Apprentissage robuste de distance par géométrie riemannienne

Antoine COLLAS[1], Arnaud BRELOY[2], Guillaume GINOLHAC[3], Chengfang REN[1], Jean-Philippe OVARLEZ[1,4],

[1]CentraleSupélec SONDRA, Université Paris-Saclay, 3 rue Joliot Curie, 91190 Gif-sur-Yvette, France,

[2]LEME, Universié Paris-Nanterre, 50 Rue de Sèvres, 92410 Ville-d'Avray, France,

[3]LISTIC, Université Savoie Mont-Blanc, 5 Chemin de bellevue, Annecy-Le-Vieux, 74940 Annecy, France,

[4]DEMR, ONERA, Université Paris-Saclay, 8 Chemin de la Hunière, 91123 Palaiseau, France,

antoine.collas@centralesupelec.fr

Résumé – L'*apprentissage de distance* propose d'optimiser la distance de *Mahalanobis* pour des problèmes de classification. Plusieurs algorithmes associés à ce problème s'interprètent comme des estimateurs d'une matrice de covariance. Partant de ce constat, nous proposons un nouveau problème d'estimation appelé *Robust Geometric Metric Learning* (*RGML*) qui vise à estimer une matrice de covariance par classe et leur barycentre riemannien. Deux algorithmes associés à ce problème d'estimation sont développés : *RGML gaussien* et *RGML Tyler*. Ces algorithmes optimisent deux fonctions de coût sur la variété riemannienne des matrices symétriques définies positives et sa sous-variété à déterminant unitaire. Enfin, les algorithmes proposés sont appliqués sur des jeux de données réels. De bonnes performances ainsi qu'une robustesse aux données mal étiquetées sont obtenues.

Abstract – *Metric learning* proposes to optimize the *Mahalanobis* distance for classification problems. Several algorithms associated with this problem are interpreted as estimators of a covariance matrix. Based on this observation, we propose a new estimation problem called *Robust Geometric Metric Learning* (*RGML*) which aims to estimate a covariance matrix per class and their Riemannian barycenter. Two algorithms associated with this estimation problem are developed: *RGML Gaussian* and *RGML Tyler*. These algorithms optimize two cost functions on the Riemannian manifold of symmetric positive definite matrices and its submanifold of unitary determinant. Finally, the proposed algorithms are applied on real datasets. Good performance as well as robustness to mislabeled data are obtained.

1 Introduction

De nombreux algorithmes de classification et de partitionnement reposent sur une distance entre les données. Ces algorithmes incluent les classiques *K-means*, *Classifieur du centroïde le plus proche*, et *k-plus proches voisins*. La définition de cette distance a une importance cruciale puisqu'elle détermine la similarité entre points. En pratique, ces algorithmes reposent le plus souvent sur la distance euclidienne qui est $d_{\boldsymbol{I}_p}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$ pour $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^p$. Cependant, cette distance est sujette à plusieurs problèmes. Un exemple est lorsque deux classes ont une variance élevée le long d'un axe commun : dans cette configuration, deux données de la même classe peuvent être éloignées l'une de l'autre, tandis que deux données de deux classes différentes peuvent être très proches.

Afin de trouver une distance plus pertinente pour la classification, le problème de l'*apprentissage de distance* a été proposé. Il vise à trouver une distance de Mahalanobis

$$d_{\boldsymbol{A}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{A}^{-1} (\boldsymbol{x}_i - \boldsymbol{x}_j)} \qquad (1)$$

qui rapproche les données de même classe et éloigne les données de classes différentes. L'*apprentissage de distance* consiste à formaliser un problème d'optimisation par rapport à $\boldsymbol{A} \in \mathcal{S}_p^+$, l'ensemble des matrices symétriques définies positives (SDP) d'ordre p, pour déterminer au mieux cette distance.

