Metric Learning with Submodular Functions

Jiajun Pan, Hoel Le Capitaine

Metric Learning with Submodular Functions

Jiajun Pan and Hoel Le Capitaine

University of Nantes, LS2N UMR CNRS 6004, Nantes, France

Abstract

Most of the metric learning mainly focuses on using single feature weights with L_p norms, or the pair of features with Mahalanobis distances to learn the similarities between the samples, while ignoring the potential value of higher-order interactions in the feature space. In this paper, we investigate the possibility of learning weights to coalitions of features whose cardinality can be greater than two, with the help of set-functions. With the more particular property of submodular set-function, we propose to define a metric for continuous features based on Lovasz extension of submodular functions, and then present a dedicated metric learning approach. According to the submodular constraints, it naturally leads to a higher complexity price so that we use the ξ-additive fuzzy measure to decrease this complexity, by reducing the order of interactions that are taken into account. This approach finally gives a computationally, feasible problem. Experiments on various datasets show the effectiveness of the approach.

Keywords:
Metric learning, nonlinear metric, submodular functions

1. Introduction

Since the seminal paper of Xing et al. [35], metric learning has attracted much interest in the machine learning community. Distances and metrics are widely used in the computer science and mathematics areas. The performance of many machine learning algorithms is strongly related to the chosen distance. For instance, K-means relies on distance measurements between data points, nearest-neighbours classifiers use a metric to identify the nearest neighbours and algorithms such as SVM need a good kernel to find maximum-margin hyperplane. It is now widely known that using a convenient metric, or similarity measure, in machine learning algorithms is fundamental [34, 3, 22].

A common practice consists in considering the Mahalanobis metric defined by

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

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are m-dimensional feature vectors of samples and M is a positive semi-definite matrix of parameters (otherwise, it is not a metric, but still can be used as a dissimilarity measure, see, e.g. [8]) that can be learned. Using such a metric is equivalent to perform a linear projection of the data by the matrix decomposition $M = L^T L$, and then use the conventional Euclidean distance in this new feature space. Using a linear metric allows writing simple, convex, objective functions where one of the crucial points is to maintain M positive semi-definite (PSD), which can be done by projection onto the PSD cone through eigen-decomposition. Many metric learning algorithms are based on this model, such as the Large-Margin Nearest Neighbors (LMNN) [34] and Information Theory Measurement Learning (ITML) [11].

In practice, however, the distribution of the data is often complex, so that nonlinear approaches have been proposed. In the nonlinear case of metric learning algorithms, the general principle is to use a nonlinear embedding or mapping function before the linear projection, e.g. kernelization. In [9], the authors directly model nonlinear metrics with a discriminative objective, while the authors of [7] propose a kernelization of a linear metric. In χ^2-LMNN[19], authors extend the linear metric learning approach LMNN to χ^2- distances specialized for histogram data and gave a gradient boosted LMNN for nonlinear mapping combined with a traditional Euclidean distance. Another possible approach of

learning metric for nonlinear datasets is to consider one local metric for different regions of the feature space. Learning multiple linear metrics has the capacity to capture the heterogeneity of complex tasks. Mul-LMNN (Multiple LMNN metric learning) [33] is the multiple metrics learning version of LMNN. Mul-LMNN separates the training datasets in several clusters with supervised (labels) or unsupervised (k-means) information. Then, for each cluster, a metric is learned with LMNN. Notice that, this could be considered as local metric learning for each cluster, the local distance depends on which nodes are in which cluster and the global distance could not be symmetric if the pair of data nodes are in different clusters [37]. Notice also that several nonlinear metric learning algorithms are neither linear metric with different nonlinear transformations or several linear local metrics, but a direct optimization of a nonlinear metric. A typical example is LSMD (Learning Similarity Metric Discriminatively) [10], where the authors proposed the first nonlinear form metric learning with the Siamese architecture of a CNN (Convolutional Neural Network)[26]. In this paper, authors train a pair of CNNs sharing the parameters with the selected constraints. The CNN leads to a high computational cost, but the authors demonstrate its advantage for face verification tasks.

Nonetheless, most of the metric distance learning algorithms are based on the Mahalanobis distance model, whether linear or nonlinear [3]. Having a closer look at the Mahalanobis metric shows that it consists of giving weight to all possible feature pairs. The use of the inverse of the covariance matrix for M, i.e., the historical Mahalanobis distance, implies that the weight of a feature pair is proportional to the cofactor of the features. Although the cofactor of a pair of features depends on all other pairwise covariances, the actual distance definition only considers the pairwise combination of features, whereas p-tuple-wise combinations bring a lot more information.

In this paper, we are investigating the possibility of giving (and learning) weights to coalitions of features whose cardinality can be greater than two. To this aim, we propose to consider to define a metric using a set function. A set function is a function whose the input is a set, and whose output is a scalar evaluating of the set. We propose to use set-functions as a mapping from subsets of features to a weight value used in the definition of the metric.

Let us consider a simple 3-dimensional example.

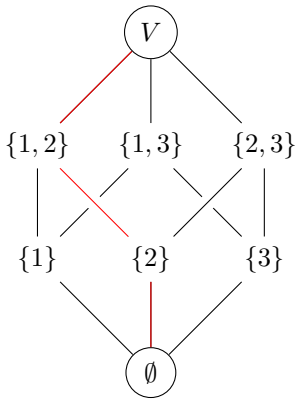


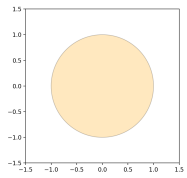
Figure 1: Hasse diagram using set-functions on a 3-dimensional problem.

In Figure 1, one can draw its corresponding Hasse diagram. If we learn a Mahalanobis metric on this entity, for each single dimension $\{1\}$,$\{2\}$,$\{3\}$ and the pairs of dimension $\{1,2\}$,$\{1,3\}$,$\{2,3\}$, the transformation matrix will give weights while the more complex coalition $\{1,2,3\}$ gets ignored. We consider a set-function $f(S)$, in this case $S \subseteq \{1,2,3\}$, so that each subset of the powerset of V can be given a weight.

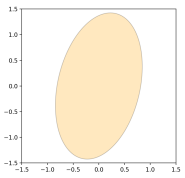
	$f(\{1\})$	$f(\{2\})$	$f(\{1,2\})$
f_1	0.5	0.5	1
f_2	0.25	0.25	0.25
f_3	0.2	0.5	0.7
f_4	0.5	0.5	0.7

Table 1: Values of the set-functions used in Figure 2.

