STT3030 - Cours #12

Arthur Charpentier

Automne 2024

Apprentissage non-supervisé

Deux problèmes standards en apprentissage non-supervisé:

- La mise-en-grappe/le regroupement : Méthodes et techniques afin de créer des groupes de variables *similaires*
- La réduction de dimension: Méthodes et techniques pour projeter les données $\mathbf{X} \in \mathcal{X}$ sur $\mathbf{Z} \in \mathcal{Z}$ où la dimension de $\mathcal{Z}$, un *score* ou un *code*, est **beaucoup** plus petite que celle de $\mathcal{X}$.

Réduction de dimension

- Beaucoup de modèles supervisés sont moins performants et lents à optimiser lorsque p est trop grand. (exemple: modèles linéaires si $p > n$)
- Il peut parfois être difficile de visualiser des données $\mathbf{X} \in \mathcal{X}$ lorsqu'elles sont de très grande dimension.
- Dans la réduction de dimension on vise à apprendre une fonction $f : \mathcal{X} \subset \mathbb{R}^p \to \mathcal{Z} \subset \mathbb{R}^d$ où $d < p$.
- On veut souvent conserver le maximum d'information pour distinguer les observations malgré la réduction de dimension.
- On doit aussi penser à la forme de f car il peut aussi être pratique d'apprendre $g : \mathcal{Z} \to \mathcal{X}$ pour "reconstruire" $\mathbf{x}$ étant donné $\mathbf{z}$.

Réduction de dimension, cas linéaire

- Si $\mathbf{M}$ est une matrice $n \times n$, $\vec{\boldsymbol{v}}$ est vecteur propore pour $\mathbf{M}$ (associé à λ) si $\mathbf{M}\vec{\boldsymbol{v}} = \lambda\vec{\boldsymbol{v}}$
- Si $\mathbf{M}$ est diagonalisable, $\mathbf{M} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$ où $\boldsymbol{\Lambda} = \text{diag}(\boldsymbol{\lambda})$, et $\mathbf{V} = [\vec{\boldsymbol{v}}_1, \cdots, \vec{\boldsymbol{v}}_n]$.
- Si $\mathbf{M}$ est inversible, $\mathbf{M}^{-1} = \mathbf{V}\boldsymbol{\Lambda}^{-1}\mathbf{V}^\top$ où $\boldsymbol{\Lambda} = \text{diag}(\boldsymbol{\lambda})$, et $\mathbf{V} = [\vec{\boldsymbol{v}}_1, \cdots, \vec{\boldsymbol{v}}_n]$.
- Théorème de **Eckart–Young–Mirsky** (Eckart and Young (1936), Mirsky (1960)):

$$\mathbf{M}^\star \in \underset{\mathbf{H}:n\times n}{\text{argmin}}\{\|\mathbf{M} - \mathbf{H}\|_F \text{ telle que rank}(\mathbf{H}) \leq k\}$$

Si $\mathbf{M} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top = \begin{bmatrix}\mathbf{V}_r & \mathbf{V}_{n-r}\end{bmatrix}\begin{bmatrix}\boldsymbol{\Lambda}_r & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Lambda}_{n-r}\end{bmatrix}\begin{bmatrix}\mathbf{V}_r & \mathbf{V}_{n-r}\end{bmatrix}^\top$, $\mathbf{M}^\star = \mathbf{V}_r\boldsymbol{\Lambda}_r\mathbf{V}_r^\top$

(qui sera unique si $\lambda_r > \lambda_{r+1}$)

- L'erreur d'approximation vérifie $\|\mathbf{M} - \mathbf{M}^\star\|_F = \sqrt{\lambda_{r+1}^2 + \cdots + \lambda_n^2}$

Réduction de dimension, cas linéaire

```
> X = decathlon[,1:5]
> (S = var(X))
                  100m  Long.jump    Shot.put    High.jump        400m
100m       0.069181098 -0.0498225 -0.07730085 -0.005761341  0.15785018
Long.jump -0.049822500  0.1001100  0.04781500  0.008292500 -0.21972500
Shot.put  -0.077300854  0.0478150  0.67968122  0.035875488 -0.13164098
High.jump -0.005761341  0.0082925  0.03587549  0.007912195 -0.01928439
400m       0.157850183 -0.2197250 -0.13164098 -0.019284390  1.33044878
> eigen(S)
$values
[1] 1.41859896 0.65892825 0.07356046 0.03064542 0.00560019
$vectors
            [,1]         [,2]        [,3]         [,4]          [,5]
[1,]  0.12953964  0.072064780 -0.52908351  0.834781670 -0.0351617763
[2,] -0.17153505 -0.007260737  0.81844653  0.542662909 -0.0786463937
[3,] -0.19601616 -0.973494784 -0.09090409  0.054681113 -0.0513029715
[4,] -0.01959857 -0.048110438  0.04120183  0.075216843  0.9949603598
[5,]  0.95655045 -0.211535436  0.20063592 -0.002988949  0.0005308868
```

