Machine Learning I

Nicolas Bourgeois

Nearest Neighbor

```
import numpy as np
from sklearn import datasets
from sklearn.neighbors import KNeighborsClassifier
iris = datasets.load_iris()
X, Y = iris.data, iris.target
np.random.seed(0)
indices = np.random.permutation(len(X))
X_train, Y_train = X[indices[:-10]], Y[indices[:-10]]
X_test, Y_test = X[indices[-10:]], Y[indices[-10:]]
knn = KNeighborsClassifier()
knn.fit(X_train, Y_train)
result = knn.predict(X_test)
print(result,Y_test)
```

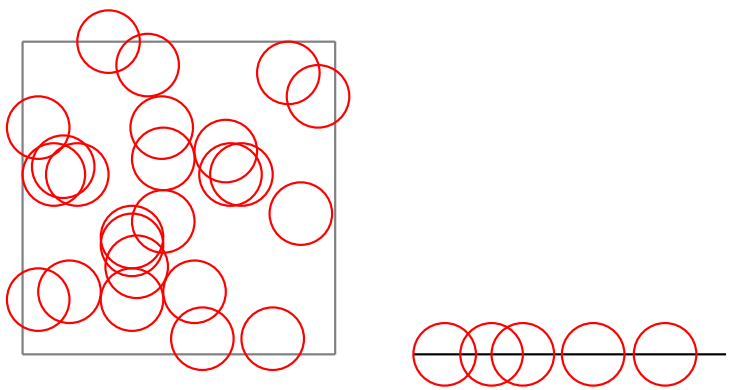
Malédiction de la dimension

Pour que l'estimateur soit efficace, il faut que la distance entre les points soit raisonnablement petite.

Malédiction de la dimension

Pour que l'estimateur soit efficace, il faut que la distance entre les points soit raisonnablement petite.

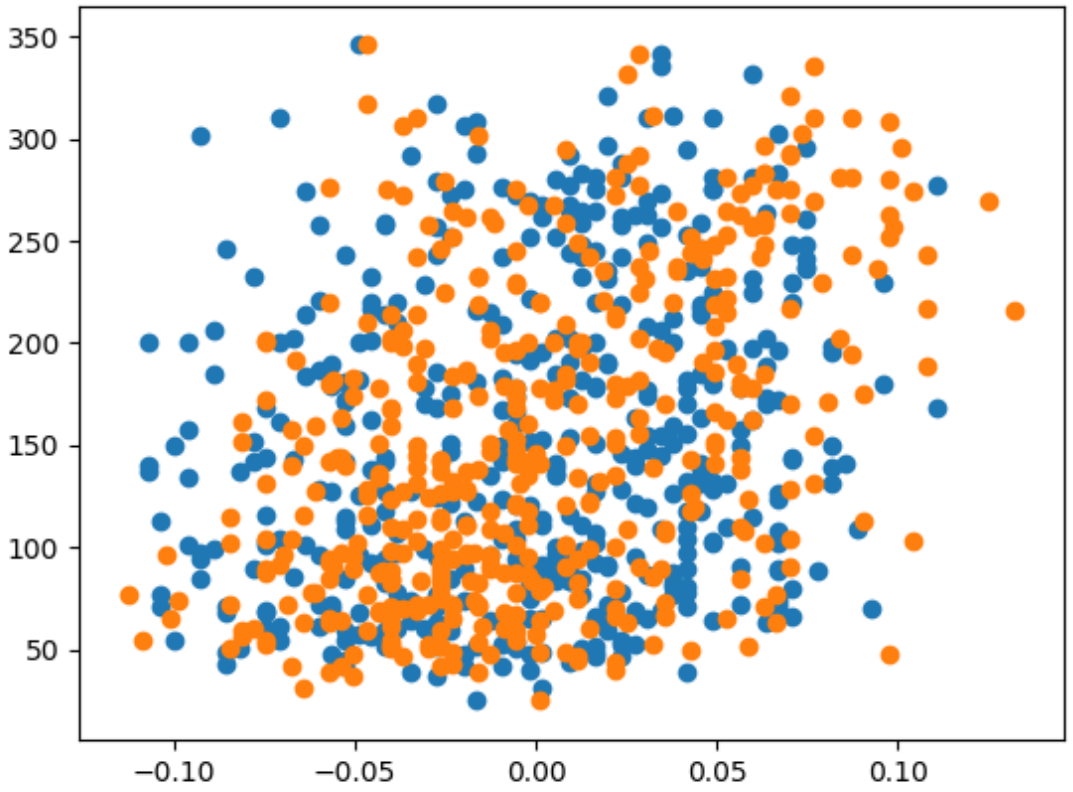
Donc le nombre de boules nécessaires pour couvrir croît exponentiellement avec la dimension (le nombre de variables).