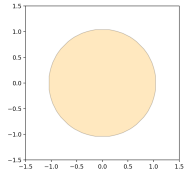
The red path in Figure 1 corresponds to one possible use of learned weights for computing our proposed distance, that is using directly Lovasz extension $L_f(\mathbf{x})$ [27]. In practice, we see that computing a distance between two objects using the proposed d_f (see Eq. 12) reduces to a weighted path from $\emptyset$ to V in the lattice. In Figure 2, the unit balls for various distances, including the one proposed (Lovasz distance) in this paper are given, using the set-functions given in Table 1. As can be seen, it allows obtaining a wide range of convex shapes, generalizing both Euclidean distance and Mahalanobis distance and it is worth to mention that using a set-function which is not submodular would provide a nonconvex shape, i.e., not a proper metric. Notice that using a modular function such as f_1 provides the Euclidean distance as expected since the Lovasz



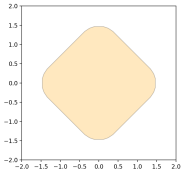
(a) Euclidean distance



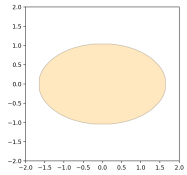
(b) Mahalanobis distance



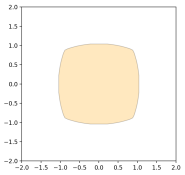
(c) Lovasz distance with f_1



(d) Lovasz distance with f_2



(e) Lovasz distance with f_3



(f) Lovasz distance with f_4

Figure 2: 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 1, respectively.

extension reduces to a linear function $f^t(\mathbf{x})$ in that case.

The shapes versatility confirms the generalization ability of such distance. However, it also opens several questions. Especially how to learn such a complex function?

Our work primarily deals with a linear programming approach where the minimum norm point algorithm is used on submodular functions. More precisely, we provide the new following contributions:

- through the use of a submodular set-function and its Lovasz extension, we propose to define a norm, and therefore a distance metric. The benefit of this formulation is that the distance allows considering high-order (up to m) interactions between features, as opposed to 1-order feature interaction for L_p norms, and 2-order feature interaction for Mahalanobis distances (Proposition 1).

- we propose an algorithm LEML (Lovasz Extension Metric Learning) that allows learning proposed metric, by using linear programming and convenient writing of submodular constraints imposed on the set-function as well as the relative constraints considered in metric learning.

- we present LEML-ξ with a way of decreasing the (time) complexity of the model while keeping/improving performances, by introducing ξ-additive set-functions, and constraint simplification by the help of the inverse set-function into the optimization problem.

The python code of our proposed algorithms can be found on `https://github.com/lynnoak/submodular_metric`.

The rest of this paper is organized as follows: In Section 2, we firstly introduce background knowledge about metric learning algorithms. Then we list several classical metric learning algorithms and related current metric learning algorithms. This section is not only a survey about metric learning but also an introduction to the algorithms used as the baseline for comparing with our proposed algorithms in Section 5experiments. In Section 3, we present the mathematical definition of the set function and the submodular function. Then further introduce their nature and related applications. This section focuses on the reasons we choose the submodular function to define the metric. Next, in

Section 4, we propose our Lovasz extension metric learning algorithm. Firstly, we recall the definition Eventually, the experiment design and result of the proposed algorithm and comparison approaches are described in Section 5. Conclusion and perspectives are given in Section 6.

2. Metric Learning

Most of the propositions of metric learning follow these 3 steps long general process [35, 34, 11, 22]:

- Metric definition: Depending on the idea or the peculiarity of the dataset, construct a new distance function $d'(x, y)$ for assessing the similarity between the samples x and y. Usually, $d'(x, y) = d(f(x), f(y))$ and $d(x, y)$ is a current distance (for example, Minkowski distance, Euclidean distance or anything similar) and $f(S)$ is a transfer function mapping the original feature space to a latent feature space.
- Constraint selection: Depending on the learning task and the availability of the target label or other feedback, the metric learning algorithm requires select constraints $\mathcal{C}$ from the information of samples to learning the new metric.
- Learning metric model: Generally, for learning a metric, a loss function 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 $\mathcal{C}$, the other one is the regularization $r(M)$ with a balance parameter λ.

For the learning model, there have been a number of approaches in the past fifteen years. A popular formulation of metric learning using the Mahalanobis distance d_M^2 (see Eq. 1) is to find M such that it minimizes $L(M) = \ell(M, \mathcal{C}) + \lambda r(M)$, where ℓ is a loss function penalizing unsatisfied constraints, with $\mathcal{C}$ the set of constraints. λ is a trade-off parameter between the regularization term and the loss, and $r(M)$ is a regularization function. If feasible, this model is generally casted as a constrained optimization problem

$$\begin{aligned} &\min r(M) \\ &\text{s.t. } \ell(M, i) \leq 0, \forall i \in \mathcal{C} \end{aligned} \tag{2}$$

Most of the metric learning algorithms are using the Mahalanobis metric definition with a different set of constraints selections and regularization terms [35, 34, 11, 22].

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 "metrics" in the metric learning algorithm only meet the above definitions under finite conditions or do not meet the third triangular inequality condition. These "metrics" can be called (dis)-similarity, and the similarity is used for special learning tasks. Despite the fact it does not define a strict proper metric, it may however also reach the desired learning objectives.

Depending on the metric learning approach, constraints are given under different forms. Among them, one can distinguish two popular ones: pairwise constraints and relative constraints.

- Pairwise constraints consists in considering pair of points that should be considered similar ($\mathcal{S}$) or dissimilar ($\mathcal{D}$) to build the complete constraint set $\mathcal{C} = \mathcal{S} \cup \mathcal{D}$

$$d(\mathbf{x}_i, \mathbf{x}_j) \leq u, (i, j) \in \mathcal{S}$$

$$d(\mathbf{x}_i, \mathbf{x}_j) \geq l, (i, j) \in \mathcal{D}$$

  Generally, samples are considered as similar if they share the same target label y, and dissimilar otherwise. More precisely, considering

a generic distance d, we require $d(\mathbf{x}_i, \mathbf{x}_j)$ to be large if (i, j) belongs to the set of dissimilar observations $\mathcal{D}$. On the other hand, we require $d(\mathbf{x}_i, \mathbf{x}_j)$ to be low if (i, j) belongs to $\mathcal{S}$. The sets $\mathcal{S}$ and $\mathcal{D}$ are used as the input constraints.