Soit un problème de classification à K classes : m données $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_m\} \subset \mathbb{R}^p$ étiquetées dans $[\![1, K]\!]$. Les données de la classe k sont notées $\{\boldsymbol{x}_{kl}\}$ et n_k paires distinctes $(\boldsymbol{x}_{kl}, \boldsymbol{x}_{kq})$ avec $l \neq q$ sont formées. L'ensemble S_k contient tous ces couples et S contient les $n_{\mathcal{S}} = \sum_{k=1}^{K} n_k$ couples de toutes les classes. Le rapport $\frac{n_k}{n_{\mathcal{S}}}$ est noté π_k. Ensuite, pour chaque classe k, n_k vecteurs $\boldsymbol{s}_{ki}$ sont formés par soustraction des paires $(\boldsymbol{x}_{kl}, \boldsymbol{x}_{kq}) \in \mathcal{S}_k$, *i.e.* $\boldsymbol{s}_{ki} = \boldsymbol{x}_{kl} - \boldsymbol{x}_{kq}$. L'ensemble $\mathcal{D}$ contient $n_{\mathcal{D}}$ paires de vecteurs qui n'appartiennent pas à la même classe. Chaque vecteur $\boldsymbol{d}_i$ est défini comme la soustraction des éléments de chaque paire dans $\mathcal{D}$, $\boldsymbol{d}_i = \boldsymbol{x}_l - \boldsymbol{x}_q$ avec $(\boldsymbol{x}_l, \boldsymbol{x}_q) \in \mathcal{D}$. $\mathcal{S}_p$ est l'ensemble des matrices symétriques d'ordre p, et $\mathcal{SS}_p^+$ est le sous-ensemble de $\mathcal{S}_p^+$ des matrices de déterminant unitaire. Enfin, $|\,.\,| : \mathcal{S}_p \to \mathbb{R}$ est l'opérateur déterminant.

1.1 État de l'art et estimation de la matrice de covariance

De nombreux travaux d'*apprentissage de distance* existent dans la littérature (voir [1] pour une étude complète). Dans ce qui suit, nous en présentons deux importants.

ITML (*Information-Theoretic Metric Learning*) [2] propose

de trouver une matrice $\boldsymbol{A}$ qui reste proche d'une matrice prédéfinie $\boldsymbol{A}_0$ tout en respectant des contraintes de similarités et de dissimilarités. La *divergence de Kullback-Leibler gaussienne* $D_{KL}(\boldsymbol{A}_0, \boldsymbol{A}) = \mathrm{Tr}(\boldsymbol{A}^{-1}\boldsymbol{A}_0) + \log|\boldsymbol{A}\boldsymbol{A}_0^{-1}|$ mesure la proximité entre $\boldsymbol{A}$ et $\boldsymbol{A}_0$. Le problème d'optimisation *ITML* est

$$\begin{aligned} \underset{\boldsymbol{A}\in\mathcal{S}_p^+}{\text{minimiser}} \quad & \mathrm{Tr}(\boldsymbol{A}^{-1}\boldsymbol{A}_0) + \log|\boldsymbol{A}| \\ & d^2_{\boldsymbol{A}}(\boldsymbol{x}_l, \boldsymbol{x}_q) \leq u, \quad (\boldsymbol{x}_l, \boldsymbol{x}_q) \in \mathcal{S}, \\ & d^2_{\boldsymbol{A}}(\boldsymbol{x}_l, \boldsymbol{x}_q) \geq l, \quad (\boldsymbol{x}_l, \boldsymbol{x}_q) \in \mathcal{D}, \end{aligned} \tag{2}$$

où $u, l \in \mathbb{R}$ sont des seuils choisis pour rapprocher des points de même classe et éloigner des points de classes différentes. En pratique, $\boldsymbol{A}_0$ est choisi comme la matrice identité ou bien la matrice de covariance empirique. Dans ce dernier cas, (2) revient à maximiser la vraisemblance gaussienne multivariée sous contraintes. Ainsi, *ITML* peut être vu comme un problème d'estimation de matrice de covariance.

