Régression et classification

Contents

1 Introduction

1.1 Contexte

A l'occasion des trois premières révolutions industrielles, des tâches, auparavant réservées au travail manuel de l'Homme, ont été automatisées. Il semble envisageable d'associer au tournant du 21ème siècle une quatrième révolution portée par l'automatisation de la capacité à prédire, essentielle pour le processus de prise de décision dans les secteurs de l'industrie, du commerce et des services.

Cette transformation se fonde sur des évolutions scientifiques et techniques majeures. Elle est ainsi associée à une discipline, le machine learning ou apprentissage automatique de modèles prédictifs par extrapolation à partir de données générées par des processus physiques, numériques ou biologiques. Ces développements algorithmiques, en particulier la redécouverte des réseaux de neurones profonds, ont révélé sous un nouveau jour leur potentiel autour des années 2010 grâce, d'une part, à la création de jeux de données volumineux dans des domaines variés comme la reconnaissance de la parole, la vision par ordinateur, les données multimedia, le traitement de la langue naturelle, la robotique, les véhicules autonomes... et, grâce d'autre part, à une croissance rapide des capacités de calcul et de stockage aux coûts toujours plus abordables.

Par ailleurs, cette automatisation des prédictions s'accompagne d'un renouveau des formes de jugement dans les processus de prise de décision avec un couplage de plus en plus fin entre d'un côté, des experts humains et de l'autre, des chaînes de traitement automatique des données qui aboutissent sur la mise en production d'algorithmes prédictifs. Pour la réussite de cette intégration, les compétences de l'ingénieur informaticien sont essentielles. Ce dernier, puisqu'il comprend en profondeur le fonctionnement et les limites des algorithmes qu'il déploie, est capable d'en mesurer les risques et les biais pour éclairer le jugement de ceux qui utiliseront ses réalisations logicielles pour prendre des décisions.

Ainsi, ce cours introductif à l'apprentissage automatique a pour objectif d'offrir des connaissances fondamentales et des compétences pratiques qui aideront l'ingénieur à tenir ce rôle essentiel.

1.2 Objectifs

La discipline de l'apprentissage automatique, ou machine learning, élabore des algorithmes et des méthodes pour découvrir des régularités dans des données multidimensionnelles afin, entre autres, d'automatiser la prédiction. Elle peut se subdiviser en trois catégories. D'abord, l'apprentissage non (ou semi) supervisé

qui s'attache à découvrir des structures dans les données non étiquetées à travers des approches comme le clustering, la réduction dimensionnelle, les modèles génératifs... Ensuite, l'apprentissage par renforcement, dans le cadre duquel un agent interagit avec son environnement en adaptant son comportement pour maximiser une fonction de récompense. Enfin, l'apprentissage supervisé, qui fait l'objet de ce module, a quant à lui pour objectif d'apprendre à prédire l'association entre un objet décrit selon plusieurs dimensions et une étiquette.

Par exemple, il peut s'agir d'associer aux quartiers d'une ville le prix médian d'un logement. Dans ce cas, un quartier peut être décrit par la proportion de zones résidentielles, le taux de criminalité, le nombre moyen de pièces par habitat, etc. Ici, nous faisons face à un problème dit de « régression » où la valeur à prédire, autrement l'étiquette associée à chaque observation, est continue. Lorsque la variable à prédire est discrète, il s'agit d'un problème dit de « classification », comme détecter un objet dans une image ou décider si une transaction bancaire risque d'être une fraude. Nous considérerons ces deux catégories de problèmes.

Ainsi, ce cours a pour objectif d'introduire quelques concepts fondamentaux de l'apprentissage supervisé et de montrer leurs interconnexions variées dans le cadre de développements algorithmiques qui permettent d'analyser des jeux de données dans une visée avant tout prédictive. Ainsi, les propositions théoriques mèneront à l'écritures de programmes qui implémentent ou utilisent quelques modèles essentiels de l'apprentissage supervisé.

Pour faciliter l'acquisition des connaissances, le cours est accompagné de notebooks manipulables, rédigés dans le langage de programmation R. Ces mises en pratique systématiques doivent permettre de faire le lien entre des concepts fondamentaux et leur application dans des projets d'analyse de données.

1.3 Premiers concepts

$\mathbf{x}^{(1)} \dots \mathbf{x}^{(P)}$ sont des vecteurs de $\mathbb{R}^N$ associés aux valeurs, aussi appelées étiquettes, $y^{(1)} \dots y^{(P)}$ de $\mathbb{R}$. Nous cherchons une fonction $f(\mathbf{x}) : \mathbb{R}^N \to \mathbb{R}$ qui modélise la relation entre les observations $\mathbf{x}$ et les étiquettes y.

La fonction f peut avoir une forme paramétrique, comme par exemple :

$$f(\mathbf{x}) = a_0 + a_1 x_1 + a_2 x_2 + \dots + a_N x_N$$

Si $P = N + 1$, les paramètres $a_0, a_1, \dots a_N$ sont solution d'un système linéaire :

$$\begin{cases} y^{(1)} & = a_0 + a_1 x_1^{(1)} + a_2 x_2^{(1)} + \dots + a_N x_N^{(1)} \\ y^{(2)} & = a_0 + a_1 x_1^{(2)} + a_2 x_2^{(2)} + \dots + a_N x_N^{(2)} \\ \dots & = \dots \\ y^{(P)} & = a_0 + a_1 x_1^{(P)} + a_2 x_2^{(P)} + \dots + a_N x_N^{(P)} \end{cases}$$

Ce système s'écrit également sous forme matricielle :

$$\begin{pmatrix} 1 & x_1^{(1)} & \dots & x_N^{(1)} \\ 1 & x_1^{(2)} & \dots & x_N^{(2)} \\ \dots & \dots & \dots & \dots \\ 1 & x_1^{(P)} & \dots & x_N^{(P)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_N \end{pmatrix} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(P)} \end{pmatrix}$$

Chaque ligne i de la matrice du terme de gauche de l'égalité ci-dessus est le vecteur ligne $\mathbf{x}^{(i)T}$ avec l'addition d'un premier terme constant qui correspond au paramètre a_0. En nommant cette matrice $\mathbf{X}^T$, le système linéaire ci-dessus s'écrit :

$$\mathbf{X}^T \mathbf{a} = \mathbf{y}$$

Soit le cas particulier où x est un scalaire et f est un polynome de degré N :

$$f(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_N x^N$$

Avec $P = N + 1$ observations et les étiquettes associées $\left(x^{(k)}, y^{(k)}\right)$, les coefficients de ce polynôme sont solution d'un système linéaire :

$$\begin{pmatrix} 1 & x^{(1)} & (x^{(1)})^2 & \dots & (x^{(1)})^N \\ 1 & x^{(2)} & (x^{(2)})^2 & \dots & (x^{(2)})^N \\ \dots & \dots & \dots & \dots & \\ 1 & x^{(P)} & (x^{(P)})^2 & \dots & (x^{(P)})^N \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_N \end{pmatrix} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(P)} \end{pmatrix}$$

La matrice du terme de gauche de l'égalité ci-dessus est traditionnellement appelée "matrice de Vandermonde".

1.4 Interpolation polynomiale sur un jeu de données synthétique

Soit un exemple de fonction non-linéaire :

```
f <- function(x) {exp(x) * cos(2*pi*sin(pi*x))}
```

Cette fonction est utilisée pour générer un jeu de données avec l'ajout d'un bruit gaussien.

