Machine Learning VIII : Minimisation du risque

Nicolas Bourgeois

1. **Outils de base**
    - Fonction de Perte
    - Risque empirique
    - Matrice de confusion
    - Courbe ROC

2. **Risque et Consistance**
    - ERM
    - Consistance
    - Exemple déroulé : kNN
    - Décomposition approximation-estimation

Fonction de Perte

Soit $LF : E \rightarrow \mathbb{R}_+$ une fonction de perte, telle que

$$LF(Y, Y) = 0$$

Et Φ une fonction d'agrégation
Objectif :

$$\min_f \Phi\left(LF(f(X_i), Y_i)\right)$$

Exercice

Exercice

Suggérez des fonctions de perte pour la régression

Exercice

Suggérez des fonctions de perte pour la classification

Solution

1)
$LF(X,Y) = (X - Y)^q$
$LF(X,Y) = \frac{|X-Y|}{|X+Y|}$

2)
$LF(X,Y) = \mathbf{1}_{X \neq Y}$
$LF(X,Y) = 1$ si $X > Y$, ϵ si $X < Y$, 0 sinon.

Risque empirique

Soit $LF : E \to \mathbb{R}_+$ une fonction de perte, telle que

$$LF(Y, Y) = 0$$

Et on fait la moyenne *sur l'échantillon observé* :
Objectif :

$$\frac{1}{N} \sum_{i \in I} (LF(f(X_i), Y_i))$$

Exercice

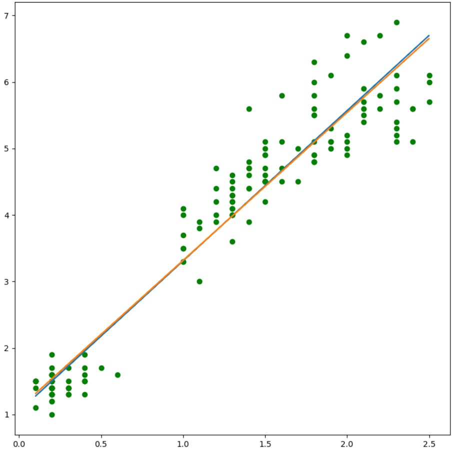
Exercice

En reprenant les données iris, programmez (sans utiliser scipy ou scikit-learn) une régression linéaire entre la longueur et l'épaisseur des pétales, en choisissant respectivement :

- $LF(X, Y) = (X - Y)^2$
- $LF(X, Y) = |X - Y|$

Affichez les deux droites sur le même graphe.

Résultat attendu



Solution

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
iris = pd.read_csv('data1.csv')
Y,X=iris.PetalLength,iris.PetalWidth
plt.scatter(X,Y,c="green")
for p in range(1,3):
    coefs,score=list(np.linspace(0,3,601)),[]
    for c in coefs:
        m=sum(Y[i]-c*X[i] for i in range(len(X)))/len(X)
        score.append(sum(abs(Y[i]-c*X[i]-m)**p for i in range(len(X)))
    s_opt=min(score)
    c_opt=coefs[score.index(s_opt)]
    m_opt=sum(Y[i]-c_opt*X[i] for i in range(len(X)))/len(X)
    plt.plot(X,c_opt*X+m_opt)
    print(c_opt,m_opt)
plt.show()
```

Exercice

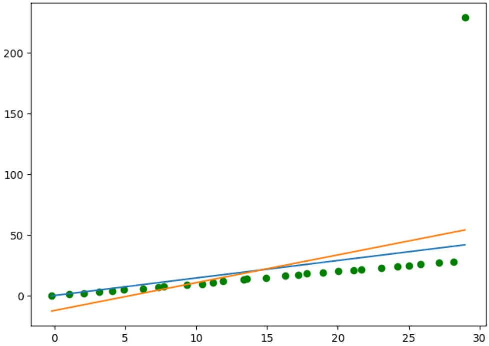
Exercice

Construisez un exemple pour lequel le choix de la fonction de perte a un gros impact sur la régression.

Solution

```
import numpy as np
import random
from matplotlib import pyplot as plt
X=[x+random.normalvariate(0,0.2) for x in range(30)]
Y=[x+random.normalvariate(0,0.2) for x in range(30)]
Y[29]+=200
plt.scatter(X,Y,c="green")
for p in range(1,3):
    coefs,score=list(np.linspace(0,3,601)),[]
    for c in coefs:
        m=sum(Y[i]-c*X[i] for i in range(len(X)))/len(X)
        score.append(sum(abs(Y[i]-c*X[i]-m)**p for i in range(len(X))))
    s_opt=min(score)
    c_opt=coefs[score.index(s_opt)]
    m_opt=sum(Y[i]-c_opt*X[i] for i in range(len(X)))/len(X)
    plt.plot(X,[c_opt*x+m_opt for x in X])
    print(c_opt,m_opt)
plt.show()
```

Solution



Principe

Dans le cas d'une classification supervisée, on considère :

$$c_{jk} = |i, g(X_i) = j \& Y_i = k|$$

Exercice

Exercice

Avec scikit-learn, entrainez un k-means sur l'iris set et produisez à la main la matrice de confusion associée.

Solution

