Machine Learning
Python, R

Nicolas Loménie
http://www.math-info.univ-paris5.fr/~lomn/

13 avril 2018

Le sur-apprentissage

R

Cet exemple va illustrer le problème du sur-apprentissage (en R).

Algorithm 1: overfitting with R

$x \leftarrow 1:10$
$y \leftarrow x + c(-0.5, 0.5)$
$plot(x, y)$
$model1 < -lm(y \sim x)$
$z < -seq(1, 10, length.out = 250)$
$lines(z, predict(model1, data.frame(x = z)), lty = 1)$

Ajouter le graphique avec un modèle polynomial en x d'ordre 3, puis 9. (Utilisez la fonction $poly(x, n)$) Commentez. Changer un peu les données d'entrées $x \leftarrow c(1:5, 10:15)$. Commentez. Appliquez vos connaissances (TP1) pour améliorer vos graphiques à votre guise.

Machine Learning avec python

Algorithm 2: sklearn and digits

Data: from sklearn.datasets import load_digits
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.cross_validation import train_test_split

$digits = load\_digits()$
$X = digits.data$
$y = digits.target$
/* Comment est fait cet ensemble d'apprentissage */
printdigits
Data: import matplotlib.pyplot as plt
$ex = X[0].reshape(8, 8)$; // pour visualiser le vecteur en matrice image
$plt.gray()$
$plt.matshow(ex)$
$plt.show()$

Machine Learning avec python

Algorithm 3: sklearn and digits

```
/* Allez chercher plus loin dans le fichier d'autres
   exemples de chiffres                                  */
X_train, X_test, y_train, y_test = train_test_split(X, y)
gb = GradientBoostingClassifier(n_estimators = 20)
gb.fit(X_train, y_train)
print(gb.estimators_[0])
/* Optionnel car nécessite graphviz (visualisation de
   graphes) - installation requise                       */
Data: from sklearn.externals.six import StringIO
from sklearn.tree import export_graphviz
from IPython.display import Image
import pydot

tree0 = gb.estimators_[0][0])
dot_data = StringIO()
export_graphviz(tree0, outfile = "dot_data")
```

Machine Learning avec python

sklearn et circles

Algorithm 4: sklearn and non-linear mapping

Data: from sklearn.datasets import make_circles
from sklearn.svm import LinearSVC
from sklearn.cross_validation import train_test_split
from sklearn.metrics import accuracy_score

$X, y = make\_circles(n\_samples = 1000, noise = 0.11, factor = 0.4)$
$X_{train}, X_{test}, y_{train}, y_{test} = train\_test\_split(X, y)$
$svc = LinearSVC()$
$svc.fit(X_{train}, y_{train})$
$y_{pred} = svc.predict(X_{test})$
$print(accuracy\_score(y_{test}, y_{pred}))$

Machine Learning avec python

sklearn et circles

Algorithm 5: sklearn and non-linear mapping

```
def phi(x1, x2) :
        return np.array([x1, x2, x1 * *2 + x2 * *2])
X_New = phi(col1deX, col2deX) /* Comment récupérer des
    colonnes d'une matrice en Python?                    */
X_train, X_test, y_train, y_test = train_test_split(X_new, y)
svc = LinearSVC()
svc.fit(X_train, y_train)
y_pred = svc.predict(X_test)
print(accuracy_score(y_test, y_pred))
```