Régression linéaire

```
from sklearn import linear_model, datasets
import matplotlib.pyplot as plt
import numpy as np
dbt = datasets.load_diabetes()
X_train,X_test=dbt.data[:-20],dbt.data[-20:]
Y_train,Y_test=dbt.target[:-20],dbt.target[-20:]
regr = linear_model.LinearRegression()
regr.fit(X_train, Y_train)
print(regr.coef_)
print(np.mean((regr.predict(X_test)-Y_test)**2))
print(regr.score(X_test, Y_test))
plt.scatter(dbt.data[:,0],dbt.target)
plt.scatter(dbt.data[:,3],dbt.target)
plt.show()
```

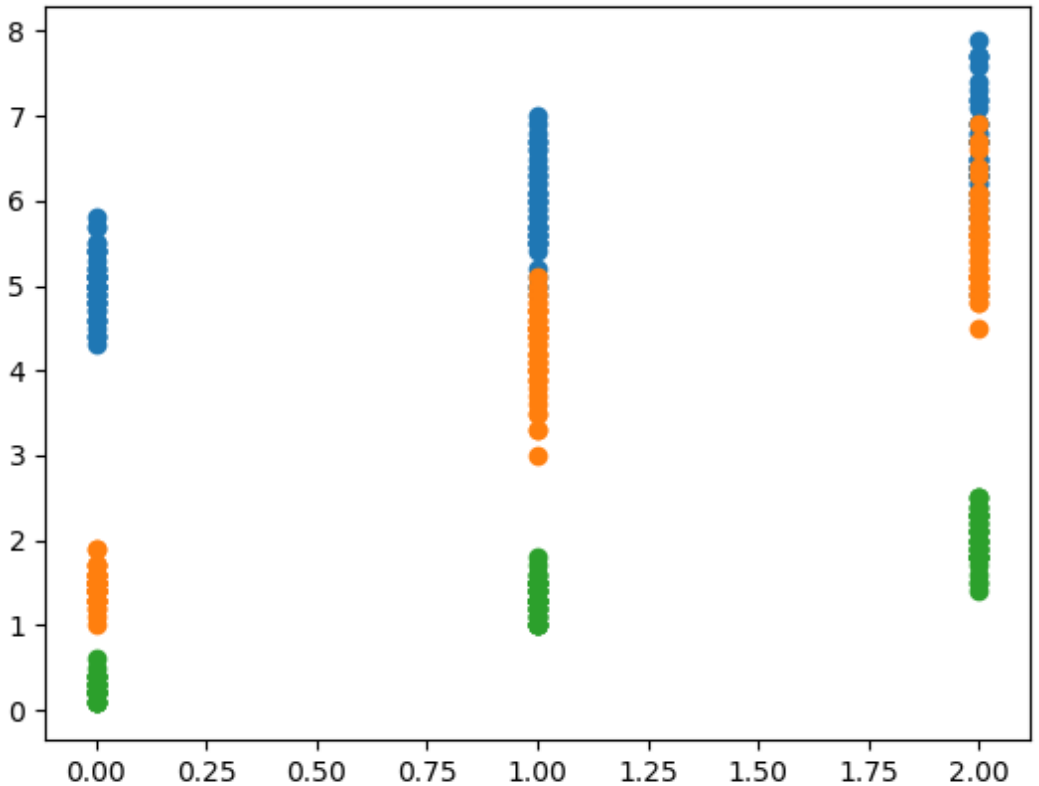
Régression linéaire



Régression logistique

```
from sklearn import linear_model, datasets
import matplotlib.pyplot as plt
import numpy as np
iris = datasets.load_iris()
X, Y = iris.data, iris.target
logistic = linear_model.LogisticRegression(C=1e5)
logistic.fit(X, Y)
print(logistic.coef_)
print(logistic.score(X, Y))
plt.scatter(Y,X[:,0])
plt.scatter(Y,X[:,2])
plt.scatter(Y,X[:,3])
plt.show()
```

Régression logistique



SVM

```
from sklearn import svm, datasets
import numpy as np
iris = datasets.load_iris()
X, Y = iris.data, iris.target
np.random.seed(55)
indices = np.random.permutation(len(X))
X_train, Y_train = X[indices[:-70]], Y[indices[:-70]]
X_test, Y_test = X[indices[-70:]], Y[indices[-70:]]
svc,svc2=svm.SVC(kernel='linear'),svm.SVC(kernel='rbf')
svc.fit(X_train, Y_train)
svc2.fit(X_train, Y_train)
results=(svc.predict(X_test),svc2.predict(X_test))
print(results[0]-Y_test,"\n",results[1]-Y_test)
```

SVM

```
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) on Windows (64 bits).
This is the Pyzo interpreter with integrated event loop for TK.
Type 'help' for help, type '?' for a list of *magic* commands.

>>> (executing file "sklearn4-svm.py")
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

>>>
```