- Relative constraints relies on relative comparisons between samples.

$$d(\mathbf{x}_i, \mathbf{x}_j) \leq d(\mathbf{x}_i, \mathbf{x}_k), \quad (i, j, k) \in \mathcal{R}$$

In this setting, we consider triples $(i, j, k) \in \mathcal{R}$, where we consider that $\mathbf{x}_i$ and $\mathbf{x}_j$ are more similar than $\mathbf{x}_i$ and $\mathbf{x}_k$. Relative triplets (i, j, k) can be trivially obtained from pairwise constraints, where we sample (i, j) from $\mathcal{S}$, and (i, k) (or (j, k)) from $\mathcal{D}$. In the sequel, we are using relative constraints $\mathcal{R}$.

Notice that, depending on the learning task and the availability of the target label or other feedback, the metric learning can be supervised or unsupervised, with different approaches to select the constraints. In this paper, we only consider the supervised case, which means we separate the similar and dissimilar constraints set with the information of the target label. However, there are many sources of information that one could use, such as rank, order, time-series, etc. Recently, other approaches have been proposed for active constraint selection in metric learning, see [23].

The other important part of the loss function is the regularization. As most of the machine learning algorithms, regularization limits the complexity of the model to avoid over-fitting and obtain better generalization [2, 4].

As shown in Table 2, the different regularizers lead to different properties. 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.

Now, we briefly introduce several metric learning algorithms that we use to compare with our approach in the experiments.

2.1. Large-Margin Nearest Neighbors

Large-Margin Nearest Neighbors (LMNN) [34] is one of the most popular method for metric learning. It relies on relative constraints

$$\mathcal{C} = \{(i, j, k) | \forall (i, j, k) \in \mathcal{C}, d(\mathbf{x}_i, \mathbf{x}_j) + m \leq d(\mathbf{x}_i, \mathbf{x}_k))\}$$

and propose a regularization

$$tr(MC) = \sum_{(i,j)\in\mathcal{S}} d_M(\mathbf{x}_i, \mathbf{x}_j)$$

based on [36] which is a linear minimization of new metrics between similar samples. The loss function is as follow:

$$L(M) = \sum_{(i,j,k)\in\mathcal{D}} [d_M(\mathbf{x}_i, \mathbf{x}_k) - d_M(\mathbf{x}_i, \mathbf{x}_j) - m]_+ + \lambda \sum_{(i,j)\in\mathcal{S}} d_M(\mathbf{x}_i, \mathbf{x}_j) \tag{3}$$

where m usually set to 1.

The two parts of this function show the two objectives: $\sum_{(i,j,k)\in\mathcal{D}}[d_M(\mathbf{x}_i, \mathbf{x}_k) - d_M(\mathbf{x}_i, \mathbf{x}_j) - m]$ indicates that the dissimilar neighbor point should be farther than the similar one; while $\sum_{(i,j)\in\mathcal{S}} d_M(\mathbf{x}_i, \mathbf{x}_j)$ tries to keep the similar neighbor as close as possible.

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. In the original version of LMNN, they select the relative constraints with the supervised information, which is Target label. Therefore, there are many extensions to the LMNN, such as for nonlinearization, localization, online learning. Among the algorithms that will be discussed in the following sections, one of them is the nonlinear version of LMNN.

2.2. Information Theoretic Metric Learning

Another popular metric learning method is Information Theory Measurement Learning (ITML) [11]. The author uses information theory to regularize the loss function with $tr(M) - \log\det(M)$ and select similarity and dissimilarity constraints. They expect to minimize the Kullback-Leibler divergence of the PSD matrix M of the learning metric and the initial matrix M_0 given the constraints. The final loss function based on $\log\det$ divergence is given by:

$$L(M) = \sum \ell(X^T M X) + \lambda(tr(M) - \log\det(M)) \tag{4}$$

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

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, log det, divergence

Table 2: The difference between different regularizations, where C is a given matrix, for example an identity matrix for nuclear-norm regularizer)

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

Like LMNN, ITML is also a classic algorithm that is inevitable in metric distance learning algorithms. It is worth mentioning that although under many of the same conditions, ITML is more computationally complex and cost longer time than LMNN, however, the ITML algorithm has greater flexibility in selecting constraints and input samples, and has higher stability. In the ITML paper, the selection of relative constraints is not only by the target label but also can be selected by other methods. Moreover, the authors not only proposed a regularization method based on KL divergence but also proposed kernel metric learning, which extended the possibility of ITML as a nonlinear learning algorithm. ITML also has many different extensions like LMNN.

2.3. Least Squares Metric Learning

Least Squares Metric Learning (LSML) [25] uses relative constraints as LMNN and the log det divergence regularization as ITML. In addition to the combination of the two papers, LSML proposes a new method of encoding constraint loss, which uses a squared hinge function and optimizes the residual sum of squares under the constraints.The squared hinge function $f_{hinge}()$ is defines as:

$$f_{hinge}(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x^2 & \text{if } x > 0 \end{cases}$$

The relative constraints is written as

$$d_M(\mathbf{x}_k, \mathbf{x}_l) \geq d_M(\mathbf{x}_i, \mathbf{x}_j), \forall (i,j,k,l) \in \mathcal{C},$$

where the constraints set $\mathcal{C}$ is selected by the pair (i,j), which is more similar than the pair (k,l). The final loss function is then written as follows:

$$L(M) = \sum_{l(i,j,k,l)\in\mathcal{C}} f_{hinge}(d_M(\mathbf{x}_i, \mathbf{x}_j) \quad (5)$$
$$-d_M(\mathbf{x}_k, \mathbf{x}_l)) + \lambda(tr(M) - \log\det(M))$$

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.

2.4. Local Fisher Discriminant Analysis

LFDA [31] 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}:

$$T_{lfda} = argmax_{T\in\mathbf{R}^{d\times m}}(T^T B' T) \quad (6)$$

subject to $T^T W' T$, and where 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, the authors did not compare with other metric 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 learning algorithms of the same period.

2.5. Geometric Mean Metric Learning

Geometric Mean Metric Learning (GMML), [40] is a new method for forming Euclidean metrics

based on the first principle of intuitionistic geometric inference. The authors use pairwise constraints, and the loss function is given by

$$L(M) = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} d_M(\mathbf{x}_i,\mathbf{x}_j) + \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} d_{M^{-1}}(\mathbf{x}_i,\mathbf{x}_j) \tag{7}$$

The main difference is that they use geodesic convexity as the generalization of ordinary (linear) convexity to (nonlinear) manifolds and metric spaces. To this aim, the authors define $A\zeta B = A^{1/2}(A^{-1/2}BA^{-1/2})^t A^{1/2}$ with $t \in [0,1]$ as the step length of a geodesic path joining A and B.

