# Table des matières

**1 Apprentissage Supervisé** **2**
1.1 Les Problèmes de Régression . . . . . 2
1.1.1 La Régression Linéaire . . . . . 2
1.1.2 Methode . . . . . 3
1.1.3 La fonction de cout . . . . . 4
1.1.4 Descente graduelle . . . . . 5

**2 Implémentations** **6**
2.0.1 Machines à Support Vecteur (SVM) . . . . . 6

# 1 Apprentissage Supervisé

## 1.1 Les Problêmes de Régression

En intelligence artificielle, la notion regression est la procedure qui, etant donne un entre x $\in$ R$^d$ et y $\in$ R, permet de determine approximativement la fonction f qui caracteise la relation qui existe entre x et y. Ainsi la determination de cette fonction implique la notion de fonction parametrique.

### 1.1.1 La Régression Linéaire

La regression lineaire est un probleme de regression auquel le model ou la fonction depend lineairement de ces parametres. Les differents type de regression lineaire sont la regeression lineaire affine, la regeression lineaire polynomiale et la la regeression lineaire a fonctions de base radiales. Il existe deux types fondamentaux de regression lineaire qui sont : la regression lineaire affine et la regression lineaire polynomiale.

1. Une regression lineaire de parametre $\theta$ est dite affine si pour tout x $\in$ R$^d$

$$\mathrm{f}_\theta(\mathrm{x}) = \theta_0 + \theta_1 x = [\theta_0, \theta_1]\begin{bmatrix}1\\x\end{bmatrix}, \qquad \text{avec } \theta_0 \in \mathrm{R} \text{ et } \theta_1 \in \mathrm{R}^d$$

Soit l'expression suivante :

$$y = \mathrm{f}_\theta(x) + \varepsilon \quad \text{avec } \varepsilon \sim N(0, \sigma^2).$$

Dans cet expression nous assumons que f est la fonction que nous allons estimer a partir de son parametre $\theta$ et qui nous permettra de faire nos prediction pour chaque element donne a partir le domaine d'entrainement. Nous noterons par f$^c$ comme etant la fonction estimee de f.

Cette expression nous amene aussi a faire l'etude de l'une des idees fondamentales les plus importante dans cette partie du cours qui est d'optimiser le parametre $\theta$. Bien qu'il existe plusieurs methodes d'optimiser cet parametre, nous allons nous interesser par la methode l'estimation du maximum de vraissemblance qui interpelle, bien sur la notion de probabilite conditionnellle.

Pour une suite de points $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ representant le domaine d'entrainement nous assumons que le $y_i$ suivent une loi normale et qu'ils sont aussi independante identiquement distribues pour estimer le parametre $\theta$ dans l'expression $y = \mathrm{f}_\theta(x) + \varepsilon$.

Alors nous avons $X = \{x_1, x_2, ..., x_n\} \in R^{n*d}$ et $y = \{y_1, y_2, ..., y_n\} \in R^n$.

Determinons le parametre $\theta^*$ qui maximise la vraisemblance

$P(y_1, y_2, .., y_n|x_1, x_2, ..x_n, \theta) = P(y|X, \theta) = \prod_i^n P(y_i|x_i, \theta)$ avec $y_i \sim N(\theta^T x, \sigma^2)$

Dans ce cas nous avons $P(y_i|x_i, \theta) = \frac{1}{\sigma\sqrt{2\pi}}\exp(-\frac{(yi-\theta^T x)^2}{2\sigma^2})$

D'ailleurs, nous savons que, la fonction logarithme est strictement croissante ce qui implique que le maximum de la vraisemblance est aussi celui de la log-vraisemblance. Ainsi, en appliquant le logarithme de la vraissemblance, nous avons

$$\log \mathrm{P}(y|X,\theta) = \sum_{i=1}^{n} \log \mathrm{P}(y_i|x_i,\theta)$$

Pour chaque $i \in \{1, 2, ..., n\}$ , $\log \mathrm{P}(y_i|x_i,\theta) = -\frac{(yi-\theta^T x)^2}{2\sigma^2} + \text{cte}$

$$\implies \log \mathrm{P}(y|X,\theta) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(yi-\theta^T x)^2 + \text{cte}$$

$$= -\frac{1}{2\sigma^2}(y-\theta X)^T(y-\theta X) + cte$$

Ainsi la dérivée partielle du logarithme de la vraissemblance par rappport à $\theta$ est alors donnée par :

$$\frac{\partial \log \mathrm{P}(y|X,\theta)}{\partial\theta} = \frac{\partial}{\partial\theta}\left(-\frac{1}{2\sigma^2}(y-\theta X)^T(y-\theta X) + cte\right)$$

$$= \frac{1}{\sigma^2}X^T(X\theta - y)$$

Maintenant le fait de resoudre l'équation $\frac{\partial \log \mathrm{P}(y|X,\theta)}{\partial\theta} = 0$ nous permettra de trouver le meilleur $\theta$. C'est exactement le parametre $\theta$ qui annule l'équation.