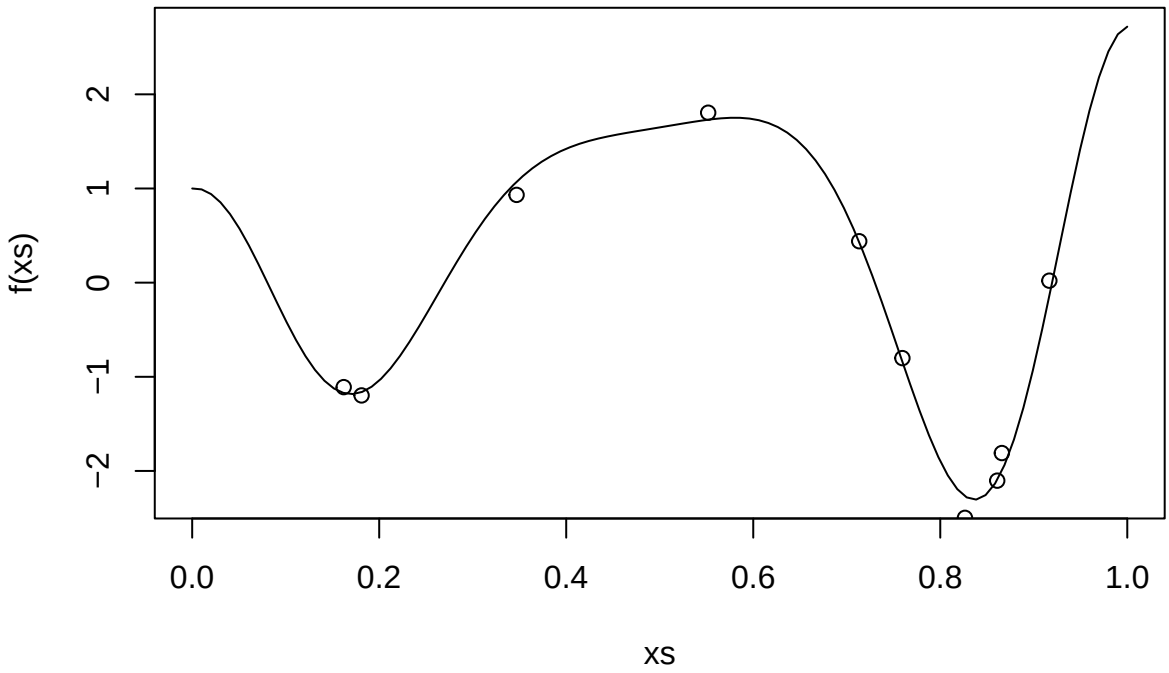
```
gendat <- function(N, sd) {
  # N: nombre d'observations à générer
  # sd: écart type d'un bruit gaussien de moyenne nulle
  X = runif(N)
  Y = f(X) + rnorm(N, mean=0, sd=sd)
  dim(X) <- c(N,1) # en général chaque observation est décrite par plusieurs
                   # variables et X est une matrice avec autant de lignes que
                   # d'observations et autant de colonnes que de variables. Sur
                   # notre exemple, chaque observation n'est décrite que par une
                   # seule variable.
  list(X = X, Y = Y)
}
```

Voici une fonction pour afficher simultanément un jeu de données et la fonction utilisée pour le générer :

```
plt <- function(data, f, ...) {
  xs = seq(0,1,length.out=100)
  plot(xs, f(xs), type="l", ...)
  points(data$X, data$Y)
}
```

Nous affichons un exemple de jeu de données.

```
set.seed(1123)
data = gendat(10,0.2)
plt(data,f)
```



La fonction ci-dessous résout le système linéaire correspondant à la matrice de Vandermonde et retourne les coefficients du polynôme résultat.

```
polyreg1 <- function(data) {
  xs <- c(data$X) # on transforme la matrice X, de dimension Nx1 sur notre
                  # exemple, en un vecteur
  vandermonde = outer(xs, 0:(length(xs)-1), "^")
  solve(vandermonde, data$Y)
}
```

La fonction ci-dessous évalue un polynôme en un point `x` étant donné ses coefficients `coef`.

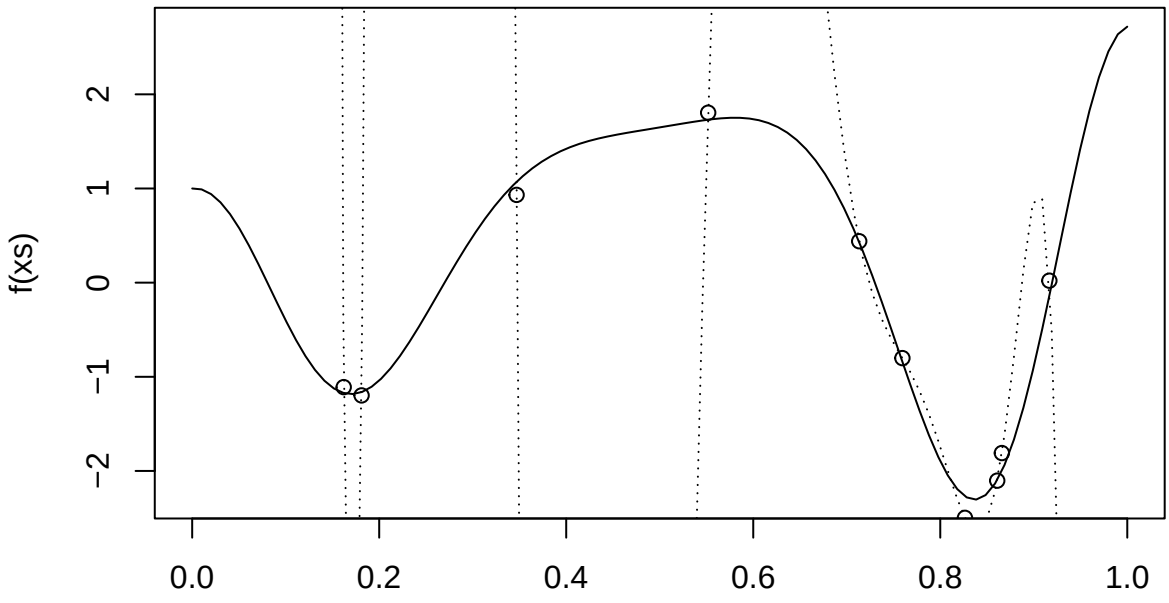
```
polyeval <- function(coef,x) {
  powers = 0:(length(coef)-1)
  f = function(y) { sum(coef * y^powers) }
  sapply(x,f)
}
```

La fonction ci-dessous ajoute au graphe courant le tracé en pointillés d'un polynôme défini par ses coefficients.

```
pltpoly <- function(coef) {
  xs = seq(0,1,length.out=100)
  lines(xs, polyeval(coef,xs), lty="dotted")
}
```

Nous affichons la fonction génératrice, le jeu de donnée et le polynôme qui passe par chaque point du jeu de données.

```
coef = polyreg1(data)
plt(data,f)
pltpoly(coef)
```



Il est improbable que ce polynôme, passant exactement par chaque observation, puisse offrir de bonnes capacités prédictives. Vérifier par exemple que, sur notre exemple synthétique, pour cinq points générés à partir de la fonction f et avec l'ajout d'un bruit gaussien (par exemple d'écart type 0.2), le polynôme découvert, de degré quatre, peut être très éloigné de la fonction génératrice. C'est un exemple du phénomène de sur-apprentissage. Pour limiter ce problème, nous cherchons à découvrir un polynôme de degré plus faible. Il ne passera pas exactement par toutes les observations, mais il prédira probablement mieux les étiquettes associées à de nouvelles observations.

2 Approche des moindres carrés

2.1 Généralisation à un espace de fonctions

Soit un espace vectoriel composé de fonctions. Une base de cet espace est un ensemble de fonctions $(f_1, f_2, \dots f_N)$ tel que toute fonction de l'espace s'exprime comme combinaison linéaire des fonctions de base.

$$f(\mathbf{x}) = a_1 f_1(\mathbf{x}) + a_2 f_2(\mathbf{x}) + \cdots + a_N f_N(\mathbf{x})$$

Pour un jeu de données $\left\{\mathbf{x^{(k)}}, y^{(k)}\right\}_{k=1}^{N}$ de taille N, les coefficients a_i sont solution d'un système linéaire.

$$\begin{pmatrix} f_1(\mathbf{x^{(1)}}) & f_2(\mathbf{x^{(1)}}) & \dots & f_N(\mathbf{x^{(1)}}) \\ f_1(\mathbf{x^{(2)}}) & f_2(\mathbf{x^{(2)}}) & \dots & f_N(\mathbf{x^{(2)}}) \\ \dots & \dots & \dots & \dots \\ f_1(\mathbf{x^{(N)}}) & f_2(\mathbf{x^{(N)}}) & \dots & f_N(\mathbf{x^{(N)}}) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_N \end{pmatrix} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(N)} \end{pmatrix}$$

Nous notons ce système linéaire $\mathbf{Ax} = \mathbf{b}$.