Ensuite, *GMML* (*Geometric Mean Metric Learning*) [3] est un algorithme important en *apprentissage de distance*. Il atteint de bonnes performances sur plusieurs jeux de données avec un coût calculatoire faible. Le problème d'optimisation *GMML* est

$$\underset{\boldsymbol{A}\in\mathcal{S}_p^+}{\text{minimiser}} \frac{1}{n_{\mathcal{S}}} \sum_{(\boldsymbol{x}_l,\boldsymbol{x}_q)\in\mathcal{S}} d^2_{\boldsymbol{A}}(\boldsymbol{x}_l, \boldsymbol{x}_q) + \frac{1}{n_{\mathcal{D}}} \sum_{(\boldsymbol{x}_l,\boldsymbol{x}_q)\in\mathcal{D}} d^2_{\boldsymbol{A}^{-1}}(\boldsymbol{x}_l, \boldsymbol{x}_q). \tag{3}$$

L'intuition derrière ce problème est que $d_{\boldsymbol{A}^{-1}}$ éloigne les points de classes différentes tandis que $d_{\boldsymbol{A}}$ rapproche les points de même classe. La formulation (3) se réécrit

$$\underset{\boldsymbol{A}\in\mathcal{S}_p^+}{\text{minimiser}} \mathrm{Tr}(\boldsymbol{A}^{-1}\boldsymbol{S}) + \mathrm{Tr}(\boldsymbol{A}\boldsymbol{D}) \tag{4}$$

où $\boldsymbol{S} = \frac{1}{n_{\mathcal{S}}} \sum_{k=1}^{K} \sum_{i=1}^{n_k} \boldsymbol{s}_{ki}\boldsymbol{s}_{ki}^T$ et $\boldsymbol{D} = \frac{1}{n_{\mathcal{D}}} \sum_{i=1}^{n_{\mathcal{D}}} \boldsymbol{d}_i\boldsymbol{d}_i^T$. Dans [3], la solution de (4) est calculée. Il s'agit du point milieu de la géodésique entre $\boldsymbol{S}^{-1}$ et $\boldsymbol{D}$, *i.e.* $\boldsymbol{A}^{-1} = \boldsymbol{S}^{-1}\#_{\frac{1}{2}}\boldsymbol{D} = \boldsymbol{S}^{-\frac{1}{2}}\left(\boldsymbol{S}^{\frac{1}{2}}\boldsymbol{D}\boldsymbol{S}^{\frac{1}{2}}\right)^{\frac{1}{2}}\boldsymbol{S}^{-\frac{1}{2}}$. Ensuite, [3] propose de généraliser cette solution par $\boldsymbol{A}^{-1} = \boldsymbol{S}^{-1}\#_t\boldsymbol{D} = \boldsymbol{S}^{-\frac{1}{2}}\left(\boldsymbol{S}^{\frac{1}{2}}\boldsymbol{D}\boldsymbol{S}^{\frac{1}{2}}\right)^{t}\boldsymbol{S}^{-\frac{1}{2}}$ avec $t \in [0, 1]$. Appliquée à des jeux de données réels, les auteurs obtiennent généralement de bonnes performances avec t petit ou nul (voir la Figure 3 de [3]) ce qui revient à $\boldsymbol{A} \approx \boldsymbol{S}$. Cette solution peut être interprétée statistiquement. Supposons que les données soient des réalisations de vecteurs aléatoires indépendants de moments un et deux dépendants de la classe,

$$\boldsymbol{x}_{kl} \overset{d}{=} \boldsymbol{\mu}_k + \boldsymbol{\Sigma}_k^{\frac{1}{2}}\boldsymbol{u}_{kl} \tag{5}$$

