai, Yuxin Peng, and Jianguo Xiao. Heterogeneous metric learning with joint graph regularization for cross-media retrieval. In *Twenty-seventh AAAI conference on artificial intelligence*, 2013.

[ZXI16] Jiaping Zhao, Zerong Xi, and Laurent Itti. metricdtw : local distance metric learning in dynamic time warping. *arXiv preprint arXiv :1606.03628*, 2016.

UNIVERSITE BRETAGNE LOIRE MATHSTIC

UNIVERSITÉ DE NANTES

Titre : Métrique d'apprentissage pour les données structurées

Mot clés : Apprentissage métrique, Fonction sous-modulaire, Apprentissage relationnel

Resumé : L'apprentissage à distance métrique est une branche de l'apprentissage par re-présentation des algorithmes d'apprentissage automatique. Nous résumons le développement et la situation actuelle de l'algorithme actuel d'apprentissage à distance métrique à partir des aspects de la base de données plate et de la base de données non plate. Pour une série d'algorithmes basés sur la distance de Mahalanobis pour la base de données plate qui ne parvient pas à exploiter l'intersection de trois dimensions ou plus, nous proposons un algorithme d'apprentissage métrique basé sur la fonction sous-modulaire. Pour le manque d'algorithmes d'apprentissage métrique pour les bases de données relationnelles dans des bases de données non plates, nous proposons LSCS (sélection de contraintes relationnelles de force relationnelle) pour la sélection de contraintes pour des algorithmes d'apprentissage métrique avec informations parallèles et MRML (Multi-Relation d'apprentissage métrique) qui somme la perte des contraintes relationnelles et les contraintes d'etiquetage. Grâce aux expériences de conception et à la vérification sur la base de données réelle, les algorithmes proposés sont meilleurs que les algorithmes actuels.

Title : Metric learning for structured data

Keywords : Metric Learning, Submodular Function, Relation learning

Abstract : Metric distance learning is a branch of re-presentation learning in machine learning algorithms. We summarize the development and current situation of the current metric distance learning algorithm from the aspects of the flat database and non-flat database. For a series of algorithms based on Mahalanobis distance for the flat da-

tabase that fails to make full use of the intersection of three or more dimensions, we propose a metric learning algorithm based on the submodular function. For the lack of metric learning algorithms for relational databases in non-flat databases, we propose LSCS(Relational Link-strength Constraints Selection) for selecting constraints for metric learning algorithms with side information and MRML (Multi-Relation Metric Learning) which sums the loss from relationship constraints and label constraints. Through the design experiments and verification on the real database, the proposed algorithms are better than the current algorithms.