

Metric Learning from Imbalanced Data

Léo Gautheron, Emilie Morvant, Amaury Habrard, Marc Sebban

▶ **To cite this version:**

Léo Gautheron, Emilie Morvant, Amaury Habrard, Marc Sebban. Metric Learning from Imbalanced Data. International Conference on Tools with Artificial Intelligence (ICTAI), Nov 2019, Portland, Oregon, United States. hal-02276288

HAL Id: hal-02276288
https://hal.science/hal-02276288

Submitted on 2 Sep 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Metric Learning from Imbalanced Data

Léo Gautheron, Emilie Morvant, Amaury Habrard and Marc Sebban

Laboratoire Hubert Curien UMR 5516, Univ Lyon, UJM F-42023, Saint-Etienne, France.

September 2, 2019

Abstract

A key element of any machine learning algorithm is the use of a function that measures the dis/similarity between data points. Given a task, such a function can be optimized with a *metric learning* algorithm. Although this research field has received a lot of attention during the past decade, very few approaches have focused on learning a metric in an imbalanced scenario where the number of positive examples is much smaller than the negatives. Here, we address this challenging task by designing a new Mahalanobis metric learning algorithm (**IML**) which deals with class imbalance. The empirical study performed shows the efficiency of **IML**.

1 Introduction

Metric learning [3, 4, 22] is a subfield of representation learning that consists of designing a pairwise function able to capture the dis/similarity between two data points. This is a key issue in machine learning where such metrics are at the core of many algorithms, like k-nearest neighbors (kNN), SVMs, k-Means, etc. To construct a dis/similarity measure suitable for a given task, most metric learning algorithms optimize a loss function which aims at bringing closer examples of the same label while pushing apart examples of different labels. In practice, metric learning is usually performed with cannot link/must link constraints—two data points x and x' should be dis/similar [10, 27, 37–40]—or relative constraints—a data point x should be more similar to another x' than to a third one x'' [23, 31, 37, 42].

In this paper, we focus on the family of metric learning algorithms that construct a Mahalanobis distance defined as

$$d_{\mathbf{M}}(x, x') = \sqrt{(x - x')^\top \mathbf{M}(x - x')}\,,$$

parameterized by a positive semidefinite matrix $\mathbf{M}$. Learning such a Mahalanobis distance leads to several nice properties: *(i)* $d_{\mathbf{M}}$ is a generalization of the Euclidean distance; *(ii)* it induces a projection such that the distance between two points is equivalent to their Euclidean distance after a linear projection; *(iii)* $\mathbf{M}$ can be low rank implying a projection in a lower dimensional latent space; *(iv)* it involves optimization problems that are often convex and thus easy to solve. The most famous Mahalanobis distance learning algorithms are likely **LMNN** (Large Margin Nearest Neighbor [37]) and **ITML** (Information-Theoretic Metric Learning [10]), which are both designed to improve the accuracy of the kNN classification rule in the latent space. The principle of **LMNN** is the following: for each training example, its k nearest neighbors of the same class (the *target neighbors*) should be closer than examples of other classes (the *impostors*). The algorithm **ITML** uses a LogDet regularization and minimizes (respectively maximizes) the distance between examples of the same (respectively different) class. We can also cite another recent Mahalanobis distance learning algorithm called **GMML** (Geometric Mean Metric Learning [40]) where the metric is computed using a closed form solution of an unconstrained optimization problem involving similar and dissimilar pairs. In light of these learning procedures, it is worth noticing that the loss functions optimized in **LMNN**, **ITML** and **GMML** (and in most pairwise metric learning methods) tend to favor the majority class as there is no distinction between the constraints involving examples of the majority class and the constraints on the minority class. This strategy is thus not well suited when dealing with imbalanced classes. An illustration of this phenomenon on the SPECTFHEART dataset from the UCI repository is shown in Figure 1. We observe that decreasing the proportion of minority examples tends to generate a metric which classifies (with a kNN rule) all the examples as the majority class, thus leading to an accuracy close to 1. On the other hand, the F1-measure [32], commonly[1] used in imbalanced settings [7, 26], decreases with the proportion of positives, showing that the classifier missed many positives, usually considered as the examples of interest.

This problem of learning from imbalanced data has been widely tackled

[1]The F1-measure is much more adapted to imbalanced scenarios since it does not involve the true negatives but considers both the false positives and the false negatives.

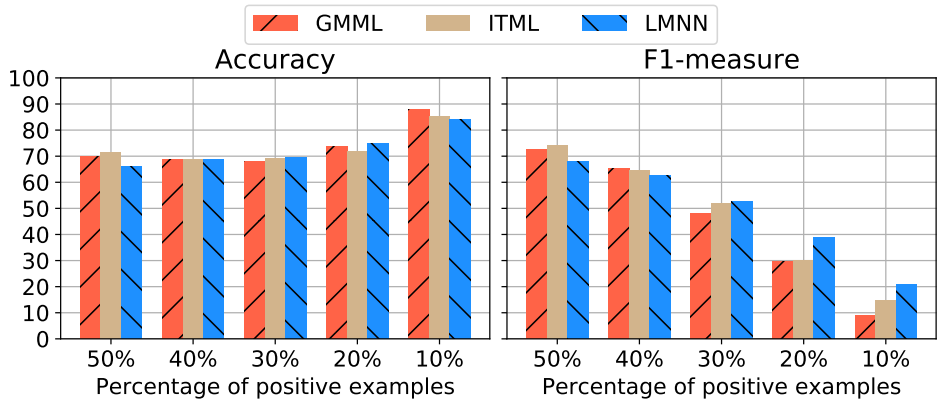


Figure 1: Illustration on the SPECTFHEART dataset of the negative impact of classic metric learning algorithms when facing an increasing imbalance in the dataset. On the left, as the proportion of minority examples decreases (the positive class), the kNN algorithm with the learned metrics tend to classify all the examples as the majority class, with an accuracy close to 100%. On the right, using the F1-measure, we note that the classifier actually miss many positives, typically considered as the examples of interest.

in the literature [5, 20]. Classic methods typically make use of over/under-sampling techniques [1, 2, 12, 14, 16, 25, 29] or create synthetic samples in the neighborhood of the minority class, *e.g.,* using SMOTE-like strategies [8, 9, 19] or resorting to adversarial techniques [11]. However, these methods may lead to over or under-fitting and are often subject to an inability to generate enough diversity, especially in a highly imbalanced scenario. Other strategies aim at addressing imbalanced situations directly during the learning process. They include cost-sensitive methods [13, 41] which require prior knowledge on the miss-classification costs, the optimization of imbalance-aware criteria [17, 28, 34] which are often non convex, or ensemble methods based on bagging and boosting strategies [18] that can be computationally expensive.

