07_ML_Experiments_and_Fairness

1. Cross-Validation Implementation

Possible Problems

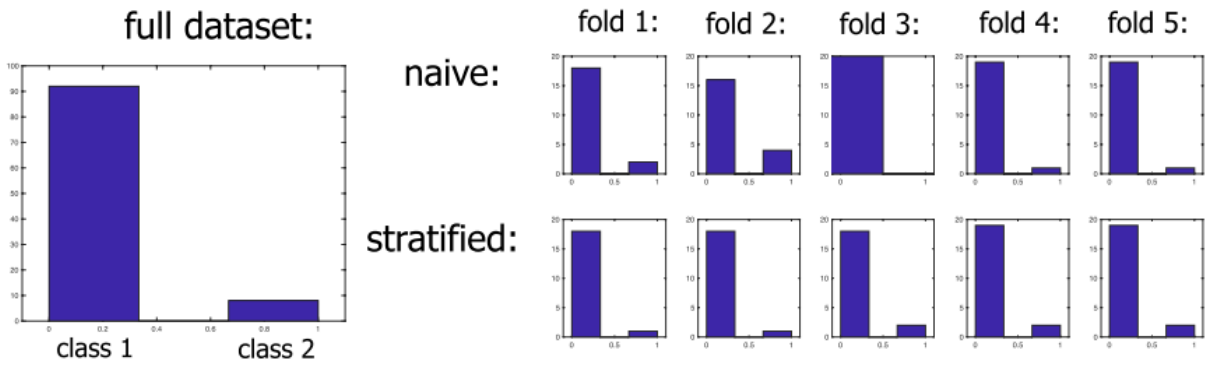
There are two classical scenarios in which the implementation of Cross-Validation need to be more careful.

- when classes are **heavily imbalanced**: which means the small class may lost in some of the folds
- when samples are **very correlated** (e,g, videos), which means the folds will contains elements from all "groups". Because element in a single group may be very similar (video frames in a given period), the learned agent actually do something not by features, just by mechanically memorize.

Stratified Cross-Validation

Try to get the same distribution in each fold

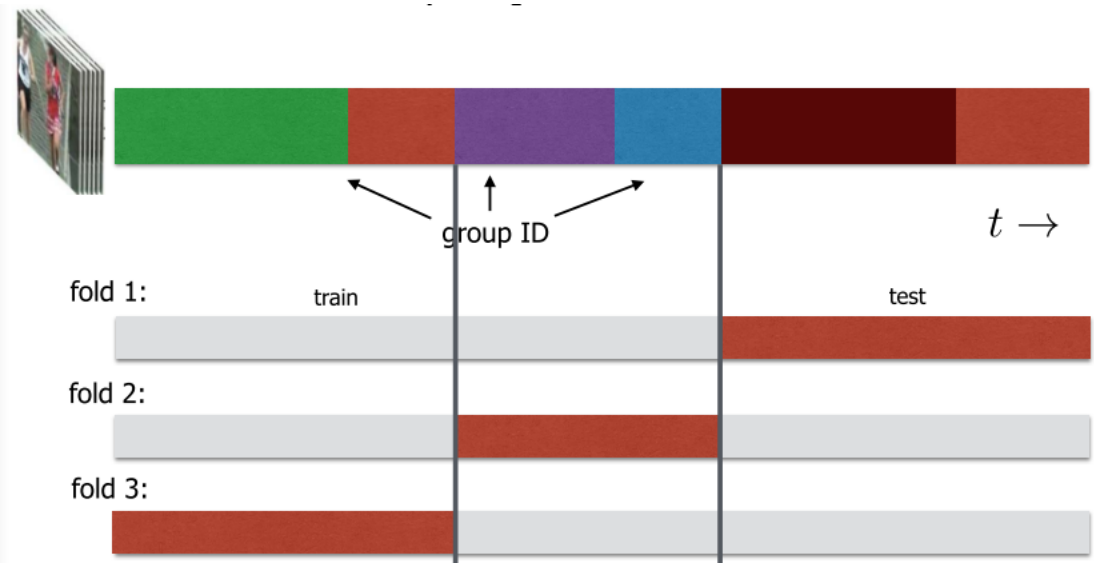
1. get the data from one class
2. split in K folds
3. combine the data from the different classes



Leave-One-Group-Out Cross-Validation

Samples within a group are heavily correlated, mixing them in training and testing gives (too) optimistic performance estimates

When generate fold, **each fold should have no elements from some groups.**



2. Paired T-test

Use T-test to test whether the performance of different model

Example

	fold 1	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
Method 1	2.1	19.1	11.0	4.2	10.6	2.8	12.9	9.1	10.2	11.8
Method 2	4.0	19.3	10.6	5.1	10.8	4.2	12.1	11.0	11.5	11.7

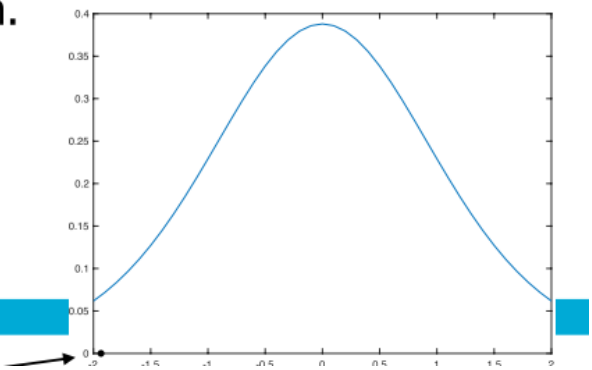
	fold 1	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
Difference	-1.9	-0.2	0.5	-0.9	-0.2	-1.4	0.9	-1.9	-1.3	0.1

- Are the averaged errors the same?
  =
  is the difference between the averaged errors zero?
- Need a test-statistic
- You can show that the variable $T = \frac{\bar{e_1} - \bar{e_2}}{\sigma_{e_1 - e_2}/\sqrt{k}}$

  has a Student-t distribution with (k-1) degrees of freedom.
- For us: T=-1.93

$$P(T \leq -1.93) = 0.0426$$

Just significant!