They finally obtain the following matrix update:

$$M = (S + \lambda M_0^{-1})^{-1}\zeta(D + \lambda M_0)$$

where $S = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top$,
$D = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top$,
and λ is the regularization parameter.

GMML has several very attractive features through the Riemannian geometry of the positive definite matrix, which has inherent geometric appeal and is easy to interpret. Compared to the widely used LMNN and ITML methods, GMML is faster than several orders of magnitude.

2.6. Gradient boosted LMNN

In [19, 20], the authors introduce gradient boosted LMNN (GB-LMNN), which are explicitly designed to be nonlinear and easy to use. GB-LMNN applies gradient-boosting to learn nonlinear mappings directly in function space.

To generalize the LMNN objective function to a non-linear transformation $\phi(x)$, the Euclidean distance with transformation is denoted as:

$$d_\phi(\mathbf{x}_i,\mathbf{x}_j) = (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^\top(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)), \tag{8}$$

and extend the loss function of LMNN as follow:

$$L(\phi) = \sum_{(i,j,k)\in\mathcal{D}} [d_\phi(\mathbf{x}_i,\mathbf{x}_k) - d_\phi(\mathbf{x}_i,\mathbf{x}_j) - m]_+ + \lambda \sum_{(i,j)\in\mathcal{S}} d_\phi(\mathbf{x}_i,\mathbf{x}_j) \tag{9}$$

The transformation $\phi(x)$ is defined with gradient boosted method as an additive function: $\phi_t(x) = \phi_{t-1}(x) + \alpha h_t(x)$ where $h_t(x) \approx \arg\min_{h\in tr} L(\phi_{t-1}(x) + \alpha h_t(x))$ initialize with the linear transformation learned by LMNN, ϕ_0. The α is the learning rate and tr denotes the set of all regression trees of limited depth r.

As an extension of LMNN, GB-LMNN takes advantage of the robustness, speed, parallelism, and insensitivity to a single additional hyperparameter.

3. Submodular Function and Extension

In this paper, we consider the possibility of giving weights to coalitions of features whose cardinality can be greater than two. To this aim, we use set-functions $f(S) : 2^V \to [0,1]$, which maps subsets S of a ground set V to unit interval values. Notice that, Notice that, in general, the definition of set-function is $f(S) : i|i \in V \to \mathbb{R}$, and the codomain of $f(S)$ is not restricted to be the unit interval but belongs to $\mathbb{R}$. This definition allows the association of weights to subsets, in our case, subsets of features. The possibility of giving a weight to each element of the Hasse diagram of the feature space allows obtaining a very flexible measure.

For assuring the condition of metric or norm, we select a special set function, that is submodular function. The mathematical definition of the submodular function is as following:

A set function $f(S)$ is submodular if and only if $\forall\mathcal{S}_1,\mathcal{S}_2 \subseteq V$,

$$f(S_1) + f(S_2) \geq f(S_1 \cup \mathcal{S}_2) + f(S_1 \cap \mathcal{S}_2) \tag{10}$$

In terms of optimization, submodular functions can be seen as the discrete equivalent of continuous convex functions. An alternative definition [32] of submodularity is given as $f(S_1 \cup \{j\}) - f(S_1) \geq f(S_2 \cup \{j\}) - f(S_2)$, 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 $f(S)$ as much as adding the same element j to a smaller set $\mathcal{S}_1$. Figure 3 shows example for submodularity as $f(\mathcal{S}_{a,b}\cup\{i\}) - f(\mathcal{S}_{a,b}) \geq f(\mathcal{S}_{a,b,c,d}\cup\{i\}) - f(\mathcal{S}_{a,b})$. This property is known as the diminishing return [30] 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 [32].

More recently, submodular functions have attracted interest in the machine learning community

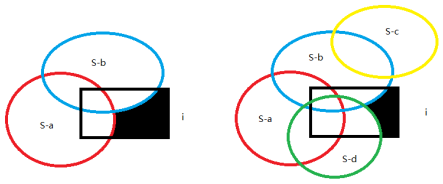


Figure 3: $f(\mathcal{S}_{a,b} \cup \{i\}) - f(\mathcal{S}_{a,b}) \geq f(\mathcal{S}_{a,b,c,d} \cup \{i\}) - f(\mathcal{S}_{a,b})$

(see, e.g., [1]), 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.

3.1. Lovasz Extension

For optimizing a submodular minimum or maximum problem, we generally apply for the set-function extension. Notice 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.

Lovasz extension [27] (also known as the Choquet integral) is one of the most popular extensions that help to solve the set-function minimization, in particular, the submodular function minimization. Lovasz extension 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:

$$L_f(\mathbf{x}) = \int_0^{+\infty} f(\{x \geq z\})dz + \int_{-\infty}^{0} [f(\{x \geq z\}) - f(V)]dz \tag{11}$$

or, in the discrete case,

$$L_f(\mathbf{x}) = \sum_{i=1}^{2^m} x_{(i)} \left[f(\{j | x_j \geq x_{(i)}\}) - f(\{j | x_j \geq x_{(i+1)}\}) \right]$$

where $x_{(i)}$ denotes a nondecreasing permutation of the input vector $\mathbf{x}$ such that $x_{(2^m)} \geq \cdots \geq x_{(1)}$ and $x_{(2^m+1)} = \infty$ 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 2 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

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

An interesting property of the Lovasz extension is its ability to draw a link between set-functions and convex functions. With [1, 27], we know that a set-function $f(S)$ is submodular, if and only if $L_f(\mathbf{x})$ is convex. 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 $f(S)$. The minimum values of $f(S)$ and f^- are equal. If S is a minimizer of $f(S)$, $1_{\mathcal{S}}$ 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 $f(S)$.

For a submodular set-function $f(S)$, the Lovasz extension $L_f(\mathbf{x})$ is non-decreasing, and the convex closure are one and the same [1]. So the minimizer of the submodular function is equal to the minimizer of the Lovasz extension.

3.2. Related Machine Learning Approaches based on Submodular Function

Through the years, there have been several propositions for both submodular function minimization [15], and the generally NP-hard submodular maximization problem requiring approximation [21]. In the machine learning area, there are already several approaches using the submodular function [1].

In [28], 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 [39], 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.

4. Lovasz Extension Metric Learning

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

In [18], they extend Bregman divergences to a specific class Lovasz Bregman divergences, in 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 (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.