Unlike the state of the art, we propose in this paper to address the problem of learning from imbalanced data by optimizing a metric suited to scenarios where the positive data are very scarce. As far as we know, very few methods were designed in this setting. We can cite, Feng *et al.* [15] propose to regularize a standard metric learning problem by using the KL-divergence between the classes. Wang *et al.* [35] use **LMNN** to learn a classic metric

and then perform a sampling on the training data to account the imbalance. However, as we will see in our experimental study, better performances can be achieved by resorting to a metric dedicated specifically to deal with the imbalance of the application at hand. In order to implicitly control the rates of false positives and false negatives, we design a new algorithm, called **IML** for Imbalanced Metric Learning, which accounts carefully the nature of the pairwise constraints (by decomposing them with respect to the labels involved in the pairs) and gives them an equal weight to account the imbalance.

The rest of the paper is organized as follows. Section 2 introduces the notations and the principle of classical Mahalanobis metric learning. Section 3 describes our algorithm **IML** which takes the form of a simple regularized convex problem. We perform an experimental study of our approach in Section 4 before concluding in Section 5.

2 Notations and Setting

In this paper, we deal with binary classification tasks where $\mathcal{X} \subseteq \mathbb{R}^d$ is a d-dimensional input space and $\mathcal{Y} = \{-1, +1\}$ is the binary output space. We further define $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ as the joint space where $z = (x, y) \in \mathcal{Z}$ is a labeled example. In supervised classification, a machine learning algorithm is provided with a learning sample $\mathcal{S} = \{z_i{=}(x_i, y_i)\}_{i=1}^n$ of n labeled examples *i.i.d.* from a fixed yet unknown distribution $\mathcal{D}$ over $\mathcal{Z}$. We assume that the learning sample is defined as $\mathcal{S} = \mathcal{S}^+ \cup \mathcal{S}^-$, with $\mathcal{S}^+$ the set of positive examples and $\mathcal{S}^-$ the set of negative examples such that the number of positives $n^+ = |\mathcal{S}^+|$ is smaller than the number of negatives $n^- = |\mathcal{S}^-|$ (we say that $+1$ is the minority class and -1 the majority one). We consider a hypothesis space $\mathcal{F}$, such that $\forall f \in \mathcal{F},\ f : \mathcal{X} \to \mathcal{Y}$. Given $\mathcal{S}$ and $\mathcal{F}$, the final objective of the learner is to find in $\mathcal{F}$ a hypothesis f (called a classifier) which behaves well on from $\mathcal{D}$, meaning that f has to classify correctly unseen data points.

In this work, we aim at constructing a Mahalanobis distance which induces a new space in which a kNN classifier will work well on both classes. The Mahalanobis distance is a type of metric parameterized by a positive semidefinite (PSD) matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$ that can be decomposed as $\mathbf{M} = \mathbf{L}^\top \mathbf{L}$, where $\mathbf{L} \subseteq \mathbb{R}^{r \times d}$ is a projection induced by $\mathbf{M}$ (where r is the rank of $\mathbf{M}$). A nice property is that the Mahalanobis distance between two points x and

x' is equivalent to the Euclidean distance after having projected x and x' in the r-dimensional space, *i.e.*,

$$\begin{aligned} d^2_{\mathbf{M}}(x, x') &= (x - x')^\top \mathbf{M} (x - x') \\ &= (\mathbf{L}x - \mathbf{L}x')^\top (\mathbf{L}x - \mathbf{L}x') . \end{aligned}$$

Mahalanobis metric learning algorithms [4,6,21] can usually be expressed as follows:

$$\min_{\mathbf{M} \succeq 0} F(\mathbf{M}) = \frac{1}{n^2} \sum_{(z,z') \in \mathcal{S}^2} \ell(\mathbf{M}, z, z') + \lambda \, Reg(\mathbf{M}), \tag{1}$$

where one wants to minimize the trade-off between a convex loss ℓ over all pairs of examples and a regularization Reg under the PSD constraint $\mathbf{M} \succeq 0$.

The major drawback of this classical formulation is that the loss gives the same importance to any pair of examples (z, z') whatever the labels y and y'. Intuitively, this is not well suited to imbalanced data where the minority class is the set of examples of interest (think, for example, about anomaly detection [7]). Note that some metric learning algorithms [37, 40] allow us to weight the role played by the must-link and cannot-link constraints, but they do not directly take into account the labels of the examples.

To tackle these drawbacks, we propose in the next section **IML**, a metric learning algorithm able to deal with imbalanced data.

3 IML: Imbalanced Metric Learning

Our algorithm is built on the simple idea consisting in decomposing further the sets of must-link and cannot-link constraints based on the two labels involved in the constraints. Each set can then be weighted differently during the optimization to reduce the negative effect of the imbalance.

Starting from (1), we need to define a loss function ℓ on which we base our **IML** algorithm; we have for all $(z, z') \in \mathcal{X}^2$ and for all $\mathbf{M} \in \mathbb{R}^{d \times d}$:

$$\ell(\mathbf{M}, z, z') = \begin{cases} \ell_1(\mathbf{M}, z, z') = \left[d^2_{\mathbf{M}}(x, x') - 1 \right]_+ & \text{if } y = y', \\ \ell_2(\mathbf{M}, z, z') = \left[1 + m - d^2_{\mathbf{M}}(x, x') \right]_+ & \text{if } y \neq y', \end{cases} \tag{2}$$

where $[a]_+ = \max(0, a)$ is the Hinge loss and $m \geq 0$ a margin parameter.