2.2 Expression matricielle

Le système linéaire $\mathbf{Ax} = \mathbf{b}$ avec $\mathbf{A} \in \mathbb{R}^{M \times N}$ n'a pas de solution quand le nombre d'observations dépasse le nombre de fonctions de base (c'est-à-dire, $M > N$). Une approche possible est alors de chercher une approximation $\mathbf{Ax} \approx \mathbf{b}$ qui minimise la somme des carrés des erreurs : $\|\mathbf{Ax} - \mathbf{b}\|_2^2$.

$$
\begin{aligned}
& \|\mathbf{Ax}-\mathbf{b}\|_2^2 \\
= & \{\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}\cdot\mathbf{x}}\} \\
& (\mathbf{Ax}-\mathbf{b})\cdot(\mathbf{Ax}-\mathbf{b}) \\
= & \{\text{Par définition du produit scalaire euclidien}\} \\
& (\mathbf{Ax}-\mathbf{b})^T(\mathbf{Ax}-\mathbf{b}) \\
= & \{\text{propriété de la transposition}\} \\
& \left(\mathbf{x}^T\mathbf{A}^T-\mathbf{b}^T\right)(\mathbf{Ax}-\mathbf{b}) \\
= & \{\text{multiplication}\} \\
& \mathbf{x}^T\mathbf{A}^T\mathbf{Ax}-\mathbf{x}^T\mathbf{A}^T\mathbf{b}-\mathbf{b}^T\mathbf{Ax}+\mathbf{b}^T\mathbf{b} \\
= & \{\mathbf{b}^T\mathbf{Ax} \text{ étant une valeur scalaire, } \mathbf{b}^T\mathbf{Ax} = \left(\mathbf{b}^T\mathbf{Ax}\right)^T = \mathbf{x}^T\mathbf{A}^T\mathbf{b}\} \\
& \mathbf{x}^T\mathbf{A}^T\mathbf{Ax}-2\mathbf{x}^T\mathbf{A}^T\mathbf{b}+\mathbf{b}^T\mathbf{b}
\end{aligned}
$$

Cette dernière expression quadratique en $\mathbf{x}$ correspond à une surface convexe. Donc son minimum peut être calculé en annulant sa dérivée (penser à une courbe $y = a + bx + cx^2$ dont l'unique extremum est atteint lorsque la pente est nulle).

$$
\begin{aligned}
& \mathbf{0} = 2\mathbf{A}^T\mathbf{Ax}-2\mathbf{A}^T\mathbf{b} \\
= & \\
& \mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b}
\end{aligned}
$$

Ainsi, quand $M > N$, la solution approximée $\mathbf{x}$, telle que $\mathbf{Ax} \approx \mathbf{b}$ par minimisation de la somme des carrés des erreurs, est la solution du système linéaire suivant où $\mathbf{A}^T\mathbf{A}$ est appelée la matrice de Gram.

$$\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b}$$

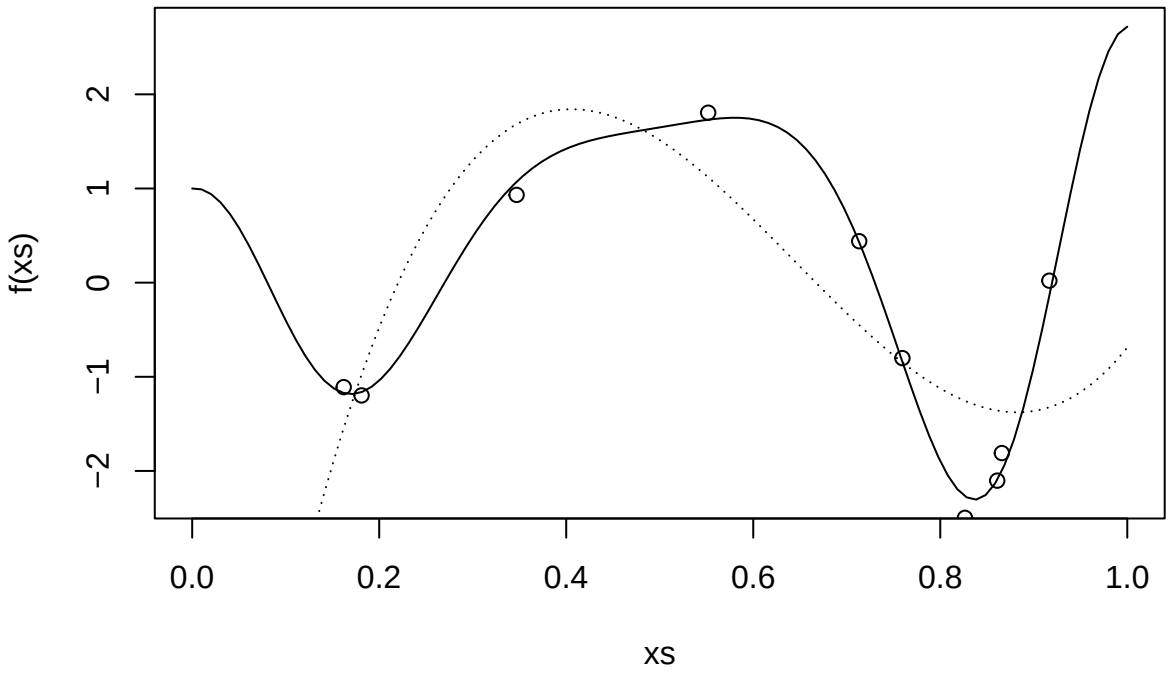
2.3 Approche des moindres carrés appliquée à la régression polynomiale

Pour un polynôme de degré $N-1$, les fonctions de bases mentionnées ci-dessus sont : $f_1(x) = 1$, $f_2(x) = x$, $f_3(x) = x^2$,..., $f_N(x) = x^{N-1}$. Elles permettent de définir la matrice des données $\mathbf{A}$ et la matrice de Gram $\mathbf{A}^T\mathbf{A}$. La fonction ci-dessous résout le système linéaire correspondant à la matrice de Gram pour un polynôme de degré fixé. Elle retourne les coefficients du polynôme résultat.

```
polyreg2 <- function(data, degre) {
  xs <- c(data$X)
  A = outer(xs, 0:degre, "^")
  gram = t(A) %*% A
  solve(gram, as.vector(t(A) %*% data$Y))
}
```

Sur notre exemple synthétique, nous affichons la fonction génératrice, le jeu de donnée et le "meilleur" polynôme de degré 3.

```
coef = polyreg2(data,3)
plt(data,f)
pltpoly(coef)
```



Ce polynôme de degré trois modélise mieux la fonction génératrice inconnue que celui de degré quatre qui ne commettait aucune erreur sur les données observées.

3 Régularisation de Tikhonov

3.1 Problèmes linéaires mal posés

Avec moins d'observations que de fonctions de base ($M < N$), le système $\mathbf{Ax} = \mathbf{b}$ ne possède pas de solution unique. Même quand $M \geq N$, le système linéaire peut posséder une solution approchée préférable à la solution optimale. C'est en particulier vrai quand plusieurs observations sont très proches à un coefficient multiplicatif près (on parle de colinéarités). Par exemple, soit le système linéaire suivant :

$$\begin{pmatrix} 1 & 1 \\ 1 & 1.00001 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0.99 \end{pmatrix}$$

Sa solution est $\mathbf{x}^T = (1001, -1000)$. Cependant, la solution approchée $\mathbf{x}^T = (0.5, 0.5)$ semble préférable. En effet, la solution optimale a peu de chance de bien s'adapter à de nouvelles observations (par exemple, l'observation $(1, 2)$ serait projetée sur l'étiquette -999).

3.2 Ajout de contraintes de régularité

Ainsi, lorsqu'il faut choisir entre plusieurs solutions, il peut être efficace d'exprimer une préférence envers celles dont les coefficients (ou paramètres) ont de faibles valeurs. Cela consiste par exemple à minimiser $|x_1| + |x_2| + \ldots$ (aussi noté $\|\mathbf{x}\|_1$, la "norme 1") ou encore $x_1^2 + x_2^2 + \ldots$ (aussi noté $\|\mathbf{x}\|_2^2$, le carré de la "norme 2"). Dans ce dernier cas, il s'agit de résoudre un nouveau problème de minimisation :

$$\begin{aligned} \min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_2^2 + \alpha \|\mathbf{x}|_2^2 \\ avec \quad 0 \leq \alpha \end{aligned}$$

C'est encore un problème de minimisation quadratique en $\mathbf{x}$ dont le minimum se découvre par annulation de la dérivée.

$$
\begin{aligned}
\mathbf{0} &= 2\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{A}^T\mathbf{b} + 2\alpha\mathbf{x} \\
&= \\
\left(\mathbf{A}^T\mathbf{A} + \alpha\mathbf{I}_{n\times n}\right)\mathbf{x} &= \mathbf{A}^T\mathbf{b}
\end{aligned}
$$

