Apprentissage Automatique

Rui Shibasaki

2025

1. Introduction

- L'apprentissage automatique est souvent connu comme *machine learning*
- Appelé aussi Apprentissage Statistique
- Ce domaine englobe un ensemble de modèles mathématiques issus de statistiques
- But: faire des inférences/prédictions à partir d'un ensemble de données

Inférences X Prédictions

Inférences X Prédictions

- Inférence: décrire l'environnement qui a généré l'ensemble des données.
- Prédictions: deviner comment l'environnement répond à une nouvelle donnée.
- Exemple: étant donné une série de photos d'animaux de compagnie, on souhaite distinguer les chats.
  Faire une **inférence** c'est dire que pour les chats on observera des oreilles pointues, petites, des poils long au museau, etc..
  Faire une **prédiction** c'est déterminer si une photo est une photo de chat ou pas.
- Pour l'inférence on a besoin d'une certaine **explicabilité** du modèle.

Types d'apprentissage: Supervisé X Non-supervisé

Supervisé

Souvent utilisé dans le but de prédire, l'apprentissage supervisé utilise des données labellisées. Le label étant donc ce qu'on essaye de prédire.
Exemple: les photos d'animaux de compagnie sont labellisés avec l'espèce contenue dans la photo. On sait donc qu'une photo de chat est une photo de chat puisque la photo a le label "chat".

Non-supervisé

Pour l'apprentissage non-supervisé, une telle labellisation n'existe pas. L'apprentissage se fait à l'aveugle. Dans ces cas, on cherche plus à en tirer des inférences sur les données.
Exemple: pour un ensemble de données provenant de la population d'un pays on chercherait à définir les caractéristiques de groupes de personnes qui se ressemblent.

Types d'apprentissage supervisé: Classification X Régression

Classification

Les méthodes de Classification visent à prédire une valeur qualitative.
Exemple: prédire quel animal se retrouve dans une photo est une Classification.

Régression

Les méthodes de Régression visent à predire une valeur quantitative.
Exemple: considérant les données sur les prix de vente des biens immobiliers selon leur caractéristiques, prédire le prix de vente pour une maison avec des caractéristiques particulières.

Types de méthodes: Paramétrique X Non-Paramétrique

Paramétrique

Les méthodes paramétriques considèrent que l'ensemble de données est issu d'un modèle spécifique, et ce modèle est définit par des paramètres qui vont le "customiser".

Non-Paramétrique

Les méthodes non-paramétriques ne font aucune supposition ou hypothèse sur l'environnement qui a produit l'ensemble des données.

Régression Linéaire simple

La régression Linéaire est une méthode de

- Régression
- Paramétrique
- Supervisé

- On a une mesure quantitative Y et une variable prédictive X. On estime que Y et X ont une dépendance linéaire:

$$Y \approx \beta_1 X + \beta_0$$

autrement :

$$Y = \beta_1 X + \beta_0 + \epsilon$$

- L'apprentissage consiste à trouver β_1 et β_0 qui définiront la droite la plus adaptée pour décrire la relation entre X et Y.
- Le modèle appris sera donc utilisé pour prédire Y étant donné un nouveau X.

Régression Linéaire simple

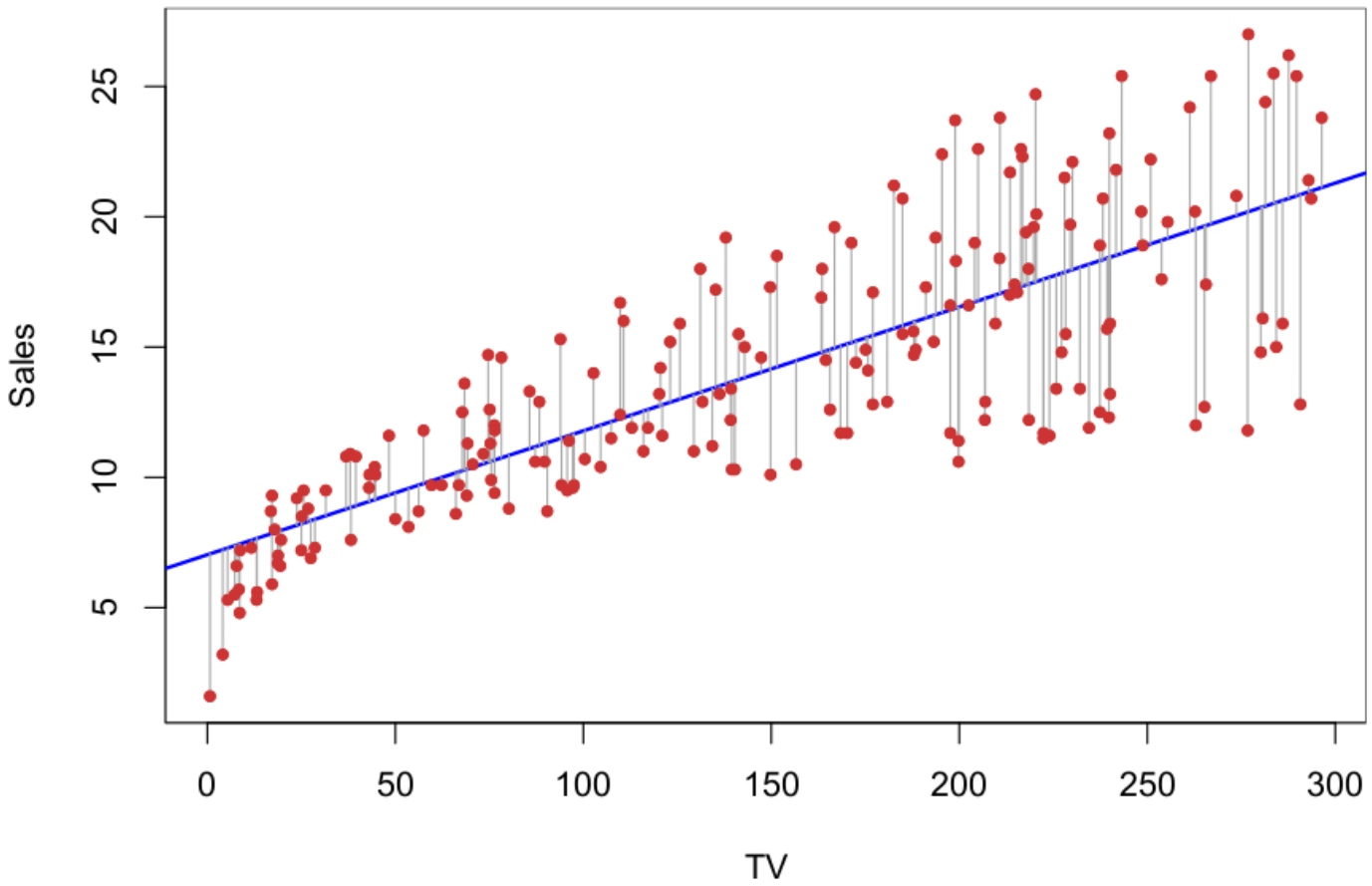


Figure: From James, G. et al. An introduction to Statistical Learning. Springer, 2023.

Régression Linéaire simple

Exemple

On dispose d'une série de données indicant les revenus de ventes après les dépences en campagne publicitaire. On estime que les revenus augmentent linéairement avec l'augmentation des dépenses publicitaires. Après avoir entraîné un modèle de régression linéaire simple avec les données diponibles on obtient

$$Y = 2X - 50$$

Pour une nouvelle campagne publicitaire, on peut prédire un revenu de 150 euros pour un budget publicitaire de 100 euros:

$$150 = 2 * 100 - 50$$

Régression Linéaire simple: Estimation des paramètres

- On utilise les données existents pour entraîner notre modèle, c'est-à-dire estimer les valeurs de β_0 et β_1. Les données sont representés par:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

- Dans l'exemple, x est un montant dépensé en publicité et y, un montant en revenu.
- Notre modèle doit minimiser la difference entre la prédiction $\hat{y}_i = \beta_0 + \beta_1 x_i$ et la valeur réel y_i, pour tout $i = 1, \ldots, n$:

$$\min \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- Cette stratégie est appelé "Least Squares". Les termes sont au carrés pour enlever le fait d'avoir des différences négatives et positives.
- Travailler avec la valeur absolue est plus complexe. La fonction ne serait pas dérivable, donc plus complexe à minimiser par les méthodes de gradient.

Régression Linéaire simple: Estimation des paramètres

Compte tenu de la définition $\hat{y}_i = \beta_0 + \beta_1 x_i$, remarquez que

$$L(\beta_1, \beta_0) = \min \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \min \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$$

La fonction $L(\beta_1, \beta_0)$ est convexe, donc atteint sont point minimal sur: $\frac{\partial L(\beta_1, \beta_0)}{\partial \beta_0} = 0$ et $\frac{\partial L(\beta_1, \beta_0)}{\partial \beta_1} = 0$. Résolvant ce système d'équations nous obtenons:

$$\beta_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

où $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$ et $\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$

- Le terme $\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$ est aussi appelé *residual sum of squares* (RSS)

Régression Linéaire simple: Précision des paramètres

- Les paramètres ont été estimés à l'aide de l'échantillon (les données diponibles), mais quelle est la précision de ses paramétres?
- Peut-on être confiant du fait qu'il existe une relation linéaire entre X et Y?
- Pour répondre à ces questions nous allons calculer l'écart-type de β:

$$SE^2(\beta_0) = \sigma^2 \left(\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \right) \quad (1)$$

$$SE^2(\beta_1) = \sigma^2 \left(\frac{1}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \right) \quad (2)$$

- Où $\sigma^2 = \text{Var}(\epsilon)$. On suppose aussi que les valeurs de ϵ_i sont décorrélées et qu'ils possèdent la même variance (c'est souvent pas le cas mais les formules restent une bonne approximation)

Régression Linéaire simple: Précision des paramètres

- Comme Var(ϵ) n'est pas connu généralement, nous estimons Var(ϵ) par le RSE (residual standart deviation)

$$RSE = \sqrt{\frac{RSS}{n-2}}$$

- Le RSE est une estimation de la variance de ϵ (l'erreur intrinsèque du modèle), considérant que $E[\epsilon] = 0$
- À partir des écart-types nous pouvons calculer des intervalles de confiance. Pour un intervalle de confiance à 95% nous avons (approximativement):

$$\beta_1 \pm 2 \cdot SE(\beta_1)$$

$$\beta_0 \pm 2 \cdot SE(\beta_0)$$

Régression Linéaire simple: Précision des paramètres

Exemple ventes/publicité

Pour l'exemple sur les campagnes publicitaires si l'intervalle de confiance à 95% (IC) pour β_0 est [210, 350], alors à un bugdet zéro pour les campagnes publicitaires on est sûr à 95% que les ventes seront en moyenne entre 210 et 350 euros.
Dans l'autre côté, si l'IC de β_1 est [0.05, 0.075], pour chaque 1000 euros de dépenses en publicité on espère une augmentation en moyenne de 50 à 75 euros en ventes.

Régression Linéaire simple: Précision des paramètres

- Les écarts-types peuvent aussi être utilisés pour les tests d'hypothèse:
  - $H_0 : \beta_1 = 0$ (il n'y a pas de relation entre x et y)
  - $H_1 = \beta_1 \neq 0$ (il y a un certain degré de relation) entre x et y
- En pratique, nous calculons la t-statistique:

$$t = \frac{\beta_1 - 0}{SE(\beta_1)}$$

(le nombre d'écart-types que β_1 est loin de zéro)

- Ensuite on calcule la valeur $p = P(|T| > |t| \mid H_0)$, considérant que T est une variable alétoire qui suit une distribution de Student à $n - 2$ degrés de liberté.
- Si $p < 0.05$ (ou 0.01, ça dépend du niveau de confiance) nous pouvons rejeter H_0 en faveur de H_1 avec un niveau de confiance de 95% (99% pour $p < 0.01$).