We illustrate in Figure 2 the behavior of the two losses ℓ_1 and ℓ_2. The idea of ℓ_1 is to bring examples of the same class at a distance less than 1

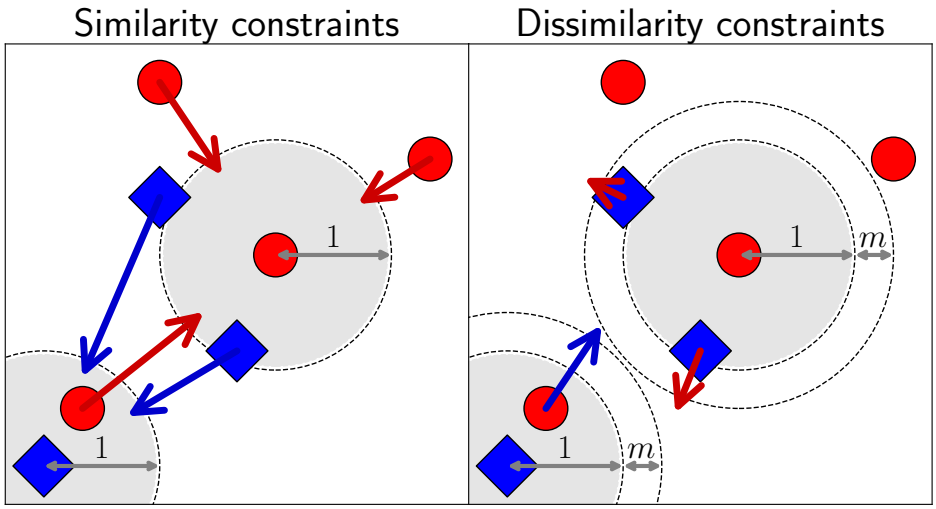


Figure 2: Illustration of the behavior of our loss ℓ defined in (2). On the left, the similarity constraints (loss ℓ_1) aim at bringing examples of the same class at a distance less than 1. On the right the dissimilarity constraints (loss ℓ_2) aim at pushing away examples of different classes at a distance larger than 1 plus a margin m.

while ℓ_2 aims to push far away examples of different classes at a distance larger than 1 plus a margin m.

In addition to inserting (2) into (1), we need to set the regularization term $Reg(\mathbf{M})$. In order to avoid overfitting, we propose to enforce $\mathbf{M}$ to be close to the identity matrix $\mathbf{I}$ such as $Reg(\mathbf{M}) = \|\mathbf{M} - \mathbf{I}\|_F^2$, with $\|\cdot\|_F$ the Frobenius norm. In other words, we aim at learning a Mahalanobis metric which is close to the Euclidean distance while satisfying the best the semantic constraints.

All things considered, our **IML** algorithm takes the form of the following

convex problem:

$$\begin{aligned}\min_{\mathbf{M}\succeq 0} F(\mathbf{M}) = & \frac{a}{4|Sim^+|}\sum_{(z,z')\in Sim^+}\ell_1(\mathbf{M},z,z') + \\ & \frac{a}{4|Sim^-|}\sum_{(z,z')\in Sim^-}\ell_1(\mathbf{M},z,z') + \\ & \frac{(1-a)}{4|Dis^+|}\sum_{(z,z')\in Dis^+}\ell_2(\mathbf{M},z,z') + \\ & \frac{(1-a)}{4|Dis^-|}\sum_{(z,z')\in Dis^-}\ell_2(\mathbf{M},z,z') + \\ & \lambda\left\|\mathbf{M}-\mathbf{I}\right\|_F^2 , \end{aligned} \tag{3}$$

where the four sets Sim^+, Sim^-, Dis^+ and Dis^- are defined as subsets of $\mathcal{S}\times\mathcal{S}$ respectively as:

$$\begin{aligned} Sim^+ &\subseteq \mathcal{S}^+\times\mathcal{S}^+ , \\ Sim^- &\subseteq \mathcal{S}^-\times\mathcal{S}^- , \\ Dis^+ &\subseteq \mathcal{S}^+\times\mathcal{S}^- , \\ \text{and}\quad Dis^- &\subseteq \mathcal{S}^-\times\mathcal{S}^+ . \end{aligned}$$

The parameter a takes values in $[0,1]$; it controls the trade-off between bringing closer the similar examples and keeping far away the examples of different classes.

The fundamental difference between our formulation and classic metric learning formulations is that we separate in our loss the set of similar pairs into two sets Sim^+ and Sim^-, and the set of dissimilar pairs into two sets Dis^+ and Dis^-. In a classic metric learning formulation, these four sets are all treated equally by giving them a weight of $\frac{1}{n^2}$. However in the presence of imbalanced data, the number of pairs in Sim^+ and Dis^+ which is in $O(n^+)$ is much smaller than in the sets Sim^- and Dis^- where the number of pairs is in $O(n^-)$. Intuitively, in the presence of imbalanced data, the terms in $O(n^+)$ will have a smaller impact on the loss function, thus, we aim at re-weighting these four sets to account the imbalance. We adopt a simple strategy consisting in giving a weight to each set that depends on its number of elements. We choose to give to the four sets a weight $\frac{1}{4|\text{set}|}$. This strategy allows us to give the same importance to the four terms in the loss function, no matter how imbalanced the data is. We will see experimentally that using this re-weighting instead of the weight $\frac{1}{n^2}$ greatly increases performances when facing increasingly imbalanced data.

If we look more closely at (3), when all pairs from $\mathcal{S} \times \mathcal{S}$ are involved, Sim^+ and Sim^- contain respectively n^+n^+ and n^-n^- pairs while Dis^+ and Dis^- contain respectively n^+n^- and n^-n^+ pairs. This means that the pairs in Dis^+ and Dis^- are symmetric and these two sets might be merged. However, metric learning rarely considers all the possible pairs as it becomes quite inefficient in the presence of a large number of examples. Possible strategies to select the pairs include a random selection of the pairs [10, 38–40] or a selection based on the nearest neighbors rule [27, 37].

For this reason, it might make sense to separate the two sets Dis^+ and Dis^- and allows us to weight them differently as *(i)* they may not consider the same subsets of pairs, and *(ii)* may not capture the same geometric information. Another interpretation of such a decomposition in an imbalanced learning setting is the following: if z' is selected as belonging to the neighborhood of z, the minimization of the four terms of (3) can be seen as a nice way to implicitly optimize with a kNN rule the true positive, false negative, false positive and true negative rates respectively.

Among the two strategies to select the pairs, the selection based on the nearest neighbors is more adapted to an imbalanced scenario as it considers k pairs for each training example both from the majority and minority classes. However, the random strategy just picks at random two examples to create a pair. Then with imbalanced data, it might be possible not to have any similar pair between two minority examples, thus focusing on the majority class. We will see experimentally that, as expected, the selection of the pairs based on the nearest neighbors rule performs better.

4 Experiments

4.1 Datasets

We provide here an empirical study of **IML** on 22 datasets coming mainly from the UCI[2] and Keel[3] repositories except for the 'SPLICE' dataset which comes from LIBSVM[4]. All datasets are normalized such that each feature has a mean of 0 and a variance of 1.