Machine learning, final remarks

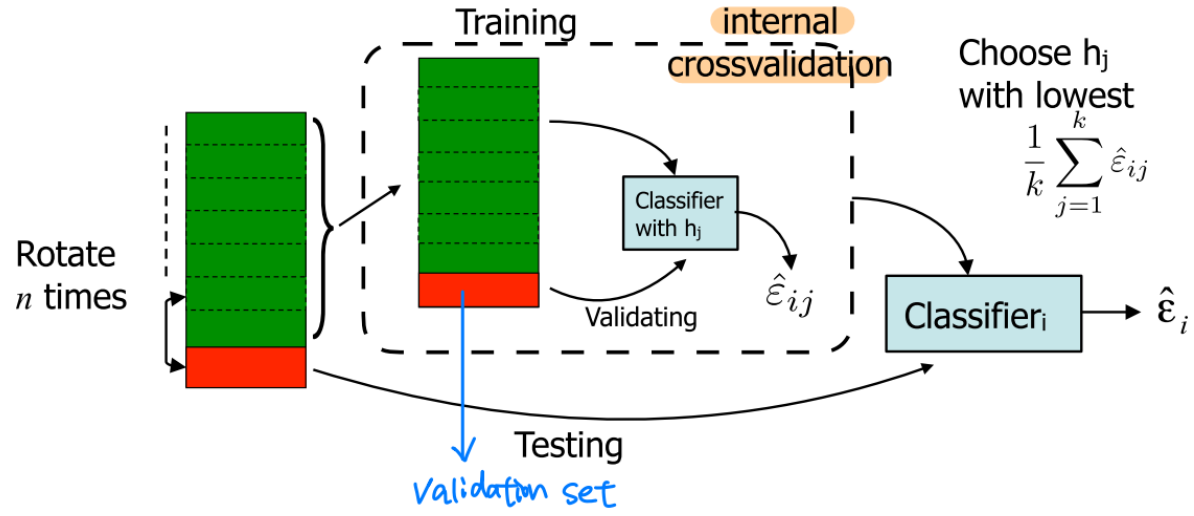
3. Hyperparameter Optimization

When implementing machine learning algorithm, we always has a lot of **hyperparameters to be optimized**. There are mainly two problem about the optimization of hyperparameters: how to **judge** a group of hyperparmeters and how to **search** for a good hyperparameter.

Judgement

Don't optimise hyperparameter by looking at the test set, that is a kind of cheating.

Instead, we should divide data into three part: tr**aining data, validation data and testing data**. More specifically, we can use **double cross-validation,** i.e. do cross-validation inside another cross-validation.



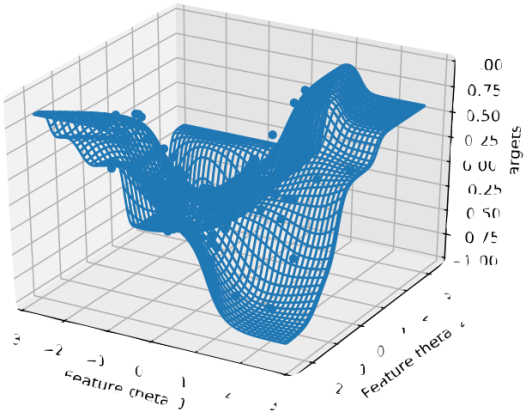
Search

Grid Search

Grid search is always the most basic algorithm.

Bayesian Optimization

For hyperparameter settings, we can first **randomly generate** groups of hyperparameter and test their results. Then we can use **Gaussian Process** to **fit the hyperparameter-performance space**. Then we can choose what we want (higher performance/lower uncertainty, etc.).



4. Presentation of Result

One important thing to be noticed that, when implementing or design a machine learning algorithm, **the minimum error often is not the most interesting thing**. We should try to understand the advantages/disadvantages.

- What errors are made? (inspect objects, inspect labels)
- What classes are problematic? (confusion matrix)

- Does adding training data help? (learning curve)
- How robust is the model

We should always repeat our experiments and give idea about **average** performance and **standard deviations**.

One important thing to be remembered: **Don't claim to have the overall 'optimal' classifier**

Summary

- Cross Validation: need be careful in some scenarios
  - heavily unbalanced: Stratified Cross-Validation
  - heavily correlated: Leave-One-Group-Out Cross-Validation
- Paired T-test: Compare different models (assume we have the test result on different fold of these different models)
- Hyperparameter Optimization
  - Compare: double cross-validation
  - Search: grid search + bayesian method (gaussian process)
- Presentation of result
  - Compare many aspects