VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

Hands-on scikit-learn Examples

Georg Zitzlsberger georg.zitzlsberger@vsb.cz

26-03-2021

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research, Development and Education

MINISTRY OF EDUCATION, YOUTH AND SPORTS

Agenda

Demo: Intel Optimized scikit-learn

Cluster Example: Mean Shift

Model Selection Example: Cross-Validation

Demo: Intel Optimized scikit-learn

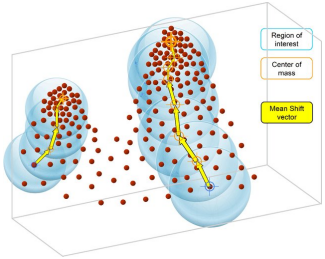
- Some functions are optimized with Intel DAAL:
  Demo with k-Means
- NumPy and SciPy are optimzied with Intel Math Kernel Library (Intel MKL):
  Demo with Kernel Prinicpal Component Analysis (KPCA)

Cluster Example: Mean Shift

- Centroid based method
- From neighborhood, find direction to increase density of points
- Shift the centroids toward higher densities based on Mean Shift vector:

$$m(x_i) = \frac{\sum_{x_j \in N(x_i)} K(x_j - x_i) x_j}{\sum_{x_j \in N(x_i)} K(x_j - x_i)}$$

- Can set number of clusters automatically
- Not for large scale problems due to repetetive searches

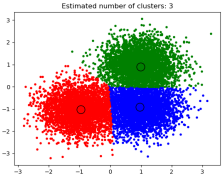


Airborne LiDAR Remote Sensing for Individual Tree Forest Inventory Using Trunk Detection-Aided Mean Shift Clustering Techniques, Chen, et al.

Cluster Example: Mean Shift

```python
import numpy as np
from sklearn.cluster import MeanShift, estimate_bandwidth
from sklearn.datasets import make_blobs

# Generate sample data
centers = [[1, 1], [-1, -1], [1, -1]]
X, _ = make_blobs(n_samples=10000,
                  centers=centers,
                  cluster_std=0.6)

# Compute clustering with MeanShift

# The following bandwidth can be automatically
# detected using
bandwidth = estimate_bandwidth(X,
                               quantile=0.2,
                               n_samples=500)

ms = MeanShift(bandwidth=bandwidth, bin_seeding=True)
ms.fit(X)
labels = ms.labels_
cluster_centers = ms.cluster_centers_

labels_unique = np.unique(labels)
n_clusters_ = len(labels_unique)

print("number␣of␣estimated␣clusters␣:␣%d" % n_clusters_)
```

```python
# Plot result
import matplotlib.pyplot as plt
from itertools import cycle

plt.figure(1)
plt.clf()

colors = cycle('bgrcmykbgrcmykbgrcmykbgrcmyk')
for k, col in zip(range(n_clusters_), colors):
    my_members = labels == k
    cluster_center = cluster_centers[k]
    plt.plot(X[my_members, 0], X[my_members, 1], col + '.')
    plt.plot(cluster_center[0],
             cluster_center[1], 'o', markerfacecolor=col,
             markeredgecolor='k', markersize=14)
plt.title('Estimated␣number␣of␣clusters:␣%d' % n_clusters_)
plt.show()
```

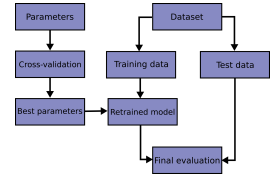


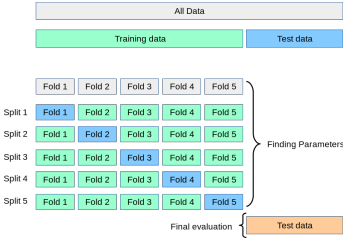
Python source code:

▸ Mean Shift Clustering

Model Selection Example: Cross-Validation

- To avoid/minimize overfitting, three data sets are used:
  - Training data: Which is repeatedly used to train a model over different sessions
  - Validation data: After each training session, evaluate the model performance
  - Testing data: After all sessions, evaluate the final model performance
- Function `train_test_split(...)` aids with creating the folds
- **Problem:** This removes quite a (large) number of samples from training and adds bias towards training and valiation data!
- **Solution:** k-fold Cross-Validation (CV)



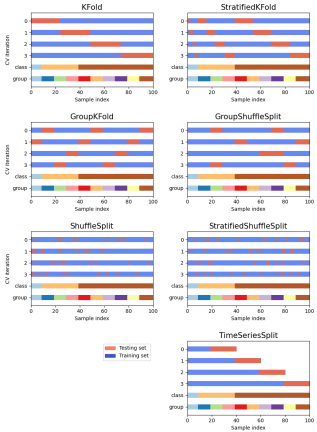


From scikit-learn documentation:

Cross-Validation Workflow

Model Selection Example: Cross-Validation

```
from sklearn.model_selection import (TimeSeriesSplit, KFold, ShuffleSplit,
                                     StratifiedKFold, GroupShuffleSplit,
                                     GroupKFold, StratifiedShuffleSplit)
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Patch
np.random.seed(1338)
cmap_data = plt.cm.Paired
cmap_cv = plt.cm.coolwarm
n_splits = 4

# Generate the class/group data
n_points = 100
X = np.random.randn(100, 10)

percentiles_classes = [.1, .3, .6]
y = np.hstack([[ii] * int(100 * perc)
               for ii, perc in enumerate(percentiles_classes)])

# Evenly spaced groups repeated once
groups = np.hstack([[ii] * 10 for ii in range(10)])

cvs = [KFold, GroupKFold, ShuffleSplit, StratifiedKFold,
       GroupShuffleSplit, StratifiedShuffleSplit, TimeSeriesSplit]
for cv in cvs:
    this_cv = cv(n_splits=n_splits)
    fig, ax = plt.subplots(figsize=(6, 3))
    plot_cv_indices(this_cv, X, y, groups, ax, n_splits)

    ax.legend([Patch(color=cmap_cv(.8)), Patch(color=cmap_cv(.02))],
              ['Testing␣set', 'Training␣set'], loc=(1.02, .8))
    # Make the legend fit
    plt.tight_layout()
    fig.subplots_adjust(right=.7)
plt.show()
```



Python source code:

▸ Visualizing cross-validation

VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

IT4Innovations National Supercomputing Center

VŠB – Technical University of Ostrava
Studentská 6231/1B
708 00 Ostrava-Poruba, Czech Republic
www.it4i.cz

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research, Development and Education

MINISTRY OF EDUCATION, YOUTH AND SPORTS