Introduction au domaine de recherche : Outils d'analyse topologique de données pour la réduction de dimension

Baptiste Collet

Septembre 2023

Contents

1 Motivations

Ces dernières années, les grands progrès en analyse de données sont surtout venus du deeplearning avec de nombreuses nouvelles architectures de réseaux de neurones adaptés à des données vectorielles.

Cependant, il est également nécessaire de prendre en compte la forme générale des nuages de points.

A l'heure du "big data", les données arrivent en grand nombre mais surtout en grande dimension, elles sont ainsi à la fois difficilement visualisables par l'humain et exploitables par des algorithmes ayant une forte complexité en la dimension.

Pour adresser ces deux problèmes, la réduction de dimension est souhaitable mais la dimension visée n'est pas la même dans les deux cas, 2 ou 3 pour de la visualisation, beaucoup plus pour de l'extraction de features.

Le plus souvent, l'objectif recherché lors d'une réduction de dimension est que les distances soient préservées, c'est à dire que deux points proches à la base le soient encore à l'arrivée et de même pour deux points éloignés.

La considération de distorsion des distances n'est pas suffisante. En effet on aurait envie que des points regroupés au début (composante connexe) le restent après réduction de dimension. De même on aurait envie que les trous et cavités soient conservés. Il s'agit en fait exactement des notions d'homologie de dimension 0,1 et 2 expliquées ci-après.

2 Eléments sur les réseaux de neurones

2.1 Le cadre de l'apprentissage supervisé

Un *réseau de neurones* est un outil permettant d'approximer des fonctions difficilement descriptibles d'un espace $\mathcal{X}$ dans un espace $\mathcal{Y}$, on notera g la fonction associée au réseau de neurones.

$\mathcal{X}$ est presque toujours un espace vectoriel, pour $\mathcal{Y}$ on parle de problème de *régression* quand c'est un espace vectoriel et de *classification* quand c'est un espace discret.

Dans le cas d'une régression g va directement vers $\mathcal{Y}$. Dans celui d'une classification, g renvoie une distribution de probabilité sur $\mathcal{Y}$.

Pour quantifier à quel point g est une bonne approximation de la fonction désirée on utilise une fonction de comparaison appelée fonction de *perte*, ou *loss*, $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$, par exemple la norme L_2 pour les régressions ou l'entropie croisée pour la classification.

On dispose généralement, d'un jeu de données que l'on découpe en trois parties (pour une raison expliquée un peu plus loin) : l'ensemble E_{train} d'entraînement, de validation E_{val} et de test E_{test}.

Un réseau de neurones dispose d'un ensemble Θ de valeurs ajustables appelées *paramètres*.

Pour trouver quelles valeurs des paramètres fournissent la fonction voulue on initialise leurs valeurs aléatoirement puis on effectue une *descente de gradient*. Soit $\hat{R}_\Theta = (\sum_{(x,y)\in E_{train}} \mathcal{L}(g_\Theta(x), y)/|E_{train}|$, le risque empirique.

On calcule le gradient ∇ de $\hat{R}_\Theta$ selon Θ et on remplace Θ par $\Theta - \alpha\nabla$ (où α estun coefficient appelé *learning rate*) pour minimiser l'erreur.

En pratique, on utilise des méthodes un peu plus complexes avec une mémoire des gradients précédents et une pénalisation de la valeur des coefficients. Les méthodes les plus connues sont Adam et RMSprop.

De fait ce que l'on veut vraiment optimiser c'est la qualité des prédictions futures pour des données pour lesquelles on connaîtra x mais pas y, une formalisation classique est de dire que les données suivent une distribution aléatoire sur $\mathcal{X} \times \mathcal{Y}$ et que l'on veut minimiser le *risque* $R_\Theta = E(\mathcal{L}(g_\Theta(X), Y))$, que l'on estime en calculant l'erreur sur E_{test}.

On dit d'un réseau, qui aurait une faible erreur d'entraînement mais une grosse erreur de test, qu'il a fait du *surapprentissage*, de l'*overfitting*. L'ensemble de validation sert pendant la phase de choix des paramètres d'optimisations pour vérifier qu'il n'y a pas d'overfitting.

2.2 Exemples d'architectures de réseaux de neurones

La forme de réseau de neurones la plus simple est le MLP (Multi-layer perceptron).
Un MLP à h couches cachées est la composition de $h+1$ produits matriciels entrecoupés d'applications d'une fonction non linéaire dîte d'activation.
Ainsi par exemple un réseau à deux couches cachées et de fonction d'activation tanh envoie un vecteur $u \in \mathbb{R}^{n_0}$ vers le vecteur $W_2(tanh(W_1 tanh(W_0 u))) \in \mathbb{R}^{n_3}$ (tanh d'un vecteur signifie tanh appliqué à chaque coordonnée).
Les paramètres ajustables sont ceux des matrices W_0, W_1, W_2.

