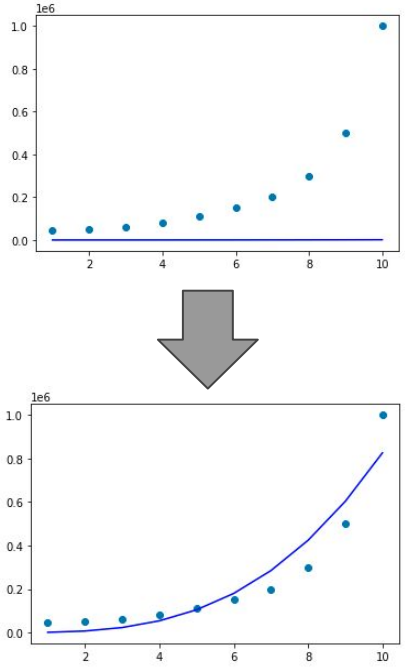
Premier modèle IA

Eva - Amaury

Sommaire

1. Rappels sur les régressions
2. Explication des fonctions
3. Evaluation des modèles
4. Comparaison scikitlearn/méthode normale
5. Conclusion

Rappel sur les régressions



Régression linéaire :

- Une variable explicative (feature)
- Une cible à déterminer (target)

Régression multiple :

- Plusieurs features
- Une target

Régression polynomiale

- Une variable mais plusieurs features (une par degré)
- Une target

Explication des fonctions

Fonction coût : mesure des erreurs du modèle

```
def fonc_cout (X, y, theta):
    '''Cette fonction renvoie une fonction coût pour une association target-feature-theta donnée.'''
    m = len(y)
    return 1/(2*m) * np.sum(((X.dot(theta)) - y )**2)
```

Gradient : vecteur représentant la pente le long de la courbe de coût

```
def grad (X, y, theta):
    '''Cette fonction retourne un gradient calculé à partir de feature(s) X, d'une cible y et d'un
    vecteur paramètre theta.'''
    m = len(y)
    return 1/m * X.T.dot(modele(X, theta) - y)
```

Explication des fonctions

Descente de gradient : optimisation du vecteur paramètre theta

```
def desc_grad (X, y, theta, alpha, nb_iter) :
    '''Cette fonction réalise l'algorithme de descente de gradient et retourne un theta optimisé
    en conséquence ainsi qu'un array numpy contenant les valeurs successives de la fonction coût
    durant l'apprentissage (pour visualisation éventuelle).'''
    evolution_cout = np.zeros(nb_iter)

    for i in range(0, nb_iter):
        theta = theta - alpha*grad(X, y, theta)
        evolution_cout[i] = fonc_cout(X,y,theta)

    return theta, evolution_cout
```

Explication des fonctions

Le modèle en lui-même : y prédit à partir du produit matriciel des features par le vecteur theta.

```
def modele (X, theta):
    '''Cette fonction retourne un vecteur colonne y prédit à partir d'une ou plusieurs feature(s)
    X et d'un theta.'''
    y_modele = np.dot(X,theta)
    return y_modele
```

Evaluation des modèles

Indicateurs de performance d'un modèle :

Coefficient de détermination R2 : part de variabilité expliquée

```
def coef_R2 (y, y_modele)
    coef_r2 = 1-((y-y_modele)**2).sum()/((y-y.mean())**2).sum()
    return coef_R2
```

Erreur quadratique moyenne: carré des écarts à la moyenne

```
from sklearn.metrics import mean_squared_error
```

La **fonction coût** peut aussi être un indicateur de performance (relatif).

Comparaison scikitlearn et méthode normale

Scikitlearn :

- Moins de risques d'erreur
- Moins d'efforts passée la prise en main de la bibliothèque
- Performance similaire sur ce type de modèles simples
- Interprétation du code plus rapide

Méthode normale :

- Permet de mieux comprendre les principes qui sous-tendent les techniques
- Beaucoup plus long et difficile
- Plus de risques d'erreurs