The authors studied the possibility of learning higher-order distances according to the submodular functions in [16], which is similar to our goal, but they are limited to binary vectors. They demonstrate the possibility of defining metrics through submodular functions and show the performance of submodular Hamming metrics for metric minimization tasks (experiment with clustering) and metric maximization tasks (experiment with diversified k-best).

In [1], the authors give another proof of the links between submodularity and convexity. Unlike our use of submodular constraints to find metrics, the author uses the support function of the submodular function as the regularizer for optimizing the loss and focuses on supervised learning tasks such as variable form or feature selection.

We first define a new metric with the Lovasz extension, and second how to learn the associated set-function for a metric learning task.

4.1. Norm and Metric

For the metric learning algorithms previously mentioned, most of the metric they learn is a distance transformation with Mahalanobis metric [35, 29, 34, 11, 2] , while several of them are not [25, 10]. Metric learning algorithms thus need to prove the defined metric satisfies the conditions of metric we mentioned in Section 2, such as the metric learning with neural networks [10]. More generally, when updating the metric matrix M, one has to ensure that it remains PSD, so that the properties of a metric hold.

Our proposed algorithm 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. The mathematical definition of 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.

Consequently, defining a metric with the Lovasz extension reduces to prove that the Lovasz extension defines a norm, given some conditions on $f(S)$. In particular, we mentioned that for a submodular function $f(S)$, its Lovasz extension and the convex closures 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 $[0, 1]^m$.

4.2. Lovasz Extension Metric

In this paper, we propose to define a norm and therefore a distance metric which could break the limits of the Mahalanobis metric.

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.*

Proof. 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 [1, 27]). 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. □

Lemma 4.1. *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:

$$d_f^2(\mathbf{x}_i, \mathbf{x}_j) = L_f\left((\mathbf{x}_i - \mathbf{x}_j)^2\right) \tag{12}$$

Notice that Equation 12 can be easily generalized on p-power Lovasz extension norms, exactly the same way as L_p norms of Euclidean spaces.

4.3. Learning Lovasz Extension Metric

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 $f(\mathcal{S})$,

$$min_f \sum_{(i,j,k)\in\mathbb{C}} \ell(i,j,k) + \lambda r(f) \tag{13}$$

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

$$\begin{aligned} &\min r(f(\mathcal{S})) \\ &\text{s.t. } \ell(i,j,k) \leq 0, \forall (i,j,k) \in \mathbb{C} \\ &\quad f(\mathcal{S}) \text{ is submodular} \end{aligned} \tag{14}$$

Algorithm 1 describes our proposition LEML (Learning Lovasz Extension Metric), and the details of generating the relative constraints and submodular constraints are in the next section.

Algorithm 1 Lovasz Extension Metric Learning

Input: X, Y, m,κ,$\gamma = 1$ **r**
Output: **f**
$C = 0$
$\mathbf{b} = 0$
Generate relative constraints from label
for n from 1 to κ **do**
  Randomly sample triple (i,j,k) from the label Y with $Y_i == Y_j$ and $Y_i ! = Y_k$
  Sort the order of differnce $(x_i - x_j)^2_{(l)}$ of (i,j) for the Lovasz extension
  Sort the order of differnce $(x_i - x_k)^2_{(l)}$ of (i,k) for the Lovasz extension
  $C_n = Sorted((x_i - x_j)^2_{(l)}) - Sorted((x_i - x_k)^2_{(l)})$
  $\mathbf{b}_n = \gamma$
end for
Generate submodular constraints
for n from κ+1 to κ+n_c **do**
  Generate a vector V_n with [1,0,-1] for submodular constraints
  $V_{S_1} = -1$, $V_{S_2} = -1$,$V_{S_1 \cup S_2} = 1$,$V_{S_1 \cap S_2} = 1$,
  $C_n = V_n$
  $\mathbf{b}_n = 0$
end for
min $\mathbf{f}^T\mathbf{r}$ subject $C\mathbf{f} \leq \mathbf{b}$
solve with linear inequality programming
return **f**

Although we are aware that one can consider sparse linear programming solutions [38] 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.

4.4. Set-function Vector and Constraints Matrix

In order to adapt to compute the Lovasz extension metric $L_f(\mathbf{x})$ 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 for a simple illustration with $|\mathcal{V}| = 3$. Let C_s be such a matrix. Consequently, the submodularity constraint of Equation 14 can be written as $C_s\mathbf{f} \leq 0$.

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

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:

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

where $(x_i - x_j)^2_{(k)}$ is the permutation defined within Equation 12,$\mathcal{V}$ is the full set of all m-dimensions of the features.

Direct manipulation of the Lovasz extension of 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:

$$
\mathbf{a}_{ij} = \begin{pmatrix}
0 - (x_i - x_j)^2_{(m)} \\
0 \\
\cdots \\
(x_i - x_j)^2_{(m)} - (x_i - x_j)^2_{m-1} \\
0 \\
\cdots \\
0 \\
\cdots \\
(x_i - x_j)^2_{(2)} - (x_i - x_j)^2_{(1)} \\
0 \\
\cdots \\
(x_i - x_j)^2_{(1)} - 0
\end{pmatrix}, \qquad (15)
$$

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

$$
C_r^T = \left(\mathbf{a}_{ij}^1 - \mathbf{a}_{ik}^1, \cdots, \mathbf{a}_{ij}^\kappa - \mathbf{a}_{ik}^\kappa\right), \qquad (16)
$$

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 $L_f(\mathbf{x})$ is linear in $f(\mathcal{S})$, with such a form of set-function, the Equation (14) can be written as a linear inequality program:

$$
\begin{aligned}
&\min \mathbf{f}^T\mathbf{r} \qquad (17) \\
&\text{s.t. } C^T\mathbf{f} + \mathbf{t} \leq 0 \\
&\qquad 0 \leq \mathbf{f} \leq 1
\end{aligned}
$$

where

$$
C = \begin{pmatrix} C_r^T \\ C_s^T \end{pmatrix}, \mathbf{t} = \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix}, \qquad (18)
$$

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 β_i for each of these constraints. We subsequently add the penalty term $\alpha \sum_{i=1}^{m} \beta_i$ to $\mathbf{f}^T\mathbf{r}$, where α is a trade-off parameter (set to 1 in our experiments). Solving the revised program gives the solution denoted as LEML afterwards.

4.5. Constraints based on ξ-additive Fuzzy Measure

So far, we get the algorithm to learn a Lovasz extension metric, however, with a higher price of computational complexity. The number of values to be learned, 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 sizeable dimensional data sets. A 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 ξ-additive fuzzy measure, see [17], 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 ξ-additive set-function as an additive set-function whose corresponding pseudo-Boolean function has a polynomial development of degree at most ξ. Interestingly, if a set-function is ξ-additive, it means that there are no interactions between subsets of more than ξ elements.