For the sake of simplicity, we have chosen binary datasets, described in Table 1 where the minority class is given by the columns "Label". Note that **IML** can easily be generalized to multiclass problems by learning one

[2] `https://archive.ics.uci.edu/ml/datasets.html`

[3] `http://sci2s.ugr.es/keel/datasets.php`

[4] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#splice`

Table 1: Description of the datasets (n: number of examples, d: number of features, c: number of classes) and the class chosen as positive (*Label*), its cardinality (n^+) and its percentage (%).

Name	n	d	c	*Label*	n^+	%	Name	n	d	c	*Label*	n^+	%
splice	3175	60	2	-1	1527	48.10%	glass	214	11	6	1	70	32.71%
sonar	208	60	2	R	97	46.64%	newthyroid	215	5	3	2, 3	65	30.23%
balance	625	4	3	L	288	46.08%	german	1000	23	2	2	300	30.00%
australian	690	14	2	1	307	44.49%	vehicle	846	18	4	van	199	23.52%
heart	270	13	2	2	120	44.44%	spectfheart	267	44	2	0	55	20.60%
bupa	345	6	2	1	145	42.03%	hayes	160	4	3	3	31	19.38%
spambase	4597	57	2	1	1812	39.42%	segmentation	2310	19	7	window	330	14.29%
wdbc	569	30	2	M	212	37.26%	abalone	4177	10	28	8	568	13.60%
iono	351	34	2	b	126	35.90%	yeast	1484	8	10	ME3	163	10.98%
pima	768	8	2	1	268	34.90%	libras	360	90	15	1	24	6.66%
wine	178	13	3	1	59	33.15%	pageblocks	5473	10	5	3, 4, 5	231	4.22%

metric per class in a standard "one-versus-all" strategy, and then applying a majority vote [30, 33].

4.2 Optimization Details

Like most Mahalanobis metric learning algorithms, **IML** requires that the learned matrix $\mathbf{M}$ is PSD. There exist different methods to enforce the PSD constraint [22]. A classic solution consists in performing a Projected Gradient Descent where one alternates a gradient descent step and a (costly) projection onto the cone of PSD matrices. The advantage is that the problem remains convex [39] *w.r.t.* $\mathbf{M}$, ensuring that one will attain the optimal solution of the problem by correctly setting the projection step in the gradient descent. Another solution [36] is based on the fact that if $\mathbf{M}$ is PSD, it can be rewritten as $\mathbf{M} = \mathbf{L}^\top\mathbf{L}$. Therefore, instead of learning $\mathbf{M}$, one can enforce $\mathbf{M}$ to be PSD in a cheaper way by directly learning the projection matrix $\mathbf{L} \in \mathbb{R}^{r\times d}$ (where r is the rank of $\mathbf{M}$). This can be done thanks to a gradient descent by computing the gradient of the problem *w.r.t.* $\mathbf{L}$ (instead of $\mathbf{M}$). The implementation[5] we propose is based on this latter approach [36] where we make use of the L-BFGS-B algorithm [43] from the SciPy Python library to optimize our problem: it takes as input our initial point (the identity matrix), the optimization problem of (3), and its gradient, then it performs a gradient descent that returns the projection matrix $\mathbf{L}$ minimizing (3). To prevent us from tuning r and finding the best r-dimensional projection space, we set $r = d$ in the experiments. Indeed, our main objective here is to learn a robust metric and not to get a sparse

[5]The code is available here: `https://leogautheron.github.io`

solution.

As discussed at the end of Section 3, the pairs of examples considered by **IML** in its four terms are chosen using the nearest neighbors rule. Indeed, we noted experimentally that the algorithms using this strategy (**LMNN** [37] and **IML**) perform better than the ones using a random selection strategy (**ITML** [10] and **GMML** [40]).

4.3 Experimental setup

All along our experiments, we use a 3NN classifier (like in **LMNN**) after projection of the training and test data using the metric learned. The metrics considered in the comparative study are the Euclidean distance and the ones learned by **GMML** [40], **ITML** [10], **LMNN** [37] and **IML**. For each dataset, we generate randomly 20 stratified splits of 30% training examples and 70% test data (same class proportions in training and test) and report the mean results over the 20 splits. The parameters are tuned by 5-fold cross-validation on the training set through a grid search using the following parameter ranges: for **LMNN**, $\mu \in \{0, 0.05, \ldots, 1\}$ (k is fixed to 3); for **ITML**, $\gamma \in \{2^{-10:10}\}$; for **GMML** $t \in \{0, 0.05, \ldots, 1\}$; and for **IML**, $m \in \{1, 10, 100, 1000, 10000\}$, $\lambda \in \{0, 0.01, 0.1, 1, 10\}$ and $a \in \{0, 0.05, \ldots, 1\}$ (k is also fixed to 3). For **IML**, we select randomly without replacement 100 combinations of hyper-parameters and use the one giving the best validation performance.

4.4 Experiments

First experiment—without data pre-processing We start by applying the experimental setup described above and we report the results in Table 2. On average, the F1-measure of 72.1% obtained by **IML** is the best in comparison to 70.8% for **LMNN**, 70.1% for **ITML**, 69.3% for **GMML** and 67.3% for the Euclidean distance. Overall, **IML** shows also the best average rank of 1.48. We note that **IML** generally gives better performances on the datasets considered no matter how much they are balanced or not. This means that our re-weighting scheme of the pairs can not only improve the performances in an imbalanced setting but can be also competitive in more classic scenarios.

Second experiment—with data pre-processing To address imbalanced data issues, classic machine learning algorithms typically resort to

Table 2: Average F1-measure ± standard deviation over 20 splits obtained using different metric learning algorithms.

