Introduction à l'apprentissage statistique - 2022

François Landes

December 4, 2023

Contents

1 Réduction dimensionnelle : cas de l'Analyse en Composantes Principales, ACP (Principal Components Analysis, PCA) (aperçu en CM6, vu à fond plus tard)

L'analyse en composantes principales (en anglais, *Principal Components Analysis*, PCA) peut-être vue comme un des nombreux pré-traitements disponibles dans la panoplie du Machine-Learner. Mais c'est un outil en fait beaucoup plus général et puissant, qui est plus ancien que l'avènement des méthodes de ML modernes.

En dépit de son âge, cette méthode n'a pas tellement vieilli: de par son élégance, ou son côté "bien fondée formellement", elle reste d'actualité.

La PCA est une méthode d'**apprentissage non supervisé**. Elle fait partie des méthodes dites de **réduction dimensionnelle** (*dimensional reduction*).

La PCA peut aussi être vue comme une méthode de visualisation de données en grande dimension. Cependant, ce serait ignorer sa puissance de la limiter à ce rôle. D'autres méthodes, moins contrôlées mais assez puissantes, comme *t-SNE*, sont bien adaptées pour la visualisation de données en grande D.

La PCA peut aussi être vue comme une méthode de compression des données. La reconstruction (décompression) est très facile.

1.1 Intuition sur : réduction dimensionnelle

De façon générale (et non seulement pour la PCA), les méthodes de réduction dimensionnelle consistent à réduire la dimension de l'espace d'entrée (l'espace des features, de dimension D), vers une autre dimension, plus petite, D' (ou p). Concrètement, cela consiste à chercher une application non injective Φ:

$$\Phi : \mathbb{R}^D \longrightarrow \mathbb{R}^{D'}, \text{ avec} D' < D \tag{1}$$

Pourquoi est-ce intéressant? Il y a différentes raisons, qui se recoupent entre elles. Réduire la dimension des donnée:

- Permet de supprimer les redondances dans l'information d'entrée. Par exemple, lorsque 2 attributs sont identiques en valeur, ou de valeur proportionnelle, il n'y a pas d'intérêt à les garder en doublon (c'est même nuisible, a priori).
- Permet de réduire le "bruit" (c'est le nom qu'on donne aux informations "inutiles"... c'est une vaste question de savoir ce que ça veut dire, précisément, et il est très difficile de distinguer le bruit du signal).
- On évite ainsi la malédiction des grandes dimensions. En effet, en général (mais pas nécessairement), plus les dimensions sont grandes, plus on a de paramètres à ajuster et on se retrouve facilement dans des cas de sur-apprentissage.
- On évite donc le sur-apprentissage: en effet, qui dit moins de dimensions en entrée dit, en général, moins de paramètres dans le modèle.
- Cela aide donc l'apprentissage: en termes de vitesse (a priori, c'est le cas, on peut probablement imaginer des cas pathologiques où ça ralentit l'apprentissage).
- Cela aide donc l'apprentissage: en termes de performance: ça dépend. On peut limiter la performance en supprimant les données, mais on peut aussi l'améliorer par la diminution du sur-apprentissage.

1.2 PCA: idée

Ici on ne verra qu'une réduction dimensionnelle: la PCA.

Dans ce cas particulier, l'application Φ est une projection (application linéaire) de l'espace $\mathbb{R}^D$ dans le sous-espace $\mathbb{R}^{D'}$. On projette les données sur un hyper-plan de dimension D', qui a une certaine orientation dans l'espace parent $\mathbb{R}^D$. Dans la figure 1, on voit ce que cela peut signifier, dans un cas à petite dimension, ou on peut encore dessiner. On voit aussi, ou bien on rappelle que, une projection est aussi le choix d'un nombre D' de combinaisons linéaires des D attributs d'entrée. Chaque dimension/nouvel attribut, après PCA, sera une combinaison linéaire des attributs d'entrée.

On notera en général $\vec{x}' = P\vec{x}$ les données après projection ($P_{(D,D')}$ étant la matrice de projection, ou la matrice des coefficients des combinaisons linéaires, si vous voulez). Dans le cas particulier où $D' = 1$, x' est un nombre et $x' = \vec{x}.\vec{P}$ où . représente le produit scalaire.

La question qui se pose maintenant est: comment choisir un bon hyper-plan sur lequel ou voudrait projeter ? Le choix fait, dans le cadre de la PCA, est de prendre l'hyperplan **tel que la variance des données, après projection, est maximale**. On favorise donc les dimensions de grande variance (et leurs combinaisons linéaires). On verra que **ce choix est équivalent à un autre choix**, qui serait celui de vouloir **minimiser l'erreur de reconstruction** (en norme L2, c'est-à-dire de minimiser l'erreur quadratique moyenne de reconstruction).

L'idée (naïve) ici est donc de **garder les attributs qui varient le plus** (car se sont ceux qui portent le plus d'information). Par ailleurs, si les données sont très hétérogènes, on peut les standardiser un minimum avant d'effectuer la PCA (pas sûr pendant le cours).

1.3 PCA: démonstration du bien-fondé de la méthode

1.3.1 Variance des données après projection

Calculons donc la variance des données, après projection. Pour commencer, on ne s'intéresse qu'à une direction de projection. Soit $\vec{u}_i$ le vecteur qui caractérise l'axe de la projection. Les données projetées deviennent donc de simples nombres, qu'on peut noter $x'_n = \vec{u}_i^T \vec{x}_n = \sum_d u_{id} x_{nd}$.

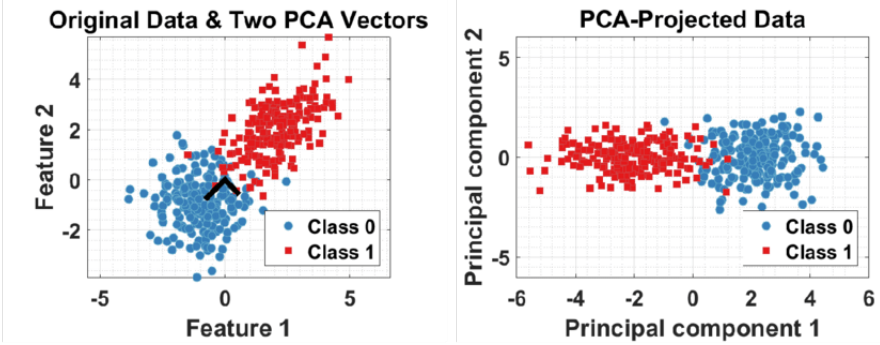

Figure 1: PCA de données à $D \geq 2$ dimensions (on n'en voit que 2, les 2 premières, mais on peut imaginer qu'il y en a d'autres, orthogonales aux premières) vers $D' = 2$ dimensions. Si on imagine le cas $D = 2$, on fait alors juste une rotation. Si on avait choisi $D' = 1$, il ne resterait pour représenter les données que l'axe des abscisses, celui de la première composante: dans l'image de droite, il faudrait imaginer que l'axe des ordonnées a été écrasé, qu'il n'en reste que la valeur moyenne (ou 0).