De nombreuses autres architectures ont été développées. Par exemple, on voit que si l'on voulait prédire quelques valeurs associées à une image, il faudrait au moins autant de paramètres que de pixels, or il faut un nombre de paires de données comparables au nombre de paramètres, ceci serait prohibitif. Pour palier à cela on peut prendre en compte les propriétés de symétrie de la fonction et faire en sorte que le réseau ait les mêmes symétries. C'est l'idée derrière les CNN, dont le fameux AlexNet de 2012, qui est considéré comme le point de départ de la dernière révolution du deeplearning.

Une autre architecture est l'autoencodeur. C'est une composition de deux réseaux de neurones (souvent des MLP), l'encodeur de $\Phi : \mathbb{R}^n \to \mathbb{R}^l$ et d'un décodeur $\Psi : \mathbb{R}^l \to \mathbb{R}^n$. L'espace intermédiaire, $\mathbb{R}^l$, généralement beaucoup plus petit est appelé espace latent, on le note parfois Z.

L'objectif en général est que Ψ soit "autant que possible" l'inverse de Φ. Plus formellement on souhaite en général que $\sum_{x \in X_{train}} ||\Psi(\Phi(x)) - x||$ soit minimal. Cependant l'utiliser comme fonction de perte en tant que tel ne marche pas très bien. C'est pourquoi de nombreuses alternatives ont été proposées dont les VAE (Variational AutoEncoder). Souvent on souhaite de plus certaines propriétés sur les valeurs dans l'espace latent d'où la variété d'autoencodeurs existants.

3 Eléments d'analyse topologique de données

3.1 Notion de complexes simpliciaux et homologie

Pour décrire à un ordinateur (donc de manière discrète), un objet continu comme une variété, on utilise la notion de complexe simplicial qui généralise la notion 2d de triangulation.

Définition 1. *Un d-simplexe σ est l'enveloppe convexe de $d+1$ points, appelés sommets, affinément indépendants d'un espace ambiant $\mathbb{R}^n$.*

Une face τ d'un simplexe σ est un sous-simplexe construit à partir d'un sous-ensemble des $d+1$ sommets de σ, on note $\tau \leq \sigma$

Lorsque sa dimension est i on note τ_i

Définition 2. *Un complexe simplicial $\mathcal{T}$ est un ensemble de simplexes σ, qui vérifie les deux propriétés suivantes :*

- *Si σ' est une face d'un simplexe σ de $\mathcal{T}$ alors σ' appartient aussi à la triangulation*
- *L'intersection de deux simplexes du complexe est soit nulle soit une face commune aux deux simplexes*

Définition 3. *Une p-chaîne d'un complexe simplicial $\mathcal{T}$ est une somme formelle de p-simplexe de $\mathcal{T}$.*

L'ensemble des p-chaînes d'un complexe $\mathcal{T}$ forme un groupe, noté $C_p(\mathcal{T})$ (isomorphe à une puissance de $Z/2Z$) où l'opération de groupe consiste à prendre le Xor des éléments des deux chaînes.

On peut associer à chaque p-simplexe σ son bord $\partial\sigma$: l'ensemble de ses faces de dimension $p-1$. On peut faire de même avec les p-chaînes. On appelle cette opération l'opérateur de bord. Le bord du bord d'un simplexe est vide, en effet une face $\tau = \sigma\backslash\{a,b\}$ apparaît à la fois dans le bord de $\sigma\backslash\{a\}$ et de $\sigma\backslash\{b\}$, les deux contributions s'annulent. Puisque que ∂ appliqué deux fois envoie vers 0, c'est une dérivation et on peut donc considérer l'homologie associée et qui s'appelle l'homologie simpliciale.

Définition 4. *On appelle bord (respectivement cycles) les éléments de Im ∂ (respectivement Ker ∂), on note $B_p(\mathcal{T})$ (respectivement $Z_p(\mathcal{T})$) le groupe associé.*

Le groupe d'homologie est le quotient $H_p(\mathcal{T}) = Z_p(\mathcal{T})/B_p(\mathcal{T})$.

Il est isomorphe à un $(Z/2Z)^r$ et le rang r est appelé nombre de betti β_p

Ces nombres de betti ont un sens intuitif en petite dimension. Par exemple pour l'homologie zéro, on voit qu'une famille de générateurs est de prendre un sommet par composante connexe. β_0 est donc le nombre de composantes connexes. De même l'homologie 1 correspond aux "trous" et à la notion de genre, pour l'homologie 2 on parle plus souvent de cavités.

3.2 Homologie persistante et descripteur topologique

L'analyse topologique de données est un domaine relativement nouveau qui s'appuie sur des objets comme les diagrammes de persistence qui s'appuient eux-mêmes sur la notion de complexes simpliciaux.