Dataset	Euclidean	LMNN	ITML	GMML	IML
splice	76.3 ± 0.7	76.3 ± 1.3	79.7 ± 1.4	86.5 ± 0.8	**87.3** ± 0.6
sonar	69.2 ± 5.3	69.2 ± 5.0	70.6 ± 5.9	70.6 ± 6.5	**72.8** ± 5.0
balance	87.4 ± 1.8	90.4 ± 1.3	93.0 ± 1.4	89.8 ± 1.3	**93.1** ± 2.2
australian	79.9 ± 1.7	80.9 ± 2.5	**82.0** ± 1.9	81.7 ± 2.0	82.0 ± 2.0
heart	76.8 ± 2.1	76.9 ± 3.6	76.8 ± 2.9	74.8 ± 3.2	**77.0** ± 3.0
bupa	49.8 ± 4.4	**52.0** ± 5.3	51.3 ± 4.8	50.1 ± 5.0	51.2 ± 5.8
spambase	85.3 ± 0.9	86.8 ± 0.8	87.8 ± 1.0	88.4 ± 0.8	**89.3** ± 0.6
wdbc	94.2 ± 1.3	94.4 ± 1.3	94.3 ± 1.1	93.5 ± 1.7	**94.9** ± 1.3
iono	67.8 ± 6.7	72.0 ± 5.4	73.4 ± 5.4	70.8 ± 3.9	**74.1** ± 4.0
pima	56.2 ± 1.9	56.7 ± 3.0	57.5 ± 3.0	55.9 ± 3.3	**57.6** ± 3.0
wine	94.9 ± 2.2	95.5 ± 2.9	**96.3** ± 3.3	96.0 ± 2.9	95.5 ± 2.4
glass	66.0 ± 3.4	**67.2** ± 3.5	62.6 ± 5.2	63.6 ± 5.2	66.1 ± 3.7
newthyroid	83.4 ± 4.2	**90.6** ± 2.6	89.8 ± 5.2	88.1 ± 5.2	90.5 ± 4.7
german	35.3 ± 2.8	37.1 ± 3.9	37.4 ± 3.3	37.3 ± 3.9	**38.0** ± 3.5
vehicle	80.5 ± 2.4	90.1 ± 1.7	90.2 ± 2.4	**92.6** ± 1.0	91.4 ± 1.9
spectfheart	34.8 ± 12.3	29.3 ± 11.6	34.4 ± 7.9	39.1 ± 8.4	**45.3** ± 6.4
hayes	44.9 ± 13.2	52.7 ± 10.8	55.4 ± 8.7	**57.2** ± 12.5	56.0 ± 11.4
segmentation	81.8 ± 2.4	80.8 ± 3.1	79.6 ± 3.0	85.3 ± 2.1	**86.6** ± 2.0
abalone	22.6 ± 2.1	21.7 ± 1.7	21.2 ± 3.0	22.1 ± 2.1	**22.6** ± 1.6
yeast	73.2 ± 2.3	73.6 ± 2.5	74.2 ± 3.1	74.9 ± 2.8	**74.9** ± 3.0
libras	48.4 ± 15.1	56.1 ± 16.3	65.5 ± 15.3	**68.3** ± 12.2	66.8 ± 13.4
pageblocks	71.9 ± 3.0	**73.7** ± 2.9	69.7 ± 5.1	71.8 ± 3.2	72.4 ± 3.0
Mean	67.3 ± 4.2	69.3 ± 4.2	70.1 ± 4.3	70.8 ± 4.1	**72.1** ± 3.8
Average Rank	4.30	3.26	3.00	2.96	1.48

over/under-sampling techniques [1] or create synthetic samples in the neighborhood of the minority class—for example by using SMOTE-like strategies [8]. We now aim at studying the behavior of those methods when used as a pre-process of the metric learning procedures. We consider the results of Table 2 as baselines. We compare them to the performances obtained after performing prior to metric learning an over-sampling using SMOTE and a Random Under Sampling (RUS) strategy of the negative data. We use the implementations of these methods from the Python library *imbalanced-learn* [24].

The results obtained are reported in Table 3a for SMOTE and in Table 3b for RUS. Note that the results from Tables 2, 3a and 3b were computed using the same training/test splits and the same validation folds and are

Table 3: Average F1-measure ± standard deviation over 20 splits using different metric learning algorithms, but with a pre-processing of the data.

(a) With a smote pre-processing [8] until $n^+ = n^-$.

Dataset	Euclidean	LMNN	ITML	GMML	IML
splice	74.9 ± 0.9	76.4 ± 1.3	79.6 ± 1.2	86.3 ± 0.8	**87.3** ± 0.8
sonar	72.6 ± 4.2	71.2 ± 4.2	72.6 ± 4.5	73.0 ± 6.4	**75.4** ± 4.2
balance	87.4 ± 1.9	89.7 ± 1.9	92.1 ± 1.4	89.6 ± 1.5	**92.4** ± 2.3
australian	80.3 ± 1.6	81.1 ± 1.8	**82.4** ± 1.5	80.7 ± 3.2	81.8 ± 2.0
heart	77.3 ± 2.0	**77.4** ± 3.7	75.7 ± 4.1	75.0 ± 2.6	77.3 ± 2.0
bupa	54.1 ± 3.1	**55.9** ± 4.1	53.9 ± 3.7	55.4 ± 3.4	55.2 ± 3.2
spambase	85.9 ± 0.7	87.2 ± 0.8	87.4 ± 0.9	88.5 ± 0.6	**89.4** ± 0.7
wdbc	93.4 ± 1.3	93.6 ± 1.5	94.0 ± 1.4	93.5 ± 2.2	**94.4** ± 1.5
iono	78.4 ± 2.6	78.3 ± 3.9	77.5 ± 3.8	77.7 ± 3.7	**79.4** ± 3.7
pima	60.1 ± 2.6	60.3 ± 2.8	60.8 ± 2.2	60.1 ± 2.1	**61.3** ± 1.8
wine	92.7 ± 2.8	94.7 ± 3.0	**96.3** ± 2.6	95.3 ± 3.0	96.2 ± 3.1
glass	66.6 ± 2.9	**67.2** ± 3.9	64.6 ± 3.2	66.1 ± 4.0	64.8 ± 6.4
newthyroid	87.6 ± 3.5	89.6 ± 4.1	**91.6** ± 3.2	88.7 ± 4.0	91.0 ± 3.3
german	46.3 ± 2.2	46.0 ± 2.3	46.4 ± 1.8	45.4 ± 3.5	**46.6** ± 1.9
vehicle	80.6 ± 2.1	89.5 ± 2.1	89.9 ± 3.1	**92.0** ± 1.6	91.0 ± 2.3
spectfheart	47.4 ± 2.3	49.1 ± 4.4	46.7 ± 6.9	41.9 ± 8.5	**49.6** ± 6.0
hayes	68.0 ± 6.8	**69.6** ± 7.4	67.8 ± 7.8	64.4 ± 7.6	67.7 ± 8.7
segmentation	82.0 ± 1.9	81.5 ± 1.8	81.6 ± 2.2	83.8 ± 2.9	**84.5** ± 3.0
abalone	**32.3** ± 0.7	31.7 ± 1.1	31.9 ± 0.8	31.4 ± 1.7	31.9 ± 1.1
yeast	65.9 ± 2.9	68.3 ± 2.6	70.4 ± 2.8	67.1 ± 3.7	**70.5** ± 2.9
libras	68.3 ± 8.1	69.3 ± 10.9	69.7 ± 13.8	76.7 ± 8.5	**77.9** ± 12.2
pageblocks	62.0 ± 2.9	61.6 ± 3.5	55.5 ± 4.0	61.5 ± 4.1	**62.5** ± 3.7
Mean	71.1 ± 2.7	72.2 ± 3.3	72.2 ± 3.5	72.5 ± 3.6	**74.0** ± 3.5
Average Rank	3.78	3.09	3.09	3.43	1.61

(b) With a Random Under Sampling of the negative examples until $n^- = n^+$.