On va prouver que la variance des données le long de cet axe, $\sigma_i'^2$, est donnée par: $\sigma_i'^2 = \vec{u}_i^T C \vec{u}_i$, où C est la matrice de covariance des données. (L'idée de la démonstration est de se concentrer sur une direction, ici $\vec{u}_i$). On a déjà parlé de la covariance, mais on est sympa, on rappelle la formule:

$$C = \frac{1}{N} \sum_n^N (\vec{x}_n - \langle \vec{x} \rangle)(\vec{x}_n - \langle \vec{x} \rangle)^T \tag{2}$$

Pour rappel, C est clairement une matrice de taille (D, D) (somme de N produits matriciels de matrices $(D, 1)$ par matrices $(1, D)$) et $\langle \vec{x} \rangle$ représente le vecteur moyenen de $\vec{x}$ (la moyenne empirique). On calcule la variance des données après projection sur $\vec{u}_i$, notée $\sigma_i'^2$:

$$\sigma_i'^2 = \frac{1}{N} \sum_n (x'_n - \langle x' \rangle)^2 \qquad = \text{Définition de la variance} \tag{3}$$

$$= \frac{1}{N} \sum_n (\vec{u}_i^T \vec{x}_n - \langle \vec{u}_i^T \vec{x} \rangle)^2 \tag{4}$$

$$= \frac{1}{N} \sum_n (\vec{u}_i^T \vec{x}_n - \langle \vec{u}_i^T \vec{x} \rangle)(\vec{u}_i^T \vec{x}_n - \langle \vec{u}_i^T \vec{x} \rangle)^T \tag{5}$$

Où on a utilisé que $a_i^2 = a_i a_i^T$, même si a_i est de taille (1,1).

$$= \frac{1}{N} \sum_n (\vec{u}_i^T \vec{x}_n - \langle \vec{u}_i^T \vec{x} \rangle)(\vec{x}_n - \langle \vec{x} \rangle)^T (\vec{u}_i^T)^T \tag{6}$$

On a utilisé que $(AB)^T = B^T A^T$.

$$= \frac{1}{N} \sum_n \vec{u}_i^T (\vec{x}_n - \langle \vec{x} \rangle)(\vec{x}_n - \langle \vec{x} \rangle)^T \vec{u}_i \tag{7}$$

$$= \vec{u}_i^T \quad \frac{1}{N} \sum_n (\vec{x}_n - \langle \vec{x} \rangle)(\vec{x}_n - \langle \vec{x} \rangle)^T \quad \vec{u}_i \tag{8}$$

$$\sigma_i'^2 = \vec{u}_i^T \quad C \quad \vec{u}_i \tag{9}$$

CQFD.

1.3.2 Calcul de la meilleure direction

Comme on l'a dit, **le choix fait dans la méthode de la PCA est celui de maximiser la variance des données projetées, donc trouver le $\vec{u}_i$ qui maximise $\sigma_i'^2$**. A priori, on pourrait écrire que le meilleur vecteur de projection est donc :

$$(?) \quad \vec{u}_i^* = \underset{\vec{u}_i \in \mathbb{R}^D}{\text{argmax}} \left(u_i^T C \vec{u}_i \right) \quad (?) \tag{10}$$

Cependant, si on résout ce problème d'optimisation, on obtient la solution triviale et non intéressante, qu'il faut prendre un vecteur $\vec{u}_i$ de la norme la plus grande possible (une matrice de Covariance est par définition semi-définie positive, c.a.d. que ses valeurs propres sont positives ou nulles). Ce n'est pas intéressant, car on fait une projection, ce qui nous intéresse c'est de trouver le bon angle de projection, pas de multiplier nos données par ∞ pour augmenter leur variance.

On décide donc de se restreindre aux vecteurs de norme 1, c'est-à-dire de ne regarder que l'orientation de $\vec{u}_i$: on impose $\vec{u}_i^2 = 1$ On traite donc du problème d'optimisation sous contrainte suivant:

$$\vec{u}_i^* = \underset{\vec{u}_i \in \mathbb{R}^D, \vec{u}_i^2 = 1}{\operatorname{argmax}} \left(u_i^T C \vec{u}_i \right) \tag{11}$$

Qui peut se réécrire, en attribuant à la contrainte $u_i^2 = 1 \Leftrightarrow (1 - u_i^2) = 0$ le multiplicateur[1] λ:

$$\vec{u}_i^* = \underset{\vec{u}_i \in \mathbb{R}^D}{\operatorname{argmax}} \left(u_i^T C \vec{u}_i + \lambda(1 - \vec{u}_i^2) \right) \tag{12}$$

On note:

$$F(\vec{u}_i) = u_i^T C \vec{u}_i + \lambda(1 - \vec{u}_i^2) \tag{13}$$

$$\vec{u}_i^* = \underset{\vec{u}_i \in \mathbb{R}^D}{\operatorname{argmax}} \left(F(\vec{u}_i) \right) \tag{14}$$

Ce problème de maximisation est assez simple, et il se trouve qu'on peut le résoudre exactement. La méthode est classique: on cherche le zéro de la fonction que est dans le argmax, c'est-à-dire on cherche à résoudre $\vec{\nabla}_{\vec{u}_i} F(\vec{u}_i) = \vec{0}$. Ce n'est pas évident, mais le gradient de $u^T C u$ se calcule, pour une matrice C symmétrique (et donc telle que $C^T = C$): $\vec{\nabla}_{\vec{u}}(\vec{u}^T C \vec{u}) = 2C\vec{u}$. En fait, pour faire ce calcul, on repart de la définition intiale, c'est-à-dire de $\vec{u}^T C \vec{u} = \frac{1}{N} \sum_n^N (\vec{u}^T \vec{x} - \langle \vec{u}^T \vec{x} \rangle)^2$. Pour alléger la notation, on va supposer, temporairement, que les données sont centrées, c'est-à-dire que $\langle \vec{x} \rangle = \vec{0}$. Ça ne change pas la valeur de la matrice de covariance, et ça allège l'écriture.

$$\vec{\nabla}_{\vec{u}}(\vec{u}^T C \vec{u}) = \vec{\nabla}_{\vec{u}} \frac{1}{N} \sum_n^N (\vec{u}^T \vec{x}_n - \cancel{\langle \vec{u}^T \vec{x}_n \rangle})^2 \tag{15}$$

$$= \frac{1}{N} \sum_n^N \vec{\nabla}_{\vec{u}} (\vec{u}^T \vec{x}_n)^2 \tag{16}$$

$$= \frac{1}{N} \sum_n^N 2\vec{x} (\vec{u}^T \vec{x}_n)^1 \qquad \text{car } \frac{\partial}{\partial s} f^2(s) = 2 \frac{\partial f}{\partial s} f(s) \tag{17}$$

$$\text{Puisque } \vec{u}^T \vec{x} \text{ est un nombre,} \vec{u}^T \vec{x}_n = (\vec{u}^T \vec{x})^T = \vec{x}_n^T \vec{u} \tag{18}$$