avec $\boldsymbol{\mu}_k \in \mathbb{R}^p$, $\boldsymbol{\Sigma}_k \in \mathcal{S}_p^+$, $\boldsymbol{u}_{kl}$ le vecteur aléatoire centré reduit de la classe k, *i.e.* $\mathbb{E}[\boldsymbol{u}_{kl}] = \boldsymbol{0}$ et $\mathbb{E}[\boldsymbol{u}_{kl}\boldsymbol{u}_{kq}^T] = \boldsymbol{I}_p$ si $kl = kq$, $\boldsymbol{0}_p$ sinon. Il s'ensuit que $\boldsymbol{s}_{ki} \overset{d}{=} \boldsymbol{\Sigma}_k^{\frac{1}{2}}(\boldsymbol{u}_l - \boldsymbol{u}_q)$. Par conséquent, la matrice de covariance de $\boldsymbol{s}_{ki}$ est égale à deux fois la matrice de covariance de la classe k, $\mathbb{E}[\boldsymbol{s}_{ki}\boldsymbol{s}_{ki}^T] \overset{d}{=} 2\boldsymbol{\Sigma}_k$. Il en résulte que, en espérance, $\boldsymbol{S}$ est deux fois la moyenne arithmétique des matrices de covariance des différentes classes,

$$\mathbb{E}[\boldsymbol{S}] = \frac{1}{n_{\mathcal{S}}} \sum_{k=1}^{K} \sum_{i=1}^{n_k} \mathbb{E}[\boldsymbol{s}_{ki}\boldsymbol{s}_{ki}^T] = 2\sum_{k=1}^{K} \pi_k\boldsymbol{\Sigma}_k. \tag{6}$$

L'utilisation de $\boldsymbol{S}$ dans la distance (1) rappelle en conséquence l'étape de préblanchiment des données de l'analyse discriminante linéaire.

1.2 Motivations et contributions

D'après la Section 1.1, *GMML* s'interprète comme une méthode à 2 étapes qui calcule, premièrement, la matrice de covariance empirique de chaque classe et, deuxièmement, leur moyenne arithmétique. Au lieu de faire ces 2 étapes, nous proposons dans ce papier, un nouvel algorithme, *Robust Geometric Metric Learning* (*RGML*) qui fera l'estimation jointe régularisée et robuste des matrices de covariance et de leur moyenne riemannienne.

Cette idée a initialement été proposée dans [4]. La contribution de ce papier est quadruple : **1)** *RGML* est formulé pour l'*apprentissage de distance*, **2)** *RGML* fait usage de la distance riemannienne sur $\mathcal{S}_p^+$, **3)** l'optimisation repose sur les géométries riemanniennes de $\mathcal{S}_p^+$ et $\mathcal{SS}_p^+$ [5, 6], **4)** nous montrons que cette méthode est robuste et performante sur des données réelles partiellement mal étiquetées.

2 Formulation du problème

2.1 Formulation générale de *RGML*

La formulation du problème d'optimisation *RGML* est

$$\underset{\theta\in\mathcal{M}_{p,K}}{\text{minimiser}} \left\{ h(\theta) = \sum_{k=1}^{K} \pi_k \left[\mathcal{L}_k(\boldsymbol{A}_k) + \lambda d^2(\boldsymbol{A}, \boldsymbol{A}_k)\right] \right\} \tag{7}$$

où $\theta = (\boldsymbol{A}, \{\boldsymbol{A}_k\})$, $\mathcal{M}_{p,K} = \left(\mathcal{S}_p^+\right)^{K+1}$, $\mathcal{L}_k$ est une fonction de coût d'estimation de la matrice de covariance dépendant de $\{\boldsymbol{s}_{ki}\}$, $\lambda > 0$ et d est une distance entre deux matrices. Dans les prochaines sous-sections, deux fonctions $\mathcal{L}_k$ seront considérées : la log-vraisemblance négative gaussienne et la fonction de coût de Tyler. Une fois que (7) est résolu, la matrice centrale $\boldsymbol{A}$ est utilisée dans la distance de Mahalanobis (1).

Nous continuons en expliquant plus en détails la fonction de coût (7). Pour celà, nous regardons les effets à variables fixées. Tout d'abord, pour une matrice centrale $\boldsymbol{A}$ fixe, (7) se réduit à k problèmes séparables

$$\underset{\boldsymbol{A}_k}{\text{minimiser}} \mathcal{L}_k(\boldsymbol{A}_k) + \lambda d^2(\boldsymbol{A}, \boldsymbol{A}_k), \tag{8}$$

dont les solutions sont des estimations régularisées de $\{\boldsymbol{\Sigma}_k\}$.