Dataset	Euclidean	LMNN	ITML	GMML	IML
splice	75.9 ± 0.7	76.2 ± 1.2	79.5 ± 1.6	86.5 ± 0.6	**87.4** ± 0.7
sonar	70.4 ± 5.2	69.9 ± 5.2	70.2 ± 5.7	73.1 ± 6.3	**73.2** ± 4.5
balance	87.5 ± 1.5	90.1 ± 1.8	**92.8** ± 1.5	90.1 ± 1.3	92.8 ± 1.9
australian	80.4 ± 1.7	81.5 ± 2.3	82.2 ± 1.6	81.7 ± 2.2	**82.3** ± 2.3
heart	**77.4** ± 2.0	77.3 ± 1.9	76.6 ± 2.5	75.8 ± 3.3	77.3 ± 2.6
bupa	53.8 ± 4.1	**55.7** ± 3.6	54.6 ± 4.1	54.1 ± 4.8	54.5 ± 3.9
spambase	85.0 ± 1.0	86.2 ± 1.1	86.8 ± 1.2	88.1 ± 1.1	**88.7** ± 0.8
wdbc	93.7 ± 1.2	93.2 ± 2.1	93.6 ± 1.8	92.9 ± 1.6	**94.7** ± 1.4
iono	73.1 ± 5.2	74.8 ± 3.3	75.4 ± 3.4	73.3 ± 4.1	**75.6** ± 3.4
pima	60.8 ± 2.7	60.9 ± 2.4	**62.2** ± 1.7	60.5 ± 2.1	60.5 ± 1.9
wine	91.2 ± 2.6	92.7 ± 3.8	**94.2** ± 3.8	94.1 ± 2.8	93.5 ± 3.1
glass	64.6 ± 3.5	**64.6** ± 3.1	61.1 ± 4.9	63.2 ± 4.5	64.2 ± 5.1
newthyroid	86.6 ± 4.6	90.4 ± 3.0	91.1 ± 4.9	91.4 ± 5.0	**92.1** ± 2.9
german	46.7 ± 1.6	**47.5** ± 1.7	47.3 ± 2.3	46.9 ± 2.5	46.4 ± 1.8
vehicle	74.0 ± 3.1	85.5 ± 3.1	87.7 ± 2.6	**89.7** ± 1.6	88.5 ± 2.6
spectfheart	44.2 ± 3.9	45.5 ± 5.4	46.7 ± 4.6	42.6 ± 8.0	**49.2** ± 5.0
hayes	63.4 ± 9.0	67.0 ± 7.7	64.7 ± 6.7	**67.7** ± 7.4	63.4 ± 8.2
segmentation	64.6 ± 3.1	64.3 ± 2.9	65.7 ± 3.4	70.4 ± 2.4	**74.6** ± 1.9
abalone	**32.8** ± 1.1	31.6 ± 1.0	32.5 ± 1.3	32.5 ± 1.4	32.3 ± 1.5
yeast	57.2 ± 4.5	59.7 ± 3.8	60.9 ± 3.8	60.8 ± 4.6	**61.8** ± 3.0
libras	34.3 ± 10.6	36.5 ± 12.6	38.2 ± 12.2	35.6 ± 10.9	**41.5** ± 11.3
pageblocks	46.8 ± 3.7	48.3 ± 4.5	43.0 ± 5.2	**50.2** ± 4.8	49.1 ± 4.1
Mean	66.6 ± 3.5	68.2 ± 3.5	68.5 ± 3.7	69.1 ± 3.8	**70.2** ± 3.3
Average Rank	4.04	3.30	2.65	2.96	2.04

thus comparable. In each of the three settings considered, **IML** obtains the best results showing that it is more appropriate for improving the F1-measure. We also note that SMOTE allows one to increase significantly the performances of all methods, while there is no gain with RUS in comparison with an approach without sampling.

This increase of performance suggests that SMOTE and **IML** are more complementary than competitors with different objectives (re-balancing for the former and representation learning for the latter).

Third experiment—increasing the imbalance We now aim at showing the efficiency of our method by artificially increasing and decreasing the imbalance. For a given dataset, we create a maximum of 10 synthetic variants where the percentage of minority examples is in $\{50\%, 40\%, 30\%, 20\%, 10\%, 5\%, 4\%, 3\%, 2\%, 1\%\}$. To create a synthetic variant of a dataset with a percentage of minority examples higher than in the original dataset, we apply a random under sampling of the majority class until the desired percentage is reached. Similarly, to create a synthetic variant with a smaller percentage of minority examples, we apply a random under sampling of the minority class. Note that we create the synthetic variant of the dataset only if it contains at least 20 minority examples. For example, for the dataset SPECTFHEART, we cannot go under 10% of minority examples. Due to the small number of minority examples present in the more

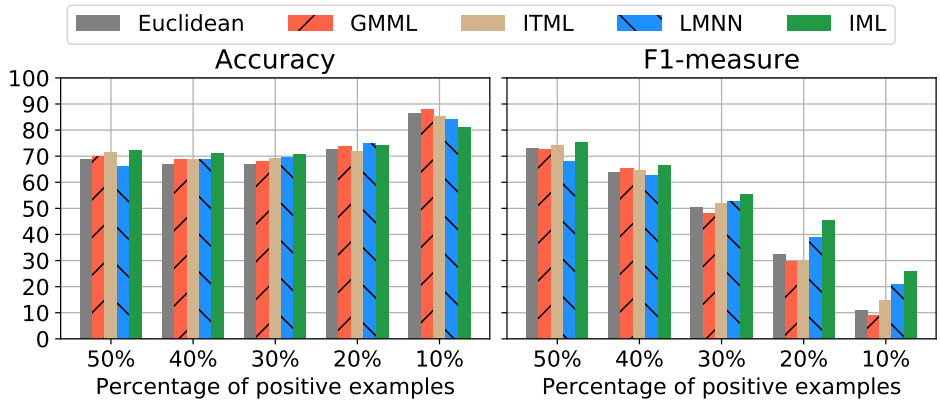


Figure 3: Mean Accuracy and F1-measure over 20 splits on the SPECTFHEART dataset by artificially increasing the imbalance. We compare state-of-the-art metric learning algorithms with our proposed method **IML**.

imbalanced synthetic variants of the datasets, we split them into 50% training and 50% test examples. We report the mean results over 20 iterations where at each iteration we recompute the synthetic variants of the dataset and the train/test splits.