Therefore, ξ-additive set-functions restrict their values to sets S for which we have $|S| \leq \xi$. This drastically reduces the number of variables required to fully define the set-function $f(S)$, going from 2^m to 2^ξ, with $\xi << m$. Note that this definition differs from the proposition of [5], 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 $f(S)$ (also known as Mobius Transform, see [17]), defined as

$$f^{-1}(S) = \sum_{T \subset S} (-1)^{|S \setminus T|} f(T), \text{ for all } S \subseteq V \quad (19)$$

vanishes for subsets whose cardinal is greater than ξ.

The Lovasz extension can be written using the inverse function as

$$L_f(\mathbf{x}) = \sum_{T \subseteq V} f^{-1}(T) \min_{i \in T} x_i, \quad (20)$$

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

$$f(S) = \sum_{T \subset S} f^{-1}(T) \quad (21)$$

In order to obtain a metric, the set-function $f(S)$ must remain submodular (see Proposition 1). One can write the submodular constraint imposed on $f(S)$ with its inverse $f^{-1}(S)$. 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 2. *Let $f : 2^{|V|} \rightarrow [0,1]$ be a set-function and $f^{-1}(S) : 2^{|V|} \rightarrow [0,1]$ its inverse function defined by Equation 19, then a) and b) are equivalent.*

a) $f(S)$ is a submodular set-function,

b) $\sum_{T \subset S_1 \cup S_2, T \not\subset S_1, T \not\subset S_2} f^{-1}(T) \leq 0$.

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

$$f(S_1 \cup S_2) + f(S_1 \cap S_2) \leq f(S_1) + f(S_2) \quad (22)$$

for S_ξ in V. Using Equation 19, we write
$f(S_1 \cap S_2) = \sum_{T \subset S_1 \cap S_2} f^{-1}(T)$,
and developing $f(S_1 \cup S_2)$ gives
$\sum_{I(T) \neq \emptyset} f^{-1}(T) + \sum_{T \subset S_1 \cup S_2, T \not\subset S_1, T \not\subset S_2} f^{-1}(T)$.

Consequently, Equation 22 is satisfied if and only if

$$\begin{aligned} &\sum_{T \subset S_1 \cap S_2} f^{-1}(T) + \sum_{I(T) \neq \emptyset} f^{-1}(T) \\ &+ \sum_{T \subset S_1 \cup S_2, T \not\subset S_1, T \not\subset S_2} f^{-1}(T) \\ &\leq \sum_{T \subset S_1} f^{-1}(T) + \sum_{T \subset S_2} f^{-1}(T) \end{aligned} \quad (23)$$

holds.

Finally, it is straightforward to obtain
$\sum_{T \subset S_1 \cup S_2, T \not\subset S_1, T \not\subset S_2} f^{-1}(T) \leq 0$
from Equation 23, 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 ξ-additive set-functions, i.e. we restrict the computation of the inverse set-function f^{-1}, given by Equation 19, on sets S for which $|S| \leq k$ holds. In that case, this is even reduced to $\frac{1}{8}2^\xi(\xi^2-\xi)$, where $\xi << m$, typically lower than

10. Combining this reduction and a ξ-additive set function leads to a tractable problem.

The solution obtained with this ξ-additive approach, and implementing submodular constraints following Proposition 2., is denoted as LEML-ξ hereafter.

5. 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. We use 8 data sets from the UC Irvine Machine Learning Repository [24], and their main characteristics are given in Table 4. All the experiments are done on a MacBook Pro with 2.3 GHz Intel Core i5 CPU, 16 GB 2133 MHz LPDDR3 RAM and Python 3.7. The only exception is GMML [40], that were run on Matlab.

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

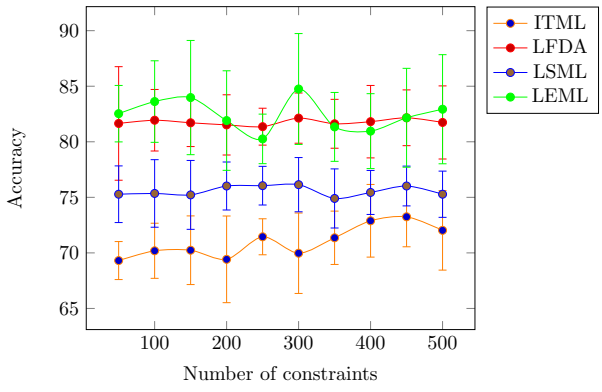
Table 4: UCI datasets used in the experiments. c indicates the number of classes.

In the first part of the experiments, we are using LEML(Lovasz Extension Metric Learning), that is using all possible orders of feature interactions. In particular, the only constraints are related to submodularity and relative constraints. We compare the results obtained with the proposed method LEML against several state-of-the-art linear and non-linear metric learning algorithms: LMNN [34], ITML [11], LSML [25], LFDA [31], GMML[40] and GB-LMNN [19, 20]. Finally, we also give the results obtained without metric learning, which is the Euc. that means Euclidean distance for $M = Id$.

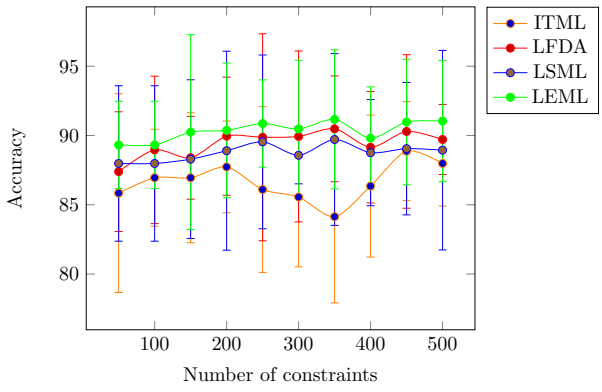
We compute the accuracy of each model on the task of K-nearest neighbours classification, using cross-validation. All the accuracy and the cost time are the averages of 10 runs. The number of neighbours K is selected between $[3, 5]$, and the folds of cross-validation are set between $[3, 15]$ according to the experience which are similar to LMNN [34], ITML [11] and GB-LMNN [19, 20]. We tested different values, without significant modification on the comments that can be drawn from the results. The given results are obtained with $K = 5$ and 10-folds cross-validation. In order to build the set $\mathcal{R}$ of relative constraints, we use the labels and randomly select the triples of objects $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ 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 κ, in LMNN [34] they use all supervised information and limit with maximum number of iteration; in ITML [11] it could be a user-select number of constraints and limit with maximum number of iteration; in LSML [25] they choose 100 for all dataset and compared algorithms; in GMML [40] and GB-LMNN [19, 20] 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), we set the same maximum number of constraints, which $\kappa = 100$, and as shown in Figure 4, after testing several different sets, the ranks are similar.