$$= \frac{1}{N} \sum_n^N 2\vec{x}_n (\vec{x}_n^T \vec{u}) \tag{19}$$

$$= 2 \frac{1}{N} \sum_n^N (\vec{x}_n \vec{x}_n^T) \vec{u} \tag{20}$$

$$= 2C\vec{u} \tag{21}$$

Par ailleurs, il est facile de vérifier que $\vec{\nabla}_{\vec{u}} \lambda \vec{u}^2 = 2\lambda \vec{u}$. On peut donc terminer la recherche du argmax($F(\vec{u}_i)$):

$$\vec{\nabla}_{\vec{u}_i} F(\vec{u}_i) = \vec{0} \Leftrightarrow \tag{22}$$

$$\vec{0} = 2C\vec{u} - 2\lambda\vec{u} \tag{23}$$

$$C\vec{u} = \lambda\vec{u} \tag{24}$$

C'est l'équation aux valeurs propres ! Comme c'est joli ! C'est un résultat assez profond. Il y a donc **autant de maxima locaux de la fonction $F(\vec{u}_i)$ qu'il y a de couples** ($\lambda_i, \vec{u}_i$ (couples de valeur propre-vecteur propre). La norme des vecteurs propre est imposée par notre contrainte $\vec{u}^2 = 1$. La direction (le signe

[1] Ici, le signe à mettre devant la contrainte s'obtient en réfléchissant sur la direction qu'on veut contenir: en fait, ce qu'on veut, c'est que $(1 - u_i^2) \geq 0$, mais cette inégalité va être saturée car les plus grand u_i^2 sont favorisés. D'où l'ajout de $\lambda(1 - \vec{u}_i^2)$ à la fonction, et non pas son opposé, $\lambda(\vec{u}_i^2 - 1)$.

devant $\vec{u}$) est arbitraire, ce qui est normal pour une projection, car une projection dépend de la droite sur laquelle on projète, mais pas de son orientation. Par contre, il y a plein de valeurs propres... laquelle faut il prendre ? Hé bien, il faut prendre celle qui réalise le maximum global de $F(\vec{u}_i)$.

Maintenant qu'on sait qu'on s'intéresse aux vecteurs $\vec{u}$ de norme 1 et tels que $C\vec{u} = \lambda\vec{u}$ (c'est-à-dire les vecteurs propres), on peut réécrire F, en prenant un couple $(\lambda_i, \vec{u}_i)$ arbitraire:

$$F(\vec{u}_i)_{|\vec{\nabla}_{\vec{u}_i} F(\vec{u}_i)=\vec{0}} = \vec{u}_i^T C\vec{u}_i + \lambda(1 - \vec{u}_i^2) \tag{25}$$

$$= \vec{u}_i^T \lambda_i \vec{u}_i + 0 \tag{26}$$

$$= \lambda_i \vec{u}_i^T \vec{u}_i \tag{27}$$

$$= \lambda_i \tag{28}$$

Plutôt propre, non ? Donc en fait, **la variance des données projetées est exactement égale à la valeur propre correspondant à la direction propre qu'on choisit pour projeter**. Une autre façon de le voir est de juste calculer la variance projetée, sans passer par F: $\sigma_i^2 = \vec{u_i}^T(C\vec{u_i}) = \vec{u_i}^T\lambda_i\vec{u_i} = \lambda_i\vec{u_i}^2 = \lambda_i$.

Conclusion: la meilleure direction, parmi toutes les directions, est celle donnée par le vecteur propre qui est associé à la plus grande valeur propre.

1.3.3 Directions suivantes

En pratique, on ne souhaite pas conserver que 1 composante, mais plutôt projeter depuis un espace de dimension D vers un espace de dimension $D', 1 < D' < D$. On projette donc non pas sur une droite (un seul vecteur) mais plutôt sur un hyper-plan, dont l'orientation est définie par les vecteurs qui sont dans ce plan. Pensez par exemple à une paire de vecteurs (non colinéaires) dans un espace à $D = 3$: ces deux vecteurs définissent naturellement un plan (à $D' = 2$ dimensions).

Les données **après projection sur un hyper-plan** de dimension $D'2$, par exemple, s'écrivent de la façon suivante. Appelons $\vec{u}_1 \in \mathbb{R}^D, \vec{u}_2 \in \mathbb{R}^D$ les deux vecteurs qui définissent ce plan. Appelons $\vec{e}_1 = (1, 0), \vec{e}_2 = (0, 1)$ les deux vecteurs de la base interne au plan (ces vecteurs définissent un repère interne au plan, qui permet de se promener dedans). Alors la projection s'écrit:

$$\vec{x}' = (\vec{x}^T\vec{u}_1)\vec{e}_1 + (\vec{x}^T\vec{u}_2)\vec{e}_2 \tag{29}$$

L'écriture générale est $\vec{x}' = \sum_{d'=1}^{D'}(\vec{x} \cdot \vec{u}_{d'})\vec{e}_{d'}$.

Quels sont les directions qui définissent cet hyper-plan ? On ne refait pas tout le raisonnement, mais au vu de la partie précédente, **la 2ème meilleure direction est celle donnée par le vecteur propre qui est associé à la plus grande valeur propre**. Et ainsi de suite pour la 3ème, la 4ème, etc.

On aboutit alors à la remarque suivante. Si on diagonalise la matrice de covariance, C, on a : $C = U\Lambda U^{-1}$, avec Λ (se lit Lambda majuscule, comme λ) la matrice diagonale des valeurs propres, et U la matrice de passage vers l'espace où C est diagonalisée.

$$\Lambda = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_D \end{pmatrix} \qquad U = \begin{pmatrix} \begin{pmatrix} . \\ \vec{u}_1 \\ . \end{pmatrix} & \begin{pmatrix} . \\ \vec{u}_2 \\ . \end{pmatrix} & ... & \begin{pmatrix} . \\ \vec{u}_D \\ . \end{pmatrix} \end{pmatrix} \tag{30}$$

On ne rentre pas dans les détails, mais il est légitime de supposer qu'une matrice de covariance de données "normales" sera diagonalisable [2] , avec tous les vecteurs propres distincts deux à deux. Par définition de C, elle est semi-définie positive, c'est-à-dire que toutes ses valeurs propres sont positives ou nulles. Comme toujours lors d'une diagonalisation, U **est constituée des vecteurs propres mis côte à côte**, dans le même ordre que les λ_i dans Λ. Ici, on choisit d'**ordonner les λ_i par valeur décroissante** (elles sont toutes positives et réelles).

La matrice de projection sur l'hyper-plan de dimension D' s'obtient alors (souvenez vous qu'on a trié les λ_i par ordre décroissant) en gardant les D' premières directions propres:

$$P = \begin{pmatrix} \begin{pmatrix} . \\ \vec{u}_1 \\ . \end{pmatrix} & \begin{pmatrix} . \\ \vec{u}_2 \\ . \end{pmatrix} & ... & \begin{pmatrix} . \\ \vec{u}_{D'} \\ . \end{pmatrix} \end{pmatrix} \tag{31}$$

[2] Le sous-espace des matrices diagonalisables dans $\mathbb{C}$ est dense dans l'espace des matrices (à valeurs dans $\mathbb{C}$. Ceci n'est pas vrai dans $\mathbb{R}$, mais à part pour de données pathologiques, une matrice de covariance sera diagonalisable.