The results for the SPECTFHEART dataset (already used in the introduction of this paper) are reported in Figure 3. We see that like the other algorithms, **IML** shows the same drop of F1-measure when increasing the imbalance, which shows the difficulty of learning from imbalanced data. However, it is important to notice that the drop of performances of **IML** is the smallest among all algorithms.

To confirm the efficiency of **IML** when facing imbalance data on a wide range of datasets, we present in Figure 4 the results of the same experiments by averaging the results over all the datasets. We observe the same behavior as for the SPECTFHEART dataset. Again, it is worth noticing that **IML** is always more robust while facing imbalanced classes.

Fourth experiment—analyzing why IML is better than the other metric learning algorithms on imbalanced data When we described **IML** in Section 3, we presented two strategies to deal with the imbalance. The first one, which is already used by some existing metric learning methods, is a selection of the similar and dissimilar pairs based on the nearest neighbor rule. The second one which we proposed in this paper is to weight differently the set of pairs based on the labels of the two examples compos-

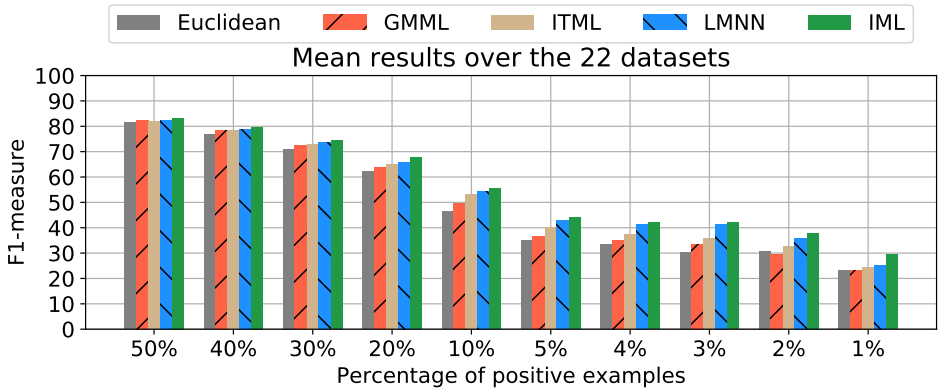


Figure 4: Mean F1-measure over 20 splits and over the 22 datasets by artificially increasing the imbalance. We compare state-of-the-art metric learning algorithms with our proposed method **IML**.

ing the pairs. To see the impact of these two strategies, we compare in this last experiment **IML** with two variants.

The variant called **ML2** considers the loss of **IML** but without the re-weighting of the set of pairs. Its loss is defined as follows:

$$\begin{aligned}\min_{\mathbf{M}\succeq 0} F(\mathbf{M}) \;= a \sum_{(z,z')\in Sim^+} \ell_1(\mathbf{M},z,z')+ \\ a \sum_{(z,z')\in Sim^-} \ell_1(\mathbf{M},z,z')+ \\ (1-a) \sum_{(z,z')\in Dis^+} \ell_2(\mathbf{M},z,z')+ \\ (1-a) \sum_{(z,z')\in Dis^-} \ell_2(\mathbf{M},z,z')+ \\ \lambda \left\|\mathbf{M}-\mathbf{I}\right\|_F^2, \end{aligned} \tag{4}$$

where the difference with (3) is that we no longer multiply each of the four sets by $\frac{1}{4|\text{set}|}$.

The variant called **ML1** considers the same loss as **ML2**, but we select randomly the pairs of examples. In order to use the same number of pairs in **ML1** and **IML**, we draw randomly $2nk$ pairs for **ML1** since **IML** considers k similar pairs and k dissimilar pairs per training example.

The results of this experiment are reported in Figure 5. When the classes are balanced with 50% of minority examples, we observe that **IML** and its

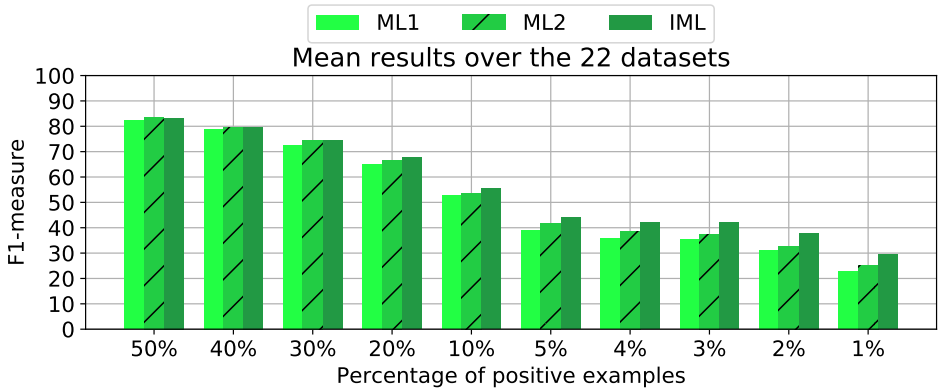


Figure 5: Mean F1-measure over 20 splits and over the 22 datasets by artificially increasing the imbalance. We compare two variants of **IML**. The variant **ML2** removes one component that allows **IML** to deal with the imbalance: the re-weighting of the pairs. **ML1** removes in addition another component to deal with the imbalance: instead of selecting the similar and dissimilar pairs with the nearest neighbor rule, they are selected randomly by **ML1**.

two variants present the same performances. When increasing the imbalance, as expected **IML** tends to be better than **ML2** itself better than **ML1**. This shows that our two strategies to deal with imbalanced data do not degrade the results on balanced data and that they are complementary to improve the performances in an imbalanced setting.