Régression Linéaire simple: Précision du modèle

- Le RSE calculé précédemment $RSE = \sqrt{\frac{RSS}{n-2}}$ peut aussi servir pour mesurer la précision du modèle prédictif. C'est une certaine mesure de performance. On utilise aussi le *mean squared error:* $MSE = \frac{RSS}{n}$
- L'inconvéniant du RSE c'est qu'il donne une valeur absolue (on a pas une base de comparaison)
- Il est souvent intéressant de se baser sur le R^2 pour mesurer la performance d'un modèle:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y}_i)^2}$$

- Nous savons donc que R^2 est surement une valeur dans l'intervalle [0,1].
- Une valeur de R^2 proche de zero indique que le modèle peut être est faux. Au contraire, une valeur proche de 1 indique que le modèle exprime bien la relation entre X et Y.

Régression Linéaire Multiple

- On peut généraliser la régression linéaire pour le cas où plusieurs variables prédictives existent:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \epsilon$$

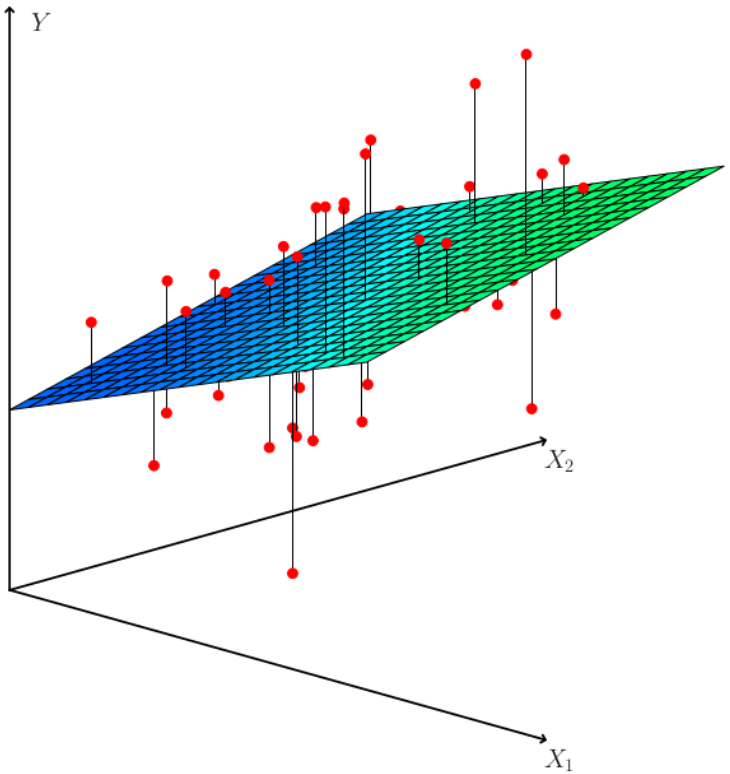


Figure: From James, G. et al. An introduction to Statistical Learning. Springer, 2023.

Régression Linéaire Multiple

Soit X la matrice $n \times (p+1)$ dont chaque ligne est un vecteur de données (avec un 1 en première position). De même, soit y le vecteur de dimension n contenant les sorties de l'ensemble d'entraînement. Nous pouvons alors écrire la somme des résidus au carré comme (RSS) est définie comme :

$$RSS(\beta) = (y - X\beta)^T(y - X\beta).$$

En différenciant par rapport à β, nous obtenons :

$$\frac{\partial RSS}{\partial \beta} = -2X^T(y - X\beta).$$

En supposant que X a un rang de colonne complet et que X^TX est donc définie positive:

$$X^T(y - X\beta) = 0.$$

Cela nous donne la solution unique :

$$\hat{\beta} = (X^TX)^{-1}X^Ty.$$

Régression Linéaire Multiple

La matrice de variance-covariance des estimateurs des paramètres des moindres carrés est donnée par :

$$\text{Var}(\hat{\beta}) = (X^T X)^{-1}\sigma^2. \tag{3}$$

Typiquement, on estime la variance σ^2 par :

$$\hat{\sigma}^2 = \frac{1}{n-p-1}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{4}$$

Ses valeurs peuvent donc être utilisées pour construire des intervalles de confiance et test d'hypothèse pour chaque paramètre $\beta_1, \ldots, \beta_p$ comme vu précédemment.

Régression Linéaire Multiple

Nous pouvons aussi nous demander s'il n'y pas de relation entre y et les variables prédictives dans l'ensemble: tous les coefficients de régression sont nuls. Nous testons l'hypothèse nulle :

$$H_0 : \beta_1 = \beta_2 = \cdots = \beta_p = 0$$

contre l'alternative :

$$H_a : \text{au moins un } \beta_j \text{ est non nul.}$$

Ce test d'hypothèse est effectué en calculant la statistique F :

$$F = \frac{(TSS - RSS)/p}{RSS/(n-p-1)}$$

où, comme dans la régression linéaire simple :

$$TSS = \sum (y_i - \bar{y})^2 \quad \text{et} \quad RSS = \sum (y_i - \hat{y}_i)^2.$$

Régression Linéaire Multiple

Parfois, nous voulons tester si un sous-ensemble particulier de q coefficients est nul. Cela correspond à l'hypothèse nulle :

$$H_0 : \beta_{p-q+1} = \beta_{p-q+2} = \cdots = \beta_p = 0,$$

où, par commodité, nous avons placé les variables choisies pour l'omission à la fin de la liste. Dans ce cas, nous ajustons un second modèle qui utilise toutes les variables sauf ces q dernières. Supposons que la somme des carrés des résidus pour ce modèle soit RSS_0. Alors, la statistique F appropriée est :

$$F = \frac{(RSS_0 - RSS)/q}{RSS/(n-p-1)}$$

Un F qui omet une seule variable du modèle, en laissant toutes les autres — c'est-à-dire $q = 1$, est équivalant à la statistique t fournissant des informations sur la relation entre chaque prédicteur individuel et y, après ajustement pour les autres prédicteurs.

Régression avec K-Nearest-Neighbors

- Nous avons vu la regression avec une méthode Paramétrique (least-squares)
- La méthode de régression par les K voisins les plus proche (K-Nearest-Neighbors) est un exemple de méthode d'apprentissage supervisée **non-paramétrique**. Ne nous n'avons pas des paramètres à estimer.
- Étant donné une valeur de K et un point de prédiction x_0, la régression KNN identifie d'abord les K observations d'entraînement les plus proches de x_0, représentées par N_0. Ensuite, elle estime $f(x_0)$ en utilisant la moyenne de toutes les réponses d'entraînement dans N_0. Autrement dit,

$$\hat{f}(x_0) = \frac{1}{K} \sum_{i \in N_0} y_i.$$

Régression avec K-Nearest-Neighbors (Exemple)

- Regardons l'exemple suivant. Les données suivent une tendance linéaire donné par la ligne noire.
- La régression linéaire donne la ligne en tirés bleue.

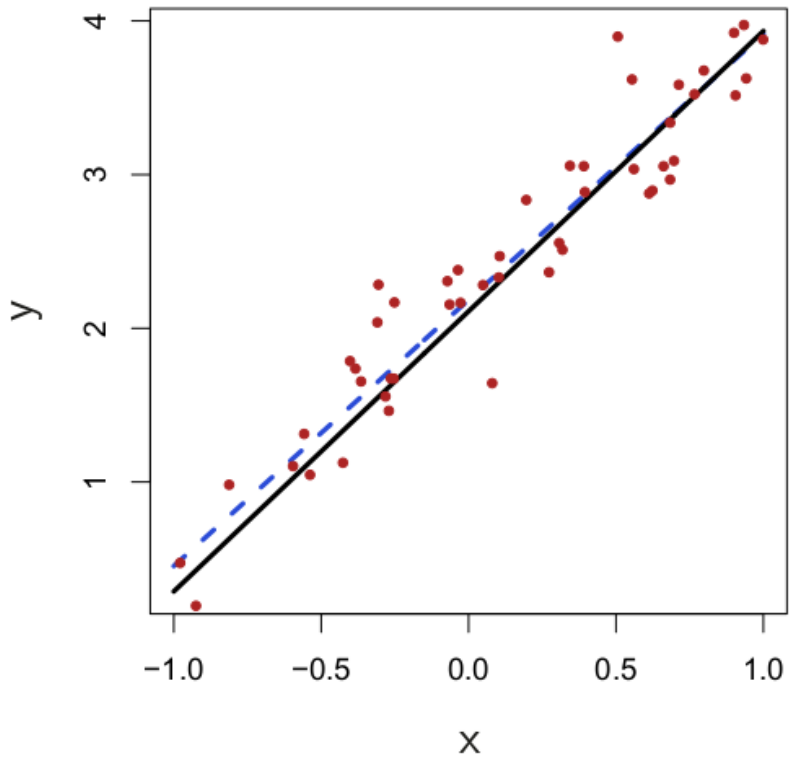


Figure: From James, G. et al. An introduction to Statistical Learning. Springer, 2023.

Régression avec K-Nearest-Neighbors (Exemple)

- $K = 1$ pour la figure à gauche et $K = 9$ pour la figure à droite.

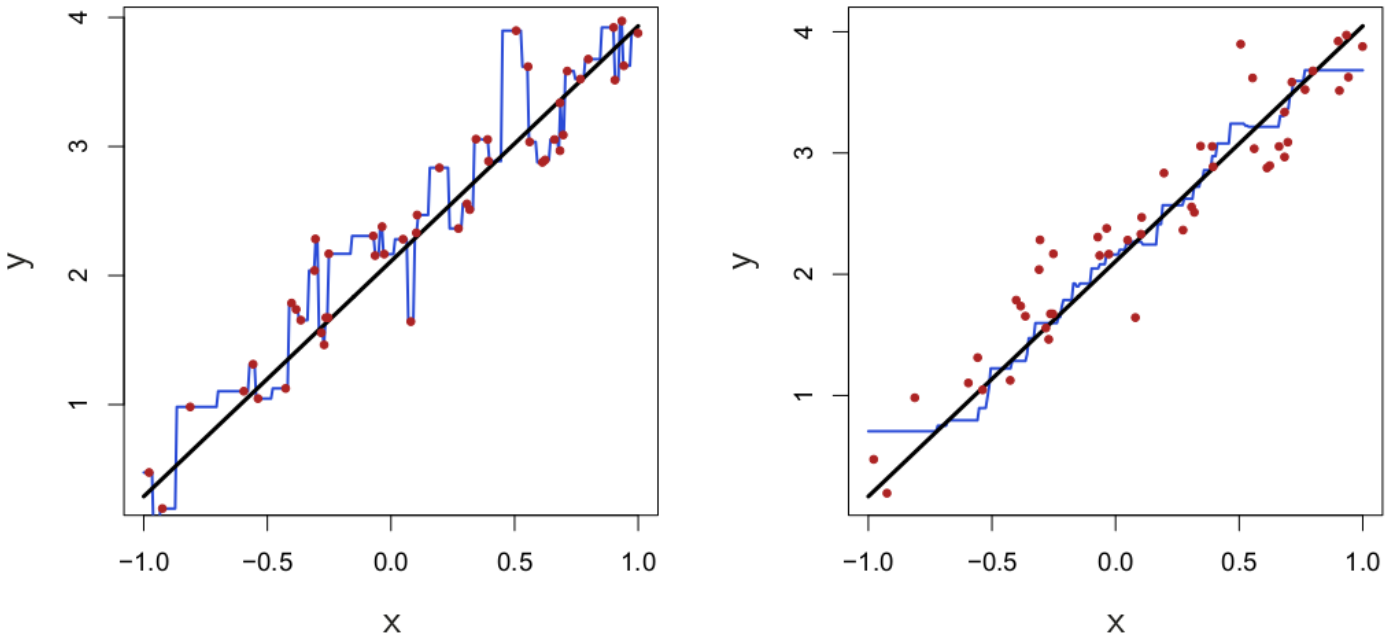
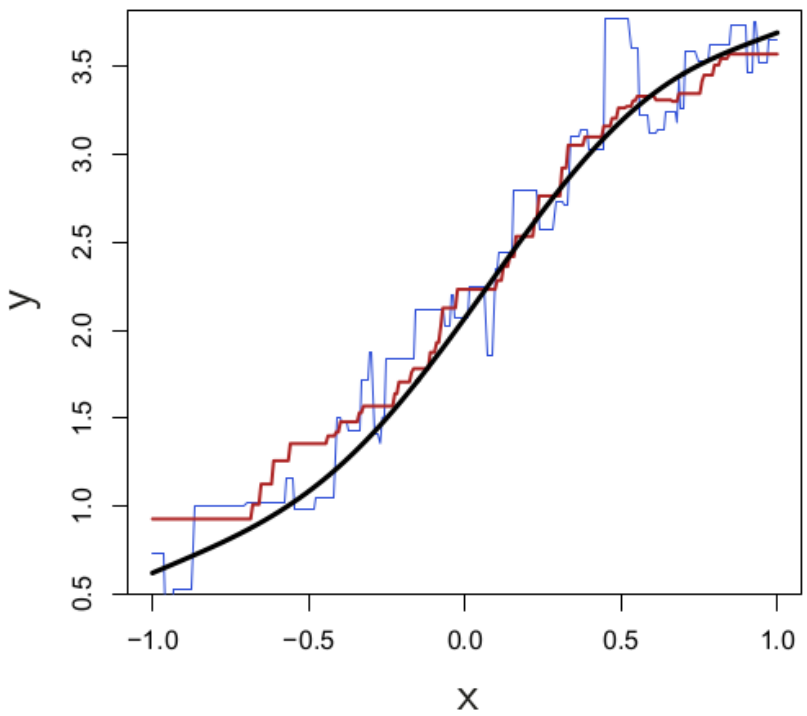


