UE : Réduction de la dimension
Projet proposé par : L. Labiod et M. Nadif **MLDS 2020**

Apprentissage Profond pour la Réduction de Dimension

1 Contexte du sujet et objectif

Apprendre des données massives est une tâche très difficile. La réduction de la dimensionnalité est une thématique bien étudiée dans la littérature et elle utilisée pour apprendre une nouvelle représentation des données à partir de données originales. Malheureusement, ces méthodes classiques de réduction de dimension ne parviennent pas à apprendre avec précision une nouvelle représentation de données lorsque les données sont volumineuses. Ces méthodes souffrent de la malédiction de la dimensionnalité, des données fortement corrélées, données clairsemées, bruyantes et hétérogènes.

Ces dernières années, certains travaux se sont concentrés sur la combinaison de l'apprentissage profond et les technique de réduction de dimension pour apprendre une représentation plus précise des données. Habituellement, ces méthodes sont combinées séparément avec des des objectifs différents, ce qui limite leurs capacités d'apprentissage. Pour les données volumineuses, ces méthodes qui optimisent différents critères sont souvent utilisées séparément; un autoencodeur (AE) suivi d'une réduction de dimension . Cette approche présente un inconvénient majeur, à savoir le manque de connexion entre les deux tâches et affecte donc grandement la qualité de la représentation finale.

Un autoencodeur est un réseau neuronal artificiel utilisé pour l'apprentissage non supervisé d'un codage efficace. L'objectif d'un autoencodeur est d'apprendre une représentation (codage) d'un ensemble de données, généralement dans le but de réduire la dimensionnalité. Si des activations linéaires sont utilisées, ou seulement une seule couche cachée sigmoïde, la solution optimale d'un autoencodeur est fortement liée à l'analyse en composantes principales (ACP). Avec des contraintes de dimensionnalité et de parcimonie appropriées, les autoencodeurs peuvent apprendre des projections de données plus intéressantes que l'ACP ou d'autres techniques de base. Les auto-encodeurs, par contre, ne sont pas linéaires par nature et peuvent apprendre des relations plus complexes entre des unités visibles et cachées. De plus, ils peuvent être empilés, ce qui les rend encore plus puissants. Récemment, l'Autoencoder et ses variantes combinées de manière séquentielle à des méthodes de réduction de dimension pour des fins de visualisation ou de clustering ont démontré leurs capacités à extraire des caractéristiques pertinentes et en fournissant une réduction de dimension intéressante.

L'objectif de ce tp est de contribuer au développement d'un cadre unifié et flexible pour les approches simultanées qui combinent l'apprentissage profond via une architecture Autoencoder et les techniques de réduction de dimension telles que LLE (Locally Linear Embedding), ISOMAP, et EIGENMAP, etc.

2 Travail à effectuer

Pour remédier à l'inconvénient des méthodes traditionnelles de réduction de dimension, nous proposons une nouvelle fonction objectif régularisée. La méthode proposée, peut être vue comme une procédure recherchant simultanément une nouvelle représentation des données contenant le maximum d'informations (en utilisant un Deep AE), et un graphe de similarité caractérisant au mieux la proximité entre le points (en utilisant LLE). cette methode consiste dans l'optimisation du problème suivant :

$$\min_{\theta_1,\theta_2,\mathbf{S}} \|\mathbf{X} - g_{\theta_2}(f_{\theta_1}(\mathbf{X}))\|^2 + \lambda \|f_{\theta_1}(\mathbf{X}) - \mathbf{S} f_{\theta_1}(\mathbf{X})\|^2 \tag{1}$$

θ_1, θ_2 sont respectivement les paramètres des blocs encodeur et decodeur de l'AE. $\mathbf{S}$ est la matrice des poids calculés avec LLE. Cette fonction objectif se décompose en deux termes, le premier correspond à la fonction objectif d'un Autoencodeur et le second teme correspond à la fonction objectif de la méthode LLE (voir

le papier de Sam Roweis et al. pour plus de détails sur l'algorithme LLE). La mise à jour de $\mathbf{S}$ se fait en optimisant le second terme, les poids optimaux sont ceux qui minimisent la fonction de coût :

$$\mathcal{E}(\mathbf{S}) = \sum_i ||f_{\theta_1}(x_i) - \sum_j s_{ij} f_{\theta_1}(x_j)||^2$$

sous la contrainte: la somme de chaque ligne de la matrice de pondération $\mathbf{S}$ est égale à 1: $\sum_j s_{ij} = 1$.

Nous construisons $\mathbf{S}$ en utilisant la même stratégie que LLE. Nous supposons que chaque individu avec ses k voisins les plus proches se trouve sur une variété localement linéaire et qu'il peut être reconstruit par une combinaison linéaire de ses k plus proches voisins. Ainsi, $\mathbf{S}$ est une matrice avec s_{ij} indique la contribution du point j dans la reconstruction de i; par conséquent il est égal à 0 si i et j ne sont pas voisins.

Dans ce projet, il s'agit d'implémenter un algorithme itératif simple, optimisant une fonction objective appropriée. Cet algorithme s'appuie sur deux étapes de mise à jour selon le schéma décrit dans le pseudo-code ci-après. Le projet comporte les étapes suivantes :

Input: data matrix $\mathbf{X}$;
repeat
(a) - Update $\mathbf{\Theta_1}$ and $\mathbf{\Theta_2}$ using Deep AE
(b) - Update $\mathbf{S}$ using the same strategy as in LLE
until convergence
Calculate:
- $\mathbf{M} = (\mathbf{I} - \mathbf{S})^\top (\mathbf{I} - \mathbf{S})$
- $\mathbf{B} = eigs(\mathbf{M})$
Output:
$\mathbf{S}$: weights matrix, $f_{\theta_1}(\mathbf{X})$: Encoding matrix, and $\mathbf{B}$: embedding matrix

---

1. Se familiariser avec les methodes de réduction de dimension (linéaire, non linéaire, non linéaire avec apprentissage profond) sous python.

2. Tester les méthodes PCA, MDS, LLE, Eigenmaps, ISOMAP et Deep AE sur les jeux de données synthétiques FCPS.

3. Implementer l'algorithme d'apprentissage profond (présenté plus haut) qui combine les Autoencodeurs et LLE.

4. Evaluer cet algorithme en termes des performances de clustering et de visualization en utilisant les nouvelles représentations $\mathbf{B}$ et $f_{\theta_1}(\mathbf{X})$.

5. Comparer cet algorithme avec au moins 3 méthodes de votre choix sur les tables données ci-dessous Table 1.

Table 1: Tables de données Images.

Tables de Données	**Caracteristiques**				
	Type	**#samples**	**#features**	**#classes**	**Sparsity(%)**
Coil20	Image	1440	1024	20	34.38
Coil100	Image	7200	1024	20	0
ORL	Image	400	1024	40	0
Yale	Image	165	1024	15	30.54
USPS	Image	9298	256	10	0
PIE	Image	2856	1024	68	8.53
MNIST	Image	70000	784	10	80.85

3 Références

- Sam Roweis and Lawrence Saul. Nonlinear dimensionality reduction by locally linear embedding. Science, v.290 no.5500 , Dec.22, 2000.
  https://cs.nyu.edu/~roweis/lle/

- G. E. Hinton nd R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks
  https://www.cs.toronto.edu/~hinton/science.pdf

- M Belkin, P Niyogi . Laplacian eigenmaps for dimensionality reduction and data representation - Neural computation, 2003 - MIT Press
  www2.imm.dtu.dk/projects/manifold/Papers/Laplacian.pdf