Apprendre des distributions sur les variétés riemanniennes

Auto encodeur pour matrices SPD

Charlotte BOUCHERIE, Florian YGER, Thibault DE SURREL

LITIS

2025

Table des matières

Table des matières

Utilité matrices SPD

- Matrices de covariances
- Utilisés dans differentes applications de ML : vision par ordinateur, imagerie cérébrale, interface cerveau/ordinateur (EEG)

A Riemannian Network for SPD Matrix Learning

- Introduction d'une architecture de réseau qui conserve les propriétés des matrices définies positives pour le Deep Learning
- 3 couches différentes : BiMap, ReEig, LogEig
- On va se baser sur ces couches pour notre auto-encodeur

Différents travaux

DreamNet: A Deep Riemannian Manifold Network for SPD Matrix Learning

- Méthodologie pour créer des réseaux profonds

Riemannian Multinomial Logistics Regression for SPD Neural Networks

- Adaptation de la régression logistique pour les matrices SPD
- Nouvelle couche spécifique pour la classification

Solutions existantes en géométrie euclidienne

- Aplatissement de l'espace tangent
- Approximation de la distance : distance de Frobenius
  $L = ||A - B||_F = \sqrt{\sum_{i,j}(A_{ij} - B_{ij})^2}$

Problèmes

- Ne préserve pas la courbure de l'espace
- Résultats non-optimaux avec distance différente
- Effet de gonflement *(swelling effect)* : les déterminants de l'interpollation des matrices applattis

On veut donc prendre en compte la courbure et la non-linéarité de l'espace des matrices SPD

Géométrie riemannienne

Métrique de Riemann *(AIRM)* : $\delta_R^2(X, Y) = ||\log(X^{-1/2} Y X^{-1/2})||_F^2$

- Permet de mesurer la similarité entre deux matrices SPD tout en respectant la structure
- On l'utilisera dans notre AE dans le modèle, dans le coût et dans la confiance.

Ce qu'on fait

- Auto-encodeur pour matrices SPD
- Couche pour faire les opérations inverses de l'auto encodeur
- Comparaison de différents modèles
- Incidence de la distance pour l'erreur de reconstitution

Table des matières

Principes de base de l'auto-encodeur

- Réseau de neurones
- Apprentissage non supervisé : mesure de l'erreur de reconstitution
- Réduction de dimensions
- Apprends les paternes sous-jacents
- Utilisés pour les modèles génératifs

$$\phi : \mathcal{X} \to \mathcal{F} \text{ , l'encodeur}$$
$$\psi : \mathcal{F} \to \mathcal{X} \text{ , le décodeur}$$
$$\phi, \psi = \underset{\phi,\psi}{\arg\min} ||X - (\psi \circ \phi)X||^2$$

Perte de reconstitution

- Pour chaque matrice, on calcule la distance riemannienne avec sa reconstitution.

$$\phi : \mathcal{X} \to \mathcal{F}$$
$$\psi : \mathcal{F} \to \mathcal{X}$$
$$\phi, \psi = \arg\min_{\phi,\psi} \delta_R^2(X, \psi(\phi(X))) = \arg\min_{\phi,\psi} ||\log(X^{-1/2}\psi(\phi(X))X^{-1/2})||_F^2$$

Confiance

- Pour chaque matrice, on prend ses k matrices les plus proches dans l'espace de sortie et ses matrices les plus proches dans l'espace d'arrivée.
- La distance est la même utilisée que pour calculer notre fonction de coût.
- On pénalise proportionellement au rang différent dans l'espace d'entrée.
- On ne pénalise pas les rapprochements.

$$T(k) = 1 - \frac{2}{nk(2n-3k-1)} \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i^k} \max(0, (r(i,j) - k))$$

Précision

- On utilise la MDM (Minimum Distance to Mean) pour connaître la précision avant la reconstitution de nos matrices.
- Pour chaque classe, un centroïde est estimé selon notre distance.
- On compare la précision initiale avec la précision finale.

Couche BiMap

- La fonction de cette couche est de générer des matrices SPD plus compactes et plus discrimantes.
- On a donc une couche qui effectue une application bilinéaire f_b pour transformer les matrices initiales en nouvelles matrices de dimension inférieure.

$$X_k = f_b^{(k)}(X_{k-1}; W_k) = W_k X_{k-1} W_k^T$$

W_k est de rang plein pour garantir que X_k reste SPD.