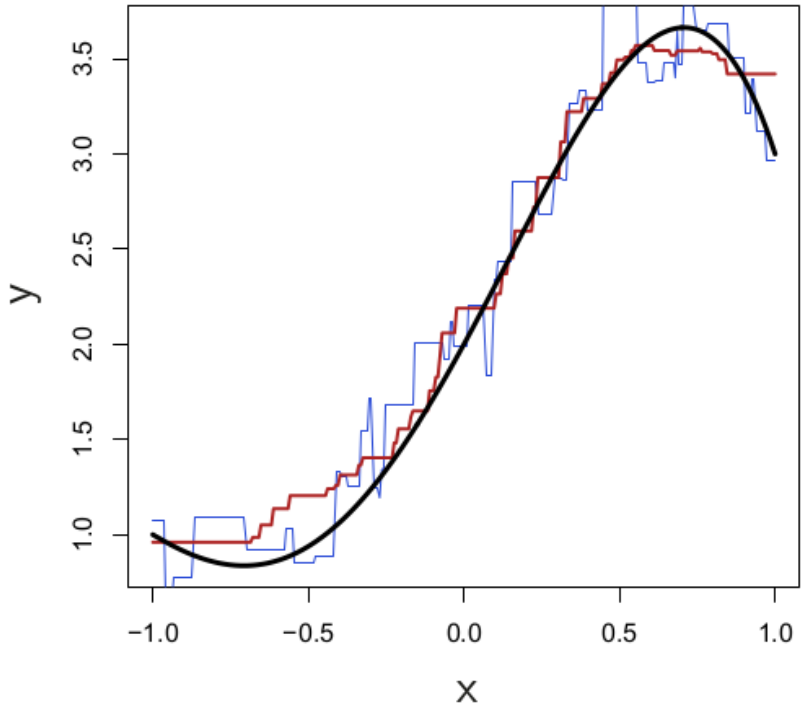
Figure: From James, G. et al. An introduction to Statistical Learning. Springer, 2023.

- On note que le modèle avec $K = 1$ est très flexible et matche parfaitement les données observées. Néanmoins on peut s'attendre à ce que les prédictions avec ce modèle soient très mauvaises.
- Ce phénomène s'appelle **Overfitting** et doit être évité.

Régression avec K-Nearest-Neighbors

- Régression KNN pour une relation non-linéaire entre x et y.





- Grace à sa flexibilité on s'adapte plus au moins bien au différentes relations entre x et y.
- Un modèle linéaire dans ce cas, ne serait pas pertinent.
- Ces problématiques peuvent être interprétées du point de vue du trade-off entre le **Biais** et la **Variance** du modèle.

Trade-Off Biais/Variance

- Pour un modèle de prédiction nous souhaitons que la différence $(y - \hat{y})^2$ soit la plus petite possible.
- Autrement dit, on souhaite que pour une donnée quelconque x_0 sa différence entre la prédiction et la vrai valeur aient une espérance faible.
- L'espérance de cette différence est notée $\mathbb{E}[(y_0 - \hat{y}_0)^2]$
- Cette espérance peut se décomposer de la forme (preuve en TD):

$$\mathbb{E}[(y_0 - \hat{y}_0)^2] = \text{Var}(\hat{y}_0) + [\text{Biais}(\hat{y}_0)]^2 + \text{Var}(\varepsilon)$$

- Le terme $\text{Var}(\varepsilon)$ est intrinsèque au modèle et ne peut pas être réduit.
- Par contre nous souhaitons réduire la variance et le biais du modèle.

Trade-Off Biais/Variance

Variance

- $Var(\hat{y})$ fait référence à la mesure dans laquelle $\hat{y}$ changerait si nous changeons les données d'entraînement.
- Idéalement, l'estimation de y ne devrait pas trop varier entre les ensembles d'entraînement.
- Plus la variance est grande, plus les prédictions dépendrons des données d'entraînement.
- En général, les méthodes statistiques plus flexibles ont une variance plus élevée.

Trade-Off Biais/Variance

Biais

- Biais($\hat{y}$) fait référence à l'erreur introduite en approximant un problème réel, qui peut être extrêmement complexe, par un modèle beaucoup plus simple.
- Par exemple, la régression linéaire suppose qu'il existe une relation linéaire entre y et $x_1, x_2, \ldots, x_p$.
- Il est peu probable qu'un problème réel ait une relation strictement linéaire, et ainsi, effectuer une régression linéaire entraînera inévitablement un certain biais dans l'estimation de y.
- Dans le cas non linéaire, peu importe le nombre d'observations d'entraînement disponibles, il ne sera pas possible de produire une estimation précise en utilisant la régression linéaire. Donc, la régression linéaire entraîne un biais élevé dans cet exemple.
- En général, les méthodes plus flexibles entraînent un biais plus faible.

Trade-Off Biais/Variance

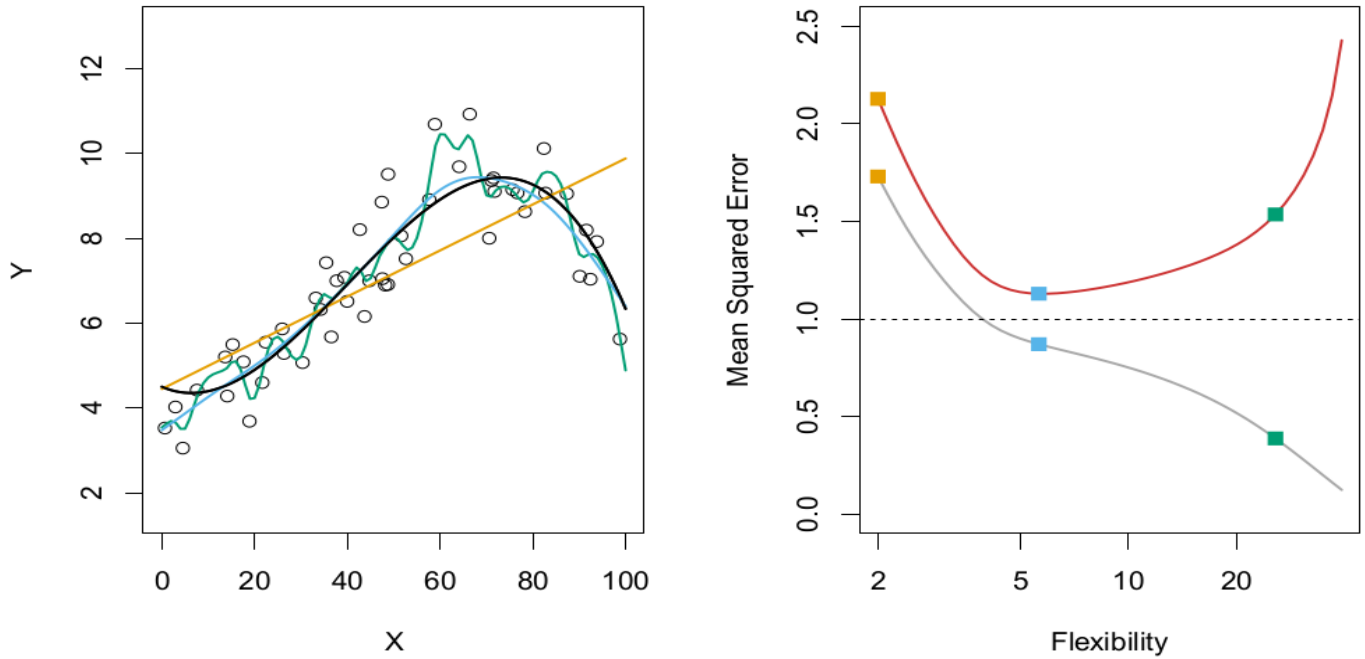


Figure: From James, G. et al. An intro. to Stat. Learning. Springer, 2023.

- Figure de gauche:
  - Noir: le meilleur modèle.
  - Bleu et Vert: des modèles flexibles (comme KNN)
  - Jaune: Régression linéaire
- Figure de droite:
  - Gris: erreur entraînement.
  - Rouge: erreur prédiction.

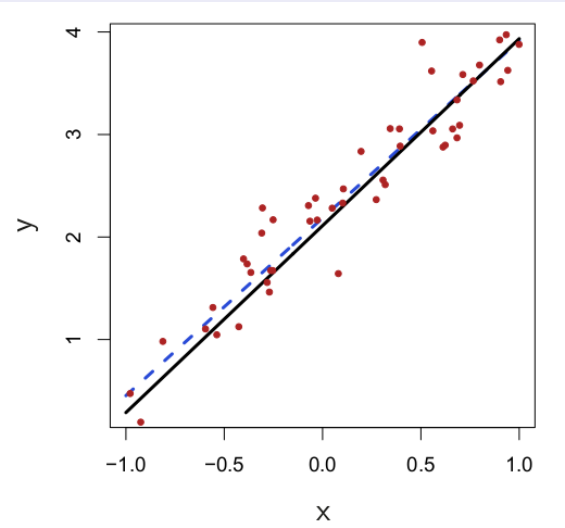
Fléau de la dimension (Curse of dimentionality)

- Un autre aspect à prendre en compte est la quantité de donnée, comparé aux nombre de variables prédictives.
- Les méthodes comme le KNN souffrent énormement du manque de données.
- Plus le nombre de variables (p) est grand ($X = [x_1, x_2, \ldots, x_p]$), plus le besoin pour un grand nombre de données est grand.
- Autrement dit, le nombre d'observations pour X de dimension p nécessaires à l'entraînement grandit exponentiellement avec l'augmentation de p.
- Ce problème est appelé *Curse of dimentionality*

Fléau de la dimension (Curse of dimentionality)

Exemple

Dans l'exemple suivant, considérons l'ensemble d'entrainement constitué seulement des deux points extrêmes (en haut à droite et en bas à gauche)



Comment changerait le modèle linéaire?
Que donnerait le modèle KNN?

