Build Your First Classifier

1 Introduction

Learning Goals

Familiarize yourself with scikit-learn (sklearn) and learn to

- implement a basic ML process using a MinDist classifier
- correctly sample data and know the difference between training and test data
- design reproducible experiments
- generate basic features

2 Minimum Distance Classifier

Training

Compute the class means $\boldsymbol{\mu}^{(c)}$ from the training data and store those. Those vectors are the representatives of each class.

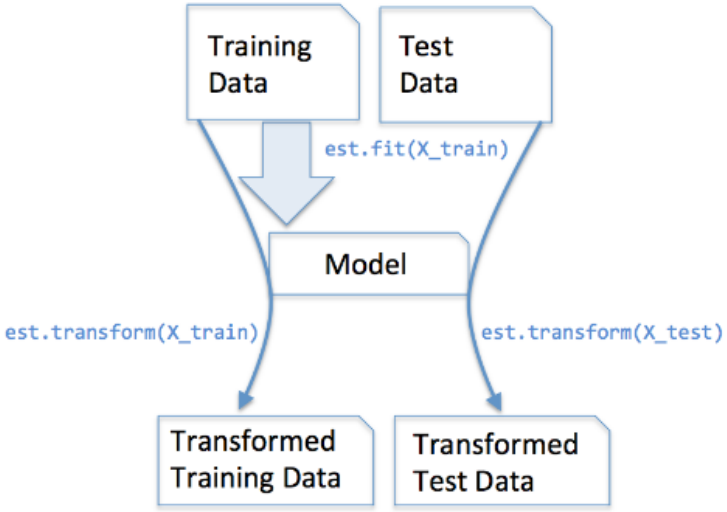
Testing

For each data $\boldsymbol{x}$ in test, predict the class by the class of the closest representative w.r.t. Euclidean distance,

$$\pi(\boldsymbol{x}) = \arg\min_{c \in \mathcal{L}} \left\| \boldsymbol{x} - \boldsymbol{\mu}^{(c)} \right\|,$$

where $\|\cdot\|$ denotes the Euclidean (standard) norm.

Recall the principles of the Minimum Distance (MinDist) classifier from the lecture. Take a look at scikit-learn's git repo to verify the classifier's implementation. Note that in scikit-learn, the MinDist classifier is called **Nearest Centroid classifier**.



Source: stackoverflow.com

3 Sklearn Essentials

A lot of sklearn objects such as preprocessors, vectorizers or scalers implement the following three methods:

- `fit`: fit the object to the current data, i.e. estimate statistics, model distributions, generate parameters etc.
- `transform`: apply model to the data using the extracted parameters from the fit-step
- `fit_transform`: short-hand method that calls and transform on the same data with just one call

Take, for example, the `StandardScaler`, which has two scaling parameters: **mean** and **standard deviation**. The scaling parameters should be **estimated from the training data** and **reused on the test data**. We want to test whether the assumptions we made from the training data StandardScaler, i.e. are valid also on the test data.
TLDR: Don't use `fit` on new data!

4 Tasks

Basic MinDist Classification

1. Load the wine data again, reusing the code from Lab 1. Only X and y are needed.
2. Implement the ML cycle using MinDist as classifier directly applied to the (unscaled) wine data.

```
from sklearn.neighbors import NearestCentroid
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

# TODO: apply sampling
mindist = NearestCentroid()
mindist.fit(X_train, y_train)
y_hat = mindist.predict(X_test)
# TODO: evaluate the performance
```

3. What accuracy score does your classifier achieve? Compare with your colleagues!
4. Does setting a random state change your results?

The following steps are intended to be implemented on top of each other, i.e. add sampling, add feature extraction, add scaling.

Sampling

1. Take a look into the sklearn documentation of the train_test_split method to find out the default split ratio, i.e. how much data is used for the training and test set, respectively.
2. Change the default setting to a ratio of 7:3 for training and test set.
3. Rerun the classification.

Feature Extraction

1. **Remove highly correlated features:** Identify the feature with highest correlation values throughout the dataset. Remove this feature from your dataset and rerun the classification. Does the accuracy score change?
2. Can you think of other feature extraction or generation methods?

Scaling

1. Apply standard scaling to the data and rerun the classification.
2. Does the accuracy change and if yes, why?

5 Homework

Reusing the same `random_state` throughout, track your experiments and note down the accuracies:

Short Description of Training Setting (*)	**Accuracy**
most basic setting (see listing above)	
highest correlating feature removed	
...	

(*) This could include: the classifier used, train-test split ratio, the preprocessing methods applied, any type of feature extraction or generation, ...

In the next labs, you will update this table with new figures, so make sure to save it (either directly in the notebook, as Excel file or on a sheet of paper)

Quiz

Take the quiz in Moodle. As some questions refer to the work discussed in this lab session, please make sure to have your have your code up and running!