Réduction de dimension, cas linéaire

```
> V = eigen(S)$vectors
> L = eigen(S)$values
> S2 = V[,1:2] %*% diag(L[1:2]) %*% t(V[,1:2])
> S2
             [,1]        [,2]        [,3]         [,4]        [,5]
[1,]  0.027226858 -0.03186688 -0.08224777 -0.005886076  0.16573543
[2,] -0.031866883  0.04177598  0.05235596  0.004999280 -0.23175441
[3,] -0.082247774  0.05235596  0.67896698  0.036310818 -0.13029417
[4,] -0.005886076  0.00499928  0.03631082  0.002070054 -0.01988855
[5,]  0.165735427 -0.23175441 -0.13029417 -0.019888551  1.32748735
> eigen(S2)
eigen() decomposition
$values
[1]  1.418599e+00  6.589283e-01  8.416465e-17  1.213437e-18 -4.970307e-18
```

i.e. $\lambda_1^\star = \lambda_1$ et $\lambda_2^\star = \lambda_2$ (puis $\lambda_j^\star = 0$ ensuite car $\mathbf{M}^\star$ est de rang 2)

Réduction de dimension, cas linéaire

et $\vec{\boldsymbol{v}}_1^\star = \vec{\boldsymbol{v}}_1$ et $\vec{\boldsymbol{v}}_2^\star = \vec{\boldsymbol{v}}_2$

```
$vectors
            [,1]         [,2]        [,3]         [,4]        [,5]
[1,]  0.12953964  0.072064780  0.18992045  0.000000000  0.97054437
[2,] -0.17153505 -0.007260737 -0.95876891  0.082115571  0.21105027
[3,] -0.19601616 -0.973494784  0.06538091  0.047692650  0.08565232
[4,] -0.01959857 -0.048110438 -0.07672140 -0.995472557  0.02120132
[5,]  0.95655045 -0.211535436 -0.18582670  0.004102648 -0.07560154

> sum( (S-S2)^2 )
[1] 0.006381646
> sum(L[3:5]^2)
[1] 0.006381646
```

i.e. l'erreur d'approximation est $\|\mathbf{M}-\mathbf{M}^\star\|_F = \sqrt{\lambda_{r+1}^2+\cdots+\lambda_n^2}$

Pas de mesure de succès

- Un grand défi de l'apprentissage non-supervisé est le manque d'une bonne mesure de succès.
- Comme il n'y a pas de réponse, nous ne pouvons pas utiliser l'erreur de prédiction/classification.
- Il n'y a pas de vrais groupes ni de vraie représentation de petite dimension non plus.
- Un problème est donc qu'il est difficile d'établir si nous avons raisonnablement accompli notre tâche d'apprentissage non-supervisé.
- Le succès du modèle est plus subjectif.
- L'évaluation du succès d'une technique d'apprentissage non-supervisé est un problème encore d'actualité.

Encodage et décodage

- En théorie de l'information, on parle d'une fonction de compression une fonction $f : \mathcal{X} \to \mathcal{Z}$ si $||\mathcal{Z}|| < ||\mathcal{X}||$ puisque l'on compresse des données $\mathbf{x} \in \mathcal{X}$ vers $\mathbf{z} \in \mathcal{Z}$ de plus petite dimension.
- On définit $g : \mathcal{Z} \to \mathcal{X}$ une fonction de décompression ou de reconstruction. On prend des données compressées $\mathbf{z}$ et on reconstruit des données de pleine dimension $\mathbf{x}$.
- Il est commun d'appeler la compression $\mathbf{z}$, le code. Dans ce contexte, dit que f est une fonction d'encodage et g une fonction de décodage.