Classification

- Dans la Classification nous avons en ensemble de N classes et nous devons classifier une donnée nouvelle comme appartenant à une de ces N classes.
- Pour une observation donnée $X = x_0$, la vraie classe Y suit la distribution de probabilité conditionnelle :

$$P(Y = j \mid X = x_0), \quad j = 1, 2, \ldots, N.$$

- Un modèle de classification va donc estimer ces probabilités et faire la classification selon ces probabilités.
- Nous cherchons un modèle qui fera le moin d'erreurs possible. C-à-d, pour une donné quelconque x_0, le modèle qui minimise:

$$P(Y_0 \neq \hat{y}_0 | x_0) = 1 - P(Y = \hat{y}_0 \mid X = x_0).$$

- Donc, pour n'importe quelle donnée, un modèle optimal est donc celui qui minimise l'espérance d'erreur:

$$\mathbb{E}[P(Y \neq \hat{Y}|X)] = \mathbb{E}[1 - P(Y = \hat{Y}|X)]$$

Classification: Bayes Classifier

- Un modèle de Bayes attribue chaque observation à la classe la plus probable, compte tenu de ses valeurs prédictives:

$$\hat{y}_{bayes} = \text{argmax}_j \; P(Y = j | X = x_0)$$

- Si les probabilités parfaitement sont connues, ce classificateur est optimale puisque $\mathbb{E}[1 - \max_j P(Y = j \mid X = x_0)]$ est la valeur minimale.
- Cette valeur est connue comme le *Bayes error rate*

Classification avec K-Nearest-Neighbors

- Un modèle KNN de Classification identifie d'abord les K points les plus proches de x_0 dans les données d'entraînement, représentés par N_0.
- Il estime ensuite la probabilité conditionnelle pour la classe j comme la fraction des points dans N_0 dont les valeurs de réponse sont égales à j :

$$P(Y = j \mid X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(y_i = j).$$

  où $I(y_i = j)$ est une variable indicatrice qui vaut 1 si $y_i = j$ et 0 sinon.
- Enfin, KNN attribue l'observation test x_0 à la classe ayant la plus grande probabilité

Classification avec K-Nearest-Neighbors

-Exemple avec $K = 3$

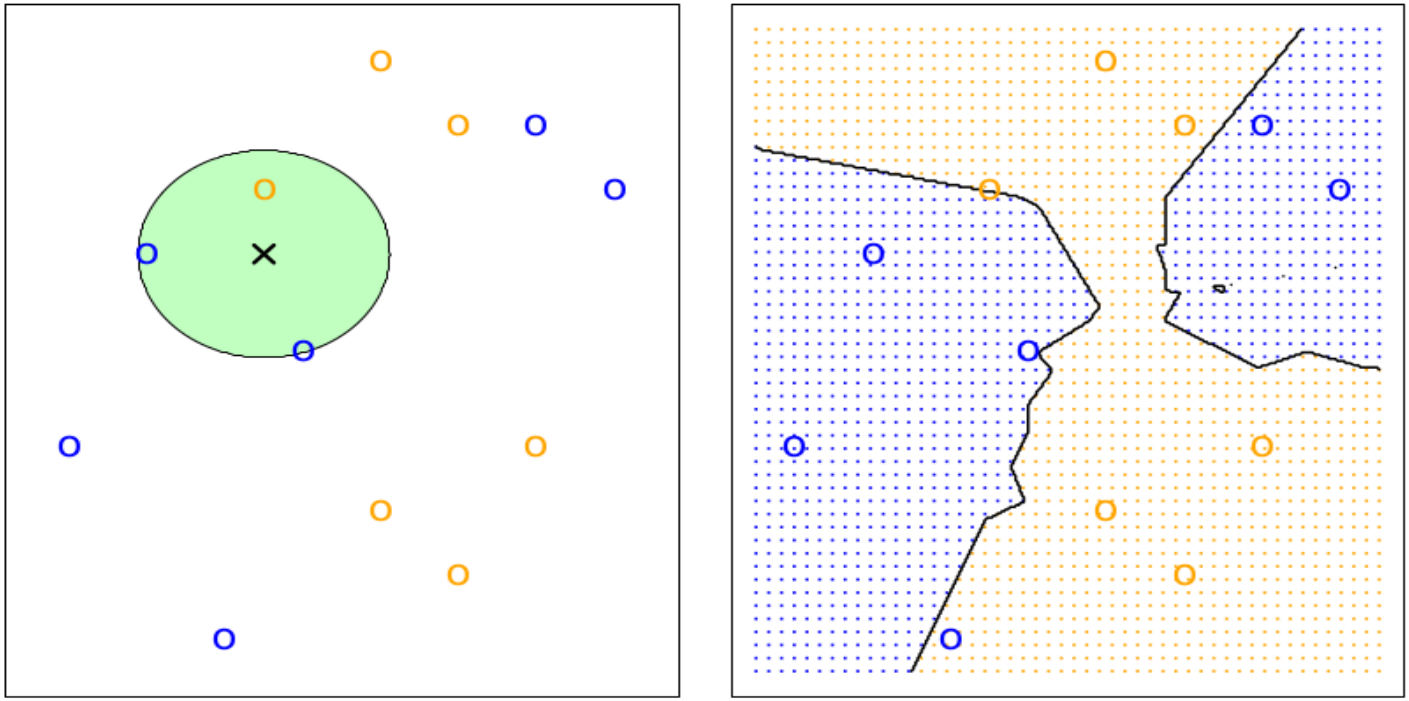


Figure: From James, G. et al. An intro. to Stat. Learning. Springer, 2023.

Classification: type d'erreurs

- Les modèles peuvent faire des classifications erronées.
- Ces erreurs sont de plusieurs types par rapport à une classe cible:

	Classe prédite		Total
Classe réelle	Autres	Cible	
Autres	Vrai Négatif (VN)	Faux Positif (FP)	N
Cible	Faux Négatif (FN)	Vrai Positif (VP)	P
Total	N^*	P^*	

Classification: type d'erreurs

Nom		Synonymes
Taux de faux positifs	FP/N	Erreur de type I, 1−Spécificité
Taux de vrais positifs	VP/P	1−Erreur de type II, puissance, sensibilité, rappel
Valeur prédictive positive	VP/P^*	Précision
Valeur prédictive négative	VN/N^*	
Spécificité	VN/N	
sensibilité	VP/P	
Erreur de type I	FP/N	
Erreur de type II	FN/P	

Classification: Thresholds

- Dans certains cas il est préférable d'avoir moins d'erreur pour une certaine classe cible, au détrimant des autres
- Par exemple: pour un modèle qui détermine si un crédit est à risque au pas, la banque va préférer faire plus d'erreur pour la classe risqué (classer des dossier comme "risqué" alors qu'ils ne le sont pas), que des erreurs pour la classe "non risqué" (classer comme "non risqué", des dossier à risque)
- Cette préférance peut être exprimer par des *Thresholds*:
- Si $P(Y = j_*|X = x_0) > t$, alors x_0 est classé dans j.
- Nous ne regardons plus seulement les probabilité maximales.

Classification: courbe ROC

- La courbe ROC nous permet de visualiser la performance d'un modèle selon tous les thresholds possibles ($0 \leq t \leq 1$)
- ROC= receiver operating characteristics.

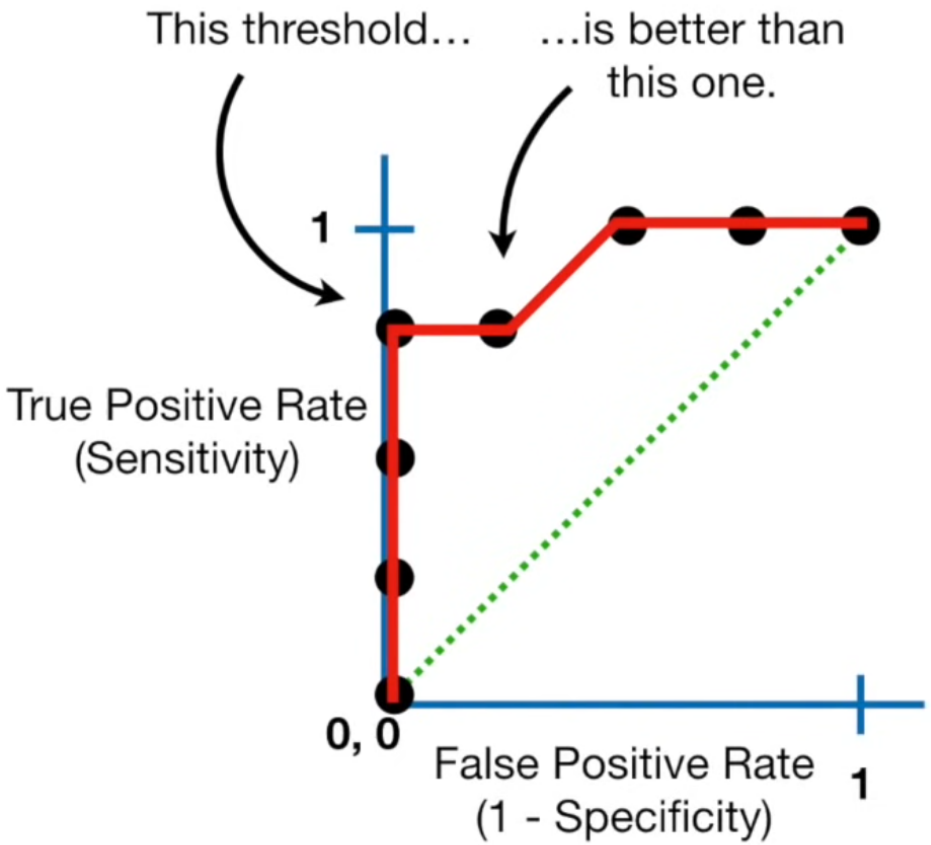


Figure: From Josh Starmer, StatQuest youtube channel, 2020.

Classification: courbe ROC

- L'aire sous la courbe donne la performance globale du modèle sous tous les thresholds.
- On peut donc comparer les modèles, selon les aires obtenues.
- Le modèle optimal est celui qui donne la courbe (0,0)-(0,1)-(1,1)

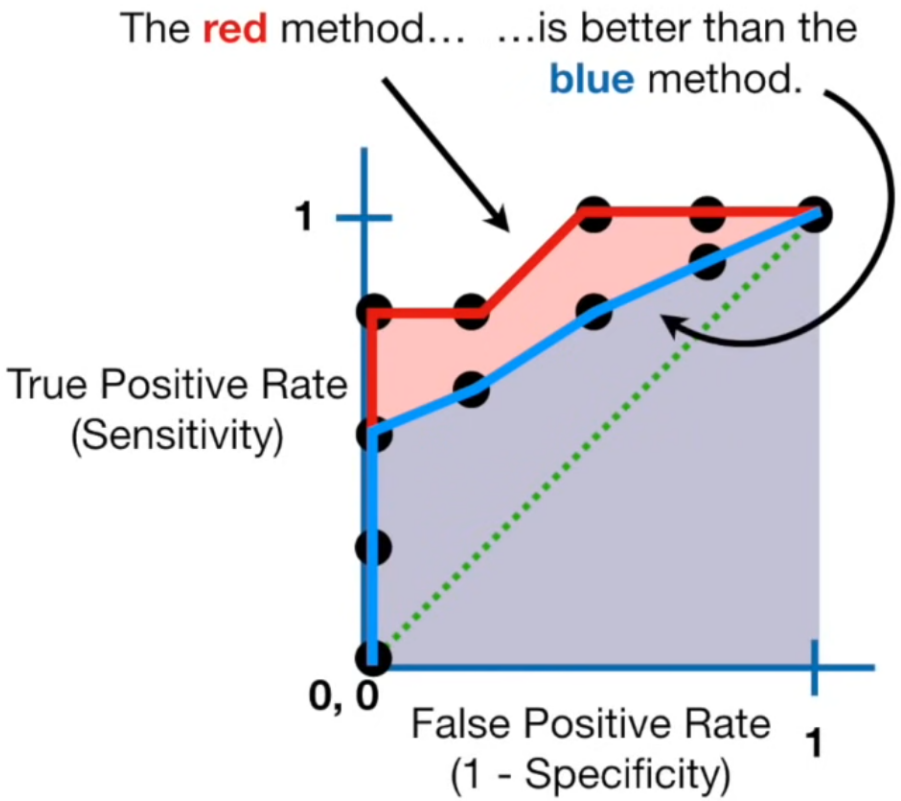


Figure: From Josh Starmer, StatQuest youtube channel, 2020.

