TP1 : Régression polynomiale

Le but de ce TP et d'approximer un jeu de données généré par la fonction suivante :

$$f(x) = 0.1 * x^3 - 0.5 * x^2 - x + 10 + bruit$$

Notez bien que la fonction objectif (supposée inconnue et à découvrir) est bruitée.

Etape 1 : générer l'ensemble d'apprentissage

Initialiser la graine aléatoire à 0 pour avoir tous les mêmes valeurs ($set.seed(0)$). Nous allons créer une fonction *generate* qui nous retourne un ensemble de n points à partir de la fonction $f(x)$. Le bruit est représenté par une variable aléatoire gaussienne de moyenne $0,5$. Pour créer cette fonction, on va suivre les étapes suivantes :

- Générer un vecteur x contenant n valeurs aléatoires uniformément dans $[-10, 10]$ (utiliser $runif$)
- Générer un vecteur y des n valeurs des points précédents selon la fonction $f(x) = 0.1 * x^3 - 0.5 * x^2 - x + 10 + bruit$ (utilisez la fonction $rnorm$).
- A partir de ces deux vecteurs, retourner une matrice de n lignes et 2 colonnes.

Utilisez la fonction *generate* pour :

- Générer un ensemble d'apprentissage composé de 15 points
- Visualiser les points dans un plot

Etape 2 : réaliser un modèle de prédiction

Nous allons postuler qu'il existe un modèle polynomial $g(x, w) = w_0 + w_1 x + w_2 x^2 + ... + w_M x^M$ avec x le vecteur des variables d'entrée, w les paramètres du modèle et M son degré.

Pour créer une régression linéaire simple avec R, on utilise :

```
> model1 <- lm(y ~ x)
> print(model1)
```

Pour une régression polynomiale de degré M, on utilise :

```
> modelM <- lm(y ~ poly(x,M))
```

— Générer 5 modèles de régression avec $M \in \{1, 2, 3, 6, 12\}$. Ensuite, générer un vecteur z contenant une séquence de 300 valeurs allant de -10 à 10. Utiliser pour cela :
`> seq(from=,to=, length.out=)`
— Réaliser des prédictions sur z avec la fonction *predict* sur chacun des 5 modèles précédents. Par exemple
`> predict(model1, data.frame(x = z))`
— Afficher chacune des prédictions sur le plot avec la fonction lines.
```
> lines(z,predict(model1, data.frame(x = z)),
 col="green",lty=1).
```
— Quel modèle colle le mieux aux données d'apprentissage ?
— Quel phénomène observe ton avec le modèle de degré 12 ?

Etape 3 : sélection du modèle

Nous allons maintenant chercher le meilleur modèle qui permet de généraliser sur l'ensemble de validation (test).

— Créer un ensemble de test de 1000 points avec la fonction *generate* de l'étape 1
— Afficher ces points avec plot

On définit l'erreur quadratique moyenne d'un modèle $g(x, w)$ sur un ensemble donné par la formule suivante :

$$EQM = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - g(x_i, w))^2}$$

— Pour chaque modèle polynomial de degré allant de 1 à 14. Calculer l'erreur sur l'ensemble d'apprentissage qu'on note (EQMA) et l'erreur sur l'ensemble de validation (test) qu'on note (EQMT).
— Afficher la courbe qui montre l'évolution des EQMA et EQMT en fonction du degrè du modèle sur un même graphique.
— Qu'observez vous pour EQMA.
— Qu'observez vous pour EQMT.
— Quel est le meilleur modèle.
— Faites varier le nombre d'exemples d'apprentissage (en l'augmentant petit à petit). Que constatez vous ?

Cas pratique (bonus)

Supposons, qu'un site e-commerce souhaite savoir si le temps de chargement des pages web de son site impacte le montant du panier moyen de l'internaute. On souhaite trouver une fonction de prédiction qui modélisera cette corrélation. Pour celà on dispose de deux fichiers "dataTrain.csv" et "dataTest.csv" contenant 700 et 300 observations respectivement. Une observation est un couple de données (temps de chargement en secondes, montants des achats en euros).

- Charger ces deux fichiers dans des dataFrame puis visualiser les données d'entraînement dans un plot.
- La corrélation est elle linéaire ? Donner le meilleur modèle qui permet de prédire le montants d'achats en fonction du temps de chargement.