Deuxièmement, pour $\{\boldsymbol{A}_k\}$ fixes, résoudre (7) calcule le barycentre riemannien des matrices $\{\boldsymbol{A}_k\}$.

$$\underset{\boldsymbol{A}\in\mathcal{S}_p^+}{\text{minimiser}} \sum_{k=1}^{K} \pi_k d^2(\boldsymbol{A}, \boldsymbol{A}_k). \tag{9}$$

Par exemple, si d est la distance euclidienne $d_E(\boldsymbol{A}, \boldsymbol{A}_k) = \|\boldsymbol{A} - \boldsymbol{A}_k\|_F$, alors le minimum de (9) est la moyenne arithmétique $\sum_{k=1}^{K} \pi_k\boldsymbol{A}_k$. Dans la suite de l'article, nous considérons la distance riemannienne sur $\mathcal{S}_p^+$ [5], c'est-à-dire

$$d_R(\boldsymbol{A}, \boldsymbol{A}_k) = \left\| \mathrm{logm}\left(\boldsymbol{A}^{-\frac{1}{2}}\boldsymbol{A}_k\boldsymbol{A}^{-\frac{1}{2}}\right) \right\|_F. \tag{10}$$

Une propriété remarquable de d_R (10) est son invariance aux transformations affines. En effet, pour tout $\boldsymbol{C}$ inversible, on a $d_R(\boldsymbol{C}\boldsymbol{A}\boldsymbol{C}^T, \boldsymbol{C}\boldsymbol{A}_k\boldsymbol{C}^T) = d_R(\boldsymbol{A}, \boldsymbol{A}_k)$. Ainsi, si $\{\boldsymbol{s}_{ki}\}$ est transformé en $\{\boldsymbol{C}\boldsymbol{s}_{ki}\}$ alors le minimum $(\boldsymbol{A}, \{\boldsymbol{A}_k\})$ de (7) devient $\left(\boldsymbol{C}\boldsymbol{A}\boldsymbol{C}^T, \left\{\boldsymbol{C}\boldsymbol{A}_k\boldsymbol{C}^T\right\}\right)$. Enfin, nous soulignons que l'optimisation de (7) est effectuée par rapport à toutes les matrices $\boldsymbol{A}$ et $\{\boldsymbol{A}_k\}$ en même temps. Ainsi, les matrices de covariance régularisées $\{\boldsymbol{A}_k\}$ sont estimées tout en calculant leur barycentre inconnu $\boldsymbol{A}$.

2.2 RGML gaussien

Pour obtenir une expression de la fonction de coût h (7), il reste à spécifier les fonctions $\mathcal{L}_k$. L'hypothèse la plus classique sur la distribution des données est l'hypothèse gaussienne (par exemple utilisée dans *ITML*). Ainsi, les premières fonctions $\mathcal{L}_k$ considérées sont les log-vraisemblances négatives gaussiennes multivariées centrées

$$\mathcal{L}_{G,k}(\boldsymbol{A}) = \frac{1}{n_k}\sum_{i=1}^{n_k} \boldsymbol{s}_{ki}^T\boldsymbol{A}^{-1}\boldsymbol{s}_{ki} + \log|\boldsymbol{A}|. \qquad (11)$$

Avec cette log-vraisemblance négative, le problème *RGML* d'optimisation (7) devient

$$\underset{\theta\in\mathcal{M}_{p,K}}{\text{minimiser}} \left\{ h_G(\theta) = \sum_{k=1}^{K} \pi_k \left[\mathcal{L}_{G,k}(\boldsymbol{A}_k) + \lambda d_R^2(\boldsymbol{A}, \boldsymbol{A}_k)\right] \right\} \quad (12)$$

et est appelé *RGML Gaussien*.