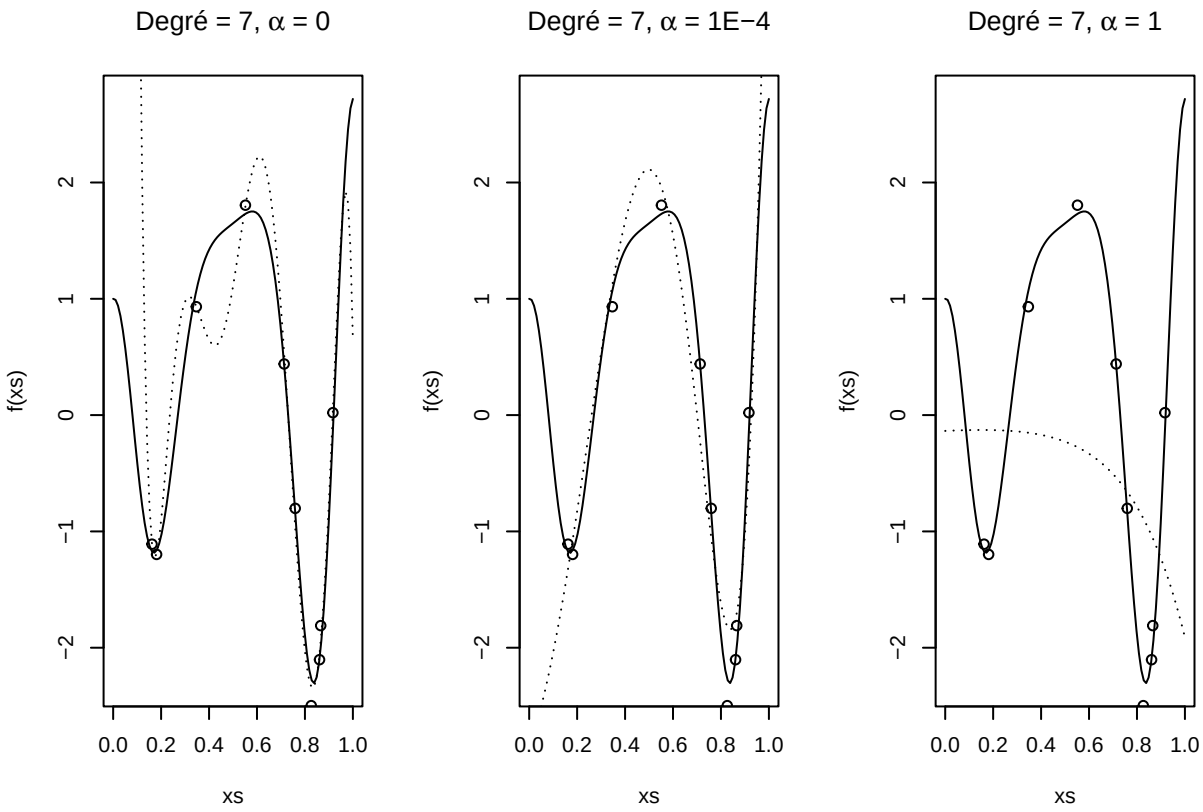
En pratique, il s'agit donc d'ajouter une petite valeur positive α aux éléments de la diagonale de la matrice de Gram. Cette approche porte plusieurs noms dont "régularisation de Tikhonov" ou "régression Ridge".

```
ridge <- function(alpha, data, degre) {
  xs <- c(data$X)
  A <- outer(xs, 0:degre, "^")
  gram <- t(A) %*% A
  diag(gram) <- diag(gram) + alpha
  solve(gram, as.vector(t(A) %*% data$Y))
}
```

3.3 Visulation de l'effet sur les paramètres de différents niveaux de régularisation

Sur notre exemple synthétique, nous affichons la fonction génératrice, le jeu de donnée et le polynôme de degré au plus sept découvert par régression ridge avec une valeur de α égale soit à 0, soit à 10^{-4}, soit à 1.

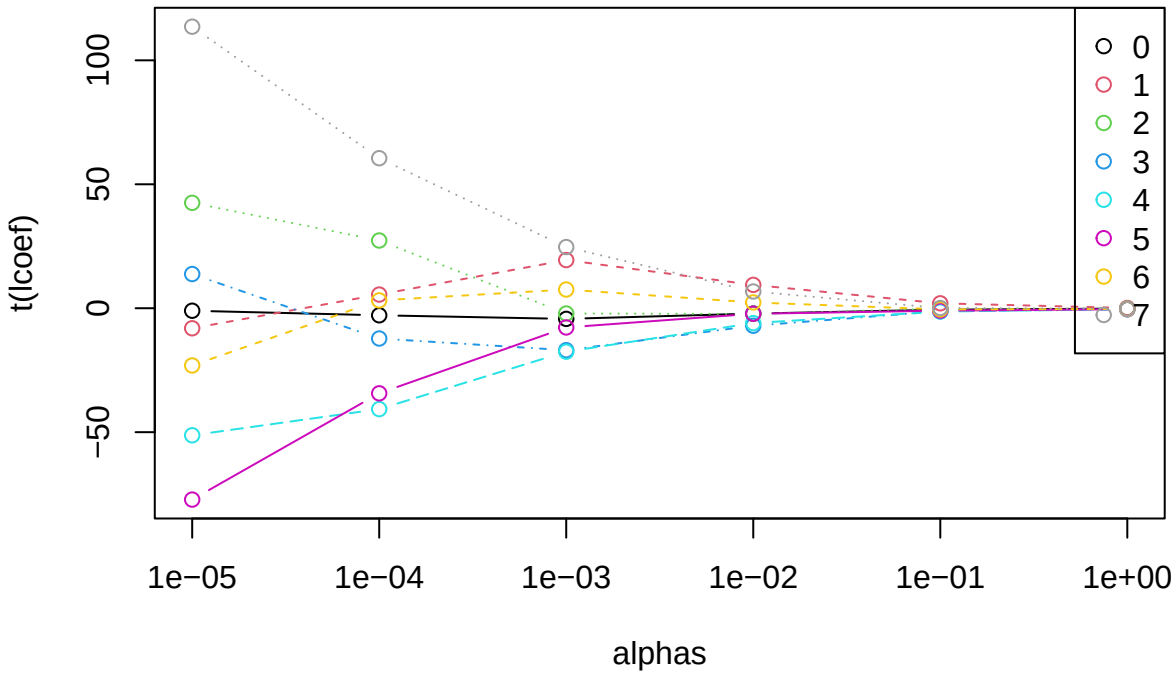
```
par(mfrow=c(1,3))
coef <- ridge(0, data, 7)
plt(data,f,main=expression(paste(plain("Degré = "), 7, plain(", "), alpha,
                                 plain(" = 0"))))
pltpoly(coef)
coef <- ridge(1E-4, data, 7)
plt(data,f,main=expression(paste(plain("Degré = "), 7, plain(", "), alpha,
                                 plain(" = 1E-4"))))
pltpoly(coef)
coef <- ridge(1, data, 7)
plt(data,f,main=expression(paste(plain("Degré = "), 7, plain(", "), alpha,
                                 plain(" = 1"))))
pltpoly(coef)
```



Plus le coefficient de régularisation α est faible, moins il contraint les paramètres (c'est-à-dire, les coefficients du polynômes) à conserver de petites valeurs et plus le polynôme découvert peut être complexe, au risque de provoquer du sur-apprentissage et donc de limiter la capacité du modèle à bien prédire les étiquettes de nouvelles observations. A l'inverse, plus le coefficient de régularisation est élevé, plus le modèle découvert sera simple, au risque de sous-apprendre en ne modélisant pas convenablement les variations propres au processus qui a généré les observations.

Pour mieux visualiser l'effet du coefficient de régularisation ridge, nous affichons les valeurs des coefficients du polynôme découvert pour différentes valeurs de α. Plus α augmente, plus les coefficients du polynôme diminuent et tendent vers 0.

```
alphas <- c(1E-5, 1E-4, 1E-3, 1E-2, 1E-1, 1)
lcoef <- sapply(alphas, ridge, data, 7)
matplot(alphas, t(lcoef), type=c("b"), pch=1, col=1:8, log="x")
legend("topright", legend = 0:7, col=1:8, pch=1)
```



4 Validation croisée

4.1 Principe de la validation croisée

Comment choisir la valeur du coefficient de régularisation α pour une régression ridge ?

Une possibilité est de diviser le jeu de données en deux parties, l'une utilisée pour apprendre le modèle prédictif, l'autre utilisée pour valider la qualité des prédictions sur des données qui n'ont pas été vues pendant la phase d'apprentissage. On parle de jeu d'entraînement et de jeu de validation ou de test. Cette méthode est appelée "validation croisée".

L'idée est de tester plusieurs valeurs de l'hyperparamètre α sur le jeu d'entraînement et de conserver celle qui donne les meilleurs résultats sur le jeu de test.