Encodage et décodage

- Un critère d'évaluation de nos fonctions d'encodage et de décodage est l'erreur de reconstruction.
- Soit $\tilde{\mathbf{x}} = g(f(\mathbf{x}))$ la reconstruction de $\mathbf{x}$ après avoir été encodé $f(\mathbf{x})$ puis décodé $g(f(\mathbf{x}))$ on désire avoir $\tilde{\mathbf{x}}$ le plus près de $\mathbf{x}$ possible.
- Un bon système d'encodage-décodage (appelé auto-encodeur), permet la compression des données (la projection vers un espace de plus petite dimension) sans trop perdre de précision.
- Un exemple concret sont les images, les formate .jpeg ou .png sont des encodages qui permettent d'enregistrer des images en prenant moins d'espace.

Encodage et décodage: minimisation de l'erreur de reconstruction

- Soit $\tilde{\mathbf{x}} = g(f(\mathbf{x}))$ la reconstruction de x, on veut apprendre une fonction f et g telle que l'on minimise:

$$\begin{aligned}
\text{ReconError} &= \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{x}_i - \tilde{\mathbf{x}}_i||^2 &(1)\\
&= \frac{1}{n}\sum_{i=1}^{n} \sqrt{\sum_{j=1}^{p}(x_{i,j} - \tilde{x}_{i,j})^2} &(2)
\end{aligned}$$

On peut essayer de minimiser la fonction objective en fonction de f et g.

PCA & neural nets: Autoencoders

$$\begin{tikzcd} \boldsymbol{X} \in \mathcal{X} \arrow[r, bend right, "f"'] & \boldsymbol{Z} \in \mathcal{Z} \arrow[r, bend right, "g"'] & \widetilde{\boldsymbol{X}} \in \mathcal{X} \end{tikzcd}$$

The error function of a nonlinear autoencoder is

$$\|\boldsymbol{X}-\widetilde{\boldsymbol{X}}\|^2 = \|\boldsymbol{X}-g\circ f(\boldsymbol{X})\|^2 = \sum_{i=1}^{n}\big(g\circ f(\boldsymbol{x}_i)-\boldsymbol{x}_i\big)^\top\big(g\circ f(\boldsymbol{x}_i)-\boldsymbol{x}_i\big)$$

$$\begin{tikzcd} \boldsymbol{X} \in \mathcal{X} \arrow[r, bend right, "\boldsymbol{P}"'] & \boldsymbol{Z} \in \mathcal{Z} \arrow[r, bend right, "\boldsymbol{P}^\top"'] & \widetilde{\boldsymbol{X}} \in \mathcal{X} \end{tikzcd}$$

The error function of a linear autoencoder is

$$\|\boldsymbol{X}-\widetilde{\boldsymbol{X}}\|^2 = \|\boldsymbol{X}-\boldsymbol{P}^\top\boldsymbol{P}\boldsymbol{X}\|^2 = \sum_{i=1}^{n}\big(\boldsymbol{P}^\top\boldsymbol{P}\boldsymbol{x}_i-\boldsymbol{x}_i\big)^\top\big(\boldsymbol{P}^\top\boldsymbol{P}\boldsymbol{x}_i-\boldsymbol{x}_i\big)$$

Autoencoders

The error function of a linear autoencoder is

$$\sum_{i=1}^{n}\left(\boldsymbol{P}^\top\boldsymbol{P}\boldsymbol{x}_i-\boldsymbol{x}_i\right)^\top\left(\boldsymbol{P}^\top\boldsymbol{P}\boldsymbol{x}_i-\boldsymbol{x}_i\right)$$

i.e.

$$\sum_{i=1}^{n}\text{trace}\left[\left(\boldsymbol{P}^\top\boldsymbol{P}-\mathbb{I}\right)\boldsymbol{x}_i\boldsymbol{x}_i^\top\left(\boldsymbol{P}^\top\boldsymbol{P}-\mathbb{I}\right)\right]$$

$$\text{trace}\left[\left(\boldsymbol{P}^\top\boldsymbol{P}-\mathbb{I}\right)\sum_{i=1}^{n}\boldsymbol{x}_i\boldsymbol{x}_i^\top\left(\boldsymbol{P}^\top\boldsymbol{P}-\mathbb{I}\right)\right]$$

the middle term is a covariance matrix, thus it is $\boldsymbol{V\Delta V}^\top$, so we recognize

$$\|\left(\boldsymbol{P}^\top\boldsymbol{P}-\mathbb{I}\right)\boldsymbol{V\Delta}^{1/2}\|_F^2$$

where $\|\cdot\|_F$ denotes the Froebenius / entrywise ℓ_2 norm of a matrix,

$$\|\boldsymbol{M}\|_F^2=\sum_{i,j}M_{i,j}^2=\text{trace}\left(\boldsymbol{M}\boldsymbol{M}^\top\right)\text{ where }\boldsymbol{M}=[M_{i,j}]$$