Paramètres dans le réseau

Nombre de filtres/canaux d'entrée *hi*, nombre de filtres/canaux de sorties *ho*, taille de la matrice d'entrée *ni*, taille de la matrice de sortie *no*

Couche ReEig

- La fonction de cette couche est d'améliorer les performances discriminantes en introduisant une non linéarité, de la même manière que ReLU.
- On introduit donc une fonction non-linéaire f_r qui corrige les matrices en posant un seuil aux valeurs propres faibles.

$$X_k = f_r^{(k)}(X_{k-1}) = U_{k-1} \max(\epsilon I, \Sigma_{k-1}) U_{k-1}^T$$

Couche LogEig/ExpEig

LogEig

La fonction de cette couche est de pouvoir appliquer la géométrie de Riemann au matrice sortante.

$$X_k = f_l^{(k)}(X_{k-1}) = \log(X_{k-1}) = U_{k-1} \log(\Sigma_{k-1}) U_{k-1}^T$$

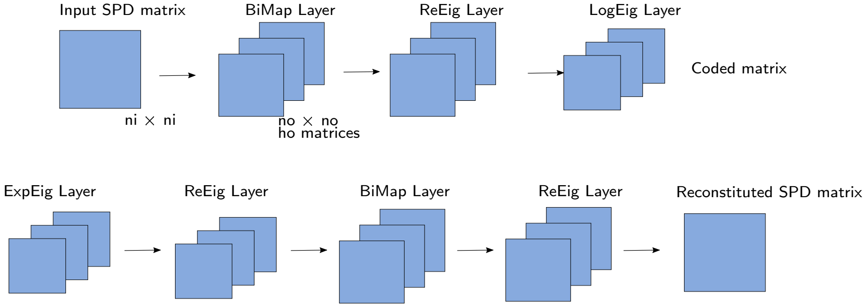
ExpEig

La fonction de cette couche est d'appliquer la fonction inverse de la couche LogEig

$$X_k = f_e^{(k)}(X_{k-1}) = \exp(X_{k-1})$$

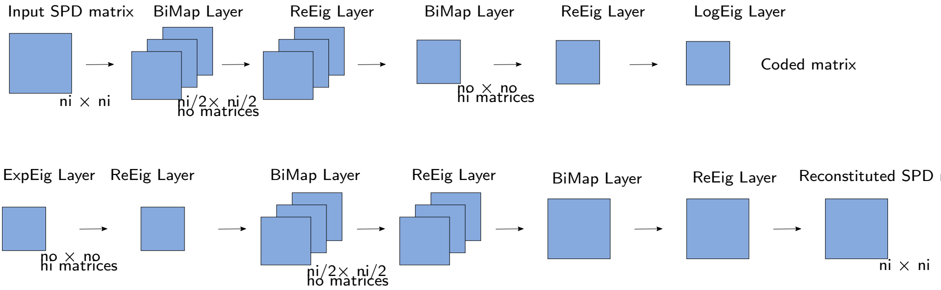
Modèle : une couche

- On a une seule couche BiMap pour l'encodeur qui va de $ni \rightarrow no$ et de $hi \rightarrow ho$.
- On regarde l'influence de la dimension de sortie et de la couche de sortie.
- Le décodeur fait l'opération inverse.



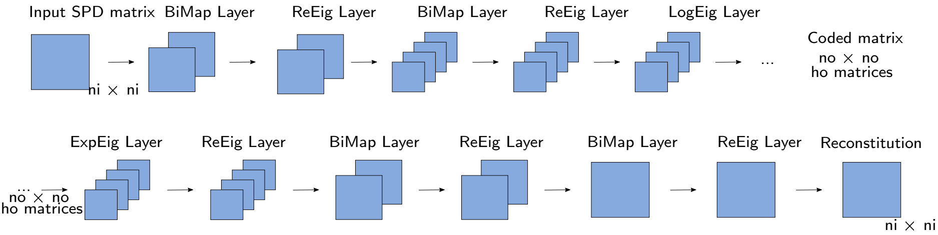
Modèle : filtres en entonnoir

- On a deux couches BiMap.
  - $ni \to ni/2$ et de $hi \to ho$.
  - $ni/2 \to no$ et de $ho \to hi$.
- On regarde l'influence du nombre de filtres intermediaires et de la dimension de sortie.
- Le décodeur fait l'opération inverse.



Modèle : plusieurs couches régulières

- Nombre de couches BiMap fixée en paramètres.
- Canaux et tailles de matrices intermédiare en fonction d'un nombre de couches fixés.
- Le décodeur fait l'opération inverse.



Modèle : taille de couches réduite de moitié

- Nombre de couches BiMap et de filtres dans les couches dépend de *ni* et *no*.
- Taille de matrice divisé par deux , nombre de filtre multiplié par deux à chaque couche.
- Le décodeur fait l'opération inverse.

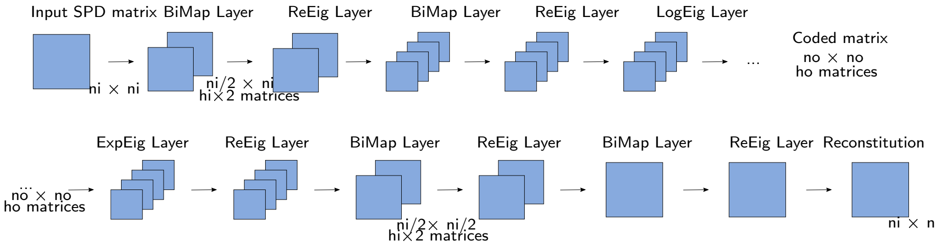


Table des matières

Données synthétiques

Données BCI

Ajout de bruits

Bruit gaussien

Bruit poivre et sel

Bruit de masquage

Résumé

- On prédit moins bien les données après le passage dans l'auto-encodeur
- On conserve les voisinages
- On compresse avec perte