```
from sklearn import datasets,cluster
iris = datasets.load_iris()
X,Y= iris.data,iris.target
km = cluster.KMeans(n_clusters=3)
km.fit(X,Y)
res = km.predict(X)
cm = [[len([i for i in range(len(Y)) if res[i]==j
  and Y[i]==k]) for j in range(3)] for k in range(3)]
print(cm)
```

exercice

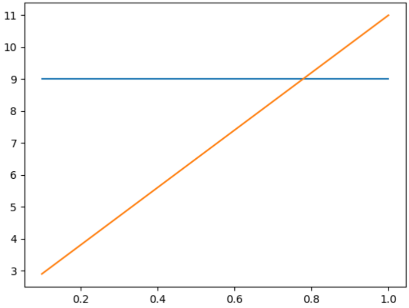
On considère deux modèles dont les matrices de confusion sont les suivantes :

40	5
4	40

43	2
9	35

Si nous considérons une loi de perte asymétrique $(1, \epsilon)$, pour quelles valeurs de ϵ le premier modèle est-il meilleur au sens du risque empirique que le second ? Faites une représentation graphique.

Résultat attendu



Solution

```
import numpy as np
from matplotlib import pyplot as plt
m1,m2=[[40,5],[4,40]],[[43,2],[9,35]]
eps=np.linspace(0.1,1,100)
y1=[m1[0][1]+x*m1[1][0] for x in eps]
y2=[m2[0][1]+x*m2[1][0] for x in eps]
plt.plot(eps,y1)
plt.plot(eps,y2)
plt.show()
```

1. **Outils de base**
   - Fonction de Perte
   - Risque empirique
   - Matrice de confusion
   - Courbe ROC

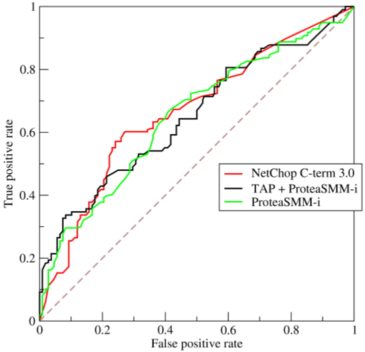
2. **Risque et Consistance**
   - ERM
   - Consistance
   - Exemple déroulé : kNN
   - Décomposition approximation-estimation

Receiver Operating Characteristic

On se place dans le cas d'une classification binaire

On trace la courbe du nombre de vrais positifs (sensibilité) sur le nombre de faux positifs (non-spécificité) par ordre décroissant de certitude.

Exemple


True positive rate
False positive rate
0
0.2
0.4
0.6
0.8
1
NetChop C-term 3.0
TAP + ProteaSMM-i
ProteaSMM-i

Exercice

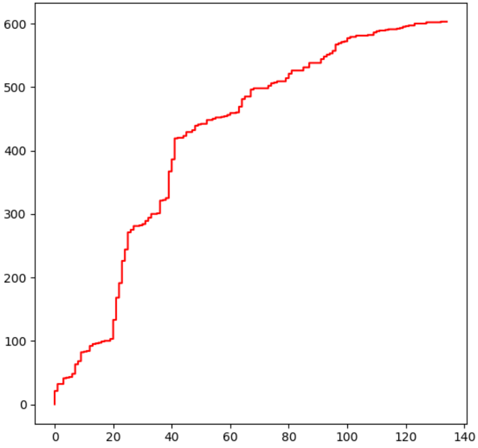
Exercice

Avec scikit-learn, entraînez un SVM à deux valeurs sur les données du titanic en prenant uniquement l'age et le prix du billet comme variable X, et bien sûr Y pour la survie. Puis produisez la courbe ROC associées.

Solution