Alternative Approach

Let $\|\cdot\|_F$ denote the Froebenius / entrywise ℓ_2 norm of a matrix,

$$\|\boldsymbol{M}\|_F^2 = \sum_{i,j} M_{i,j}^2 \text{ where } \boldsymbol{M} = [M_{i,j}]$$

Consider the following problem,

$$\min_{\boldsymbol{Y}} \{\|\boldsymbol{X} - \boldsymbol{Y}\|_F^2\} \text{ subject to } \text{rank}(\boldsymbol{Y}) = k \quad (\leq \text{rank}(\boldsymbol{X}))$$

If $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Delta}\boldsymbol{V}^\top$ then $\boldsymbol{Y} = \boldsymbol{U}_k\boldsymbol{\Delta}_k\boldsymbol{V}_k^\top$ where we keep the first k columns of $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{\Delta}$.
One can rewrite

$$\min_{\boldsymbol{P}\in\Pi} \{\|\boldsymbol{X} - \boldsymbol{P}\boldsymbol{X}\|_F^2\} \text{ subject to } \text{rank}(\boldsymbol{P}) = k$$

where Π is the set of projection matrices.
If $\boldsymbol{S} = \boldsymbol{X}^\top\boldsymbol{X}$, we can write (equivalently)

$$\max_{\boldsymbol{P}\in\Pi} \{\text{trace}(\boldsymbol{S}\boldsymbol{P})\} \text{ subject to } \text{rank}(\boldsymbol{P}) = k$$

Alternative Approach

or

$$\max_{\boldsymbol{P}\in\mathcal{P}} \{\text{trace}(\boldsymbol{SP})\}$$

where

$$\mathcal{P} = \{\boldsymbol{P} \in \Pi : \text{eigenvalues}(\boldsymbol{P}) \in \{0,1\} \text{ and } \text{trace}(\boldsymbol{P}) = k\}$$

Then $\boldsymbol{P}^\star = \boldsymbol{V}_k\boldsymbol{V}_k^\top$
Note that $\mathcal{P}$ is note convex. But one can consider

$$\mathcal{P}_\star = \{\boldsymbol{P} \in \Pi : \text{eigenvalues}(\boldsymbol{P}) \in [0,1] \text{ and } \text{trace}(\boldsymbol{P}) = k\}$$

which is the convex Hull of $\mathcal{P}$. And one can prove that the convex problem

$$\max_{\boldsymbol{P}\in\mathcal{P}_\star} \{\text{trace}(\boldsymbol{SP})\}$$

Same solution as the non-convex one (proof in Fan (1949)).

Autoencodeur: Introduction

- Un autoencodeur est un système d'encodage (de compression de données) et de décodage (de reconstruction de données) entrainer en simultané; la fonction d'encodage et de décodage vont ensemble.
- Un critère d'évaluation de nos fonctions d'encodage et de décodage est l'erreur de reconstruction.
- Soit $\tilde{\mathbf{x}} = g(f(\mathbf{x}))$ la reconstruction de $\mathbf{x}$ après avoir été encodé $f(\mathbf{x})$ puis décodé $g(f(\mathbf{x}))$ on désire avoir $\tilde{\mathbf{x}}$ le plus près de $\mathbf{x}$ possible.

Autoencodeur: Introduction

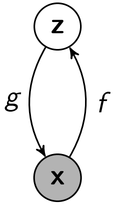
- On parle d'encodage sans perte (*lossless*) s'il y existe une fonction f et g tel que $\tilde{\mathbf{x}} = g(f(\mathbf{x})) = \mathbf{x}$.
- Souvent on parle d'un encodage avec perte (*lossy*).
- L'autoencodeur est un modèle d'apprentissage non-supervisé encore grandement étudié, à mesure que les données disponibles dans le monde moderne sont de plus en plus massives; les besoins d'algorithme pour la compression de données sont de plus en plus important!

