École nationale
de la statistique
et de l'administration
économique

université
PARIS-SACLAY

Rapport du projet de *Geometric Methods in Machine Learning*

Réduction de dimension d'images et retour de l'espace réduit à l'espace des images

Sommaire

Benjamin Phan

Mai 2018

1 Introduction

J'ai voulu lors de ce projet représenter les images dans un espace de dimension réduit (par exemple un plan). J'étais de curieux de savoir si on pouvait trouver des similarités entre les images et pouvoir les représenter visuellement. Par ailleurs, un autre de mes objectifs était de créer de nouvelles images à partir d'une base de données d'image. J'ai eu l'idée de construire un modèle permettant de passer de l'espace de dimension réduite à l'espace des images en l'entraînant sur une base d'images et leur plongement en dimension réduite. Une fois le modèle entraîné, on pourrait transformer n'importe quel point de l'espace réduit en image et espérer éventuellement retrouvé des similarités avec les images correspondants à des points proches.

J'ai utilisé dans cet objectif une base d'images d'un challenge Kaggle [1] contenant environ 80000 images de peintures ou de dessins.

2 Présentation de la démarche

On dispose de $n \approx 80000$ images en format *jpg*. Elles n'ont pas toutes les mêmes dimension alors on les redimensionne toutes à la même taille ($50 \times 50px$) et on les converti en échelle de gris pour limiter le volume de données puis chaque image est représentée sous forme de vecteur en mettant à la suite toutes les lignes. Chaque image est donc représentée par un vecteur de $\mathbb{R}^{2500}$.

Ensuite on souhaite représenter ces images dans un espace de dimension réduite. Pour ce faire on va utiliser l'algorithme ISOMAP (implémenté dans *scikit-learn*). Cependant, si on utilise directement les données brutes, la distance euclidienne serait utilisée alors que la distance euclidienne n'est peut être pas la plus adaptée pour identifier les similarités des images. À la place on utilise la distance de Mahalanobis. On rappelle que la distance de Mahalanobis entre deux vecteurs x, y est définie par :

$$d(x,y)_\Omega = \sqrt{(x-y)^T \Omega (x-y)} \tag{1}$$

avec Ω une matrice carrée de même dimension que les vecteurs. On choisit $\Omega = \Sigma^{-1}$, l'inverse de la matrice de variance-covariance de nos données. La matrice de variance-covariance étant symétrique définie positive, on peut réécrire la distance de Mahalanobis de la manière suivante :

$$d(x,y)_\Omega = \sqrt{(L(x-y))^T L(x-y)} \tag{2}$$

avec L la racine carrée de Ω.

Utiliser cette distance de Mahalanobis revient donc à utiliser la distance euclidienne sur les données transformées par L. On calcule donc la racine carrée de l'inverse de la matrice de variance-covariance et on l'applique aux données.

On utilise ensuite ISOMAP sur les données transformées. On les représente en 2D. L'exécution de l'algorithme étant excessivement longue, je n'ai utilisé qu'environ 10% des données dans l'ISOMAP. J'ai représenté les points dans le plan dans la figure 1. La structure du nuage de point a l'air un peu étoilée avec 4 branches. Cependant, on ne distingue pas de séparations suivant le style de peinture représenté. Ainsi, on peut retrouver par exemple des portraits partout dans le nuage de point.

Une fois qu'on a fait un plongement des images dans le plan, on cherche un moyen de passer du plan à l'espace des images. Pour ce faire, j'ai construit un réseau de neurone. J'ai pris 80% des images utilisées dans l'ISOMAP comme base d'entraînement et les 20% restant comme base de test. L'entrée du réseau est donc un point (résultat de l'ISOMAP) et la sortie est l'image c'est à