5 Conclusion and perspectives

In this paper, we revisit the classic formulation of metric learning algorithms that learn a Mahalanobis metric in the light of imbalanced data issues. Our method resorts to two complementary strategies to deal with the imbalance. First, unlike the state of the art methods that do not make any distinction between the pairs, we propose to decompose the usual loss with respect to the different possible labels involved in the pairs of examples. This decomposition allows us to assign specific weights to each type of pairs in order to improve the performance on the minority class. Second, contrarily to many metric algorithms that select the pairs of similar and dissimilar examples randomly, we select them based on the nearest neighbor rule. Our experimental evaluation shows that we are able to obtain better results than

state of the art metric learning algorithms in terms of F1-measure over balanced and imbalanced datasets. Last but not least, artificially increasing the imbalance in the datasets shows that our two strategies to deal with the imbalance are complementary.

We believe that our work gives rise to exciting perspectives when facing imbalanced data. Among them, we want to study how our algorithm could be adapted to learn non-linear metrics. From an algorithmic point of view, we would like to extend our method by deriving a closed form solution in a similar way as done by Zadeh *et al.* [40] to drastically reduce the computation time while maintaining good performances. Finally, Problem (3) opens the door to the derivation of generalization guarantees based for example on the uniform stability framework.

References

[1] C. Aggarwal. *Outlier Analysis*. Springer, 2013.

[2] R. Bauder, T. Khoshgoftaar, and T. Hasanin. Data sampling approaches with severely imbalanced big data for medicare fraud detection. In *ICTAI*, pages 137–142. IEEE, 2018.

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

[4] A. Bellet, A. Habrard, and M. Sebban. Metric learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 9(1), 2015.

[5] P. Branco, L. Torgo, and R. Ribeiro. A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys (CSUR)*, 49(2):31, 2016.

[6] Q. Cao, Z. Guo, and Y. Ying. Generalization bounds for metric and similarity learning. *Machine Learning*, 102(1):115–132, 2016.

[7] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[8] N. Chawla, K. Bowyer, L. Hall, and P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16(1):321–357, June 2002.

[9] N. Chawla, A. Lazarevic, L. Hall, and K. Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In *PKDD*, pages 107–119. Springer, 2003.

[10] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon. Information-theoretic metric learning. In *ICML*, 2007.

[11] G. Douzas and F. Bacao. Effective data generation for imbalanced learning using conditional generative adversarial networks. *Expert Systems with applications*, 91:464–471, 2018.

[12] C. Drummond and R. Holte. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11, pages 1–8. Citeseer, 2003.

[13] C. Elkan. The foundations of cost-sensitive learning. In *IJCAI*, pages 973–978, 2001.

[14] A. Estabrooks, T. Jo, and N. Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational intelligence*, 20(1):18–36, 2004.

[15] L. Feng, H. Wang, B. Jin, H. Li, M. Xue, and L. Wang. Learning a distance metric by balancing kl-divergence for imbalanced datasets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

[16] L. Ferreira, J. Barddal, F. Enembreck, and H. Gomes. Improving credit risk prediction in online peer-to-peer (p2p) lending using imbalanced learning techniques. In *ICTAI*, pages 175–181. IEEE, 2017.

[17] J. Frery, A. Habrard, M. Sebban, O. Caelen, and L. He-Guelton. Efficient top rank optimization with gradient boosting for supervised anomaly detection. In *ECML-PKDD*, 2017.

[18] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics*, 42(4):463–484, 2012.

[19] H. Han, W. Wang, and B. Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *ICIC*, pages 878–887. Springer, 2005.

[20] H. He and E. Garcia. Learning from imbalanced data. *IEEE TKDE*, 21(9), 2009.

[21] R. Jin, S. Wang, and Y. Zhou. Regularized distance metric learning: Theory and algorithm. In *NIPS*, 2009.

[22] B. Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2013.

[23] J. Lee, R. Jin, and A. Jain. Rank-based distance metric learning: An application to image retrieval. In *CVPR*, 2008.

[24] G. Lemaître, F. Nogueira, and C. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research*, 18(1):559–563, 2017.

[25] X. Liu, J. Wu, and Z. Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2009.

[26] V. López, A. Fernández, S. García, V. Palade, and F. Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information sciences*, 250:113–141, 2013.

[27] J. Lu, X. Zhou, Y. Tan, Y. Shang, and J. Zhou. Neighborhood repulsed metric learning for kinship verification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(2):331–345, 2014.

[28] B. McFee and G. Lanckriet. Metric learning to rank. In *ICML*, 2010.

[29] R. Pereira, Y. Costa, and C. Silla. Dealing with imbalanceness in hierarchical multi-label datasets using multi-label resampling techniques. In *ICTAI*, pages 818–824. IEEE, 2018.

[30] B. Schölkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In *KDD*, pages 252–257, 1995.

[31] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *NIPS*, 2004.

[32] C. van Rijsbergen. Further experiments with hierarchic clustering in document retrieval. *Information Storage and Retrieval*, 10(1):1 – 14, 1974.

[33] V. Vapnik. The nature of statistical learning theory. 1995.

[34] R. Vogel, A. Bellet, and S. Clémençon. A probabilistic theory of supervised similarity learning for pointwise ROC curve optimization. *ICML*, 2018.

[35] N. Wang, X. Zhao, Y. Jiang, and Y. Gao. Iterative metric learning for imbalance data classification. In *IJCAI*, 2018.

[36] K. Weinberger and L. Saul. Fast solvers and efficient implementations for distance metric learning. In *ICML*, pages 1160–1167. ACM, 2008.

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

[38] S. Xiang, F. Nie, and C. Zhang. Learning a mahalanobis distance metric for data clustering and classification. *Pattern Recognition*, 41(12):3600–3612, 2008.

[39] E. Xing, M. Jordan, S. Russell, and A. Ng. Distance metric learning with application to clustering with side-information. In *NIPS*, 2003.

[40] P. Zadeh, R. Hosseini, and S. Sra. Geometric mean metric learning. In *ICML*, 2016.

[41] B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *ICDM*. IEEE, 2003.

[42] W. Zheng, S. Gong, and T. Xiang. Person re-identification by probabilistic relative distance comparison. In *CVPR 2011*, pages 649–656. IEEE, 2011.

[43] C. Zhu, R. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM TOMS*, 23(4):550–560, 1997.