Machine Learning
(IT3190E)

Quang Nhat NGUYEN

quang.nguyennhat@hust.edu.vn

Hanoi University of Science and Technology
School of Information and Communication Technology

Academic year 2020-2021

The course’s content:

- Introduction
- **Performance evaluation of ML system**
- Supervised learning
- Unsupervised learning
- Ensemble learning
- Reinforcement learning

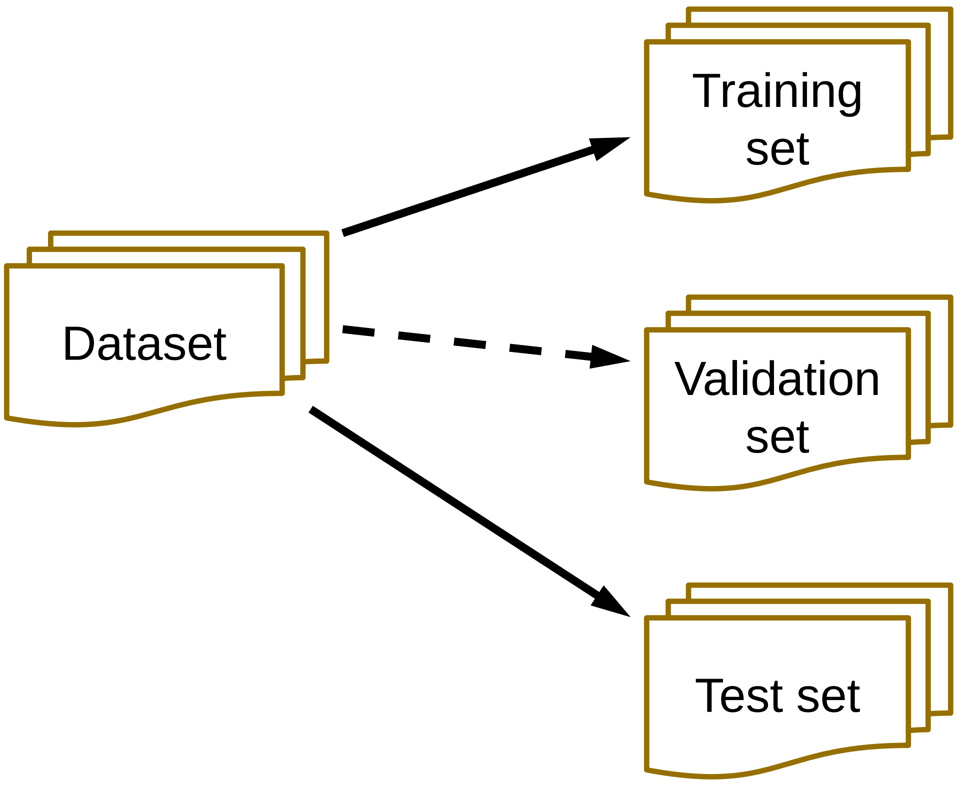
Performance evaluation (1)

- The evaluation of the performance of a ML system is usually done **experimentally** rather than analytically
  - An analytical evaluation aims at proving a system is correct and complete (e.g., theorem provers in Logics)
  - *But, it is impossible to build a formal definition of a problem to be solved by a ML system* (e.g., For a ML problem, what are correctness and completeness?)

Performance evaluation (2)

- The evaluation of the system performance should:
  - *Be done automatically by the system*, by using a set of test examples (i.e., a test set)
  - Not involve any test users
- Evaluation **methods**
  - → How to have a convincing/confident evaluation of the system performance?
- Evaluation **metrics**
  - → How to measure (i.e., to compute) the performance of the system?
  - → Different metrics for different types of problems (e.g., classification, regression, clustering)

Evaluation methods (1)



Used to train the system

Optional, and used to optimize the values of the system's hyper-parameters

Used to evaluate the *trained* system

Evaluation methods (2)

- How to get a confident/convincing evaluation of the system performance?
  - The larger the training set is, the higher the performance of the trained system is
  - The larger the test set is, the more confident/convincing the evaluation is
  - <u>Problem</u>: Very difficult (i.e., rarely) to have (very) large dataset(s)
- *The system performance depends on not only ML algorithms used*, but also:
  - Class distribution
  - Cost of misclassification
  - Size of the training set
  - Size of the test set

Evaluation methods (3)

- Hold-out (Splitting)
- Stratified sampling
- Repeated hold-out
- Cross-validation
  - *k*-fold
  - Leave-one-out
- Bootstrap sampling

Hold-out (Splitting)

- The whole dataset D is divided into 2 **disjoint subsets**
  - Training set D_train – To train the system
  - Test set D_test – To evaluate the performance of the trained sys.