Autoencodeur: un exemple simple, l'analyse en composante principale

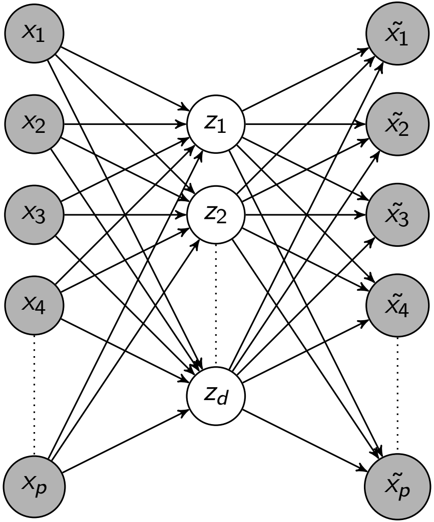
- La semaine passé on a vu que les composantes principales sont la solution de l'optimisation d'un autoencodeur simple.
- On a vu que si les fonctions d'encodage f et de décodage g sont toutes deux formées de combinaisons linéaires alors les vecteurs propres forme l'encodage optimal et la transposé de ceux-ci le décodage optimal.
- L'ACP est donc un exemple d'autoencodeur; un cas particulier d'autoencodeur.

Autoencodeur: représentation graphique

Parfois, on utilise une notation graphique très simplifié pour décire la reltation entre **x** et **z**.



Autoencodeur: représentation graphique



Autoencodeurs modernes

- Avec l'arrivé des réseaux de neurones comme approximateur de fonction universel, il était naturelle de mettre au point des algorithmes d'encodage et décodage à l'aide de réseaux de neurones.
- Bref, si un autoencodeur **généralise** l'ACP en permettant des fonctions f et g plus complexe, une fonction dde type réseaux de neurones semble être un choix intuitif pour complexifié f et g.

Réseaux de neurones: rappel

- Une fonction f peut-être approximé par un réseau de neuronnes à mesure que nous pouvons définire une fonction objective
- et que celle-ci est différentiable par rapport au paramètre qui forme le réseau f.

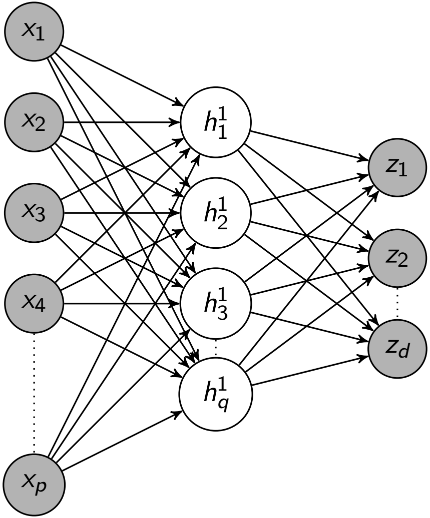
Autoencodeurs modernes

- On pourrait percevoir l'autoencodeur comme un seul géant réseaux de neurones $f: \mathcal{X} \to \mathcal{X}$.
- Par contre il est standard de plutôt percevoir le model comme étant deux réseaux de neurones $f: \mathcal{X} \to \mathcal{Z}$ et $g: \mathcal{Z} \to \mathcal{X}$ mis bout à bout.

Autoencodeurs modernes: fonction d'encodage

- Soit $f : \mathcal{X} \to \mathcal{X}$ une fonction d'encodage de type réseau de neurones alors, par exemple:
$$\mathbf{z} = f(\mathbf{x}) = \phi_2(\phi_1(\mathbf{x}_{1\times p}\mathbf{B}^{(1)}_{p\times q})\mathbf{B}^{(2)}_{q\times d})$$
- Le nombre de couches, de neuronnes et les fonctions d'activation ϕ sont des hyperparamètres déterminés préalablement,
- p et d sont déterminé par le problème,
- et les matrices $\mathbf{B}$ sont des matrices de paramètres/coefficients à apprendre (par *back-propagation*).

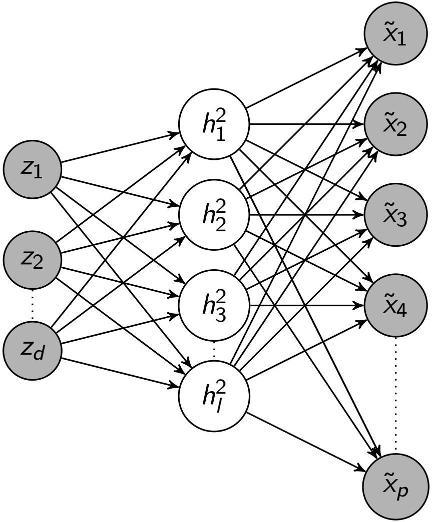
Autoencodeurs modernes: fonction d'encodage



Autoencodeurs modernes: fonction de décodage

- Soit $g : \mathcal{Z} \to \mathcal{X}$ une fonction de décodage de type réseau de neurones alors, par exemple:
$$\tilde{x} = g(\mathbf{z}) = \phi_4(\phi_3(\mathbf{z}_{1\times d}\mathbf{B}^{(3)}_{d\times l})\mathbf{B}^{(4)}_{l\times p})$$
- Le nombre de couches, de neuronnes et les fonctions d'activation ϕ sont des hyperparamètres déterminés préalablement,
- p et d sont déterminé par le problème,
- et les matrices $\mathbf{B}$ sont des matrices de paramètres/coefficients à apprendre (par *back-propagation*).