```
from sklearn import svm
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
titanic = pd.read_csv("data2.csv").loc[:,['age','fare',
    'survived']].dropna()
X,Y=titanic[['age','fare']],list(titanic.survived)
svc=svm.SVC(probability=True)
svc.fit(X,Y)
pr=[(x[0],Y[i]) for i,x in enumerate(list(svc.predict_proba(X)))]
pr.sort(key=lambda x:x[0],reverse=True)
print(pr)
TP,FP=[0],[0]
abc = np.linspace(0,len(Y),len(Y)+1)
for i in range(len(Y)):
    TP.append(TP[-1]+(1 if pr[i][1]==0 and pr[i][0]>0.5 else 0))
    FP.append(FP[-1]+(1 if pr[i][1]==1 and pr[i][0]>0.5 else 0))
plt.plot(FP,TP,color='red')
plt.show()
```

Solution



Erreur du modèle

La bonne mesure serait de minimiser :

$$D(\tilde{f}) = \mathbb{E}(LF(\tilde{f}(X), Y))$$

Mais comme on ne connaît pas la loi de (X, Y) c'est impossible.

Erreur moyenne empirique

On dispose d'un échantillon de test $\tau = (X_j, Y_j)_{j \leq n}$.
On se rabat sur l'erreur empirique :

$$\tilde{D}(\tilde{f}, \tau) = \frac{1}{n} \sum_{j \leq n} LF(\tilde{f}(X_j), Y_j)$$

Loi des Grands Nombres

Si les (X_i, Y_i) sont i.i.d. alors

$$\frac{1}{n}\sum_{j\leq n} LF(\tilde{f}(X_j), Y_j) \longrightarrow \mathbb{E}(LF(\tilde{f}(X), Y))$$

Loi des Grands Nombres

Si les (X_i, Y_i) sont i.i.d. alors

$$\frac{1}{n}\sum_{j\leq n} LF(\tilde{f}(X_j), Y_j) \longrightarrow \mathbb{E}(LF(\tilde{f}(X), Y))$$

Sous cette hypothèse, il suffit donc d'un échantillon suffisamment grand.

Exercice

Exercice

Produisez des exemples de données non i.i.d. pour lesquelles cette convergence n'existe pas.

Risque Optimal

Risque théorique associé à un estimateur :

$$D(\tilde{f}) = \mathbb{E}(LF(\tilde{f}(X), Y))$$

Risque optimal :

$$ROPT = \min_{\tilde{f}} \mathbb{E}(LF(\tilde{f}(X), Y))$$

Risque Optimal

Risque théorique associé à un estimateur :

$$D(\tilde{f}) = \mathbb{E}(LF(\tilde{f}(X), Y))$$

Risque optimal :

$$ROPT = \min_{\tilde{f}} \mathbb{E}(LF(\tilde{f}(X), Y))$$

Consistance :

$$D(\tilde{f}) \longrightarrow ROPT$$

En résumé

Si les (X_i, Y_i) sont i.i.d. alors la moyenne empirique converge vers l'espérance du modèle

$$\frac{1}{n}\sum_{j\leq n} LF(\tilde{f}(X_j), Y_j) \longrightarrow \mathbb{E}(LF(\tilde{f}(X), Y))$$

Si le modèle est consistant alors l'espérance du modèle converge vers celle du modèle optimal

$$\mathbb{E}(LF(\tilde{f}(X), Y)) \longrightarrow \min_{\tilde{g}} \mathbb{E}(LF(\tilde{g}(X), Y))$$

Algorithme

σ_X est une permutation sur (X_i) telle que :

$$\forall i, d(X_{\sigma_x(i)}, x) \leq d(X_{\sigma_x(i+1)}, x)$$

$$knn(x) = argmax_{y \in Y} \mid \{x_{\sigma_X(i)}, i \leq k, Y_{\sigma(i)} = y\} \mid$$

Résultat

Si k est constant, knn n'est pas consistant.

Si $k \to \infty$ et $k/n \to 0$, knn est consistant

Exercice

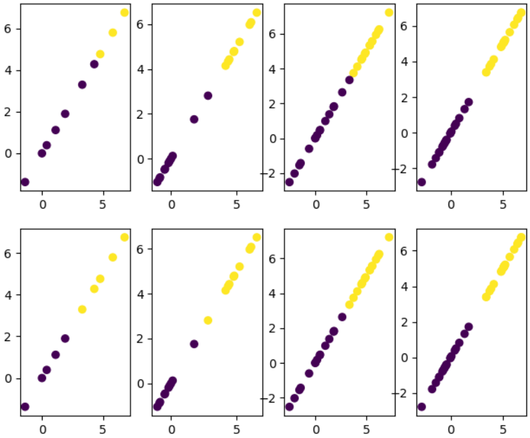
Exercice

Construisez à la main un set de variables aléatoires (X, Y) *vérifiant une relation* $Y = f(X)$*, puis un jeu d'observations biaisé* $(X_i)_{i\leq n}, (Y_i)_{i\leq n}$ *pour lequel il n'y a pas convergence de la moyenne empirique pour un 2nn.*

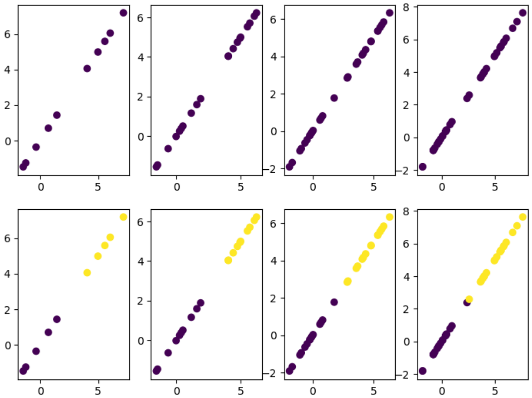
Exercice

Construisez à la main un set de variables aléatoires (X, Y) *vérifiant une relation* $Y = f(X)$*, puis un jeu d'observations iid* $(X_i)_{i\leq n}, (Y_i)_{i\leq n}$ *pour lequel il y a convergence de la moyenne empitique MAIS il n'y a pas consistance de 2nn et le modèle prédit mal.*

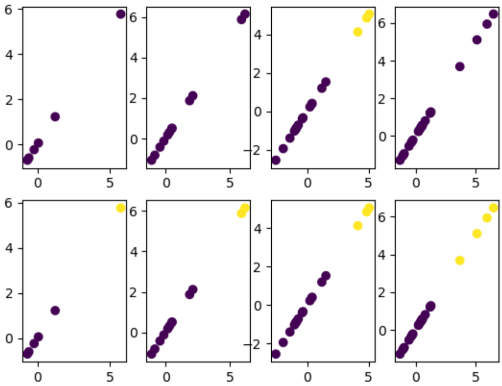
Resultat



Resultat



Resultat



Solution