Méthodes de Validation Croisée

- Jusqu'à présent nous avons vu que plusieurs modèles sont possibles pour une même application, et qu'au sein d'un même modèle nous avons plusieurs possibilités de flexibilité (nombre de voisins dans KNN par exemple).
- Il existes des méthodes pour nous aider à comparer les modèles et leurs performance selon chaque paramètre.
- Une de ces méthodes est la **Validation Croisée** ou *Cross-Validation*
- La méthode consiste en diviser l'ensemble d'entrainement aléatoirement. Une partie est utilisée pour l'entrainement et l'autre pour le test.
- Ce processus est fait plusieurs fois, ce qui génère plusieurs mesures de performance (MSE).
- Nous pouvons donc utiliser la moyenne des MSE comme valeur de comparaison entre les modèles.

K-fold Cross-Validation

- Typiquement on divise aléatoirement l'ensemble d'entrainement en k parties égales $k >= 2$ (généralement 5 ou 10 est une bonne valeur).

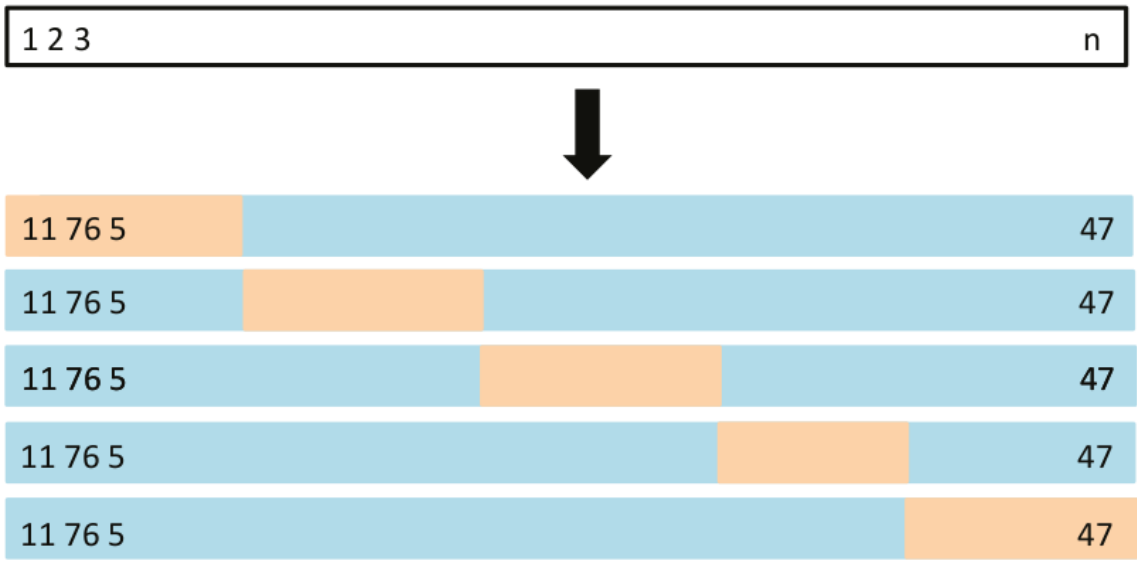


Figure: From James, G. et al. An intro. to Stat. Learning. Springer, 2023.

K-fold Cross-Validation

- Nous entrainons donc k modèles, avec à chaque fois une partie differente laissé de cotê, qui sera utilisé pour le test.
- Cela nous génères k mesures d'errer $MSE_1, \ldots, MSE_k$
- Nous pouvons donc calculer

$$CV = \frac{1}{k} \sum_{i=1}^{k} MSE_i$$

- Le modèle qui nous donne la plus petite valeur de CV est préférable.

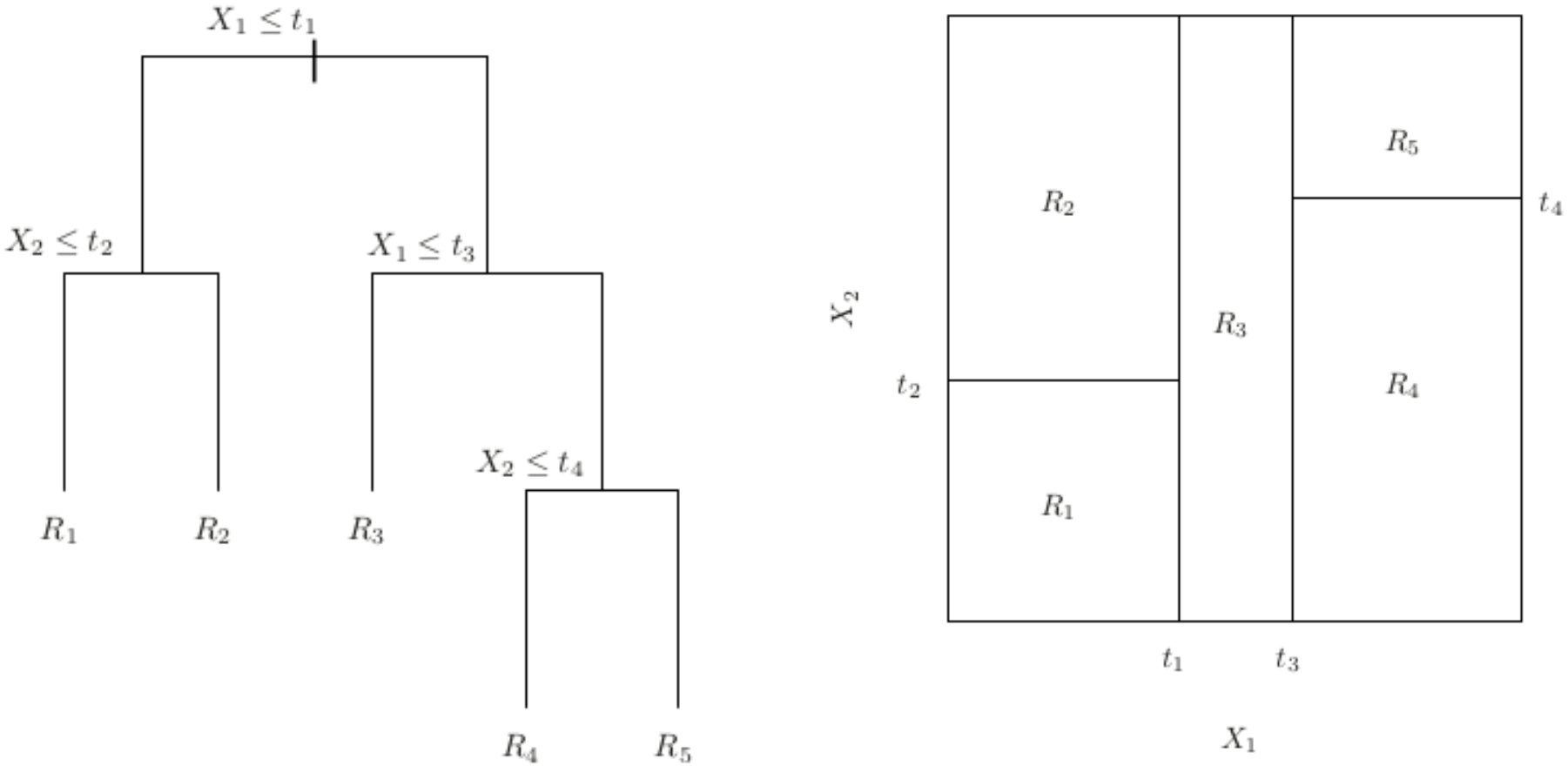
Méthodes Arborescentes

- Les méthodes arborescentes peuvent servir pour la Classification et régression. Ces sont encore des méthodes d'apprentissage supervisé (plus proche des non-paramètriques).
- Ces méthodes consiste en une partition des données d'entrainement. La prédiction pour une nouvelle donnée est donc faite à partir des données dans la partition correspondante.
- Les partitions sont établies selon des règles. Il suffit de suivre les règles pour savoir à quelle partition la nouvelle donnée appartient.
- L'objectif est de trouver les partitions $R_1, \ldots, R_J$ qui minimisent la somme des carrés résiduels (RSS), donnée par

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

où $\hat{y}_{R_j}$ est la réponse moyenne pour les observations d'entraînement dans la $j^{\text{ème}}$ région.

Méthodes Arborescentes



From James, G. et al. An intro. to Stat. Learning. Springer, 2023.

Méthodes Arborescentes: Régression

- Normalement, on construit un arbre binaire. L'algorithme va donc faire récursivement un découpage binaire.
- On choisit d'abord le prédicteur X_r et le point de coupure t tels que la division de l'espace des prédicteurs en régions $\{X|X_r < t\}$ et $\{X|X_r \geq t\}$ entraîne la plus grande réduction possible de la somme des carrés résiduels (RSS).
- Nous considérons donc tous les prédicteurs $X_1, \ldots, X_p$ et toutes les valeurs possibles du point de coupure t pour chacun des prédicteurs
- Puis nous choisissons le prédicteur et le point de coupure qui minimisent le RSS.

Méthodes Arborescentes: Régression

- Plus précisément, pour tout r et t, nous définissons la paire de demi-espaces : $R_1(r,t)$ et $R_2(r,t)$ et nous cherchons les valeurs de r et t qui minimisent l'équation suivante :

$$\sum_{i:x_i \in R_1(r,t)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(r,t)} (y_i - \hat{y}_{R_2})^2,$$

  où $\hat{y}_{R_1}$ est la réponse moyenne pour les observations d'entraînement dans $R_1(r,t)$, et $\hat{y}_{R_2}$ est la réponse moyenne pour les observations d'entraînement dans $R_2(r,t)$.
- On repète cette opération de manière récursive jusqu'à un atteindre un certain critère d'arrêt.
- Une fois les régions $R_1, \ldots, R_J$ créées, nous prédisons la réponse pour une observation en utilisant la moyenne des données d'entraînement dans la région à laquelle appartient cette nouvelle observation.

Méthodes Arborescentes: Classification

- Dans le cas de la classification, pour choisir la paire (r, t) de variable r et coupure t, nous minimisons l'indice Gini:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

où $\hat{p}_{mk}$ représente la proportion des données d'entraînement dans la $m^{\text{ème}}$ région qui appartiennent à la $k^{\text{ème}}$ classe.

$$\hat{p}_{mk} = \frac{\text{Nombre d'observations de la classe } k \text{ dans la région } m}{\text{Nombre total d'observations dans la région } m}$$

- Remarquez que l'indice est zero si tous les éléments d'une région m sont d'une même classe. L'indice mesure la pureté de la région (nœud de l'arbre).

Méthodes Arborescentes: Classification

- Une alternative à l'indice de Gini est l'**entropie**, donnée par :

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk} \tag{8.7}$$

- Puisque $0 \leq \hat{p}_{mk} \leq 1$, il en découle que $0 \leq -\hat{p}_{mk} \log \hat{p}_{mk}$.
- On peut montrer que l'entropie prendra une valeur proche de zéro si les $\hat{p}_{mk}$ sont tous proches de zéro ou proches de un.
- Ainsi, tout comme l'indice de Gini, l'entropie prendra une petite valeur si le $m^{\text{ème}}$ nœud est pur.

Réseaux de neuronnes

- Les réseaux de neurones sont devenus célèbres à la fin des années 1980, suscitant beaucoup d'enthousiasme.
- Après avoir perdu en popularité, les réseaux de neurones ont refait surface après 2010 sous le nom d'apprentissage profond (deep learning), avec de nouvelles architectures et améliorations.
- De nombreux experts estiment que ces succès sont principalement dus à la disponibilité de jeux de données d'entraînement de plus en plus volumineux.
- Grand succès avec des applications spécifiques comme les réseaux de neurones convolutifs (CNN) pour la classification d'images et les réseaux de neurones récurrents (RNN) pour les séries temporelles.

Réseaux de neuronnes simple (une couche)

- Un réseau de neurones prend un vecteur d'entrée de p variables $X = (X_1, X_2, \ldots, X_p)$ et construit une fonction non linéaire $f(X)$ pour prédire la réponse $Y = f(X)$.
- La figure montre un réseau de neurones qui modélise une réponse quantitative avec $p = 4$ prédicteurs.
- Les flèches indiquent que chaque entrée de la couche d'entrée alimente chacune des K unités cachées.
- La valeur de K est choisie par l'utilisateur (dans cet exemple, $K = 5$).