Définition 5. *Une filtration est une famille croissante de complexes simpliciaux, dont on va considérer l'homologie à chaque étape.*

Exemple la filtration des sous-niveaux :
Soit $f : \mathcal{K} \to \mathbb{R}$ un champ scalaire défini sur les sommets du complexe.
Pour une valeur de filtration x, les sommets présents sont $f^{-1}(]-\infty; x])$.
Pour les simplexes de plus haute dimension, on a une liste des simplexes possibles et on les rajoute dès que tous leurs sommets sont présents.

Définition 6. *Une autre filtration est la filtration Vietoris-Rips, qui sera la seule utilisée dans la suite, et qui consiste à relier progressivement les sommets les plus proches.*
Tous les sommets sont présents de base dans le complexe.
Pour une valeur de filtration x, une arête AB est présente dans le complexe si et seulement si $d(A, b) \leq x$
Un simplexe de plus haute dimension est présent si et seulement si toutes ses arêtes sont dans le complexe.

Définition 7. *Une manière assez commune de représenter les propriétés topologiques d'un nuage de points est le diagramme de persistence dont l'axe des abscisses correspond à la valeur de filtration pour laquelle une certaine composante homologique apparaît et l'ordonnée sert à indiquer le moment où la composante disparaît.*

Il semble naturel de vouloir comparer les diagrammes de persistences avec une notion de distance telle que deux nuages de points semblables ont des diagrammes proches. Pour cela on s'inspire du transport optimal, on essaie d'apparier les points du premier diagramme avec des points proches du second diagramme, on s'autorise à associer certains points avec la diagonale (notamment pour compenser le déséquilibre du nombre de points qui n'est pas forcément le même dans les deux diagrammes).

Il y a deux distances principalement utilisées :

- La distance de Wasserstein qui prend la valeur minimale sur les appariements de la somme des distances.
- La distance de Bottleneck qui vaut la valeur minimale sur les appariements du maximum des distances.

4 Réduction de dimension

Les méthodes de réduction de dimension les plus courantes comme PCA, t-SNE, UMAP ou MDE essaient de préserver les distances entre les points, mais elle ne préservent pas la forme générale. Pour palier à ce manque une solution est d'utiliser de l'analyse topologique de données.

Ceci a notamment été fait dans les trois papiers suivants :

- [2] qui préserve parfaitement l'homologie persistante de dimension 0 au prix potentiellement d'une distorsion considérable des distances.
- [3] qui essaie de préserver simultanément les distances et l'homologie persistante de dimension 0, c'est à priori généralisable en dimension 1
- [1] présente parmi ses applications la réduction de dimension pour laquelle ils emploient une fonction de perte combinant une norme L^2, la norme de l'article précédent et la distance Wasserstein entre les diagrammes dans l'espace de base et l'espace latent.

L'homologie de dimension 0 correspond aux composantes connexes, or pour une filtration de Vietoris-Rips ce qui "supprime" des composantes connexes, c'est le moment où le rayon atteind la moitié de la longueur d'une arête de l'arbre couvrant de poids minimal (MST en anglais).

L'algorithme du premier papier s'inspire de l'algorithme de Kruskal.
On initialise avec n ensembles correspondants chacun à un sommet du nuage de départ
Pour chaque arête, e, du MST par ordre croissant de taille, on considère les composantes E_1 et E_2 qu'elle fusionne. On positionne E_1 et E_2 à une distance supérieure au diamètre de E_1 et au diamètre de E_2 en conservant les agencements internes à E_1 et E_2 et on mémorise la nouvelle composante ainsi créée.

Les deux papiers suivants utilisent un autoencodeur et l'analyse topologique de données intervient au niveau de la loss.
Pour l'avant-dernier papier, l'algorithme considère les arêtes du MST dans l'espace de base et dans l'espace latent et vérifie si elles ont des valeurs proches. Soit X le nuage de base et Z le nuage dans l'espace latent, si u, v correspondent aux indices des sommets d'une arête du MST de X on met $(d(X_u, X_v) - d(Z_u, Z_v))^2$ dans la loss.

Pour le dernier papier, afin de prendre en compte également la dimension 1, une distance de Wasserstein sur les diagrammes est également ajoutée à la loss.

References

[1] Mathieu Carriere et al. “Optimizing persistent homology based functions”. In: *International conference on machine learning*. PMLR. 2021, pp. 1294–1303.

[2] Harish Doraiswamy et al. “Topomap: A 0-dimensional homology preserving projection of high-dimensional data”. In: *IEEE Transactions on Visualization and Computer Graphics* 27.2 (2020), pp. 561–571.

[3] Michael Moor et al. “Topological autoencoders”. In: *International conference on machine learning*. PMLR. 2020, pp. 7045–7054.