As mentioned earlier, due to the complexity of the model, our first proposition, LEML, is not able to process datasets for which the dimension is (even moderately) large. Consequently, we first use a PCA 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.

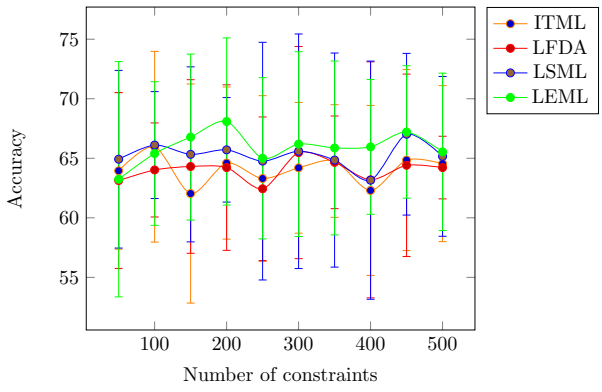
Accuracy and running time obtained on the 8 datasets for each method are given in Table 5. As can be seen, the proposed LEML generally performs better than all the other metric learning algorithms (with the notable exception of Ionosphere and Segment datasets). More precisely, given the average rank of each method, we obtain the following ranking $LEML \succ GB-LMNN \succ GMML \succ LFDA \sim LMNN \succ LSML \succ ITML \succ Euc.$. Notice that for low dimensional datasets, the running time of the proposed method is low, and quickly increases with the dimension of the data. Statistical significance of the results are assessed using a Friedman test [14] as suggested by [12].



(a) Balance



(b) Seeds



(c) Glass

Figure 4: Different numbers of constraints

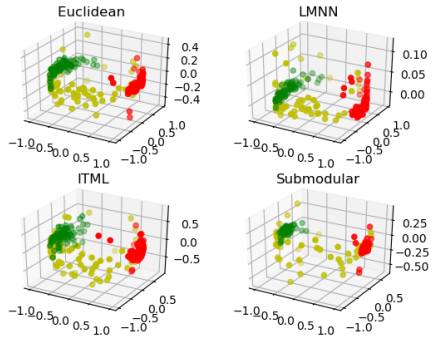


Figure 5: 3-dimensional embedding of Seeds dataset using different metrics.

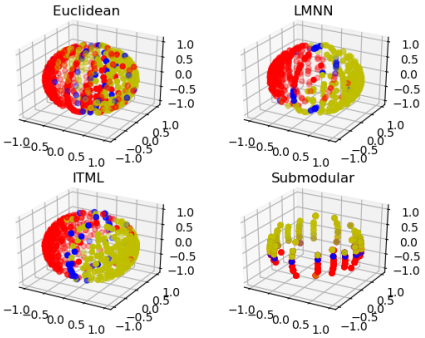


Figure 6: 3-dimensional embedding of Balance dataset using different metrics.

The value of the Friedman test, $F_F = 7.10 > F_{0.05}(8, 56)$ shows the significance of the difference between the ranks.

Figures 5 and 6 give 3D embeddings obtained with various metric learning algorithms on two 3-classes datasets, Seeds and Balance. It appears that our proposition allows better visual discrimination of the classes than existing approaches, which could be experimentally evaluated with real users.

The second part of the experiments uses the modified LEML-ξ. In particular, it uses the constraint obtained by using the result of Proposition 2 (in which submodularity constraints can be written using the inversion f^{-1}). For LEML-ξ, we use the ξ-additive constraints on $f(S)$ to decrease the complexity. In this experiment, we use the same

Dataset	Euc.	LMNN	ITML	LSML	LFDA	GMML	GB-LMNN	LEML	LEML-ξ
balance	72.66	78.86 (16.95s)	77.17 (7.35s)	73.82 (0.01s)	80.22 (0.01s)	80.34(0.42s)	68.90(0.23s)	81.02 (0.01s)	**81.12** (0.01s)
digits	93.77	94.33 (254.0s)	90.94 (0.71s)	92.83 (0.01s)	94.10 (0.07s)	94.73(1.76s)	**94.87**(4.60s)	93.89 (126.0s)	94.05 (2.60s)
glass	61.02	65.21 (4.19s)	57.24 (7.53s)	64.27 (3.95s)	58.17 (0.01s)	64.86(0.32s)	66.79(0.39s)	**68.70** (0.94s)	68.17 (0.09s)
iono	85.75	87.17 (6.58s)	85.18 (6.12s)	86.04 (0.09s)	77.18 (0.09s)	87.56(0.16s)	**94.32**(2.32s)	86.61 (150.9s)	88.60 (2.54s)
liver	66.48	63.87 (36.11s)	62.26 (7.55s)	65.70 (5.43s)	66.14 (0.02s)	64.72(0.85s)	66.48(3.52s)	66.48 (0.01s)	**66.59** (0.01s)
seeds	82.57	88.52 (2.45s)	87.62 (14.30s)	88.10 (2.71s)	89.48 (0.01s)	88.67(0.61s)	88.10(2.42s)	**90.95** (0.02s)	90.49 (0.01s)
sonar	50.87	55.69 (2.80s)	48.92 (3.11s)	51.53 (0.02s)	52.86 (0.01s)	55.83(1.04s)	**66.76**(2.07s)	56.63 (124.9s)	59.02 (2.17s)
segment	78.10	82.52 (76.75s)	80.29 (1.26s)	**85.67** (0.05s)	83.61 (0.01s)	83.34(1.03s)	83.42(2.12s)	84.38 (168.8s)	83.81 (2.76s)

Table 5: Accuracy of KNN with different metrics learning algorithm and their running time in seconds.

datasets as in the previous experiment, but we make ξ of the ξ-additive varies from 1 to $\min(10, m)$. A value of $\xi = 1$ means that there is no interaction between features, and only singletons are considered. Increasing ξ adds orders of interaction, and finally reaches the order of interaction tackled by LMEL approach without ξ-additive method. It can be seen that each time we decrease ξ, the number of free parameters of $f(S)$ is divided by 2, so that running time of the method is now very reasonable, even for quite large dimensional data. Table 5 also gives the results obtained through a grid search of ξ (last column). Interestingly, one can see that LEML-ξ often gives better results than LEML, showing that using all the m-tuple-wise combinations are not always necessary, and may even penalize the performances (e.g. balance, ionosphere, liver, and sonar).