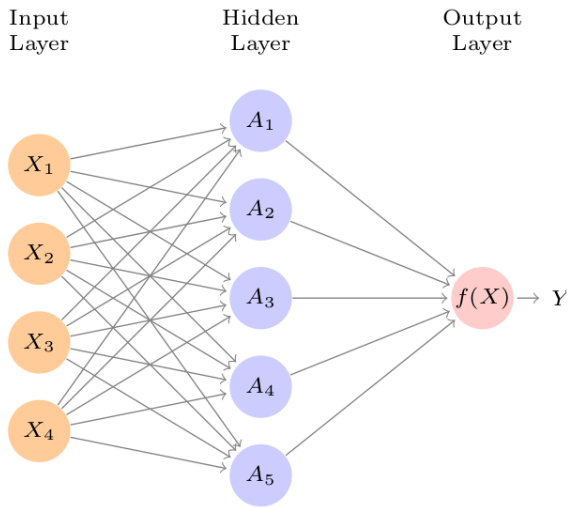


Figure: From James, G. et al. An intro. to Stat. Learning. Springer, 2023.

Réseaux de neuronnes simple (une couche)

- Le modèle de réseau de neurones a la forme:

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X)$$

- Il est construit en deux étapes:
  - Premièrement, les K activations A_k, $k = 1, \ldots, K$, dans la couche cachée sont calculées comme fonctions des caractéristiques d'entrée $X_1, \ldots, X_p$:

$$A_k = h_k(X) = g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j\right) \tag{5}$$

  - $g()$ est une fonction d'activation non linéaire qui est spécifiée à l'avance.

Réseaux de neuronnes simple (une couche)

- Chaque A_k peut être considéré comme une transformation différente $h_k(X)$ des caractéristiques originales.
- Ces K activations de la couche cachée alimentent ensuite la couche de sortie, résultant en:

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k$$

- Ceci est un modèle de régression linéaire avec $K = 5$ activations.
- Tous les paramètres $\beta_0, \ldots, \beta_K$ et $w_{10}, \ldots, w_{Kp}$ doivent être estimés à partir des données.

Réseaux de neuronnes simple (une couche)

- Souvent le choix de fonction d'activation dans les réseaux de neurones modernes est la fonction ReLU (*rectified linear unit*), qui prend la forme:

$$g(z) = \begin{cases} 0 & \text{si } z < 0 \\ z & \text{sinon} \end{cases}$$

- Une autre option utilisée est la fonction Softplus:

$$g(z) = log(1 + e^z)$$

- Le nom "réseau de neurones" dérive originellement de l'analogie entre ces unités cachées et les neurones dans le cerveau:
- Les valeurs des activations $A_k = h_k(X)$ strictement positives sont considérées comme "actives"
- Les valeurs proches de 0 sont considérées comme "silencieuses"

Réseaux de neuronnes simple (une couche)

- L'entraînement d'un réseau de neurones nécessite l'estimation des paramètres inconnus w et β.
- Pour une régression, on utilise typiquement la fonction de perte d'erreur quadratique (RSS).
- Les paramètres sont donc choisis pour minimiser:

$$\sum_{i=1}^{n}(y_i - f(x_i))^2$$

- Ces fonctions sont connus comme étant des *Loss Functions*

Réseaux de neuronnes simple (multi couches)

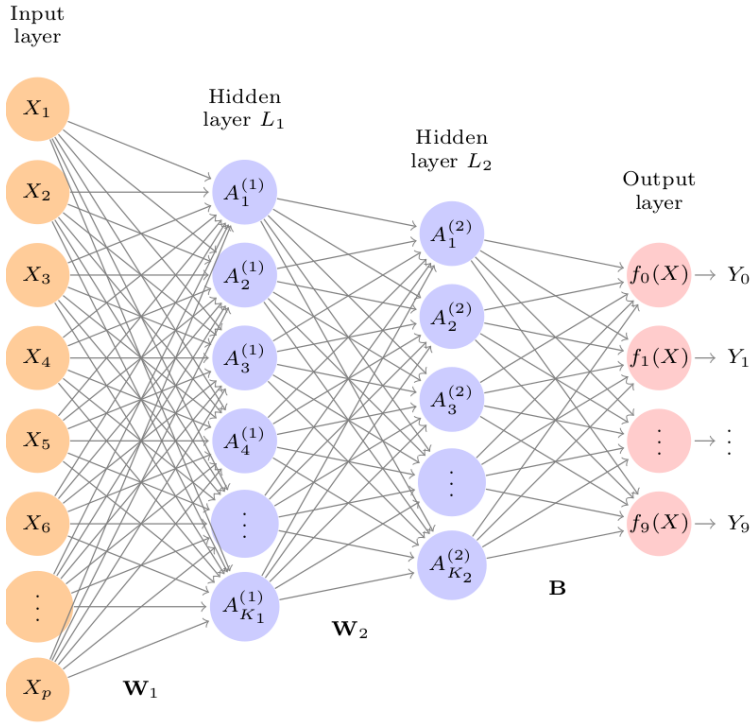


Figure: From James, G. et al. An intro. to Stat. Learning. Springer, 2023.

- Possibilité d'avoir plusieurs couches cachées (ici 2)
- Possibilité d'avoir plusieurs sorties (Par exemple pour une classification où 9 classes sont possibles, chaque Y donne la probabilité de l'input X (tout le vecteur) appartenir à la classe Y_j, $j \in \{1, \ldots, 9\}$)

Réseaux de neuronnes simple (multi couches)

- La première couche cachée est comme précédemment, pour $k = 1, \ldots, K_1$:

$$A_k^{(1)} = h_k^{(1)}(X) = g\left(w_{k0}^{(1)} + \sum_{j=1}^{p^{(1)}} w_{kj}^{(1)} X_j\right)$$

- La deuxième couche cachée traite les activations $A_k^{(1)}$ comme entrées et calcule de nouvelles activations pour $k = 1, \ldots, K_2$

$$A_\ell^{(2)} = h_\ell^{(2)}(X) = g\left(w_{\ell 0}^{(2)} + \sum_{k=1}^{K_1} w_{\ell k}^{(2)} A_k^{(1)}\right)$$

Réseaux de neuronnes simple (multi couches)

- Remarquez que chacune des activations dans la deuxième couche $A_\ell^{(2)} = h_\ell^{(2)}(X)$ est une fonction du vecteur d'entrée X.
- C'est le cas parce que bien qu'elles soient explicitement une fonction des activations $A_k^{(1)}$ de la couche L_1, celles-ci sont elles-mêmes des fonctions de X.
- Cela serait aussi le cas avec plus de couches cachées.
- Cela est la base pour effectuer l'apprentissage par le **Backpropagation**
- Nous arrivons maintenant à la couche de sortie:

$$Z_m = \beta_{m0} + \sum_{\ell=1}^{K_2} \beta_{m\ell} A_\ell^{(2)},$$

Réseaux de neuronnes simple (multi couches)

- Les fonctions de pertes deviennent: pour la régression

$$\sum_{i=1}^{n}\sum_{j=1}^{m}(y_i^j - f^j(x_i))^2$$

- Pour la classification on utilise l'entropie croisée:

$$-\sum_{i=1}^{N}\sum_{j=1}^{m} y_i^j \log f^j(x_i)$$

considérant que y_i^j et $f^j(x_i)$ sont des probabilités d'appartenance de i à la classe j.

Réseaux de neuronnes simple: apprentissage

- Reprenons le réseau à une couche: les paramètres sont $\beta = (\beta_0, \beta_1, \ldots, \beta_K)$, ainsi que chaque $w_k = (w_{k0}, w_{k1}, \ldots, w_{kp})$, pour $k = 1, \ldots, K$.
- Étant donné les observations (x_i, y_i), pour $i = 1, \ldots, n$, nous pouvons ajuster le modèle en résolvant un problème de moindres carrés non linéaires :

$$\min_{\{w_k\}_{k=1}^K, \beta} \sum_{i=1}^{n} \frac{1}{2}(y_i - f(x_i))^2,$$

où

$$f(x_i) = \beta_0 + \sum_{k=1}^{K} \beta_k g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} x_{ij}\right).$$

- L'apprentissage se fait en estimant les valeurs des paramètres $(\{w_k\}_{k=1}^K, \beta)$ tel que la fonction de perte est minimisée.

Réseaux de neuronnes simple: apprentissage

- Considérons θ le vecteur contenant tous les paramètres $(\{w_k\}_{k=1}^K, \beta)$. On peut donc réécrire la fonction de perte comme:

$$R(\theta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - f_\theta(x_i))^2,$$

- $R(\theta)$ est non convexe et donc il existe plusieurs minima. La fonction $R(\theta)$ ci-dessous est non-convexe et possède deux minima: l'un est un minimum local et l'autre est un minimum global.

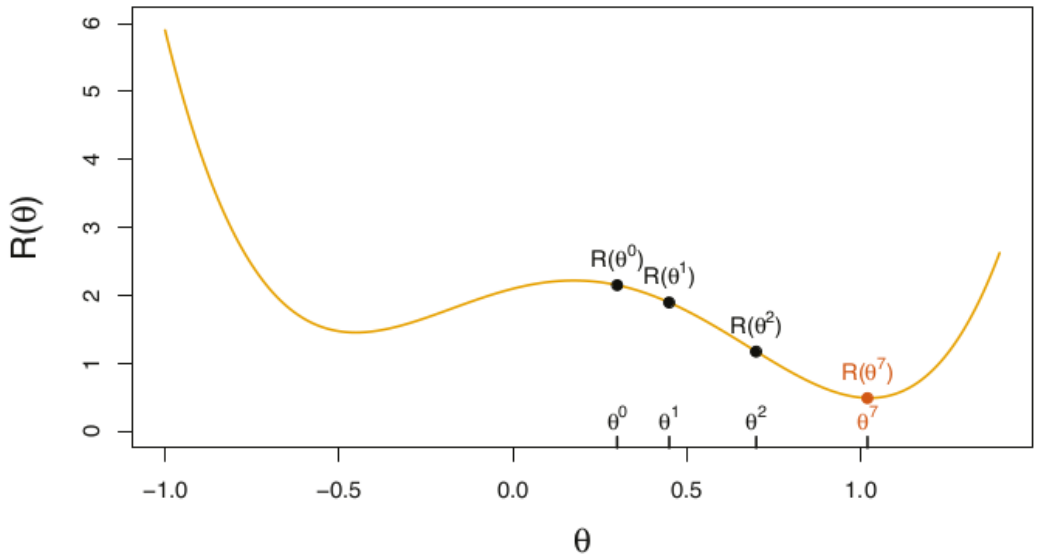


Figure: From James, G. et al. An intro. to Stat. Learning. Springer, 2023.

Réseaux de neuronnes simple: Backpropagation

- Pour résoudre le problème $min_\theta R(\theta)$ nous utilisons l'algorithme de descente du gradient.
- Remarquez que pour chaque terme de la somme nous avons:

$$R_i(\theta) = \frac{1}{2}\left(y_i - \beta_0 - \sum_{k=1}^{K} \beta_k g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} x_{ij}\right)\right)^2.$$

- Pour simplifier les expressions suivantes, nous écrivons $z_{ik} = w_{k0} + \sum_{j=1}^{p} w_{kj} x_{ij}$. Nous prenons d'abord la dérivée par rapport à β_k :

$$\frac{\partial R_i(\theta)}{\partial \beta_k} = \frac{\partial R_i(\theta)}{\partial f_\theta(x_i)} \cdot \frac{\partial f_\theta(x_i)}{\partial \beta_k} = -(y_i - f_\theta(x_i)) \cdot g(z_{ik}).$$

- Pour β_0 nous avons $\frac{\partial R_i(\theta)}{\partial \beta_0} = -(y_i - f_\theta(x_i))$

Réseaux de neuronnes simple: Backpropagation

- Ensuite, nous prenons la dérivée par rapport à w_{kj} :

$$\frac{\partial R_i(\theta)}{\partial w_{kj}} = \frac{\partial R_i(\theta)}{\partial f_\theta(x_i)} \cdot \frac{\partial f_\theta(x_i)}{\partial g(z_{ik})} \cdot \frac{\partial g(z_{ik})}{\partial z_{ik}} \cdot \frac{\partial z_{ik}}{\partial w_{kj}} = -(y_i - f_\theta(x_i)) \cdot \beta_k \cdot g'(z_{ik}) \cdot x_{ij}.$$

- Pour w_{k0} nous avons $\frac{\partial R_i(\theta)}{\partial w_{k0}} = -(y_i - f_\theta(x_i)) \cdot \beta_k \cdot g'(z_{ik})$
- Remarquez que ces derivées partielles contiennent la quantité d'erreur: $y_i - f_\theta(x_i)$.
- Ainsi, on a une fraction du résidu qui est propagée à chacun des paramètres via la règle de dérivation en chaîne
- Ce processus est connu sous le nom de rétropropagation, **Backpropagation** en anglais.