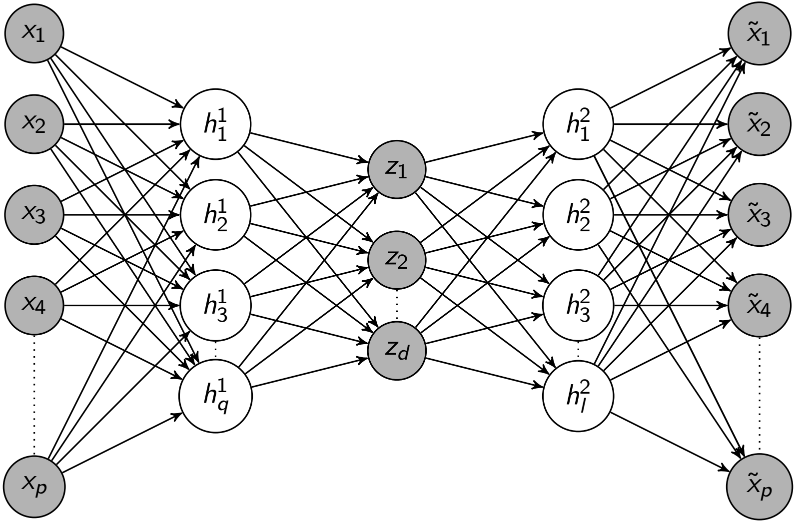
Autoencodeurs modernes: fonction de décodage



Autoencodeurs modernes

- On définit un autoencodeurs de réseaux de neurones comme étant un modèle qui met bout-à-bout les deux réseaux que nous venons de définir.
- Cela **généralise** l'ACP; les fonctions f et g permisent sont beaucoup plus complexes; elle ne sont pas de simples coombinaisons linéaires.
- Le code $\mathbf{z}$ est la sortie d'une fonction non-linéaire des entrés $\mathbf{x}$ et la reconstruction $\tilde{\mathbf{x}}$ est le résultat d'une fonction non-linéaire du code $\mathbf{z}$.

Autoencodeurs modernes: représentation graphique



Autoencodeurs modernes: fonction objective

La fonction objective ici est l'erreur de reconstruction que l'on veut minimiser:

$$\text{ReconError} = \frac{1}{n}\sum_{i=1}^{n}||\mathbf{x}_i - \tilde{\mathbf{x}}_i||^2 = \frac{1}{n}\sum_{i=1}^{n}||\mathbf{x}_i - g(f(\mathbf{x}_i))||^2$$

et nous pouvons calculer le gradient de cette fonction en fonction de chaque des paramètres $b \in B$

Autoencodeurs modernes

L'autoencodeur préserve les forces des modèles de réseaux de neurones:

- Très performant dans sa tâche
- Une fois appris, peut compresser de grandes quantités d'observation rapiddement
- Peut s'intégrer comme morceau d'un modèles plus complexe entrainer par optimisation de gradient.

Autoencodeurs modernes

L'autoencodeur préserve aussi les faiblesses des modèles de réseaux de neurones:

- Nécessite beaucoup de données
- Choisir les hyperparamètres peut être difficile
- Aucune garantie théorique.

Références

Eckart, C. and Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218.

Escofier, B. and Pagès, J. (2023). *Analyses factorielles simples et multiples*. Dunod.

Fan, K. (1949). On a theorem of weyl concerning eigenvalues of linear transformations i. *Proceedings of the National Academy of Sciences*, 35(11):652–655.

Husson, F., Lê, S., and Pagès, J. (2016). *Analyse de données avec R*. Presses universitaires de Rennes.

Lebart, L., Piron, M., and Morineau, A. (1995). *Statistique exploratoire multidimensionnelle*. Dunod.

Mirsky, L. (1960). Symmetric gauge functions and unitarily invariant norms. *The quarterly journal of mathematics*, 11(1):50–59.