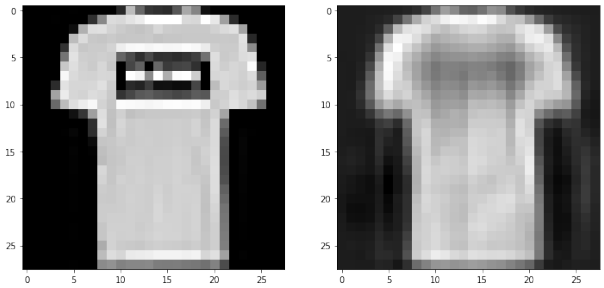

Figure 2: On passe d'une image de dimension $D = 784$ à $D' = 30$, puis on revient à l'espace d'origine pour visualisation. On a $D' = 30 \sim \sqrt{D}$ soit presque 30 fois moins, et pourtant on peut toujours reconnaître l'image.

La matrice P n'est pas carrée, elle est de taille (D, D') (D lignes, D' colonnes). Comme on l'a dit plus haut, la donnée $\vec{x}_n$ projetée s'obtient alors ainsi: $\vec{x}'_n = \sum_{d'=1}^{D'} (\vec{u}_{d'} \cdot \vec{x}_n)\vec{e}_{d'}$. Ceci est en fait l'écriture d'une multiplication matricielle:

$$\vec{x}_{n,transformed} = (\vec{x}_n)' = (\vec{x}_n - \langle\vec{x}\rangle) \cdot P \tag{32}$$

Ces vecteurs sont de dimension D'.

1.3.4 Décompression

La transformée inverse s'obtient ainsi:

$$\vec{x}_{n,decompressed} = \vec{x}_{n,transformed} \cdot P^T + \langle\vec{x}\rangle \tag{33}$$

On obtient donc de nouveau des vecteur de dimension D.

On voit dans cette expression en quoi c'est une expression avec perte. En effet, on a $\vec{x}_{n,decompressed} = ((\vec{x}_n - \langle\vec{x}\rangle) \cdot P) \cdot P^T + \langle\vec{x}\rangle$. L'opérateur PP^T est une matrice de dimension (D, D), mais qui intuitivement, "passe par une dimension intérmédiaire de dimension D'". Mathématiquement, on pourrait constater que PP^T n'a que D' directions indépendantes (valeurs propres distinctes).

1.4 PCA: résumé de l'algorithme

L'algorithme de la PCA est donc le suivant :

1. Calculer la matrice de covariance C (cela se fait en un temps linéaire en la taille des données) : $C = \frac{1}{N}\sum_{n=1}^{N}(\vec{x}_n - \langle\vec{x}\rangle)(\vec{x}_n - \langle\vec{x}\rangle)^T$
2. Diagonaliser C. Cela se fait en un temps $O(D^3)$, a priori (voir remarque plus bas). Cela revient à chercher une matrice de passage U telle que
$$C = U\Lambda U^{-1}$$
où Λ est une matrice diagonale dont les coefficients diagonaux sont les valeurs propres de C rangées dans l'ordre décroissant.
3. Garder seulement les D' premières valeurs propres (on s'intéresse uniquement aux plus grandes valeurs propres qui permettent d'obtenir plus grande variance, que l'on cherche justement à maximiser) : $P = (\vec{u}_1, ..., \vec{u}_{D'})_{D,D'}$
4. Projeter (transformer) : $\vec{x}_{n,transformed} = (\vec{x}_n - \langle\vec{x}\rangle).P$

Par ailleurs si on ne cherche que les D' plus grandes valeurs propres, on peut le faire en un temps $O(D'D^2)$ au lieu de $O(D^3)$, ce qui est très appréciable (on part du principe que $D' \leq D$). Il existe aussi des solutions encore approximatives plus rapides avec résultats légèrement aléatoires. D'autant que dans le cas de la projection ce n'est pas grave d'avoir quelques erreurs.

Une fois cet algorithme appliqué, on peut effectuer la transformation inverse pour revenir à l'espace de départ (c'est une compression avec perte).

Attention : Quand on effectue une PCA sur une une image de dimension D, on obtient D' nombres, mais on n'a plus du tout a faire a une image, la donnée n'est plus visualisable – on peut cependant la décompresser ensuite pour revenir à une image.

1.5 Ressources

Concernant ce Chapitre, vous avez:

- (ressource bilingue)
  `https://gitlab.inria.fr/flandes/ias/-/blob/master/PCA-intuitive-showcase.ipynb`
- (ressource bilingue)
  `https://gitlab.inria.fr/flandes/ias/-/blob/master/PCA-proof-of-exact-computation.ipynb`
- (en anglais)
  `https://plot.ly/ipython-notebooks/principal-component-analysis/`
- Bishop [**?**] chapter 12, page 561-570 (ou plus si vous êtes intéressé.e.s).

2 Modèles Bayésiens (CM7+8, séances 6+8)

Les techniques décrites ici peuvent être vues comme des *estimations de densité*.

Lorsqu'on souhaite "fitter" les paramètres d'une densité de probabilité sur un ensemble de données, la méthode la plus naturelle consiste à choisir les paramètres tels que les données soient "les plus vraisemblables" ou autrement dit, qu'elles soient "réalistes", c.a.d. qu'il soit crédible que ces données aient été générées par cette distribution, avec ces paramètres.

Cette approche s'appelle l'estimation par le Maximum de Vraisemblance (des données). On cherche ainsi à trouver les paramètres optimaux θ^* qui maximisent:

$$\theta^* = \text{argmax}_\theta(\mathbb{P}(X|\theta)) \tag{34}$$

Notation: l'exposant .* est souvent utilisé pour désigner la solution d'un problème d'optimisation.

Notation: le chapeau $\hat{u}$ est souvent utilisé pour désigner l'*estimateur* d'une quantité a priori inconnu. On lira $\hat{u}$ "u chapeau" ou bien "estimateur de u".

Notation: on dit que la variable aléatoire (v.a.) X est paramétrée par les paramètres θ.

2.1 MLE: 1 variable à 1 dimension

Prenons un exemple concret, par exemple la taille en cm d'un certain nombre N d'étudiants. On dispose de N points de données, à 1 dimension, c.a.d que ce sont des nombres réels. On suppose que les tirages sont indépendants. Si X_n est la v.a. associée au n-ième tirage (indiquée ci dessous), la v.a. des N tirages est alors la v.a. produit, notée X:

$$\forall n \in [|1, ..., N|], X_n \sim X_1 \tag{35}$$

$$X \sim (X_1, X_2, \dots, X_N) \tag{36}$$

Notations: $X \sim Y$ signifie que la v.a. X suit la même loi que la v.a. Y. L'indépendance entre deux v.a. X_1 et X_2 est notée $X_1 \perp\!\!\!\perp X_2$. Ici on a $X_i \perp\!\!\!\perp X_j, \forall i \neq j$. Ici, les virgules signalent un "ET" logique.

On dispose de données, qu'on note plutôt $X_{(N,1)}$ que X, afin de les distinguer de la variable aléatoire X. Les données sont la *réalisation* d'une certaine loi (ou bien quelque chose qu'on arrivera jamais à décrire mathématiquement, selon les problèmes). Les données $X_{(N,1)}$ sont donc une liste: $X_{(N,1)} = (x_1, x_2, \dots, x_N)$, avec $x_n \in \mathbb{R}$.