4.2 Découpage du jeu de données en jeu d'entrainement et jeu de validation

Nous écrivons une fonction qui permet de diviser le jeu de données en un jeu d'entraînement et un jeu de validation. Nous lui passons en paramètre le jeu de données initial et la proportion des données conservées pour l'entraînement.

```
xvalpart <- function(data,p) {
  n <- nrow(data$X)
  nentr <- round(p*n)
  entridx <- sample(1:n, nentr, replace=FALSE)
  list(entr  = list(X = data$X[entridx,,drop=FALSE],  Y = data$Y[entridx]),
       valid = list(X = data$X[-entridx,,drop=FALSE], Y = data$Y[-entridx]))
}
```

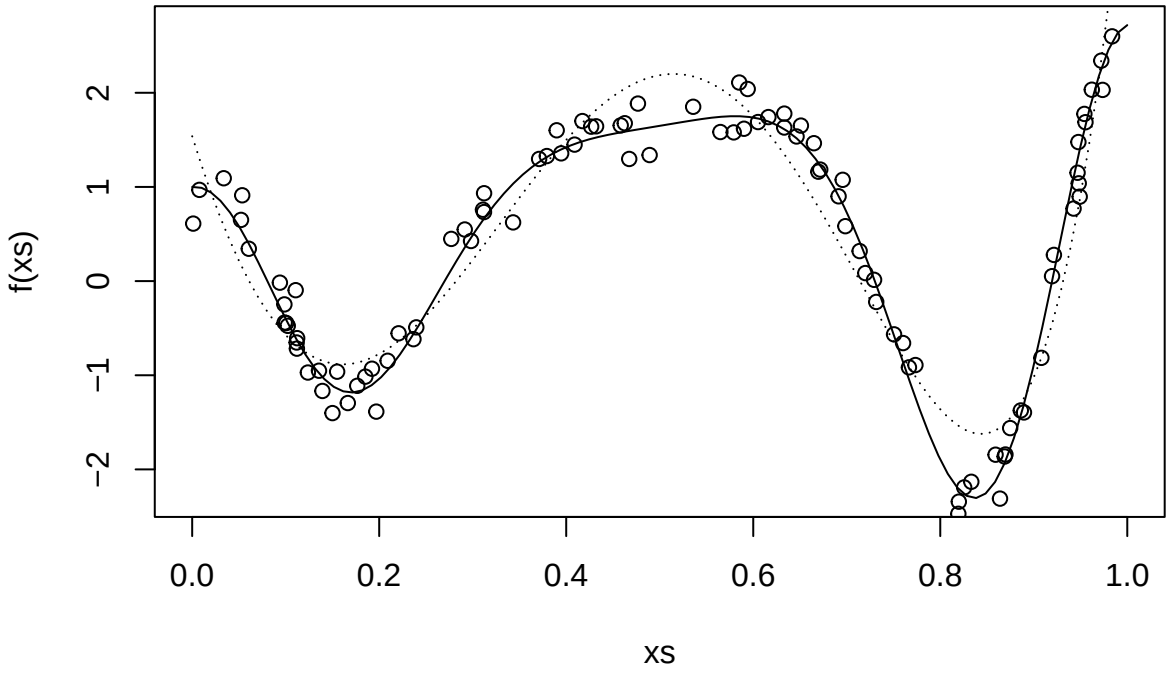
4.3 Application de la validation croisée à la régularisation de Tikhonov

Nous écrivons ensuite une fonction qui apprend plusieurs modèles de régression ridge sur le jeu d'entraînement pour des valeurs différentes de l'hyperparamètre α et qui retourne les coefficients du meilleur modèle avec la moyenne de la valeur absolue des erreurs commises par ce modèle sur le jeu de validation.

```
xvalridge <- function(alphas, data, degre, p) {
  tmp <- xvalpart(data,p)
  lcoef <- sapply(alphas, ridge, tmp$entr, degre)
  pred <- sapply(split(lcoef,col(lcoef)), polyeval, tmp$valid$X)
  meanabserrs <- colMeans(abs(pred - tmp$valid$Y))
  minmeanabserr <- min(meanabserrs)
  minidx <- which(meanabserrs == minmeanabserr)
  list(meanabserr = minmeanabserr, coef = lcoef[,minidx], alpha = alphas[minidx])
}
```

Nous générons un nouveau jeu de données composé de 100 observations et nous calculons par validation croisée un polynôme de degré au plus égal à 5 qui modélise au mieux ces données.

```
data = gendat(100,0.2)
xvalres <- xvalridge(alphas, data, 5, 0.8)
plt(data,f)
pltpoly(xvalres$coef)
```



5 Décomposition en valeurs singulières (SVD)

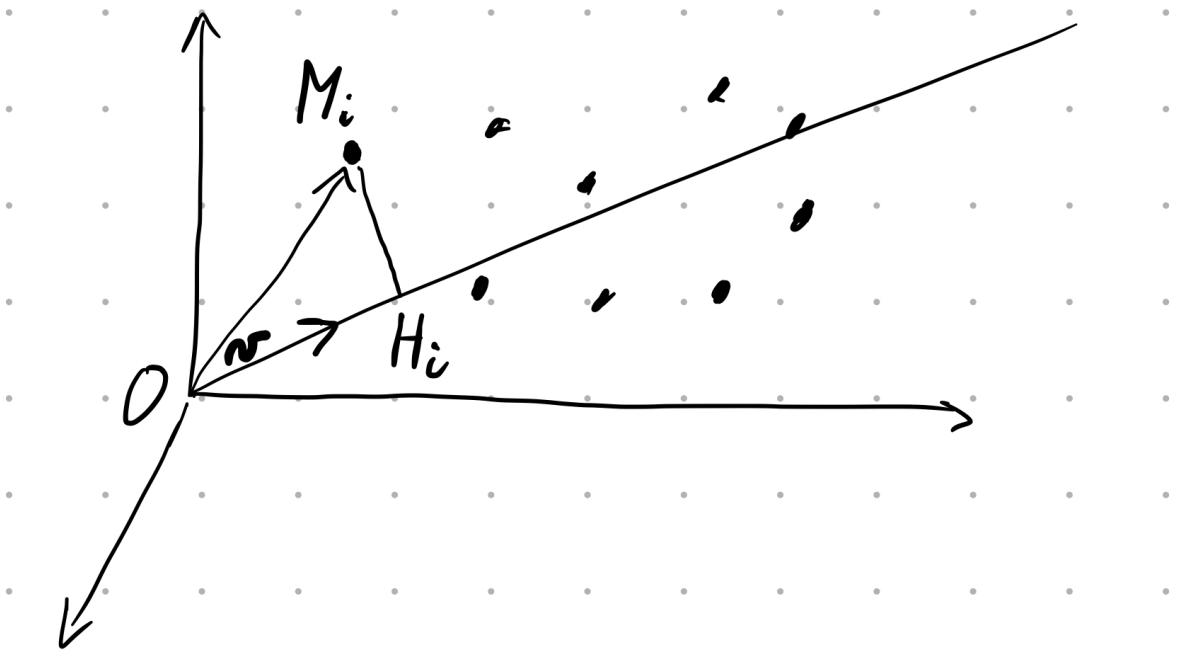
Nous présentons une méthode générique de compression d'un tableau rectangulaire de données. Elle possède de nombreuses applications. En particulier, elle permet de calculer en un seul entraînement les régressions ridge pour toutes les valeurs possibles de l'hyperparamètre α.

5.1 Recherche d'un sous-espace qui minimise les carrés des écarts à l'espace d'origine

Soit un tableau de données $\mathbf{X}$ composé de N observations en lignes, chacune décrite par P variables en colonnes. x_{ij} est la valeur de la variable j pour l'observation i. Les vecteurs lignes forment N points de l'espace $\mathbb{R}^P$. Les vecteurs colonnes forment P points de l'espace $\mathbb{R}^N$. Nous cherchons à projeter le nuage des points lignes sur un sous-espace $\mathcal{H} \subset \mathbb{R}^P$, tout en minimisant les déformations.

Pour commencer, nous considèrons le meilleur sous-espace $\mathcal{H}$ à une dimension, c'est-à-dire une droite définie

par son vecteur directeur unitaire $\mathbf{v}$ ("unitaire" signifie que sa norme est égale à 1, soit $\mathbf{v}^T\mathbf{v} = 1$). Soit M_i un des N points de $\mathbb{R}^P$. A ce point correspond le vecteur $\mathbf{OM_i}$ aussi noté $\mathbf{x_i}$ car ses coordonnées sont données par la i-ème ligne de $\mathbf{X}$. Soit H_i la projection de M_i sur la droite $\mathcal{H}$.