The results for varying ξ are given in Figure 7. According to these results, one can draw the following comments. As can be expected, increasing ξ 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 ξ concerning a given loss function, or optimal ξ that trades-off accuracy for computation. Nonetheless, experimental results show that the LEML-ξ allows successfully consider large scale problem (outperforms other metric learning algorithms, both in accuracy and running time), by choosing a sufficiently low value of ξ.

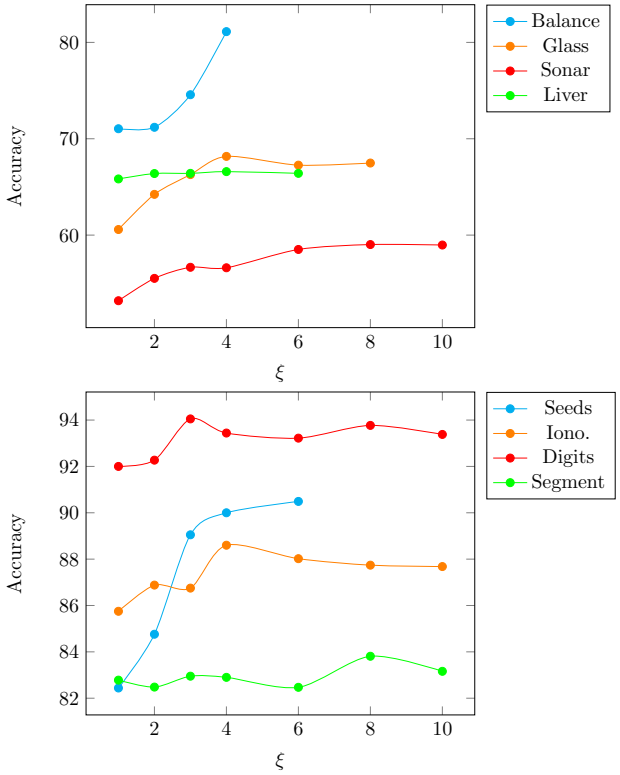


Figure 7: Evolution of the LEML-ξ classification performance using ξ-additive constraints, where ξ is varying from 1 (single feature weighting) to $\min(10, m)$.

6. Conclusions and Future Works

In this paper, we present a new metric distance based on the Lovasz extension of a submodular set-function and give the necessary conditions for defin-

ing a proper metric. Then, we present a linear program allowing them to learn this metric and some variations around the constraints imposed on the set-function. Experiments show the efficiency of the proposition on rather low dimensional datasets, by outperforming state-of-the-art metric learning approaches in terms of accuracy. Potential future work will consist of improving the algorithm, in special, by proposing online updates through stochastic gradient descent in order to scale well with the dimension of the data. We also plan to add a regularization term on the set-function $\mathbf{f}$ into the objective function. We will also study generalization bounds of the proposition, following recent work in [6]. One more interesting topic of research is related to representation learning, also linked with metric learning, using submodular functions as presented in [13]. Further investigations can be conducted in this domain through the use of deep models.

References

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

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

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

[4] Peter J Bickel, Bo Li, Alexandre B Tsybakov, Sara A van de Geer, Bin Yu, Teófilo Valdés, Carlos Rivero, Jianqing Fan, and Aad van der Vaart. Regularization in statistics. *Test*, 15(2):271–344, 2006.

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

[6] Qiong Cao, Zheng-Chu Guo, and Yiming Ying. Generalization bounds for metric and similarity learning. *Machine Learning*, 102(1):115–132, 2016.

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

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

[9] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, volume 1, pages 539–546. IEEE, 2005.

[10] 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.

[11] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *ICML*, pages 209–216. ACM, 2007.

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

[13] Brian W Dolhansky and Jeff A Bilmes. Deep submodular functions: Definitions and learning. In *NIPS*, pages 3396–3404, 2016.

[14] 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.

[15] 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.

[16] Jennifer A Gillenwater, Rishabh K Iyer, Bethany Lusch, Rahul Kidambi, and Jeff A Bilmes. Submodular hamming metrics. In *NIPS*, pages 3141–3149, 2015.

[17] Michel Grabisch. *Set Functions, Games and Capacities in Decision Making*. Springer, 2016.

[18] Rishabh Iyer and Jeff Bilmes. The lovász-bregman divergence and connections to rank aggregation, clustering, and web ranking. *arXiv preprint arXiv:1308.5275*, 2013.

[19] 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.

[20] Dor Kedem, Zhixiang Eddie Xu, and Kilian Q Weinberger. Gradient boosted large margin nearest neighbors.

[21] Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability: Practical Approaches to Hard Problems*, 3(19):8, 2012.

[22] Hoel Le Capitaine. A relevance-based learning model of fuzzy similarity measures. *IEEE Transactions on Fuzzy Systems*, 20(1):57–68, 2011.

[23] Hoel Le Capitaine. Constraint selection in metric learning. *Knowledge-Based Systems*, 146:91–103, 2018.

[24] M. Lichman. UCI machine learning repository, 2013.

[25] 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.

[26] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[27] László Lovász. Submodular functions and convexity. In *Mathematical Programming The State of the Art*, pages 235–257. Springer, 1983.

[28] James Petterson and Tibério S Caetano. Submodular multi-label learning. In *Advances in Neural Information Processing Systems*, pages 1512–1520, 2011.

[29] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. *Advances in neural information processing systems (NIPS)*, page 41, 2004.

[30] Tasuku Soma and Yuichi Yoshida. A generalization of submodular cover via the diminishing return property on the integer lattice. In *Advances in Neural Information Processing Systems*, pages 847–855, 2015.

[31] 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.

[32] Donald M Topkis. *Supermodularity and complementarity*. Princeton university press, 1998.

[33] 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.

[34] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10(Feb):207–244, 2009.

[35] 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.

[36] 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.

[37] Han-Jia Ye, De-Chuan Zhan, Xue-Min Si, Yuan Jiang, and Zhi-Hua Zhou. What makes objects similar: A unified multi-metric learning approach. In *NIPS*, pages 1235–1243, 2016.

[38] 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.

[39] Jiaqian Yu and Matthew Blaschko. Learning submodular losses with the lovasz hinge. In *ICML*, pages 1623–1631, 2015.

[40] Pourya Zadeh, Reshad Hosseini, and Suvrit Sra. Geometric mean metric learning. In *International Conference on Machine Learning*, pages 2464–2471, 2016.