Réseaux de neuronnes simple: Backpropagation

- La gradient est alors composé de :

$$\nabla R(\theta) = \left[\sum_{i=1}^{n} \frac{\partial R_i(\theta)}{\partial \beta_0}, \cdots, \sum_{i=1}^{n} \frac{\partial R_i(\theta)}{\partial \beta_K}, \sum_{i=1}^{n} \frac{\partial R_i(\theta)}{\partial w_{10}}, \cdots, \sum_{i=1}^{n} \frac{\partial R_i(\theta)}{\partial w_{Kp}} \right]$$

- Remarquez que la nombre de données (n) peut être très grand et le calcul du gradient peut donc prendre un peu de temps.
- Au lieu de calculer le gradient en utilisant tous les n observations, on utilise un *minibatch*: un échantillon plus petit dans l'ensemble des n observations.
- Le gradient calculé selon ce batch est donc une estimation du vrai gradient.
- Cette méthode est donc appelé la descente du grandient stochastique (**Stochastic Gradient Descent**)

Apprentissage non-supervisée: Analyse en composantes principales (PCA)

- L'analyse en composantes principales nous permet d'apprendre sur la relation entre les variables observées.
- Cette relation est décrite par une combinaison linéaire des variables (appelée **composante**)
- La première composante principale d'un ensemble de variables $X_1, X_2, \ldots, X_p$ est la combinaison linéaire normalisée des variables :

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + \cdots + \phi_{p1} X_p$$

qui possède la plus grande variance. Par **normalisée**, nous entendons que $\sum_{j=1}^{p} \phi_{j1}^2 = 1$. On normalise afin d'éviter une variance infinie.

- Nous appelons les coefficients $\phi_1 = (\phi_{11}, \phi_{21}, \ldots, \phi_{p1})^T$ les **poids** (*loadings* en anglais).

Apprentissage non-supervisée: Analyse en composantes principales (PCA)

- Remarquez que, ce qui nous intéresse est la relation entre les variables, donc leur magnitude n'a pas d'importance.
- On normalise les valeurs en les divisant par l'écart-type de l'échantillon
- Ces valeurs sont ensuite centrées sur zéro pour pouvoir obtenir la composante.

$$\tilde{x}_i = (x_i - \bar{x})/\hat{\sigma}$$

Apprentissage non-supervisée: Analyse en composantes principales (PCA)

- Pour déterminer les valeurs du vecteur ϕ_1 on doit résoudre le problème:

$$\text{maximize}_{\phi_{11},\ldots,\phi_{p1}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1} \tilde{x}_{ij} \right)^2 \right\} \text{ subject to } \sum_{j=1}^{p} \phi_{j1}^2 = 1.$$

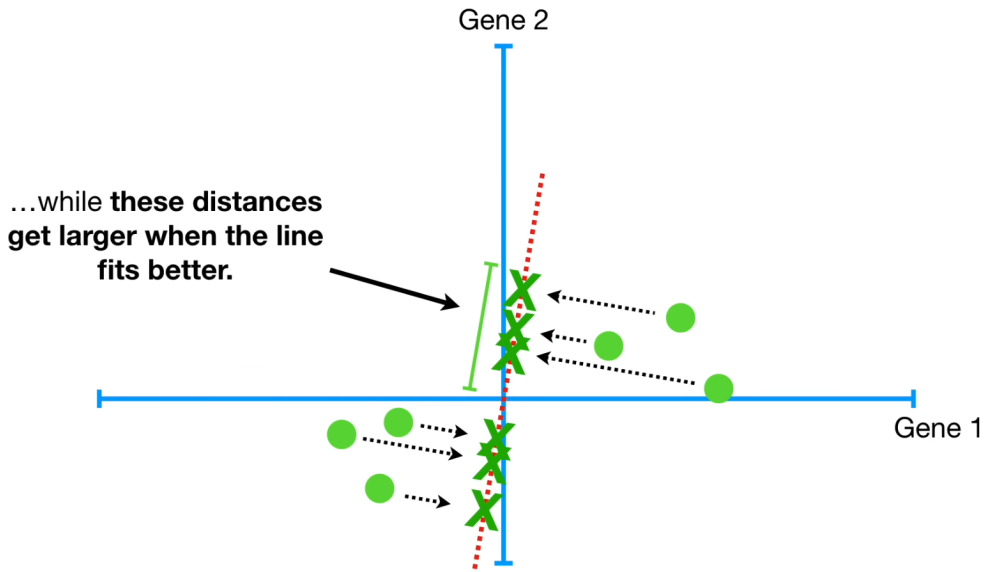


Figure: From Josh Starmer, StatQuest youtube channel, 2020.

Apprentissage non-supervisée: Analyse en composantes principales (PCA)

- Il est possible de calculer au plus $c = \min\{n-1, p\}$ composantes.
- Pour calculer les composantes suivantes le même problème est résolu avec un contrainte supplémentaire: la nouvelle composante doit être perpendiculaire à toutes les autres.

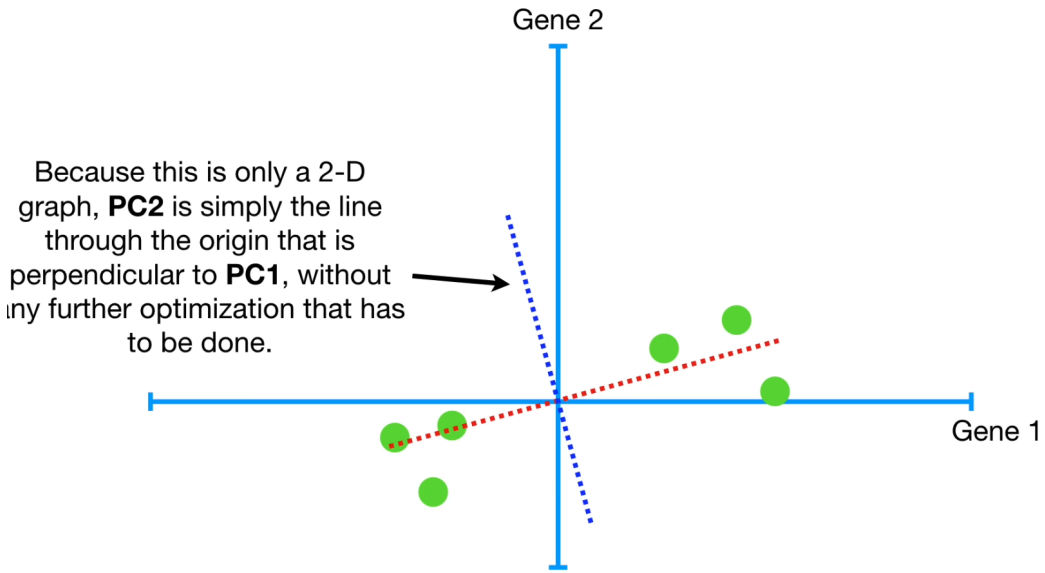


Figure: From Josh Starmer, StatQuest youtube channel, 2020.

Apprentissage non-supervisée: Analyse en composantes principales (PCA)

- Une fois que les composantes sont calculées, elles nous permettent la visualisation des données dans une dimension plus petite. Par exemple 2D.
- Cela nous permet l'identification de clusters ou similitudes entre données.

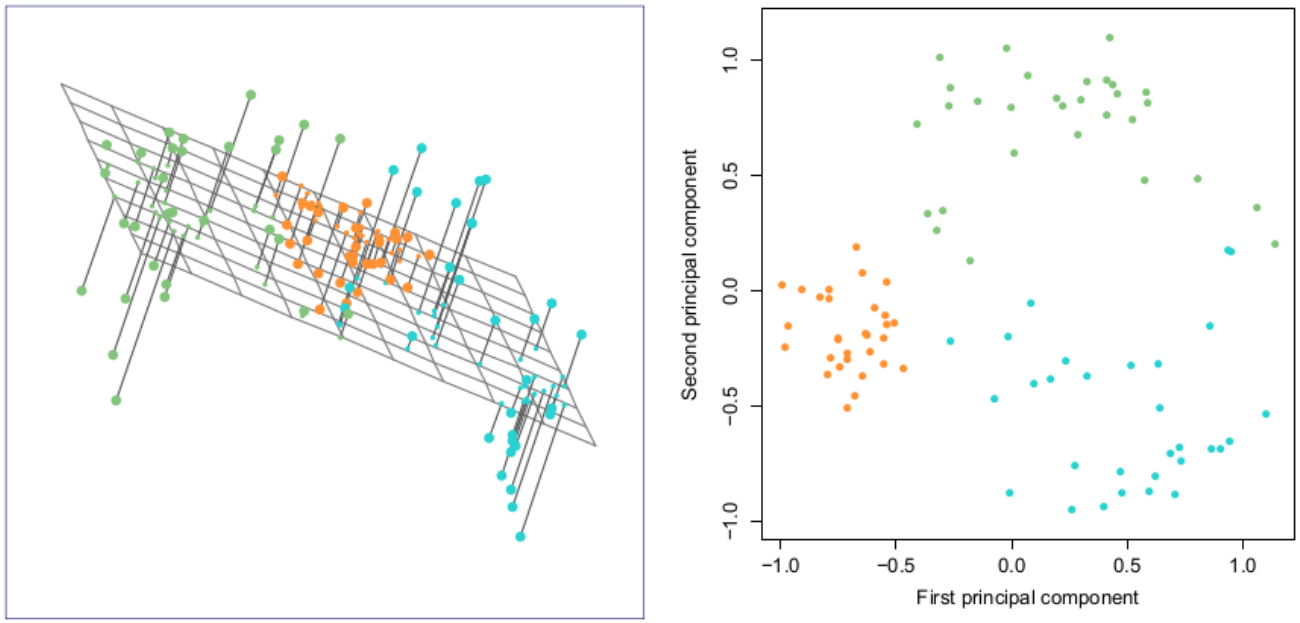


Figure: From James, G. et al. An intro. to Stat. Learning. Springer, 2023.