2.3 RGML Tyler

Lorsqu'une petite proportion des échantillons a un comportement aberrant, les méthodes d'estimation robuste de la matrice de covariance constituent un choix privilégié. Dans un contexte de classification, ce cas se produit lorsque les données sont mal étiquetées. Un estimateur classique est l'estimateur de Tyler [7] qui minimise la fonction de coût suivante

$$\mathcal{L}_{T,k}(\boldsymbol{A}) = \frac{p}{n_k}\sum_{i=1}^{n_k} \log\left(\boldsymbol{s}_{ki}^T\boldsymbol{A}^{-1}\boldsymbol{s}_{ki}\right) + \log|\boldsymbol{A}|. \qquad (13)$$

Une remarque importante est que (13) est invariant par facteur d'échelle de $\boldsymbol{A}$. En effet, pour tout $\alpha > 0$, il est facile de vérifier que $\mathcal{L}_{T,k}(\alpha\boldsymbol{A}) = \mathcal{L}_{T,k}(\boldsymbol{A})$. Ainsi, une contrainte de déterminant unitaire est ajoutée à (7) pour fixer les échelles de $\{\boldsymbol{A}_k\}$. De plus, la distance riemannienne (10) est également celle de $\mathcal{S}\mathcal{S}_p^+$. Ainsi, nous choisissons de contraindre également $\boldsymbol{A}$ pour qu'elle soit la moyenne riemannienne de $\{\boldsymbol{A}_k\}$ sur $\mathcal{S}\mathcal{S}_p^+$. Nous désignons par $\mathcal{S}\mathcal{M}_{p,K}$ ce nouvel espace de paramètres

$$\mathcal{S}\mathcal{M}_{p,K} = \{\theta\in\mathcal{M}_{p,K}, |\boldsymbol{A}| = |\boldsymbol{A}_k| = 1, \forall k \in [\![1, K]\!]\}. \quad (14)$$

Ainsi, le problème d'optimisation (7) avec la fonction de coût de Tyler (13) devient

$$\underset{\theta\in\mathcal{S}\mathcal{M}_{p,K}}{\text{minimiser}} \left\{ h_T(\theta) = \sum_{k=1}^{K} \pi_k \left[\mathcal{L}_{T,k}(\boldsymbol{A}_k) + \lambda d_R^2(\boldsymbol{A}, \boldsymbol{A}_k)\right] \right\} \quad (15)$$

et est appelé *RGML Tyler*.

2.4 Optimisation

Les problèmes (12) et (15) sont résolus à l'aide de descentes de gradient riemanniennes [8]. Les variétés riemanniennes d'intérêt sont $\mathcal{M}_{p,K}$ et $\mathcal{S}\mathcal{M}_{p,K}$ et leurs outils d'optimisation sont hérités directement de ceux de $\mathcal{S}_p^+$ et $\mathcal{S}\mathcal{S}_p^+$ [5, 6]. Dans le reste de cette sous-section, seules quelques remarques sur l'optimisation sont données et les détails techniques sont omis pour des raisons d'espace [1]. Tout d'abord, les problèmes (12) et (15) sont géodésiquement convexes, *i.e.* convexes le long des géodésiques de $\mathcal{M}_{p,K}$ et $\mathcal{S}\mathcal{M}_{p,K}$. Par conséquent, les descentes de gradient sont peu sensibles à l'initialisation. Ensuite, la complexité d'une itération de la descence de gradient *RGML* est dominée par celle de l'évaluation du gradient, en pratique, calculé par différenciation automatique. Ainsi, nous donnons la complexité de l'évaluation de la fonction de coût : $\mathcal{O}(n_{\mathcal{S}}\, p^2 + Kp^3)$. La complexité de la forme fermée de *GMML* est $\mathcal{O}(np^2 + p^3)$ où $n = \max\{n_{\mathcal{S}}, n_{\mathcal{D}}\}$. K étant le nombre de classes (et donc peu élevé), les deux algorithmes ont des compléxités proches. Enfin, l'hyperparamètre λ dans (7) peut être choisi par validation croisée.