$\frac{\partial \log \mathrm{P}(y|X,\theta)}{\partial\theta} = X^T(X\theta - y) = 0$

$\implies X^T X\theta = X^T y$ et en supposant que la matrice $X^T X$ est inversible nous avons

$\implies \theta = (X^T X)^{-1}X^T y \qquad \square$

2. Une regression lineaire de parametre $\theta$ est dite polynomiale si pour tout x $\in \mathrm{R}^d$

$$\begin{aligned} \mathrm{f}_\theta(\mathrm{x}) &= \theta_0 + \theta_1 x^1 + ... + \theta_m x^m \\ &= [\theta_0, \theta_1, ..., \theta_m]\begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_m \end{bmatrix} \\ &= \sum_{i=0}^{m}\theta_i x^i \end{aligned}$$

Pour cette partie nous assumons que nos attribues sont representes sous la forme $y = \phi(x)^T\theta + \varepsilon$

### 1.1.2 Methode

Dans cette partie, notre hypothese ou model sera definie par :

$$y = aX + b$$

Nous savons que l'equation ci-dessus est une équation d'une ligne que vous avons étudiée au lycée. $a$ est la pente de la droite et $b$ est l'ordonnée à l'origine. Maintenant, nous allons utiliser cette équation pour entraîner notre modèle etant donné une ensemble de donnee et ensuite prédire la valeur de Y pour toute valeur de X donnée. Notre défi est dans ce cas de déterminer la valeur de $a$ et $b$, de sorte que la droite correspondant à ces valeurs soit la meilleure ligne d'ajustement.

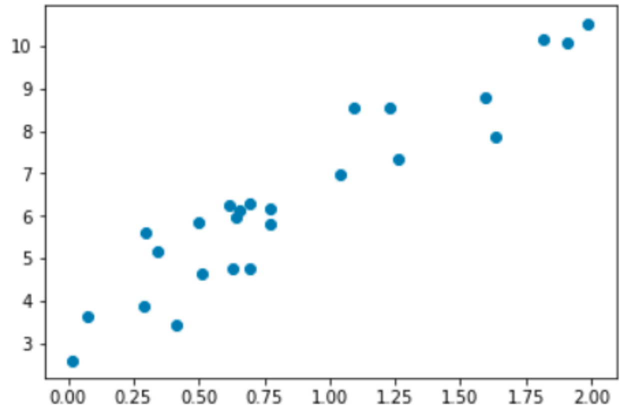


FIGURE 1 – représentation graphique d'un exemple de données dentrainement

#### 1.1.3 La fonction de cout

Dans la regression lineaire, la fonction de perte est definie comme etant la fonction qui permet de dire si les parametres $a$ et $b$ que nous venons de trouver sont les meilleurs parametres ou bien nous pouvons encore les ameliorer. Il existe differentes manieres de definir une fonction de perte mais dans ce document allons utiliser (EQM) l'erreur quadratique moyenne (MSE mean square error en anglais) pour definir notre fonction d cout. L'erreur quadratique moyenne entre le vraie $y$ et le $\hat{y}$ que nous avons predit est donne par :

$$E = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2$$

$$E = \frac{1}{n}\sum_{i=1}^{n}(y - (aX + b))^2$$

Ou n est la longueur du vecteur $y$ et $\hat{y}$

Par consequent, les parametres $a$ et $b$ qui correspondrons a la meilleure ligne d'ajustement seront tout simplement les parametres qui minimisent la fonction de perte $E$. Pour cela nous allons introduire une methode utilisee le plus souvent pour minimiser un tel cas de fonction a savoir la descente de gradient.

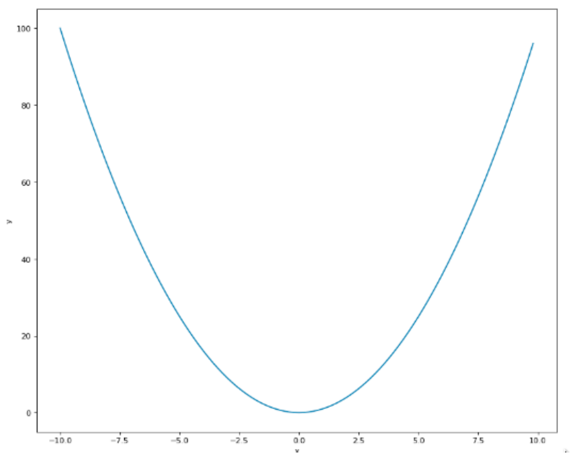

FIGURE 2 – représentation graphique d'une fonction convexe

### 1.1.4 Descente graduelle

La descente graduelle est une methode qui est beaucoup utiliser pour la minimisation de fonctions convexes.

Pour emtamer cette procedure, nous allons commencer par initialiser les $a$ par zero et $b$ par zero. Ensuite nous calculons la derivee partielle de $E$ par rapport a $a$, $\frac{\partial E}{\partial a}$ et ensuite par rapport a $b$, $\frac{\partial E}{\partial b}$.

$$\frac{\partial E}{\partial a} = -\frac{2}{n}\sum_{i=1}^{n} X_i(y_i - aX_i - b)$$
$$\frac{\partial E}{\partial b} = -\frac{2}{n}\sum_{i=1}^{n}(y_i - aX_i - b)$$

Maintenant, pour mettre a jour nos parametres nous allons repeter le processus ci-dessous jusqu'a ce que la fonction de perte soite tres proche de 0 ou egal a 0.

$$a = a - r * \frac{\partial E}{\partial a}$$
$$b = b - r * \frac{\partial E}{\partial b}$$

Les valeurs de $a$ et $b$ que nous avons trouves maintenant seront les valeurs optimales que nous noterons par $a^*$ et $b^*$ respectivement.

Alors concernant l'example de la figure **??**

# 2 Implémentations

```
class LinearRegression():
  def __init__ (self):
     pass
  def fonction_cout(self, y_vrai, y_predit):
     definit une fonction de perte et return sa valeur

  def algorithme(self, x, y, taux_apprentissage, nombre_iteration):
     - initialiser les paramtres \tehta_0 et \theta_1

     - for i in range(nombre_iteration):

           * predit,
           * calcule la fonction de cout,
           * mise a jour les parametre \tehta_0 et \theta_1
     return \tehta_0 , \theta_1

  def prediction(self, x):
      y_predit = \theta_0.T X + \theta_1
      return y_predit
```

#### 2.0.1 Machines à Support Vecteur (SVM)