On définit alors la probabilité d'avoir obtenu le tirage $X_{(N,1)}$, depuis la v.a. X:

$$\mathbb{P}(X = X_{(N,1)}) = \mathbb{P}(X_1 = x_1, X_2 = x_2, \dots X_N = x_N) \qquad \text{(Définition, toujours vraie)} \tag{37}$$

$$= \prod_n^N \mathbb{P}(X_n = x_n) \qquad \text{(Car les } X_n \text{ sont indépendants)} \tag{38}$$

Il faut maintenant rendre les choses plus concrètes en choisissant un *modèle* mathématique, c'est-à-dire une expression **explicite** pour la forme de $\mathbb{P}(X_n = x_n)$. Ici, pour aujourd'hui, on suppose que tous les points sont distribués selon la (même) loi **Gaussienne**:

$$\mathbb{P}(X_n \in [x, x + dx]) = \rho(x)\mathrm{d}x, \tag{39}$$

$$\rho(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{40}$$

Où $\theta = (\mu, \sigma)$ sont les 2 paramètres réels à *estimer* (à "fitter", en vocabulaire de Machine Learning). Attention, de façon générale, on aurait pu faire un autre choix de représentation, à la place de la Gaussienne: on aurait pu prendre une loi de Poisson, de Bernoulli, etc. C'est un *choix* de *modélisation*.

Notation: $\rho(x)$ se lit rho de x, c'est la *densité* associée à la v.a. X, qui est une v.a. continue.

L'idée de maximiser la vraisemblance des données revient alors, concrètement, à **maximiser la vraisemblance des données sachant les paramètres**, notée $\mathcal{L}$ (comme Likelihood, Vraisemblance en anglais):

$$\mathcal{L}(\theta) = \mathbb{P}(X = X_{(N,1)}|\theta) \tag{41}$$

$$\theta^* = \text{argmax}_\theta(\mathcal{L}(\theta)) \tag{42}$$

Quel que soit le choix de modélisation, il y aura toujours au moins un paramètre à *estimer* d'après les données: c'est pour cela qu'on parle d'*estimateur par le maximum de vraisemblance* (*Maximum Likelihood Estimate*, en anglais, ou EMV en Français).

Il y a une astuce qui est appliquée quasi systématiquement: le argmax d'une fonction est aussi le argmax du log de cette fonction, car log est strictement croissante:

$$\theta^* = \text{argmax}_\theta(\mathcal{L}(\theta)) = \text{argmax}_\theta(\log(\mathcal{L}(\theta))) = \text{argmax}_\theta(\ell(\theta)) \tag{43}$$

$$\ell(\theta) = \log \mathcal{L}(\theta) \quad \text{(est nommée la log-vraisemblance ou log-likelihood)} \tag{44}$$

Le reste n'est plus qu'un bête calcul. Pour le modèle Gaussien, on calcule une bonne fois $\ell(\theta)$:

$$\ell(\theta) = \log\left(\rho(X = X_{(N,1)}|\theta)\right) \tag{45}$$

$$= \log\left(\prod_n^N \rho(X_n = x_n|\theta)\right) \tag{46}$$

$$= \log\left(\prod_n^N \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_n-\mu)^2}{2\sigma^2}\right)\right) \tag{47}$$

$$= \sum_n^N \left(\log \frac{1}{\sqrt{2\pi\sigma^2}}\right) + \sum_n^N \left(\log\exp\left(-\frac{(x_n-\mu)^2}{2\sigma^2}\right)\right) \tag{48}$$

$$= -N\log(\sqrt{2\pi}\sigma) - \sum_n^N \frac{(x_n-\mu)^2}{2\sigma^2} \tag{49}$$

On cherche le maximum (extremum) d'une fonction en cherchant à annuler sa dérivée première (on pourra vérifier que la dérivée seconde est négative). Donc, au point idéal $\theta^* = (\mu^*, \theta^*)$, on aura:

$$0 = \frac{\partial}{\partial\mu}\ell(\theta) \tag{50}$$

$$0 = 0 - \sum_n \frac{2(x_n-\mu)}{2\sigma^2}(-1) \tag{51}$$

$$0 = \sum_n (x_n - \mu) \tag{52}$$

$$N\mu = \sum_n x_n \tag{53}$$

$$\mu = \frac{1}{N}\sum_n x_n \hat{=} \overline{x} \tag{54}$$

C'est-à-dire qu'on trouve que l'estimateur du paramètre μ par maximum de vraisemblance, noté μ^* ou $\hat{\mu}$, est égal à la moyenne empirique, notée $\overline{x}$. On a trouvé : $\hat{\mu} = \overline{x}$

De même pour σ :

$$0 = \frac{\partial}{\partial\sigma}\ell(\theta) \tag{55}$$

$$0 = -N\frac{1}{\sigma} - \sum_n \frac{2(x_n-\mu)^2}{2\sigma^3} \tag{56}$$

$$0 = \frac{N}{\sigma}\sigma^3 + \sum_n \frac{(x_n-\mu)^2}{\sigma^3}\sigma^3 \tag{57}$$

$$\sigma^2 = \frac{1}{N}\sum_n^N (x_n-\mu)^2 \tag{58}$$

On trouve ici encore que le meilleur estimateur du paramètre σ pour modéliser les données, σ^*, aussi noté $\hat{\sigma}$, est égal à l'écart-type empirique, $\sigma_{empirique}$ (là il n'y a pas vraiment de notation pour résumer l'expression $\sqrt{\frac{1}{N}\sum_n^N (x_n-\mu)^2}$).

Finalement, on retrouve les estimations fréquentistes, c.a.d que l'estimation du maximum de vraisemblance coïncide avec les estimations empiriques (par exemple, la variance empirique, c'est $\frac{1}{N}\sum_n^N (x_n-\mu)^2$) Mais la méthode est très générale !

2.2 1 variable à D dimensions

On précise d'abord la structure des données:
N, la taille des données d'apprentissage (le nombre d'exemples)
D, la dimension de chaque point de donnée
$X_{(N,D)}$: les données d'entraînement (par exemple, un ensemble d'images). $\vec{x}_n$ est l'image numéro n, elle correspond à D nombres réels (intensités de gris des pixels)
Θ: les paramètres à optimiser (le détail dépend du modèle choisi).

2.2.1 Dimensions indépendantes

Si les dimensions sont supposées indépendantes, alors c'est facile, tout se factorise. Par exemple, si on a des gaussiennes à plusieurs dimensions, mais avec chaque dimension indépendante, alors on a, pour un tirage $\vec{x}_n \in \mathbb{R}^D$, une densité:

$$\rho(\vec{x}_n) = \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi\sigma_d^2}} \exp\left(-\frac{(x_{nd}-\mu_d)^2}{2\sigma_d^2}\right) \tag{59}$$

Où on a noté x_{nd} la d-ième composante du vecteur $\vec{x}_n$. Quand on prend le log, ce produit là aussi devient une somme, et lorsqu'on cherche la meilleure estimation du paramètre σ_d par exemple, on obtient simplement la même chose qu'en 1D:

$$\sigma_d^2 = \frac{1}{N}\sum_{n}^{N}(x_{nd}-\mu_d)^2 \tag{60}$$

2.2.2 Dimensions corrélées

Quand les dimensions sont corrélées, on doit caractériser leur corrélation.