3 Expériences

Dans cette section, nous présentons une application de la méthode *RGML* développée dans la Section 2. En effet, nous l'appliquons sur des jeux de données du *UCI machine learning repository* [9]. Trois jeux de données de classification sont considérés : *Wine*, *Vehicle* et *Iris*. Leurs statistiques, nombre et taille des données ainsi que le nombre de classes, sont présentées dans la Table 1. Ces ensembles de données ont des classes quasiment équiprobables, *c'est-à-dire* que les classes ont des nombres de données proches. Les nombres de paires générées dans $\mathcal{S}$ et $\mathcal{D}$ sont $n_{\mathcal{S}} = n_{\mathcal{D}} = 75\, K(K-1)$ (voir [2] et [3]) .

La validation se fait selon un protocole très classique en *apprentissage de distance*. **1)** Une matrice $\boldsymbol{A}$ est estimée via une méthode d'*apprentissage de distance*. **2)** Les données $\{\boldsymbol{x}_l\}$ sont multipliées par $\boldsymbol{A}^{-\frac{1}{2}}$ pour obtenir $\{\boldsymbol{A}^{-\frac{1}{2}}\boldsymbol{x}_l\}$. **3)** Les données $\{\boldsymbol{A}^{-\frac{1}{2}}\boldsymbol{x}_l\}$ sont classées en utilisant un *k-plus proches voisins* avec 5 voisins. Ainsi, la classification est faite en utilisant la distance de Mahalanobis $d_{\boldsymbol{A}}$ (1) définie dans l'Introduction. Cette classification est répétée 200 fois via une validation croisée. La répartition des données dans les ensembles d'entrainement et de test est de $50/50\%$. L'erreur moyenne de mauvaise classification est reportée dans la Table 1. Afin de montrer la robustesse de la méthode proposée, des données mal étiquetées sont introduites. Pour ce faire, nous sélectionnons au hasard des données dans l'ensemble d'apprentissage dont les étiquettes sont ensuite changées au hasard.

Les implémentations de la validation croisée ainsi que du *k-plus proches voisins* proviennent de la bibliothèque scikit-learn [10]. La valeur choisie du paramètre λ est 0.05. Sa valeur a peu d'impact sur les performances tant qu'elle est ni trop petite ni trop grande. Ils sont comparés aux algorithmes

1. Un programme Python de ces méthodes est disponible ici : `https://github.com/antoinecollas/robust_metric_learning`.

TABLE 1 – Taux d'erreurs de classification calculés sur 3 jeux de données : Wine, Vehicle, et Iris. Les meilleurs résultats sont en **gras**. Les taux d'étiquetage erroné indiquent les taux d'étiquettes modifiées de façon aléatoire dans l'ensemble d'apprentissage.

	Wine $p = 13$, $n = 178$, $K = 3$				Vehicle $p = 18$, $n = 846$, $K = 4$				Iris $p = 4$, $n = 150$, $K = 3$			
Méthodes	Taux d'étiquetage erroné				Taux d'étiquetage erroné				Taux d'étiquetage erroné			
	0%	5%	10%	15%	0%	5%	10%	15%	0%	5%	10%	15%
Identité	30.12	30.40	31.40	32.40	38.27	38.58	39.46	40.35	3.93	4.47	5.31	**6.70**
Cov. emp.	10.03	11.62	13.70	17.57	23.59	24.27	25.24	26.51	12.57	13.38	14.93	16.68
ITML - Identité	3.12	4.15	5.40	**7.74**	24.21	23.91	24.77	26.03	3.04	4.47	5.31	**6.70**
ITML - Cov. emp.	2.45	4.76	6.71	10.25	23.86	23.82	24.89	26.30	3.05	13.38	14.92	16.67
GMML	2.16	3.58	5.71	9.86	21.43	22.49	23.58	25.11	2.60	5.61	9.30	12.62
LMNN	4.27	6.47	7.83	9.86	20.96	24.23	26.28	28.89	3.53	9.59	11.19	12.22
RGML - gaussien	**2.07**	**2.93**	5.15	9.20	**19.76**	21.19	22.52	24.21	**2.47**	5.10	8.90	12.73
RGML - Tyler	**2.12**	**2.90**	**4.51**	8.31	19.90	**20.96**	**22.11**	**23.58**	**2.48**	**2.96**	**4.65**	7.83