La longueur OH_i est le produit scalaire de $\mathbf{x_i}$ et de $\mathbf{v}$.

$$OH_i = \mathbf{x_i}\mathbf{v} = \sum_{j=1}^{P} x_{ij} v_j$$

Nous obtenons un vecteur des N projections OH_i par le produit de la matrice $\mathbf{X}$ et du vecteur $\mathbf{v}$: $\mathbf{Xv}$. En choisissant pour $\mathcal{H}$ le sous-espace qui minimise la somme des carrés des écarts, il faut minimiser $\sum_i M_i H_i^2$.

Le théorème de Pythagore donne $\sum_i M_i H_i^2 = \sum_i OM_i^2 - \sum_i OH_i^2$.

Puisque $\sum_i OM_i^2$ est indépendant de $\mathbf{v}$, nous devons maximiser

$$\sum OH_i^2 = (\mathbf{Xv})^T(\mathbf{Xv}) = \mathbf{v}^T\mathbf{X}^T\mathbf{Xv} \tag{1}$$

... sous la contrainte $\mathbf{v}^T\mathbf{v} = 1$.

Nous notons $\mathbf{v_1}$ ce vecteur directeur du meilleur sous-espace de dimension 1 pour le nuage des N observations de $\mathcal{R}^P$. Le meilleur sous-espace de dimension 2 contient $\mathbf{v_1}$ et il faut le compléter avec un second vecteur unitaire $\mathbf{v_2}$, orthogonal à $\mathbf{v_1}$ et qui maximise $\mathbf{v_2}^T\mathbf{X}^T\mathbf{Xv_2}$. Etc.

Nous montrerons plus loin que $\mathbf{v_1}$ est le vecteur propre de $\mathbf{X}^T\mathbf{X}$ associé à la plus grande valeur propre λ_1, cette dernière étant justement le maximum de la forme quadratique de l'équation (1). De même, $\mathbf{v_2}$ est le vecteur propre de $\mathbf{X}^T\mathbf{X}$ associé à la seconde plus grande valeur propre λ_2. Etc.

Nous venons de voir que les vecteurs orthogonaux $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}$ forment une base d'un sous-espace k-dimensionnel sur lequel peuvent être projetés les N vecteurs lignes de $\mathbf{X}$ pour minimiser les carrés des écarts à l'espace d'origine qui était de dimension (au plus) $max(N, P)$.

Nous construirons de même des vecteurs orthogonaux $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_k}$ qui forment une base d'un sous-espace k-dimensionnel sur lequel peuvent être projetés les P vecteurs colonnes de $\mathbf{X}$ pour minimiser les carrés des écarts à l'espace d'origine qui était de dimension (au plus) $max(N, P)$.

Nous trouverons de façon similaire que $\mathbf{u_1}$ est le vecteur propre de $\mathbf{X}\mathbf{X}^T$ associé à la plus grande valeur propre λ_1 (cette dernière étant exactement la même que celle associée à $\mathbf{v_1}$).

Nous montrerons ensuite que la matrice $\mathbf{X}$ peut s'écrire comme une somme de matrices de rang 1 qui sont les produits des vecteurs $\mathbf{u}$ et $\mathbf{v}$ (on se place dans la situation où $N > P$ et $\mathbf{X}$ est de rang P). Il s'agit de la *décomposition en valeurs singulières* de $\mathbf{X}$.

$$\mathbf{X} = \sum_{\alpha=1}^{P} \sqrt{\lambda_\alpha} \mathbf{u}_\alpha {\mathbf{v}_\alpha}^T \tag{2}$$

Nous obtenons une reconstitution approchée de rang k de $\mathbf{X}$ en ne conservant dans l'équation (2) que les k plus grandes valeurs singulières $\sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots, \sqrt{\lambda_k}$ et en annulant toutes les autres.

5.2 Illustrations du SVD

5.2.1 Compression d'une image

Comme premier exemple, considérons l'image d'un hortensia en fleurs.

Nous convertissons cette image, originellement au format JPEG, en un format en niveaux de gris, simple à manipuler, appelé PGM (*Portable Grey Map*). Dans ce format non compressé, l'image est représentée par une matrice dont chaque valeur, comprise entre 0 et 1, représente le niveau de gris d'un pixel. Nous opérons cette transformation grâce au programme *imagemagick* avec la commande `convert hortensia.jpg hortensia.pgm`.

Ensuite, nous appliquons la décomposition en valeurs singulières à cette matrice qui représente l'image en niveaux de gris. Nous utilisons le résultat du SVD pour reconstituer une version compressée de l'image en ne conservant qu'un certain nombre des plus grandes valeurs singulières.

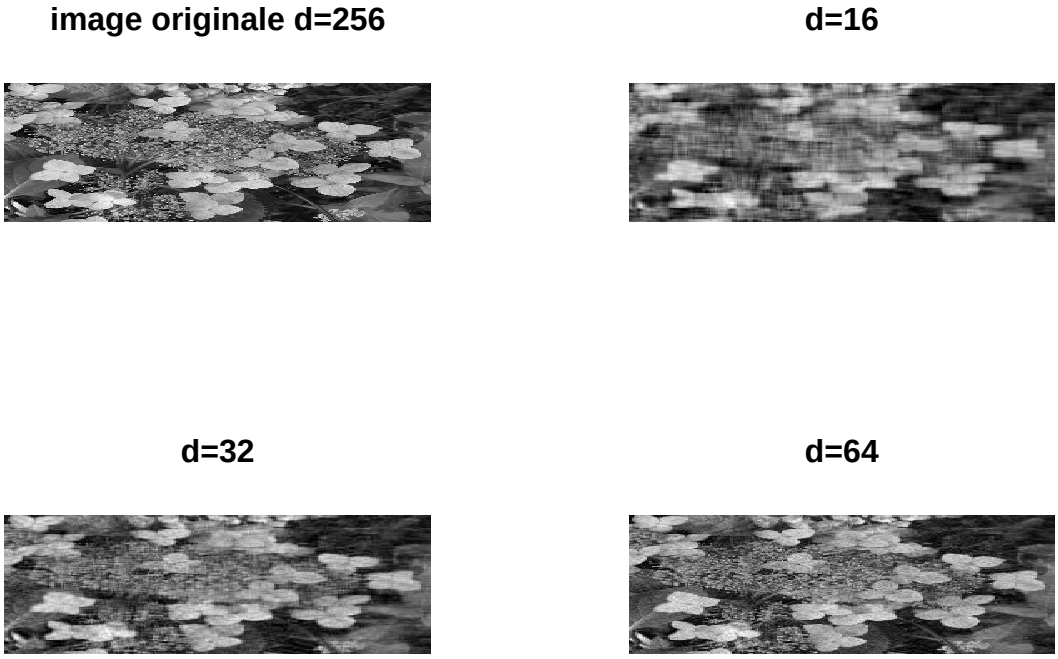
```
library(pixmap) ; # chargement du package pixmap qui peut être installé avec la
                  # commande install.packages("pixmap");
```

```
img = read.pnm("images/hortensia.pgm");
X = img@grey; # chaque entrée de la matrice représente une intensité de gris
              # comprise entre 0 et 1
rm(img);

compress_SVD <- function(X, nd) {
  svdX <- svd(X);
  svdX$u[,1:nd] %*% diag(svdX$d[1:nd]) %*% t(svdX$v[,1:nd]);
}

print_grey_img <- function(Y,...) {
  Y[Y<0]<-0; # après reconstruction approchée à partir du résultat du svd le
  Y[Y>1]<-1; # domaine [0,1] de la matrice initiale peut ne plus être respecté
  image(t(apply(Y,2,rev)), col=grey(seq(0,1,length=256)), axes=FALSE, ...)
}

par(mfrow=c(2,2));
print_grey_img(X, main="image originale d=256");
print_grey_img(compress_SVD(X,16), main="d=16");
print_grey_img(compress_SVD(X,32), main="d=32");
print_grey_img(compress_SVD(X,64), main="d=64");
```



5.3 Métrique définie par une matrice définie non négative

Avant de résoudre ce problème d'optimisation, nous rappelons quelques propriétés des matrices symétriques définies non négatives. Une matrice définie non négative (souvent noté PSD pour "positive semi-definite") $\mathbf{M}$ représente une métrique car elle définit un produit scalaire et donc une géométrie, c'est-à-dire qu'elle donne un sens aux notions de distance et d'angle.