Une caractérisation de la corrélation entre les différentes dimensions se fait, au premier niveau, par la matrice de covariance. On peut voir l'ensemble des tirages, $X_{(N,D)}$, comme un ensemble de tirages simultanés de lois distinctes: le n-ième tirage correspond au tirage simultané des variables aléatoires $X_{n1}, X_{n2}, \dots X_{nD}$ (Dans le rappel de cours ci dessous, on a X et Y, ici ça correspond par exemple à X_{n1}, X_{n2}).

La matrice de covariance est donc la matrice des covariances des v.a. associées à chaque dimension (on pourrait les appeler les X_d, mais c'est un abus de notation), elle est de taille (D, D). En loi, sa composante d, d' est définie par:

$$K_{d,d'} = Cov[X_d, X_{d'}] = \mathbb{E}[(X_d - \mathbb{E}[X_d])(X_{d'} - \mathbb{E}[X_{d'}])] \tag{61}$$

Son estimateur statistique est:

$$\hat{K}_{d,d'} = \frac{1}{N}\sum_{n}^{N}(x_{n,d} - \overline{X_d}(x_{n,d'} - \overline{X_{d'}})] \tag{62}$$

où $\langle X_d \rangle = \frac{1}{N}\sum_n^N x_{n,d}$ dénote la moyenne empirique (qui se trouve être l'estimateur de l'espérance, ce n'est pas un hasard).

La diagonale de cette matrice sera constituée par la liste des variances de chaque dimension.

2.2.3 Caractérisation des corrélations: la Covariance (rappel du poly de maths de L2)

Lorsque deux v.a. X, Y sont corrélées (non indépendantes), la loi de leur somme, $Z = X + Y$, n'est pas déterminée de façon simple à partir des lois de chacune. En particulier, les moments d'ordre k de Z ne sont pas toujours égaux à la somme des moments d'ordre k de X et Y.
Le moment d'ordre zéro est toujours égal à 1, c'est la normalisation: $\mathbb{E}[Z^0] = 1 \neq \mathbb{E}[X^0] + \mathbb{E}[Y^0] = 1 + 1$.
Le premier moment, lui, est linéaire: $\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y]$ (que X, Y soient indépendantes ou pas !).
Le second moment de $Z = X + Y$ est **une première façon de caractériser la corrélation** entre X et Y.
On définit ainsi la **covariance** de X, Y:

$$Cov(X,Y) = \mathbb{E}\Big[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])\Big] \tag{63}$$

Moralement, la covariance correspond au produit de X et Y: on peut retenir que, en gros, $Cov(X,Y) \approx \mathbb{E}[XY]$. En gros seulement, car il faut d'abord enlever le premier moment pour que cette égalité fonctionne réellement. On a soustrait les espérances de chaque v.a. afin d'enlever la partie "triviale" (constante) des deux variables. Et pourquoi vouloir calculer le produit ? Parce que c'est l'opération la plus simple, qui soit tout de même non triviale (or l'addition est triviale, puisque le premier moment est linéaire, lui).

Cet aspect générique est également apparent par le fait que la covariance apparait dans $V[Z] = V[X+Y]$, soit le deuxième moment de la somme.

$$V[X+Y] = V[X] + V[Y] + 2\ Cov(X,Y) \tag{64}$$

Propriétés:

- $X \perp\!\!\!\perp Y \implies Cov(X,Y) = 0$
- $Cov(X,Y) = 0 \not\implies X \perp\!\!\!\perp Y$
- $Cov(X,X) = V[X]$
- $Cov(X,Y) = Cov(Y,X)$
- $Cov(aX,Y) = aCov(X,Y)$
- $Cov(X+a,Y) = Cov(X,Y)$
- $Cov(X+Y,W) = Cov(X,W) + Cov(Y,W)$

Coefficient de corrélation: Si le second moment de X est fini, et le second moment de Y aussi ($V[X] < \infty, V[Y] < \infty$), le **coefficient de corrélation** de X, Y est :

$$Corr(X,Y) = \frac{Cov(X,Y)}{\sigma[X]\sigma[Y]} \tag{65}$$

On remarque que même si X, Y ont chacun des unités (par exemple des cm et des kg), le coefficient de corrélation est **adimensionné** (il n'a pas d'unités).

Le coeff. de corrélation varie entre -1 (anti corrélation complète, on peut essayer de prouver que $X = -Y$ en loi) et 1 (corrélation complète, on peut essayer de prouver que $X = Y$ en loi). Mais attention, si il vaut 0, cela ne signifie pas nécéssairement que $X \perp\!\!\!\perp Y$!

2.2.4 Cas notable: Gaussienne multi-variée

Le cas de la Gaussienne multi-variée est un cas de variable aléatoire dotée d'une covariance fixée qui revêt un intérêt particulier, car dans ce cas, la covariance caractérise parfaitement les corrélations. Sa densité s'écrit:

$$f_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\boldsymbol{x}) = \frac{1}{(2\pi)^{N/2}\left|\boldsymbol{\Sigma}\right|^{1/2}}\ \exp\left[-\frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{\mu}\right)^{\top}\boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}\right)\right] \tag{66}$$

Ici, pour changer, on a adopté la notation plus mathématicienne/informaticienne (moins explicite, plus compacte): les vecteurs sont en gras, les matrices aussi. Ici $\boldsymbol{\Sigma}$ est la matrice de covariance de la variable aléatoire associée aux $\boldsymbol{x}$, elle est de dimension (D, D). On note qu'on a besoin de $\boldsymbol{\Sigma}^{-1}$, il faut donc qu'elle soit inversible, et par ailleurs la notation $|\boldsymbol{\Sigma}| = det\boldsymbol{\Sigma}$ indique le déterminant (qui doit être non nul).

Allez jouer avec le script "exemples-Matrices-de-Covariance+Genereration-de-Gaussiennes" pour voir quelques gaussiennes multi-variées en 2D, déjà. (en gros, il suffit d'utiliser la méthode "X = np.random.multivariate_normal(mu, cov, (N))", où mu est le vecteur des moyennes, et cov la matrice de covariance des données souhaitée.

2.3 K variables à D dimensions

On peut supposer que les données sont issues de différentes sources (des variables aléatoires), c.a.d. qu'elles sont un mélange de différentes lois de probabilités. Ce cas est plus complexe. On peut d'abord traiter le cas, très simple, où chaque source produit des données X, mais qu'on détient en même temps le label y ou k qui permet d'identifier la source. C'est le cas ci dessous.

2.4 K variables à D dimensions, avec label connu: le Modèle Bayésien Naïf

Une fois qu'on a compris l'idée du calcul du MLE (Maximum Likelihood Estimate), on peut en faire un modèle d'apprentissage supervisé, très facilement. C'est un modèle un peu pauvre, mais qui est la brique de base pour des modèles plus complexes, plus expressifs. Il est donc important de bien comprendre cette brique de base.