```
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
from matplotlib import pyplot as plt
for n in range(1,5):
    X = np.concatenate((np.random.normal(0,1,n*5),
        np.random.normal(5,1,n*5)))
    Y = np.concatenate((np.full(n*5,0),np.full(n*5,1)))
    X_train, Y_train = X[:n*4].reshape(-1,1),Y[:n*4]
    knn = KNeighborsClassifier(n_neighbors=2)
    knn.fit(X_train, Y_train)
    fX = knn.predict(X.reshape(-1,1))
    plt.subplot(2,4,n)
    plt.scatter(X,X,c=list(fX))
    plt.subplot(2,4,n+4)
    plt.scatter(X,X,c=list(Y))
plt.show()
```

Solution

```
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
from matplotlib import pyplot as plt
for n in range(1,5):
    X = np.concatenate((np.random.normal(0,1,n*5),
        np.random.normal(5,1,n)))
    Y = np.concatenate((np.full(n*5,0),np.full(n,1)))
    I = list(np.random.randint(0,n*6-1,n*3))
    X_train,Y_train = X[I].reshape(-1,1),Y[I]
    knn = KNeighborsClassifier(n_neighbors=2)
    knn.fit(X_train,Y_train)
    fX = knn.predict(X.reshape(-1,1))
    plt.subplot(2,4,n)
    plt.scatter(X,X,c=list(fX))
    plt.subplot(2,4,n+4)
    plt.scatter(X,X,c=list(Y))
plt.show()
```

Minimiseur du reste empirique

Pour :

$$\tilde{D}(\tilde{f},\tau) = \frac{1}{n}\sum_{j\leq n} LF(\tilde{f}(X_j), Y_j)$$

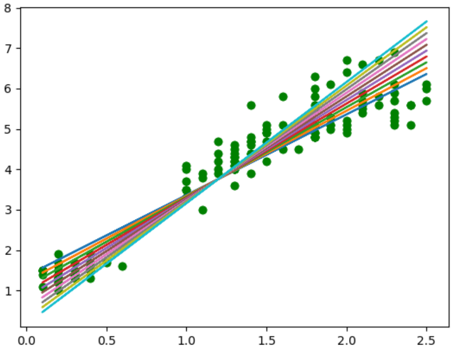
On peut prendre :

$$\tilde{F} = argmin_{\tilde{f}} \tilde{D}(\tilde{f},\tau)$$

Décomposition du risque

$$D(\tilde{F}) - ROPT = D(\tilde{F}) - \min_{\tilde{g} \in Z} \mathbb{E}(LF(\tilde{g}(X), Y))$$
$$+ \min_{\tilde{g} \in Z} \mathbb{E}(LF(\tilde{g}(X), Y)) - ROPT$$

exemples



exercice

Sur l'exemple de la régression iris précédente, suggérez (et implémentez) des solutions pour réduire, soit l'approximation au détriment de l'estimation, soit l'inverse.