Apprentissage non-supervisée: Analyse en composantes principales (PCA)

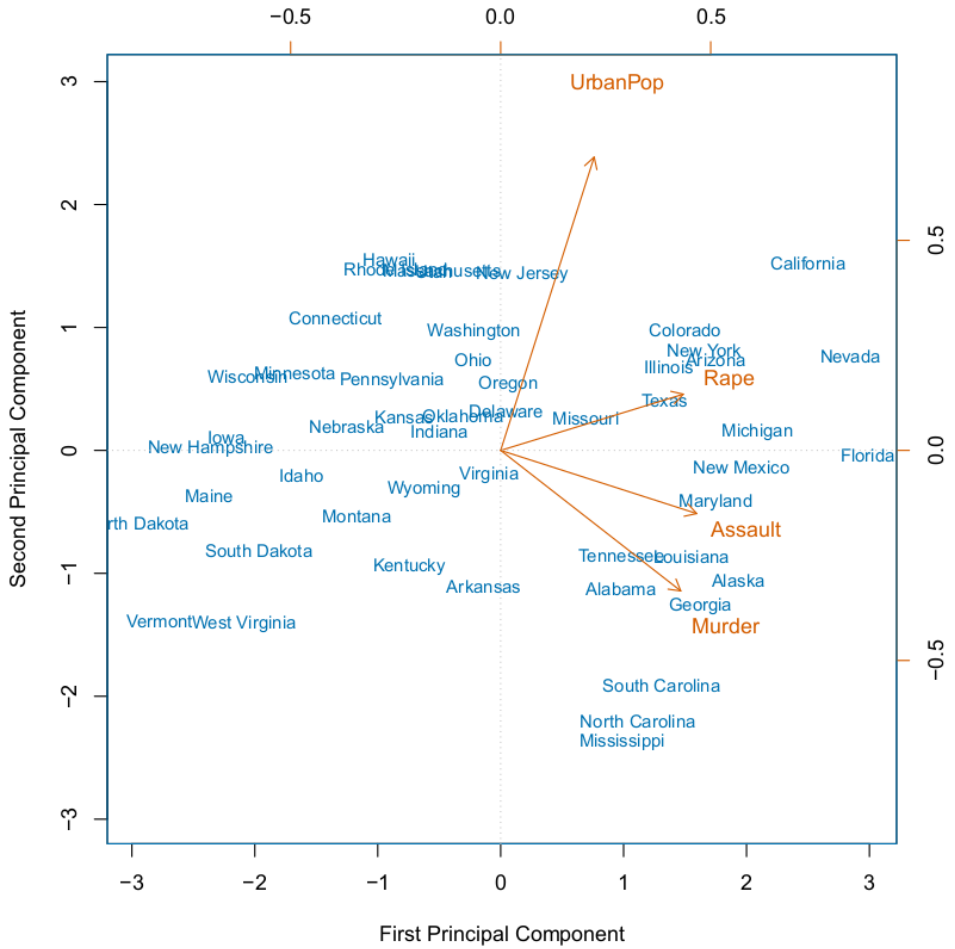


Figure: From James, G. et al. An intro. to Stat. Learning. Springer, 2023.

Apprentissage non-supervisée: Analyse en composantes principales (PCA)

- Comme dit précédemment, il est possible de calculer au plus $c = \min\{n-1, p\}$ composantes.
- Alors pourquoi s'intéresser qu'aux deux/trois premières pour la visualisation?
- Cette question est répondue en regardant la quantité de la variance totale présente en chaque composante.
- Si une bonne partie de la variance est contenu dans 2 ou 3 composantes, on peut considérer que ces composantes suffisent pour bien comprendre la relation entre les variables.

Apprentissage non-supervisée: Analyse en composantes principales (PCA)

- La variance totale présente dans un ensemble de données (en supposant que les variables ont été centrées pour avoir une moyenne nulle) est définie comme

$$TotalVar = \sum_{j=1}^{p} \text{Var}(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2$$

- Et la variance dans chaque composante est donné par la valeur de solution des problèmes résolus pour les définir:

$$PCVar_m = \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{jm} \tilde{x}_{ij} \right)^2 \right\}$$

- Pour chaque composante m nous avons donc une portion de la variance total donné par:

$$\frac{PCVar_m}{TotalVar}$$

Apprentissage non-supervisée: Analyse en composantes principales (PCA)

- On peut tracer le *Scree plot* utilisant les ratios de variance pour une meilleure visualisation:

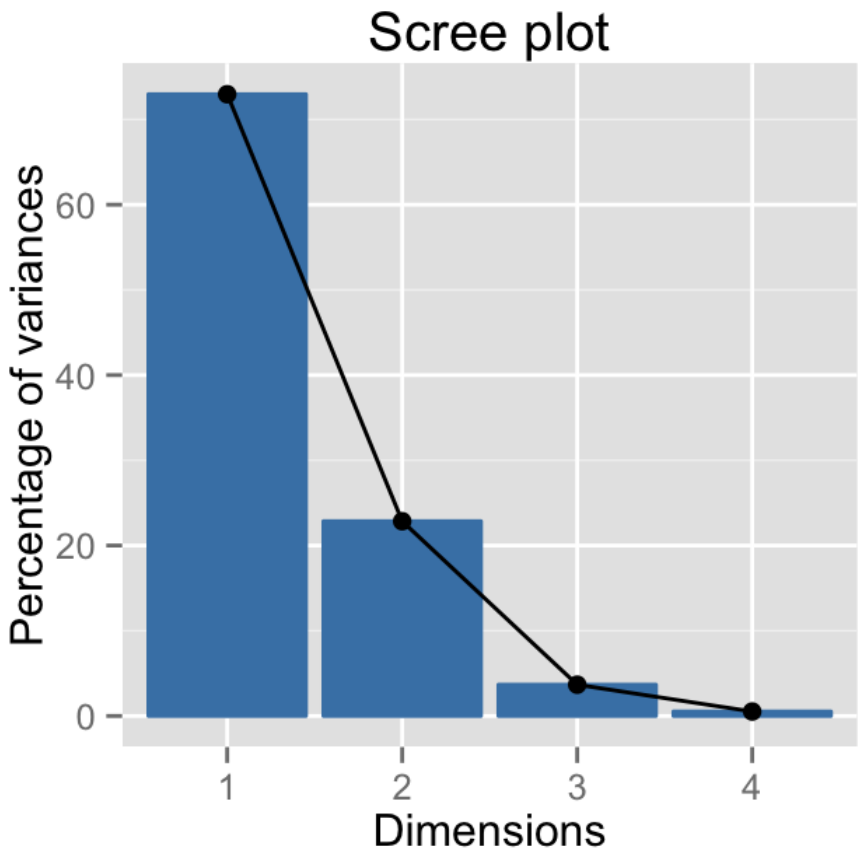


Figure: From Statistical tools for high-throughput data analysis

Apprentissage non-supervisée: Clustering K-Means

- Contrairement à PCA, le K-means est une méthode de clustering.
- Tant le clustering que l'PCA cherchent à simplifier les données, mais leurs mécanismes sont différents :
  - Le PCA cherche à trouver une représentation de plus petit dimension, qui explique une grande partie de la variance des observations;
  - Le clustering cherche à identifier des sous-groupes homogènes parmi les observations.
- Le K-means cherche à diviser l'ensemble des observations en K sous-ensembles disjoints.
- Soit $C_1, \ldots, C_K$ des ensembles contenant les indices des observations dans chaque cluster. Ces ensembles satisfont deux propriétés :
  1. $C_1 \cup C_2 \cup \cdots \cup C_K = \{1, \ldots, n\}$. En d'autres termes, chaque observation appartient à au moins un des K clusters.
  2. $C_k \cap C_{k'} = \emptyset$ pour tout $k \neq k'$. Autrement dit, les clusters sont disjoints : aucune observation n'appartient à plus d'un cluster.

Apprentissage non-supervisée: Clustering K-Means

- On cherche à determiner une partition de K sous-ensembles qui donne une solution optimal pour le problème:

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} W(C_k)$$

où

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2,$$

et $|C_k|$ désigne le nombre d'observations dans le cluster C_k.

- Ce problème n'est pas facile, mais on peut avoir une bonne solution utilisant un algorithme glouton.

Apprentissage non-supervisée: Clustering K-Means

Algorithm K-Means Clustering

1: Attribuer aléatoirement un numéro, de 1 à K, à chacune des observations. Ceux-ci servent de premières affectations de clusters pour les observations.
2: Répéter jusqu'à ce que les affectations de clusters ne changent plus :
3: (a) Pour chacun des K clusters, calculer le centroïde du cluster. Le centroïde du k-ème cluster est le vecteur des moyennes des p caractéristiques pour les observations dans le k-ème cluster.
4: (b) Assigner chaque observation au cluster dont le centroïde est le plus proche (où "proche" est défini par la distance euclidienne).

- La qualité de solution dépendra de la partition aléatoire initiale. Il faut tourner l'algorithme plusieurs fois, avec une partition initiale différente à chaque fois.
- On prend la partition finale que donne la meilleur valeur de solution.

Apprentissage non-supervisée: Clustering K-Means

Une exécution de l'algorithme

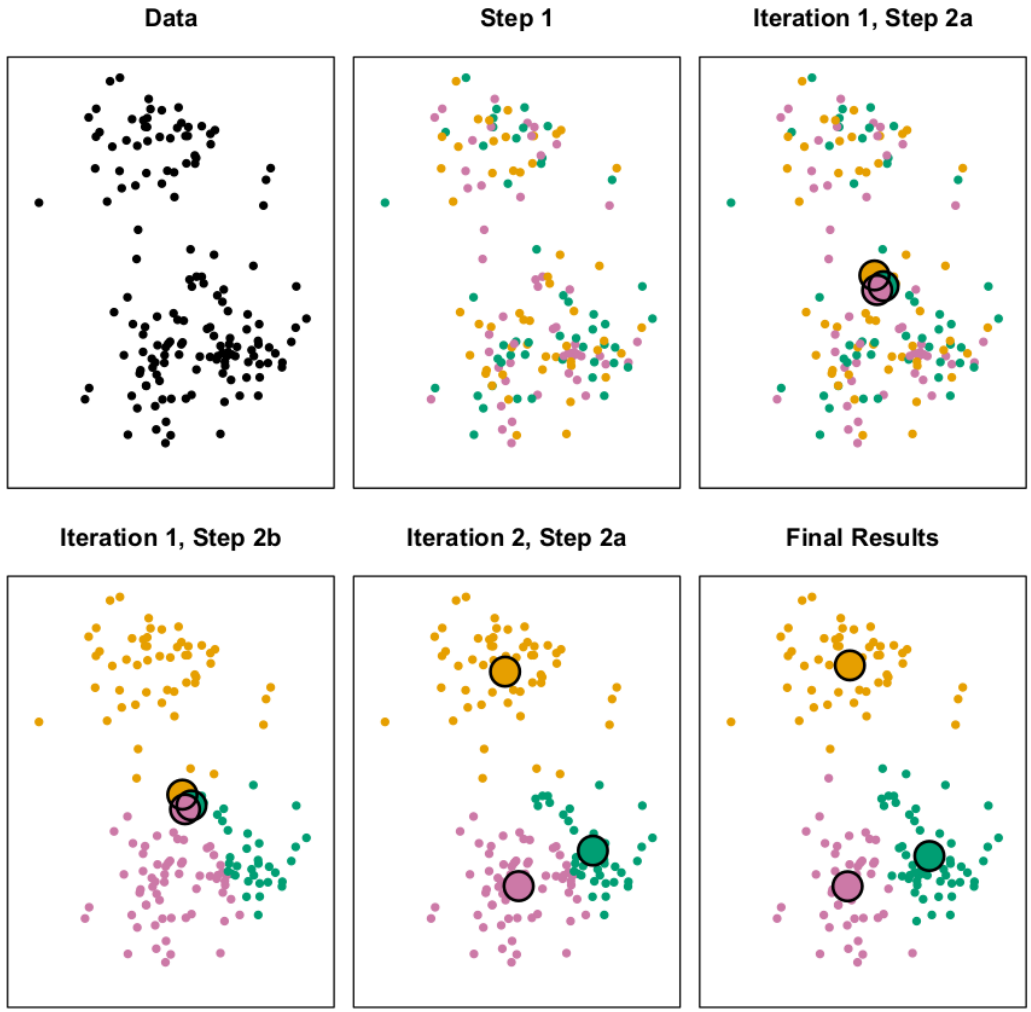


Figure: From James, G. et al. An intro. to Stat. Learning. Springer, 2023.

Apprentissage non-supervisée: Clustering K-Means

Différentes partitions initiales peuvent donner des solutions différentes:

Figure: From James, G. et al. An intro. to Stat. Learning. Springer, 2023.

Conclusion

- Dans ce cours nous avons vu plusieurs algorithmes d'apprentissage automatique.
- Quelques algorithmes supervisés pour la classification, d'autres pour la régression
- Ainsi que deux examples d'algorithmes non-supervisés.
- Il en existes beaucoup d'autres comme par example les:
  - *Support Vector Machines* et la régression logistique pour l'apprentissage supervisé
  - Le clustering hiérarchique pour le non-supervisé.