Par ailleurs, cela fait un bon exercice sur la manipulation d'indices et de sommes ou produits assez simples.

Ici, pour être concret, on choisira un modèle de Bernoulli. On rappelle que la distribution de proba de Bernoulli peut s'écrire: $\mathbb{P}(X=1)=p, \mathbb{P}(X=0)=1-p$, ce qui peut se résumer sous la forme plus dense: $\mathbb{P}(X=x)=p^x(1-p)^{1-x}$. (on utilise le fait que x vaut forcément 0 ou 1, et que $A^0=1, \forall A$).

2.4.1 Définition du problème, et structure des données

On peut imaginer qu'on s'intéresse à de la classification supervisée d'images en noir et blanc.

On précise la structure des données: N, la taille des données d'apprentissage.
D, la dimension des données (= nombre de pixels)
K, le nombre de classes présentes dans les données à classifier
$X_{(N,D)}$: les données d'entraînement (typiquement, un ensemble d'images). $\vec{x}_n$ est l'image numéro n, elle correspond à D nombres réels (intensités de gris de pixels)
$y_{(N)}$: les classes des données ($y[n]$ est la classe de la donnée numéro n (typiquement, à valeurs dans $[1, ..., K]$)
Θ: les paramètres à optimiser (le détail dépend du modèle choisi, mais il y a au moins K paramètres, probablement KD ou plus.

2.4.2 Apprentissage des paramètres du modèle

On souhaite maximiser la vraisemblance de données, pour un modèle choisi, c.a.d. pour une forme de $\mathbb{P}(X=(\vec{x})_n|y=k,\Theta)$ fixée (par exemple, modèle Gaussien ou de Bernoulli, ou une autre forme explicite de la relation entre $\mathbb{P}(X=(\vec{x})_n|y=k,\Theta)$ et les variables $x_{n,d}, y, \Theta_{k,d}$. Intuitivement, le terme de vraisemblance peut être remplacé par crédibilité, degré de réalisme: on se demande: "quels sont les paramètres Θ qui font qu'observer telle ou telle donnée (image) est crédible/réaliste ?". C'est donc assez raisonnable de maximiser cette chose là.

Maximiser la vraisemblance de données revient à apprendre les valeurs des paramètres Θ telles que $\mathcal{L}(\Theta)=\mathbb{P}(X=(\vec{x})_n|y=k,\Theta)$ soit maximale. On peut appeler Θ^* la solution de ce problème d'optimisation:

$$\Theta^* = argmax_\Theta(\mathcal{L}(\Theta)) \tag{67}$$

On développe cette expression, dans le but d'obtenir une forme explicite dont la dérivée ne soit pas trop dure à calculer, et par indépendance des variables aléatoires $X_n, n \in [\![1, N]\!]$, on a:

$$\begin{aligned}
\Theta^* &= argmax_\Theta\left(\log(\mathcal{L}(\Theta))\right) && (68)\\
&= argmax_\Theta\left(\log \mathbb{P}(X=(\vec{x})_n|y_{(n)},\Theta)\right) && (69)\\
&= argmax_\Theta\left(\log \prod_n^N \mathbb{P}(X_n=\vec{x}_n|y_n=k,\Theta)\right) && (70)\\
&= argmax_\Theta\left(\sum_n^N \log \mathbb{P}(X_n=\vec{x}_n|y_n=k,\Theta)\right) && (71)
\end{aligned}$$

À ce stade on est bien obligés de choisir une forme explicite pour l'expression un peu abstraite $\mathbb{P}(X_n=\vec{x}_n|y=k,\Theta)$. C'est ce qu'on appelle un *modèle* des données.

Les modèles Bayésiens **naïfs** correspondent à supposer que les features d'entrée sont **indépendantes** ! C'est une énorme simplifcation ! Dans la vraie vie, dans les données que l'ont rencontre typiquement, il y a des corrélations entre les features (c'est ce qu'on voit par exemple avec la PCA). Concrètement, l'indépendance des features signifie que pour chaque sample $\vec{x}_n$ (point de donnée), chaque valeur $(x_n)_d$ est indépendante des autres (des autres $(x_n)_{d'}$, c.a.d. des autres pixels de la même image). On écrit donc:

$$\mathbb{P}(X_n=\vec{x}_n|y_n=k,\Theta)=\prod_d^D \mathbb{P}(X_{n,d}=x_{n,d}|y_n=k,\Theta) \tag{72}$$

Il y a ici un petit abus de notation: on écrit $X_{n,d} = x_{n,d}$, le terme de gauche est la variable aléatoire $X_{n,d}$, le terme de droite est la valeur prise dans la réalisation de cette v.a., que l'on nomme aussi (c'est un léger abus), $x_{n,d}$.

On note $\mathbb{1}_{y_n=k}$ la fonction qui vaut 1 quand $y_n = k$ et 0 le reste du temps. C'est un moyen rapide d'imposer $k = y_n$ partout dans l'équation.

Dans le cas où le modèle de la distribution des features d'entrée est choisi Bernoulli, pour chaque feature, cela s'écrit, plus concrètement: $\mathbb{P}(X_n = \vec{x}_n | y_n = k, \Theta) = \prod_d^D p_{k,d}^{x_{n,d}} (1 - p_{k,d})^{(1-x_{n,d})} \mathbb{1}_{y_n=k}$. Ici les paramètres Θ sont, concrètement, les $p_{k,d}$.

$$\Theta^* = argmax_\Theta \left(\sum_n^N \log \Big(\prod_d^D p_{k,d}^{x_{n,d}} (1 - p_{k,d})^{(1-x_{n,d})} \Big) \mathbb{1}_{y_n=k} \right) \tag{73}$$

$$= argmax_\Theta \left(\sum_n^N \sum_d^D \log \Big(p_{k,d}^{x_{n,d}} (1 - p_{k,d})^{(1-x_{n,d})} \Big) \mathbb{1}_{y_n=k} \right) \tag{74}$$

$$= argmax_\Theta \left(\sum_n^N \sum_d^D \left[x_{n,d} \log(p_{k,d}) + (1 - x_{n,d}) \log(1 - p_{k,d}) \right] \mathbb{1}_{y_n=k} \right) \tag{75}$$

Pour trouver le maximum de cette fonction en Θ, on dérive par rapport à Θ (ici, concrètement, $p_{k,d}$) et on cherche la valeur de $p_{k,d}$ telle que la dérivée soit nulle:

$$0 = \frac{\partial \log \mathcal{L}(\Theta)}{\partial p_{k,d}} \tag{76}$$

$$= \frac{\partial}{\partial p_{k,d}} \Big(\sum_n^N \sum_{d'=1}^D \left[x_{n,d'} \log(p_{k,d'}) + (1 - x_{n,d'}) \log(1 - p_{k,d'}) \right] \mathbb{1}_{y_n=k} \Big) \tag{77}$$

