Régression linéaire :

* Fonction de prédiction :

predict(x) = theta0 + (theta1 \* x)

* Fonction de coût :

(m = la len des données)

m

J(theta0, theta1) = 1/2m E (predict(xi) - yi)2

i=0

predict(x) étant la fonction qui va prendre les thetas pour prédire :

Donc la fonction de coût:

m

J(theta0, theta1) = 1/2m E ((theta0 + (theta1 \* xi)) - yi)2

i=0

Les dérivées partielles sont :

(lr = alpha)

m

lr/lr\*theta\_0 J(theta\_0, theta\_1) = 1/m E (h(xi) - yi)

i=0

m

lr/lr\*theta\_1 J(theta\_0, theta\_1) = 1/m E (h(xi) - yi) \* xi

i=0

Donc calcul theta\_0, theta\_1:

m

theta\_0 = lr \* 1/m E (predict(xi) - yi)

i=0

m

theta\_1 = lr \* 1/m E (predict(xi) - yi) \*xi

i=0

Par comparaison, régression logistique

* Fonction de prédiction:

predict(x) = g(theta\*\*T \* x)

avec:

g(z) = 1 / (1+e\*\*-z)

Donc **predict(x) = 1 / (1 + exp(-(theta.T \* x))**

-> theta: un par classifier classifier.

**-> T correspond au transpose d’une matrice comme le fait numpy**

**-> e means np.exp**

Pour prédire, on fait passer x par les 4 classifieurs, donc on utilise 4 fois predict() avec les 4 thetas returned par les dérivés: x appartient à la classe pour laquelle cette prédiction est la plus élevée.

* Fonction de coût:

m

J(theta) = -1/m E yi \* log(predict(xI)) + (1 - yi)log(1 - predict(xi))

i=1

-> pour 4 thetas.

* La dérivée partielle est :

m

lr/lr\*thetaj J(theta) = 1/m E (predict(xi) - yi) \* xj \*\*i

i=1

j représente ici le classifieur d’une classe : thetaj et xi ==> classifieur gryffondor et data gryffondor.

* donc pour calculer un theta:

m

thetaj = lr \* 1/m E (predict(xi) - yi) \* xj \*\*i

i=1

On fait passer X par les 4 classifieurs, donc on utilise 4 fois predict() avec les 4 thetas : x appartient à la classe pour laquelle cette prédiction est la plus élevée.

4 x 13 theta par feature