Pattern Recognition
and Applications Lab

Parameter Estimation

Machine Learning – Course Laboratory

Battista Biggio

battista.biggio@diee.unica.it

Luca Didaci

didaci@diee.unica.it

Dept. Of Electrical and Electronic Engineering

University of Cagliari, Italy

University
of Cagliari, Italy

Department of
Electrical and Electronic
Engineering

Ex. 1: Testing performance on unseen data

1. Sampling a training and a testing set (from the same underlying distribution), e.g., splitting data x,y at random in x_tr, y_tr, x_ts, y_ts
   - `from sklearn.model_selection import train_test_split`

2. Normalizing training and test data (using parameters estimated on training data!)
   - `from sklearn.preprocessing import MinMaxScaler`

3. **Estimating classifier parameters on training data (today!)**

4. Fitting the classifier on training data
   - `clf.fit(x_tr, y_tr)`

5. Predicting the class labels of testing data - `clf.predict(x_ts)`

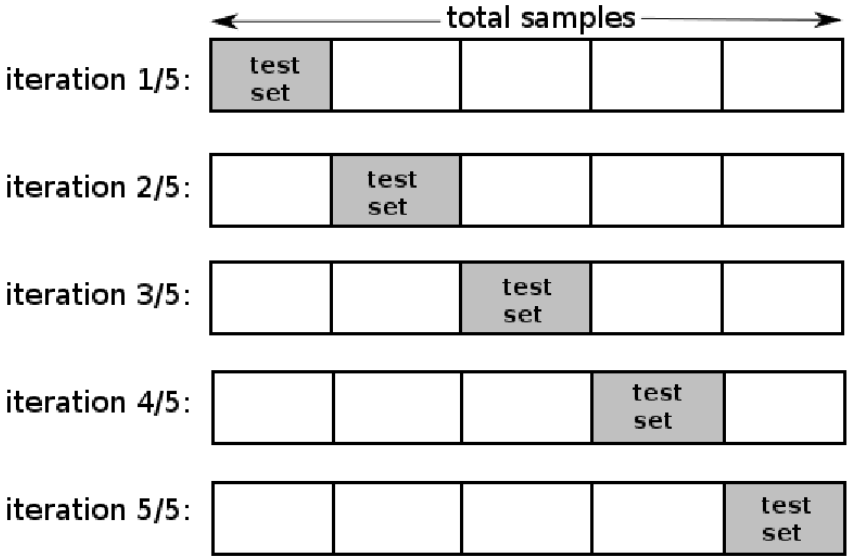
6. Evaluating accuracy or classification error

Ex. 1: Testing performance on unseen data

- In lab03/Ex_03.py, we used train_test_splits to create random data partitions
- **Ex. 1: use ShuffleSplit instead**
- http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.ShuffleSplit.html#sklearn.model_selection.ShuffleSplit
- *# define data splitter*
  splitter = ShuffleSplit(n_splits=5, train_size=0.4, random_state=0)
- and pass it to run_rep() instead of num_rep

Ex. 2: Parameter estimation with xval

- K-Fold Cross-validation (xval)
  - splits data into K folds
  - K-1 folds are used for training, 1 for testing
  - the process is repeated K times to estimate performance with a given set of classifier parameters (changing the test fold at every iteration)



1. The process is repeated for every set of classifier parameters
2. The best set of parameters is eventually selected
3. Classifier is trained using such parameters on the complete training set

Ex. 2: Parameter estimation with xval

- Consider the previous exercise, on the two-moon dataset
- Add cross-validation to tune the C parameter of the SVM classifier
- Use the following grid-search values for C:
  {0.001, 0.01, 0.1, 1, 10, 100}
- Add a grid search over the same values for the kernel parameter gamma
- How to deal with a variable number of parameters to optimize?
  - Each classifier has a different parameter set (C for linear SVMs, C and gamma for the SVM with the RBF kernel)
  - We should write a for loop for each parameter...

Ex. 3: Digits

- Consider now the digit data
- Select splitter (for the tr-ts splits) and normalizer
- Select a set of classifiers
- Define their parameters to optimize (grid values)
- Estimate their best values with xval on the training set (for each tr-ts split)
- Estimate the (average) performance of each selected classifier on the test data (for each tr-ts split)

- **This is the complete implementation of the design procedure for a machine-learning system**

- Any other dataset could also be considered
  - The aforementioned procedure is not specific to the digit data

Lessons Learned

- Parameter estimation using k-fold cross validation

Student challenges:

1. Modify the function performance_estimation to handle a different metric than accuracy (e.g., AUC)
   - **Hint**: metric should be an object, passed as a parameter to the function performance_estimation

2. Implement a function that plots the average accuracy computed on a 5-fold cross validation vs different values of C for a linear SVM, on some dataset of your choice

3. Implement a function that plots the average accuracy computed on a 5-fold cross validation vs different values of C and gamma for an RBF SVM, on some dataset of your choice
   - This is a 2D plot. Accuracy should be displayed in colors. x- and y-axis are respectively C and gamma values