Or, la dérivée s'annule pour tout $d' \neq d$, donc:

$$0 = \sum_n^N \left[\frac{x_{n,d}}{p_{k,d}} - \frac{1 - x_{n,d}}{1 - p_{k,d}} \right] \mathbb{1}_{y_n=k} + \sum_{d' \neq d}^D (0) \tag{78}$$

$$0 = \sum_n^N \left[x_{n,d}(1 - p_{k,d}) - (1 - x_{n,d}) p_{k,d} \right] \mathbb{1}_{y_n=k} \tag{79}$$

$$= \sum_n^N \left[x_{n,d} - p_{k,d} \right] \mathbb{1}_{y_n=k} \tag{80}$$

Ce qui donne, en distribuant la somme:

$$\sum_n^N p_{k,d} \mathbb{1}_{y_n=k} = \sum_n^N x_{n,d} \mathbb{1}_{y_n=k} \tag{81}$$

$$p_{k,d} \sum_n^N \mathbb{1}_{y_n=k} = \sum_n^N x_{n,d} \mathbb{1}_{y_n=k} \tag{82}$$

$$p_{k,d} = \frac{\sum_n^N x_{n,d} \mathbb{1}_{y_n=k}}{\sum_n^N \mathbb{1}_{y_n=k}} \tag{83}$$

Ce qui correspond à faire la moyenne des pixels numéro d des images de classe k: en effet le dénominateur est simplement une façon de compter le nombre d'images dans la classe k.

On peut remarquer que dans ce cas présent, on a pu faire les calculs exactement jusqu'au bout, et donc l'apprentissage se fait d'un coup, il n'y a pas d'itérations.

On peut aussi remarquer que le résultat final est extrêmement simple: on fait simplement la moyenne empirique du pixel numéro d de toutes les images étant de la classe k, et cela nous donne le paramètre p_{kd} associé à ce pixel. Dans le cas où il y aurait des corrélations entre les pixels, modélisées par des lois choisies, le calcul peut vite devenir beaucoup plus complexe, voire infaisable analytiquement.

2.4.3 Fonction de décision

Dans cette partie, on va utiliser la formule de Bayes, dans une forme un peu généralisée:

$$\mathbb{P}(A|B,C)\mathbb{P}(B|C) = \mathbb{P}(A,B|C) \tag{84}$$

Pour un nouveau point de donnée (données de validation ou test), on connaît $\vec{x}$ et on souhaite prédire y. On connaît désormais les paramètres Θ. On choisit la classe qui maximise la vraisemblance des données.

Intuitivement, nous choisissons la classe k comme celle qui correspond le plus probablement à l'image observée $\vec{x}$, en supposant qu'elle soit générée par les paramètres que nous avons appris (le vecteur (p_k), de dimensions D).

Mathématiquement, cela correspond à:

$$\begin{aligned} k^* &= argmax_k(\mathbb{P}(y=k|\vec{x},\Theta)) && (85)\\ &= argmax_k\left(\frac{\mathbb{P}(y=k\cap\vec{x}|\Theta)}{\mathbb{P}(\vec{x}|\Theta)}\right) && (86)\\ &= argmax_k\left(\frac{\mathbb{P}(\vec{x}|y=k,\Theta)\mathbb{P}(y=k|\Theta)}{\mathbb{P}(\vec{x}|\Theta)}\right) && (87)\\ &= argmax_k\mathbb{P}(\vec{x}|y=k,\Theta)\mathbb{P}(y=k|\Theta) && (88)\end{aligned}$$

La dernière ligne s'obtient en remarquant que le dénominateur ne dépend pas de k, donc il n'a pas d'importance pour déterminer l'emplacement du maximum.

Le terme $\mathbb{P}(\vec{x}|y=k,\Theta)$ a déjà été vu plus haut, il dépend du modèle choisi (Bernoulli Naïf, Gaussien Naïf ou autre).

Le terme $\mathbb{P}(y=k|\Theta)$ correspond à la probabilité d'observer une image de classe k, connaissant les paramètres appris (Θ) mais sans prendre connaissance de l'image, $\vec{x}$. C'est à dire en fait la probabilité qu'une image tirée au hasard soit de la classe k, parmi les images de *l'ensemble d'apprentissage*. C'est donc une quantité (K nombre réels) qu'il faut aussi *apprendre* la fréquence d'apparition de la classe k:

$$\pi_k = P_k = \mathbb{P}(y=k|\Theta) = \frac{\sum_n^N \mathbb{1}_{y_n=k}}{\sum_n^N 1} \tag{89}$$

On voit que cet "apprentissage" se fait d'un coup, sans itérations. C'est simplement le calcul de la fréquence de la classe k dans le training set. Cette quantité est souvent notée π_k ou P_k.

Concrètement, on a donc:

$$k^* = argmax_k\mathbb{P}(\vec{x}|y=k,\Theta)\mathbb{P}(y=k|\Theta) = argmax_k\Big(\prod_{d=1}^{D} p_{k,d}^{x_d}(1-p_{k,d})^{(1-x_d)}\frac{\sum_n^N \mathbb{1}_{y_n=k}}{N}\Big) \tag{90}$$

Numériquement, il est préférable de maximiser le log de ce nombre, pour éviter les erreurs d'arrondis. Il faudra cependant penser à prendre garde à ce que aucun des $p_{k,d}$ ne soit nul. Si tel est le cas (et ce sera très probablement le cas, dans MNIST ou ce genre de dataset très simple), on doit corriger ce problème numérique, en remplacant:

$$\begin{aligned} k^* &= \text{argmax}_k\left(\log\prod_{d=1}^{D}\left[p_{k,d}^{x_d}(1-p_{k,d})^{(1-x_d)}\right] + \log\frac{\sum_n^N \mathbb{1}_{y_n=k}}{N}\right) && (91)\\ &= \text{argmax}_k\left(\sum_{d=1}^{D}\left[(x_d)\log(p_{k,d}) + (1-x_d)\log(1-p_{k,d})\right] + \log\pi_k\right) && (92)\end{aligned}$$

par

$$k^* = \text{argmax}_k\left(\sum_{d=1}^{D}\left[(x_d)\log(p_{k,d}+\varepsilon) + (1-x_d)\log(1-p_{k,d}+\varepsilon)\right] + \log\pi_k\right) \tag{93}$$

Avec $\varepsilon = 10^{-8}$ par exemple.

2.4.4 Pre-processing

Ici, on a utilisé un modèle de Bernoulli. Implicitement, cela suppose que les entrées sont à valeur dans l'ensemble $\{0, 1\}$. Si on veut appliquer cet algo sur des données (images en noir et blanc), que faudra-t il faire comme pré-processing ?

2.4.5 Prior

Imaginez qu'on vous donne un ensemble d'apprentissage déséquilbré, où une classe est surreprésentée. Supposez que vous savez que dans les données, en général (donc, dans l'ensemble de test, en particulier), les données sont réparties de manière égale entre toutes les classes. Que pouvez-vous changer dans la fonction de décision, pour en tenir compte ? Il s'agit d'un exemple très simple de *prior*.

2.4.6 Modèle Gaussien Naïf

Exercice: Calculer les équations d'apprentissage pour le modèle gaussien naïf.

2.4.7 Modèle Gaussien, mais pas si naïf

En gros: et si on mettait un modèle avec une matrice de covariance non diagnonale entre les features ?