1. https ://www.kaggle.com/c/painter-by-numbers/data

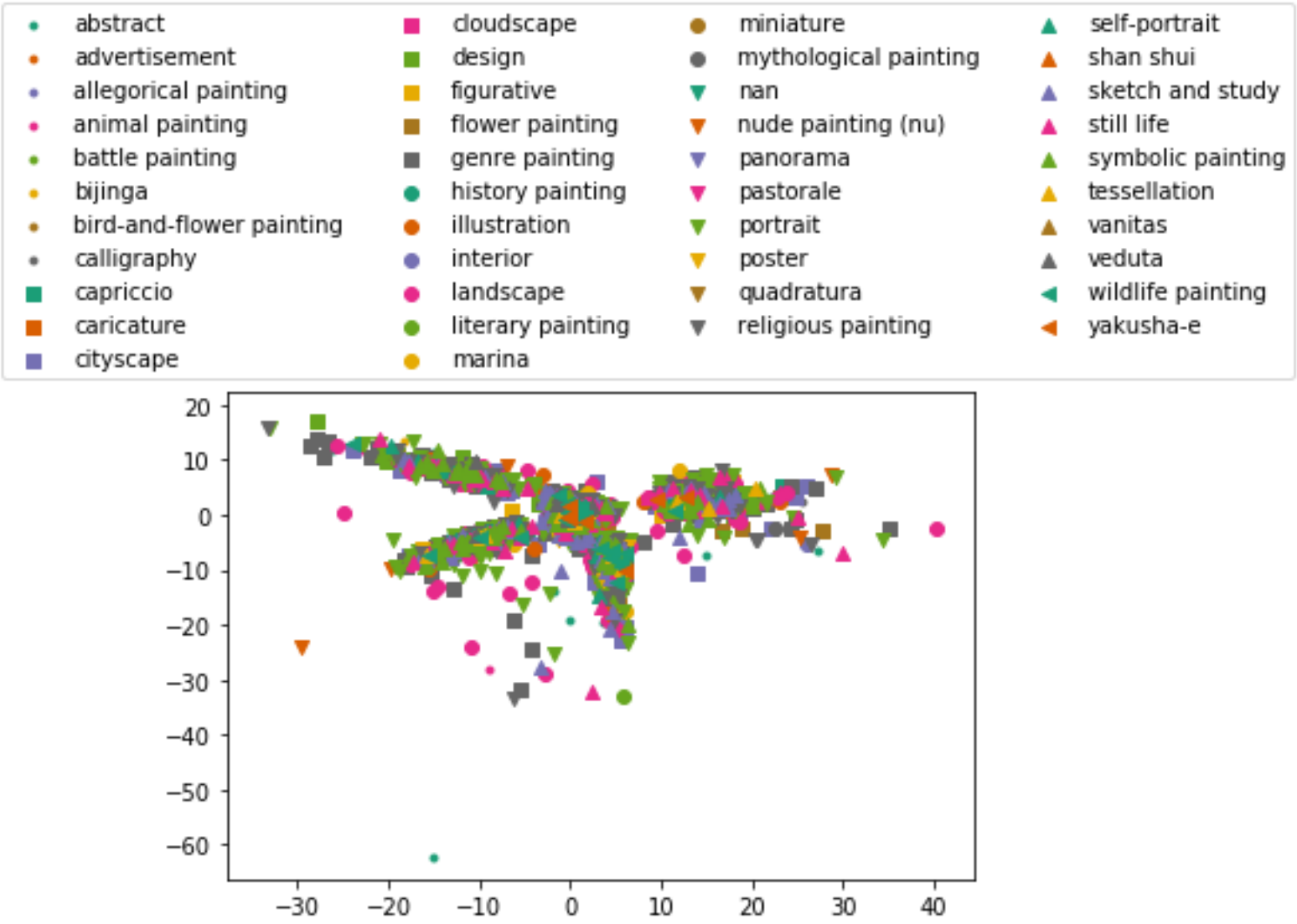

FIGURE 1 – Représentation des images de peinture dans le plan par ISOMAP

dire une matrice 50×50 représentant les pixels.

Puisque les images fournies dans l'algorithme ISOMAP avaient été transformés par la matrice L (équation 2), la sortie doit être de la même forme qu'une image transformée par L. Ainsi, j'ai mis comme la fonction tanh comme fonction d'activation à la sortie du réseau de neurone ce qui donne des coefficients dans $[-1, 1]$ puis je transforme ce nombre de sorte à ce qu'il soit dans l'intervalle de valeurs du pixel correspondant dans la base d'images transformées par L. Concrètement, pour p_i, le i-ème pixel, on note min_i et max_i les valeurs minimales et maximales de ce pixel dans la base de donnée transformée par L. Lorsque la sortie nous fournis une prédiction $\hat{p}_i \in [-1, 1]$ on applique la transformation $(\hat{p}_i(max_i + min_i) + max_i - min_i)/2$ de sorte à revenir dans l'intervalle $[min_i, max_i]$. Ainsi, on peut utiliser la distance de Mahalanobis dans la fonction de perte du réseau de neurone en mesurant la distance euclidienne entre la sortie et l'image correspondante dans la base transformée par L.
Si on veut revenir à une image non transformée par L, il suffit de multiplier par L^{-1}, i.e. ΣL (Σ et L ayant déjà été calculés au préalable). Il est toutefois possible que les valeurs des pixels finaux ne soient pas entre 0 et 255 puisque la prédiction du réseau de neurone n'est pas vraiment une image qui a été transformée par L.

Le réseau de neurone est très élémentaire, il contient une couche intermédiaire. J'ai procédé par validation croisée (*cross-validation*) pour choisir les paramètres de pénalisation L_2 ainsi que le nombre de neurones. Malheureusement, les performances du réseau de neurones sont assez décevantes puisque les images de sorties sont toutes très similaires

3 Conclusion

Les résultats ne sont pas très probants. Il est apparemment difficile de représenter les similarités entre images. Plusieurs pistes d'amélioration sont toutefois possibles.
La distance de Mahalanobis n'est peut être pas la plus adaptée dans notre problème, on pourrait essayer d'autres distance. Par exemple la métrique de Wasserstein (*earth mover distance*) pourrait être pertinente pour la comparaison d'images. Les méthodes à noyau peuvent aussi s'avérer utile dans le traitement d'image. On pourrait aussi envisager d'autres méthodes de réduction de dimension. Par exemple, le *kernel PCA* permettrait de faire une réduction de dimension avec une grande liberté possible grâce à utilisation de noyau.

Par ailleurs, j'ai été limité par la vitesse de calcul de mon ordinateur. J'ai utilisé des images en échelle de gris avec $50 \times 50px$ ce qui est très réduit. En utilisant des images plus grandes avec éventuellement de la couleur, il serait peut-être plus facile de capter les similarités et les différences entre les images.

4 Annexe

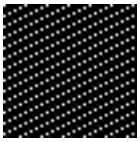

FIGURE 2 – Exemple de sortie du réseau de neurone en prenant un point correspondant à une image plongée par ISOMAP

FIGURE 3 – Image à l'origine du point correspondant

Références

[1] Przemysaw & Spurek and Jacek Tabor, *Optimal Rescaling and the Mahalanobis Distance*, May 11, 2013 Journal of Applied Statistics

[2] Kaggle : painter by numbers challenge https ://www.kaggle.com/c/painter-by-numbers/data

[3] Scikit-learn : Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011