Le produit scalaire défini par $\mathbf{M}$ entre deux vecteurs $\mathbf{x}$ et $\mathbf{y}$ pourra se noter $< \mathbf{x}, \mathbf{y} >_{\mathbf{M}} = \mathbf{x}^T\mathbf{M}\mathbf{y}$. Pour que ce produit scalaire soit valide, $\mathbf{M}$ doit respecter la contrainte $\mathbf{x}^T\mathbf{M}\mathbf{y} \geq 0$ pour tout $\mathbf{x}$ et $\mathbf{y}$.

Une fois donné un produit scalaire, les notions liées de norme et d'angle suivent~: $\|\mathbf{x}\|_{\mathbf{M}} = \sqrt{< \mathbf{x}, \mathbf{x} >_{\mathbf{M}}}$ et

$cos^2(\mathbf{x},\mathbf{y}) =< \mathbf{x},\mathbf{y} >_\mathbf{M} /\|\mathbf{x}\|_\mathbf{M}\|\mathbf{y}\|_\mathbf{M}$.

La géométrie euclidienne "classique" (associée à la base canonique) correspond à l'emploi de la matrice identité pour métrique, $\mathbf{M} = \mathbf{I}$.

Aussi, les valeurs propres d'une matrice PSD sont toutes positives~:

$$
\begin{aligned}
& \mathbf{x}^T\mathbf{M}\mathbf{x} \geq 0 \\
= & \{\lambda \text{ est une valeur propre de } \mathbf{M}\} \\
& \mathbf{x}^T\lambda\mathbf{x} \geq 0 \\
= & \{\text{utilisation de la métrique identité}\} \\
& \lambda\|x\|^2_\mathbf{I} \geq 0 \\
\Rightarrow & \{\|x\|^2_\mathbf{I} \geq 0\} \\
& \lambda \geq 0
\end{aligned}
$$

Soit $\mathbf{V}$ la matrice des vecteurs propres de $\mathbf{M}$ et $\mathbf{L}$ la matrice diagonale de ses valeurs propres positives.

$$
\begin{aligned}
& \mathbf{M}\mathbf{V} = \mathbf{V}\mathbf{L} \\
= & \\
& \mathbf{M} = \mathbf{V}\mathbf{L}\mathbf{V}^{-1} \\
= & \{\mathbf{S} = \sqrt{\mathbf{L}}\} \\
& \mathbf{M} = \mathbf{V}\mathbf{S}\mathbf{S}\mathbf{V}^{-1} \\
= & \{ \quad \text{Les vecteurs propres distincts sont orthogonaux deux à deux : } \mathbf{V}^{-1} = \mathbf{V}^T. \\
& \quad \mathbf{S} = \mathbf{S}^T. \text{On pose } \mathbf{T} = \mathbf{V}\mathbf{S}.\} \\
& \mathbf{M} = \mathbf{T}\mathbf{T}^T
\end{aligned}
$$

Ainsi, toute matrice définie non négative $\mathbf{M}$ peut être décomposée sous la forme $\mathbf{M} = \mathbf{T}\mathbf{T}^T$ où $\mathbf{T}$ est une transformation linéaire composée d'un changement d'échelle des axes du repère ($\mathbf{S}$) et d'une rotation ($\mathbf{V}$).

Nous remarquons par exemple que l'équation d'un cercle selon la métrique $\mathbf{M}$ correspond à celle d'une ellipse selon la métrique identité $\mathbf{I}$:

$$
\begin{aligned}
& \|\mathbf{x}\|^2_\mathbf{M} = c \\
= & \{\text{C'est l'équation d'un cercle, c est une constante.}\} \\
& < \mathbf{x},\mathbf{x} >_\mathbf{M} = c \\
= & \{\text{Par définition du produit scalaire.}\} \\
& \mathbf{x}^T\mathbf{M}\mathbf{x} = c \\
= & \{\mathbf{M} = \mathbf{T}^T\mathbf{T}\} \\
& \mathbf{x}^T\mathbf{T}^T\mathbf{T}\mathbf{x} = c \\
= & \{\text{Par définition du produit scalaire, on retrouve le cercle déformé en une ellipse par } \mathbf{T}\} \\
& \|\mathbf{T}\mathbf{x}\|^2_\mathbf{I} = c
\end{aligned}
$$

Ainsi, une matrice symétrique définie non négative projette le cercle unité sur une ellipse dont les axes orthonormaux sont les vecteurs propres et les longueurs des axes sont les valeurs propres.

Enfin, l'angle formé entre $\mathbf{x}$ et $\mathbf{Mx}$ est inférieur ou égal à $\pi/2$ radians :

$$
\begin{aligned}
& \mathbf{x}^T\mathbf{Mx} \geq 0 \\
= & \{\text{Géométrie du produit scalaire. On note } \theta \text{ l'angle entre les deux vecteurs.}\} \\
& \|\mathbf{x}\|_\mathbf{I} \, \|\mathbf{Mx}\|_\mathbf{I} \, cos(\theta) \geq 0 \\
\Rightarrow & \\
& |\theta| \leq \pi/2
\end{aligned}
$$

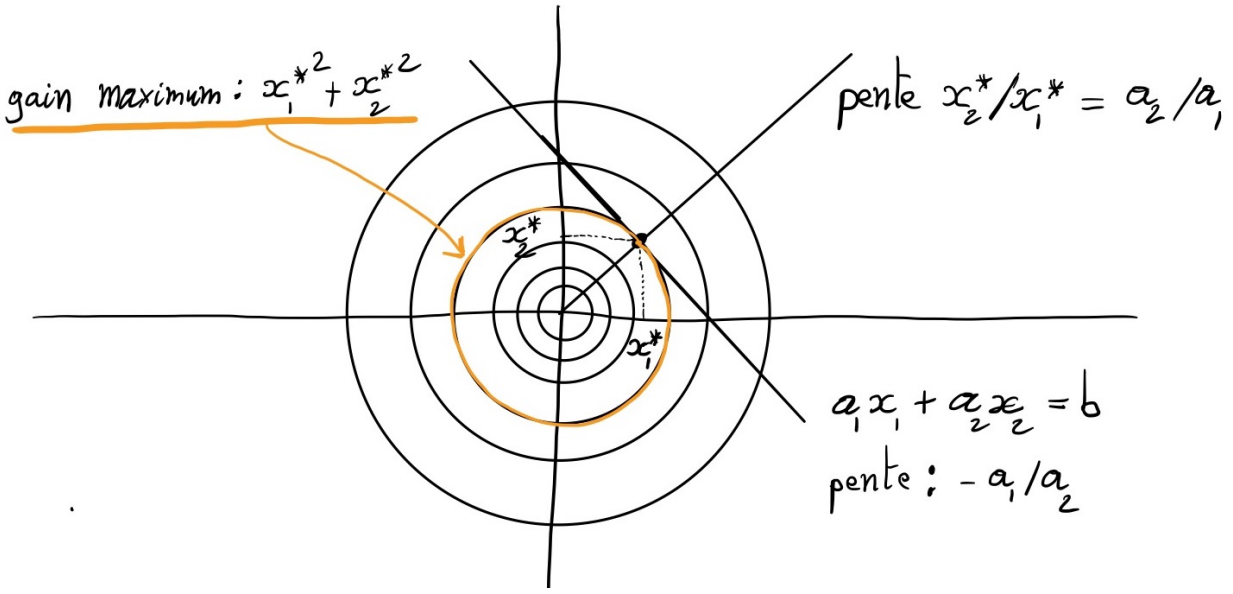
Ainsi, le vecteur résultat $\mathbf{Mx}$ est contraint de rester du même côté que $\mathbf{x}$ de l'hyperplan perpendiculaire à $\mathbf{x}$ qui sépare l'espace en deux. Nous comprenons que le concept de matrice symétrique définie non négative est une généralisation pour un espace multidimensionnel du concept de nombre positif sur la droite réelle.

5.4 Optimisation sous contraintes par la méthode des multiplicateurs de Lagrange

5.4.1 Présentation de la méthode sur un exemple

Avant de pouvoir présenter la preuve de la décomposition en valeurs singulières, nous devons introduire une stratégie souvent très utile pour résoudre un problème d'optimisation sous contraintes : la méthode dite des multiplicateurs de Lagrange.

Prenons un exemple simple à deux dimensions. Nous cherchons à maximiser $F(\mathbf{x}) = x_1^2 + x_2^2$ sous la contrainte $K(\mathbf{x}) = b$, avec $K(\mathbf{x}) = a_1x_1 + a_2x_2$. C'est-à-dire que le point solution $\mathbf{x}^*$ doit être situé sur la ligne K.



