Emmanuel CHRISTOPHE **Julien MORIN**
Adrien PILLET **Bastien MULLER**

Réduire la dimension des données grâce aux réseaux de neurones.

- **Introduction :**

Diminuer la dimension des données facilite la classification (Reconnaissance de formes), la visualisation (on peut facilement représenter la répartition des données graphiquement lorsqu'elles sont projetées dans un espace de faible dimension) et enfin le stockage de données de hautes dimensions.

La méthode classique pour réduire la dimension des données est l'analyse en composantes principales (ACP). L'ACP consiste à trouver les directions de plus grande variance dans le jeu de données, à partir desquelles on construit un nouveau repère dans lequel on projette les données.

Cet article présente une généralisation non linéaire de l'ACP utilisant un réseau de neurones multicouche "encodeur" (qui réduit la dimension des données) en série avec un réseau de neurones "décodeur" similaire (pour récupérer les données initiales). L'ensemble est appelé "auto-encodeur".

- **Principe de l'auto-encodeur :**

Il s'agit d'une méthode **pseudo-réversible** (ce qui permet le stockage de données sous forme de dimensionnalité moindre). On entraîne conjointement l'encodeur et le décodeur afin de minimiser les différences entre l'image reconstruite et l'originale.

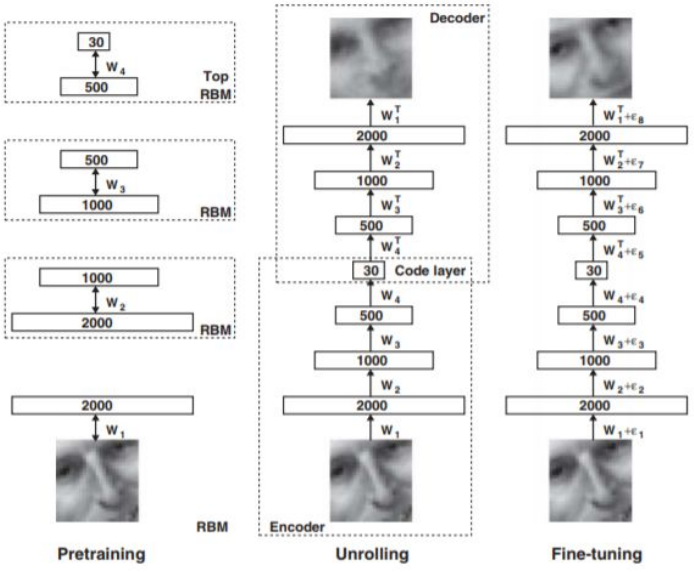

Figure-1 *Schéma du fonctionnement de l'auto-encodeur*

Les liaisons entre les neurones de deux couches successives sont pondérées. Afin d'ajuster ces poids et d'obtenir un modèle performant on utilise la rétropropagation du gradient. Pour que cette méthode donne de bons résultats il faut que les poids choisis à l'initialisation soient proches d'une solution convenable. Dans le cas contraire, les solutions trouvées correspondent à des minima locaux et ne sont pas optimales. L'algorithme utilisé ici travaille sur une couche de caractéristiques à la fois.

L'article décrit donc une méthode efficace pour une bonne initialisation des poids, ce qui permet à des réseaux auto-encodeurs profonds d'apprendre des codes de dimension peu élevée qui fonctionnent mieux que l'ACP pour réduire la dimensionnalité des données.

- **Méthode pour initialiser les poids :**

Il y a donc un problème : il est difficile d'optimiser les poids initiaux pour des auto-encodeurs non-linéaires qui possèdent plusieurs couches cachées. La seule solution pour avoir une descente du gradient qui fonctionne est d'avoir de **bons poids initiaux**, ce qui nécessite un algorithme dit de **"pré-training",** qui apprend une couche après l'autre.

On présente en entrée du système une image représentée comme un vecteur binaire. Chaque paire de couches de neurones successives peut être modélisée par une **machine de Boltzmann restreinte (RBM)**, c'est à dire un système à deux couches mettant en relation des neurones cachés et des neurones visibles. Dans notre cas, les neurones cachés représentent les caractéristiques (features) et les neurones visibles représentent les pixels (ce qu'il y a en entrée) . On définit alors une énergie d'activation pour une Machine de Boltzmann Restreinte de la manière suivante:

$$E = -\left(\sum_{i,j} w_{ij}\, x_i\, h_j + \sum_i b_i\, x_i + \sum_j c_j h_j\right)$$

Avec:

- $\mathbf{w_{ij}}$ entre le neurone i et le neurone j ;
- $\mathbf{x_i}$ est l'état $\in \{0.1\}$, du neurone visible i ;
- $\mathbf{h_j}$ est l'état du neurone caché j;
- $\mathbf{b_i}$ et $\mathbf{c_j}$ sont respectivement les biais des neurones $\mathbf{x_i}$ et $\mathbf{h_j}$.

Ainsi, comme on l'observe sur la **Figure-1**, notre auto-encodeur est un empilement de RBM et il est possible d'approcher les poids w_{ij} optimaux en trouvant le minimum de la fonction Énergie de chacune des RBM constituant notre auto encodeur. L'algorithme assigne à chaque image possible une énergie, qu'il est possible d'augmenter (par ajustement des poids et des biais) afin de ne pas choisir des images fabriquées ou distordues. On effectue plusieurs opérations afin de permettre un apprentissage sur les poids et les biais qui fonctionne bien (même s'il ne suit pas exactement la méthode du gradient).

- **Ajout de couches supplémentaires de caractéristiques binaires :**

Après avoir entraîné une première couche, on utilise les données pour raffiner le modèle et apprendre une deuxième couche. On peut ensuite répéter le processus autant de fois que souhaité, chaque couche augmentant les performances, en faisant attention d'initialiser les poids correctement et de ne pas diminuer le nombre de caractéristiques entre chaque couche. Il s'agit d'une manière très efficace pour le pré-entraînement des poids.

- **"Déploiement" du modèle :**

Après avoir pré-entraîné les multiples couches détectrices de caractéristiques, on "déploie" le modèle afin de créer les encodeurs/décodeurs initialisés à des poids identiques. Le raffinement global ne s'effectue alors plus de manière stochastique mais de manière déterministe à l'aide de calculs de probabilités, pour un raffinement des poids optimal au sens de la reconstruction.

- **Résultats :**

Des auto-encodeurs construits selon le modèle exposé ici ont été testés et comparés à des modèles basés sur l'analyse en composantes principales pour la reconstruction d'images et la classification de données, notamment sur des bases de données largement utilisées dans la littérature pour évaluer la performance des algorithmes de classification telles que la base de données MNIST constituée de chiffres manuscrits. Dans chaque cas les réseaux de neurones ont montré des performances nettement meilleure que l'ACP.

- **Conclusion :**

Maintenant que l'on peut initialiser les poids de l'auto-encodeur de manière convenable, on possède un réseau de neurone bien plus efficace que l'analyse en composante principale pour la réduction non linéaire de la dimension des données. Cette méthode peut de plus être appliquée à de très grandes bases de données étant donné que le "prétraining" et le raffinement du modèle grandissent linéairement en temps et en espace avec le nombre d'entrainements, et que l'on possède aujourd'hui des puissances de calcul suffisantes.