

Machine learning en Python: scikit-learn

M. Ben & R. Tavenard

scikit-learn : documentation

- La documentation est de très grande qualité, utilisez-la (les exemples de ces diapos en sont issus) !

<https://scikit-learn.org/stable/index.html>

Préparation des données

- En scikit-learn:
 - X est un numpy array de taille (n, p)
n individus en dimension p
 - y est un numpy array de taille (n,) ou (n, p_out)
p_out : nombre de sorties attendues
- Pre-processing
 - <https://scikit-learn.org/stable/modules/preprocessing.html>
 - Les méthodes de pre-processing sont des objets de classe Transformers (cf. Transparent suivant)

Transformers

- Méthodes :
 - fit(X)
 - transform(X)
 - fit_transform(X)

```
>>> from sklearn.preprocessing import StandardScaler
>>> data = [[0, 0], [0, 0], [1, 1], [1, 1]]
>>> scaler = StandardScaler()
>>> scaler.fit(data)
>>> print(scaler.mean_)
[0.5 0.5]
>>> print(scaler.transform(data))
[[-1. -1.]
 [-1. -1.]
 [ 1.  1.]
 [ 1.  1.]]
>>> print(scaler.transform([[2, 2]]))
[[3. 3.]]
```

Transformers

- Appliquer des pre-processing différents selon les colonnes :
ColumnTransformer

```
numeric_features = ['age', 'fare']
cat_features = ['embarked', 'sex', 'pclass']

preprocessor = ColumnTransformer(
    transformers=[
        ('num', SimpleImputer(strategy='median'), numeric_features),
        ('cat', SimpleImputer(strategy='constant', fill_value='missing'), cat_features)])
```

Estimators

- Méthodes :
 - `fit(X[, y])`
 - `predict(X)`
 - `predict_proba(X)` (pas toujours...)

```
>>> X = [[0], [1], [2], [3]]
>>> y = [0, 0, 1, 1]
>>> from sklearn.neighbors import KNeighborsClassifier
>>> neigh = KNeighborsClassifier(n_neighbors=3)
>>> neigh.fit(X, y)
KNeighborsClassifier(...)
>>> print(neigh.predict([[1.1]]))
[0]
>>> print(neigh.predict_proba([[0.9]]))
[[0.66666667 0.33333333]]
```

Pipeline

- Objets qui permettent d'enchaîner des Transformers / Estimators

```
from sklearn import datasets
from sklearn.decomposition import PCA
from sklearn.linear_model import SGDClassifier
from sklearn.pipeline import Pipeline

logistic = SGDClassifier(loss='log', penalty='l2', early_stopping=True,
                        max_iter=10000, tol=1e-5, random_state=0)
pca = PCA(n_components=30)
pipe = Pipeline(steps=[('pca', pca), ('logistic', logistic)])

digits = datasets.load_digits()
X_digits = digits.data
y_digits = digits.target

pipe.fit(X_digits, y_digits)
```

Sélection de modèle

- But : comparer plusieurs modèles / choisir des valeurs d'hyper-paramètres
- GridSearchCV

```
from sklearn import svm, datasets
from sklearn.model_selection import GridSearchCV

iris = datasets.load_iris()

parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
svc = svm.SVC(gamma="scale")

clf = GridSearchCV(svc, parameters, cv=5)
clf.fit(iris.data, iris.target)
```


scikit-learn et le parallélisme

- Parallélisation « haut niveau » en scikit-learn
 - Argument `n_jobs`
 - Parallélisation gérée par `joblib`
 - Possibilité de choisir le backend de parallélisation (ex : `dask`)

```
from sklearn.cluster import DBSCAN
clustering = DBSCAN(eps=3, min_samples=2, n_jobs=2).fit(X)
```

```
from joblib import parallel_backend
with parallel_backend('dask', n_jobs=2):
    clustering = DBSCAN(eps=2, min_samples=2, n_jobs=2).fit(X)
```

scikit-learn et le parallélisme

- Parallélisation « bas niveau » utilisant OpenMP ou BLAS
 - Parallélisation des opérations de base (opérations codées en C, Cython, ou calculs matriciels)
 - Activée par défaut lorsque possible (avec nombre de threads maximal)

```
from threadpoolctl import threadpool_limits
```

```
with threadpool_limits(user_api='openmp', limits=2): #user_api='blas' existe aussi  
    kmeans = KMeans()  
    kmeans.fit(X)
```

scikit-learn et le parallélisme

- Attention : cas du Grid Search, de la cross-validation, etc.
 - 2 niveaux possibles de parallélisation
 - Nombre total de thread : produit des nombre de threads des 2 niveaux
- Sur un cluster partagé : peut réserver tous les coeurs sans s'en rendre compte
- Sur une machine perso : « over-subscription » et perte de temps car trop de threads

Conclusion

- Ce qu'il faut connaître de scikit-learn
 - Format de données attendu
 - Objets de base
 - Mécanismes de parallélisation
 - **S'habituer à lire les docs**