Sur cet exemple, les lignes de niveaux, ou contours, de F sont des cercles centrés sur l'origine du repère cartésien. La solution $\mathbf{x}*$ appartient au contour de F tangent à la contrainte K. Il faut donc que leurs gradients soient alignés : $\nabla F = \lambda \nabla K$.

$$
\nabla F = \begin{pmatrix} \partial F/\partial x_1 \\ \partial F/\partial x_2 \end{pmatrix} = \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix} \quad ; \quad \nabla K = \begin{pmatrix} \partial K/\partial x_1 \\ \partial K/\partial x_2 \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}
$$

Nous avons donc un système de trois équations à trois inconnues (x_1, x_2 et λ) :

$$\begin{cases} 2x_1 = \lambda a_1 \\ 2x_2 = \lambda a_2 \\ a_1 x_1 + a_2 x_2 = b \end{cases}$$

En résolvant ce système par simples substitutions, nous trouvons la solution :

$$\begin{cases} x_1^* = \frac{a_1 b}{a_1^2 + a_2^2} \\ x_2^* = \frac{a_2 b}{a_1^2 + a_2^2} \\ \lambda^* = \frac{2b}{a_1^2 + a_2^2} \end{cases}$$

Ce même calcul s'écrit d'une façon plus systématique en rassemblant les équations $\nabla F = \lambda \nabla K$ et $K(\mathbf{x}) = b$ par l'introduction du Lagrangien :

$$\mathcal{L}(\mathbf{x}, \lambda) = F(\mathbf{x}) - \lambda(K(\mathbf{x}) - b)$$

En effet, en annulant les dérivées partielles de $\mathcal{L}$, nous retrouvons les équations $\nabla F = \lambda \nabla K$ et $K(\mathbf{x}) = b$:

$$\begin{array}{lcl} \partial\mathcal{L}/\partial x_1 = 0 & \Leftrightarrow & \partial F/\partial x_1 = \lambda \partial K/\partial x_1 \\ \partial\mathcal{L}/\partial x_2 = 0 & \Leftrightarrow & \partial F/\partial x_2 = \lambda \partial K/\partial x_2 \\ \partial\mathcal{L}/\partial \lambda = 0 & \Leftrightarrow & K(\mathbf{x}) = b \end{array}$$

Ainsi, le problème d'optimisation sous contrainte exprimé en fonction de F et K peut s'écrire comme un problème d'optimisation non contraint, en fonction de $\mathcal{L}$, dans un espace de plus grande dimension que celui d'origine puisque s'ajoute aux dimensions de $\mathbf{x}$ celle de λ.

5.4.2 Interprétation des multiplicateurs de Lagrange

Nous allons montrer que la dérivée du gain maximum par rapport au niveau de contrainte b est égale au multiplicateur de Lagrange λ. Autrement dit, avec $M = F(\mathbf{x}^*)$, nous avons $dM/d\lambda = \lambda^*$: un petit incrément db de b produit un incrément $\lambda^* db$ du gain maximum M.

5.5 Dérivation de la décomposition en valeurs singulières

Après ces préambules sur la notion de métrique associée à une matrice définie non négative et sur la méthode d'optimisation dite des multiplicateurs de Lagrange, nous reprenons la dérivation de la décomposition en valeurs singulières.

Rappelons que nous avons noté $\mathbf{v_1}$ le vecteur directeur du meilleur sous-espace de dimension 1 pour le nuage des N observations de $\mathcal{R}^P$. Nous montrons que $\mathbf{v_1}$ est le vecteur propre de $\mathbf{X}^T\mathbf{X}$ associé à la plus grande valeur propre λ_1.

Le résultat est sans difficulté plus général. Nous le prouvons pour toute matrice symétrique $\mathbf{A}$ (non seulement $\mathbf{X}^T\mathbf{X}$) et toute métrique de $\mathcal{R}^P$ représentée par une matrice symétrique définie non négative $\mathbf{M}$ (non seulement la matrice identité $\mathbf{I}$).

Nous rappelons que $\mathbf{v}^T\mathbf{A}\mathbf{v} = \Sigma a_{i,j} v_i v_j$. Ainsi, comme $\mathbf{A}$ et $\mathbf{M}$ sont symétriques (i.e., $a_{i,j} = a_{j,i}$ et $m_{i,j} = m_{j,i}$), les dérivées partielles des formes quadratiques ont les formes suivantes :

$$\frac{\partial \mathbf{v}^T\mathbf{A}\mathbf{v}}{\partial \mathbf{v}} = 2\mathbf{A}\mathbf{v} \quad \text{et} \quad \frac{\partial \mathbf{v}^T\mathbf{M}\mathbf{v}}{\partial \mathbf{v}} = 2\mathbf{M}\mathbf{v}$$

Pour trouver le maximum de $\mathbf{v}^T\mathbf{A}\mathbf{v}$ en intégrant la contrainte unitaire sur $\mathbf{v}$ (i.e., $\mathbf{v}^T\mathbf{M}\mathbf{v} = 1$), nous annulons les dérivées du langrangien $\mathcal{L}$:

$$\mathcal{L} = \mathbf{v}^T\mathbf{A}\mathbf{v} - \lambda(\mathbf{v}^T\mathbf{M}\mathbf{v} - 1)$$

$$\begin{aligned}
&\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = 0 \\
&= \{\text{voir calcul des dérivées ci-dessus}\} \\
&2\mathbf{A}\mathbf{v} - 2\lambda\mathbf{M}\mathbf{v} = 0 \\
&= \\
&\mathbf{A}\mathbf{v} = \lambda\mathbf{M}\mathbf{v} \\
&= \{\mathbf{v}^T\mathbf{M}\mathbf{v} = 1\} \\
&\lambda = \mathbf{v}^T\mathbf{A}\mathbf{v}
\end{aligned}$$

Nous découvrons que la valeur du multiplicateur de Lagrange λ est le maximum recherché. Par ailleurs, nous avons :

$$\begin{aligned}
&\mathbf{A}\mathbf{v} = \lambda\mathbf{M}\mathbf{v} \\
&= \{\mathbf{M} \text{ est définie non négative et donc inversible}\} \\
&\mathbf{M}^{-1}\mathbf{A}\mathbf{v} = \lambda\mathbf{v}
\end{aligned}$$

Donc $\mathbf{v}$ est le vecteur propre de $\mathbf{M}^{-1}\mathbf{A}$ associé à la plus grande valeur propre λ. Nous notons cette solution $\mathbf{v_1}$ et la valeur propre correspondante λ_1.

Nous cherchons ensuite $\mathbf{v_2}$, unitaire ($\mathbf{v_2}^T\mathbf{M}\mathbf{v_2} = 1$), orthogonal à $\mathbf{v_1}$ ($\mathbf{v_1}^T\mathbf{M}\mathbf{v_2} = 0$) et qui maximise $\mathbf{v_2}^T\mathbf{A}\mathbf{v_2}$. Pour ce faire, nous annulons les dérivées du Langrangien ci-après.

$$\mathcal{L} = \mathbf{v_2}^T\mathbf{A}\mathbf{v_2} - \lambda_2(\mathbf{v_2}^T\mathbf{M}\mathbf{v_2} - 1) - \mu_2\mathbf{v_2}^T\mathbf{M}\mathbf{v_1}$$

En cours de rédaction...

6 Régression avec régularisation de Tikhonov et SVD

7 Validation croisée "un contre tous" appliquée à la régression régularisée

8 Interprétation d'un modèle de régression linéraire

8.1 Signification des coefficients

8.2 Intervalles de confiances et approche par bootstrap

8.3 Exemple sur un dataset

9 Estimateurs par maximum de vraissemblance

9.1 Application aux paramètres d'une loi normale uni-dimensionnelle

Permet d'introduire les notions de biais et de variance minimale.

9.2 Notation généralisée pour toute distribution

9.3 Vraissemblance conditionnelle pour la régression linéaire

9.4 Interprétation bayésienne de la régularisation commme distribution a priori sur les paramètres

10 Régression logistique

10.1 Vraissemblance conditionnelle pour la régression logistique

10.2 Interprétation des coefficients d'une régression logistique

10.3 Descente de gradient (stochastique) appliquée à la régression logistique

11 Relations générales entre biais, variance, nombre d'observations et complexité du modèle

12 Modèles parcimonieux et régularisation LASSO

13 Ensembles de modèles

13.1 Effets sur le biais et la variance d'une approche par ensembles de modèles

13.2 Bagging illustré avec le modèle de forêt aléatoire

13.3 Boosting ?

14 Réseaux de neurones

14.1 Neurone

14.2 Architecture en couches

14.3 Rétropropagation du gradient

14.4 Régularisation implicite par descente de gradient stochastique

14.5 Régularisation par dropout

14.6 Régularisation par batch normalization