d'*apprentissage de distance* classiques : la matrice identité, la matrice de covariance empirique calculée sur toutes les données, *ITML* [2], *GMML* [3], et *LMNN* [11]. Les implémentations de la bibliothèque metric-learn [12] sont utilisées pour les trois derniers algorithmes.

À partir de la Table 1, plusieurs observations sont faites. Tout d'abord, sur les données brutes (*c'est-à-dire* lorsque le taux de mauvais étiquetage est de 0%) le *RGML gaussien* est toujours l'algorithme le plus performant parmi ceux testés. De plus, le *RGML Tyler* est proche avec un écart maximum de 0.26% par rapport au *RGML gaussien*. Ensuite, le *RGML Tyler* est l'algorithme le plus performant lorsque le taux d'erreur d'étiquetage est de 5 % ou de 10 %. Lorsque le taux de mauvais étiquetage est de 15 %, *RGML Tyler* est l'algorithme le plus performant pour le jeu de données *Vehicle* et il n'est battu que par *ITML - identité* sur les deux autres jeux de données. Ceci montre l'intérêt de considérer la fonction de coût de Tyler (13) en présence d'un mauvais étiquetage. Enfin, les algorithmes *RGML* sont rapides. La Figure 1 montre que *RGML gaussien* et *RGML Tyler* convergent en moins de 20 itérations sur le jeu de données *Wine* (sans changement d'étiquettes).

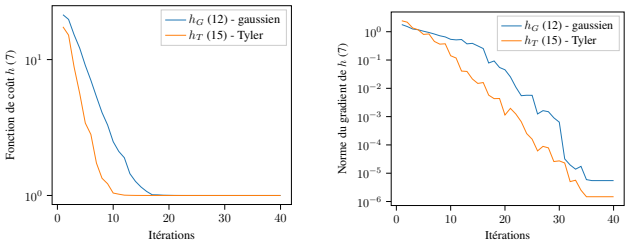

FIGURE 1 – Gauche : évolution des fonctions de coût *RGML gaussien* (12) et *RGML Tyler* (15) en fonction des itérations. Droite : évolution des normes des gradients des fonctions (12) et (15). L'optimisation est réalisée sur le jeu de données *Wine*.

Références

[1] "A tutorial on distance metric learning : Mathematical foundations, algorithms, experimental analysis, prospects and challenges," *Neurocomputing*, vol. 425.

[2] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proceedings of the 24th International Conference on Machine Learning*. New York, NY, USA : Association for Computing Machinery, 2007.

[3] P. H. Zadeh, R. Hosseini, and S. Sra, "Geometric mean metric learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. JMLR.org, 2016.

[4] E. Ollila, I. Soloveychik, D. E. Tyler, and A. Wiesel, "Simultaneous penalized m-estimation of covariance matrices using geodesically convex optimization," 2016.

[5] L. T. Skovgaard, *Scandinavian Journal of Statistics*, no. 4.

[6] X. Pennec, P. Fillard, and N. Ayache, "A riemannian framework for tensor computing," *International Journal of Computer Vision*, vol. 66, 2005.

[7] D. E. Tyler, "A Distribution-Free M-Estimator of Multivariate Scatter," *The Annals of Statistics*, vol. 15, no. 1.

[8] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.

[9] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn : Machine learning in Python," *Journal of Machine Learning Research*, vol. 12.

[11] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *The Journal of Machine Learning Research*, vol. 10.

[12] W. de Vazelhes, C. Carey, Y. Tang, N. Vauquier, and A. Bellet, "metric-learn : Metric Learning Algorithms in Python," *Journal of Machine Learning Research*, vol. 21, no. 138, 2020.