  $\rightarrow$ $D = D\_train \cup D\_test$, and usually $|D\_train| >> |D\_test|$
- Requirements:
  - Any examples in the test set D_test must not be used in the training of the system
  - Any examples used in the training of the system (i.e., those in D_train) must not be used in the evaluation of the trained system
  - The test examples in D_test should allow an unbiased evaluation of the system performance
- Usual splitting: $|D\_train|=(2/3).|D|$, $|D\_test|=(1/3).|D|$
- **Suitable if we have a large dataset *D***

Stratified sampling

- For such datasets that is small (in size) or unbalanced, the examples in the training and test sets may not be representative
- For example: There are (very) few examples for a specific class label
- Goal: The class distribution in the training and test sets should be approximately equal to that in the original dataset (D)
- Stratified sampling
  - An approach to have a balanced (in class distribution) dataset
  - Guarantee the class distributions (i.e., the percentages of examples for class labels) in the training and tests set are approximately equal
- The stratified sampling method can not be applied to a regression problem (because for that problem the system's output is a real value, not a discrete value / class label)

Repeated hold-out

- To apply the Hold-out evaluation method for multiple times (i.e., multiple runs), each one uses a different training and test sets
  - For each run, a certain percentage of the dataset D ***is randomly selected*** to create the training set (possibly together with the stratified sampling method)
  - The error values (or the values of other measure metrics) *are averaged* amongst the runs to get the final (average) error value
- This evaluation method is still not perfect
  - In each run, a different test set is used
  - There are still some overlapping (i.e., repeatedly) used examples among those test sets

Cross-validation

- To avoid any overlapping amongst the used test sets (i.e., the same examples are contained in some different test sets)
- k-fold cross-validation
  - The whole dataset D is divided into k **disjoint subsets** (called "*fold*") that have approximately equal sizes
  - For each run (i.e., of the total *k* runs), a subset is circulated to use for the test set, and the remaining (k-1) subsets are used for the training set
  - The k error values (i.e., each one for each *fold*) are averaged to get the overall error value
- Usual choices of k: 10, or 5
- Often, each subset (i.e., fold) is stratified sampling (i.e., to approximate the class distribution) prior to apply the Cross-validation evaluation method
- Suitable if we have a small or medium dataset D

Leave-one-out cross-validation

- A type of the Cross-validation method
  - The number of folds is exactly the size of the original dataset (k=|D|)
  - Each fold contains just one example
- To maximally exploit the original dataset
- No random sub-sampling
- Not possible to apply the stratified sampling method
  - → Because in each run (loop), the test set contains just one example
- (Very) high computational cost
- Suitable if we have a (very) small dataset D

Bootstrap sampling (1)

- The Cross-validation method applies sampling without replacement
  → For each example, *once selected (used) for the training set, then it cannot be selected (used) again (one more time)* for the training set
- The Bootstrap sampling method applies ***sampling with replacement*** to create the training set
  - Assume that the whole dataset D contains n examples
  - To sample with replacement (i.e., repeating) for n times for the dataset D to create the training set D_train that contains n examples
    - From the dataset D, <u>randomly</u> select an example x (but **not remove** x from the dataset D)
    - Put the example x into the training set: $D\_train = D\_train \cup x$
    - Repeat the above 2 steps for n times
  - To use the set D_train for training the system
  - To use those examples in D **but not in** D_train to create the test set: $D\_test = \{z \in D;\ z \notin D\_train\}$

Bootstrap sampling (2)

- Important notes:
  - The training set has size of *n*, and an example in *D* may **appear multi times** in *D_train*
  - The test set has size <*n*, and an example in *D* can **appear maximum to 1 time** in *D_test*
- Suitable if we have a (very) small dataset *D*

Validation set

- The examples in the test set must not be used (in any way!) in the training of the system
- In some ML problems, the system's training process includes 2 stages:
  - Stage 1: To train the system (= To learn approximately the target function)
  - Stage 2: To optimize the values of the system's hyper-parameters
- The test set cannot be used for the purpose of optimization of the system's parameters
- To divide the whole dataset D into 3 disjoint subsets: *training set*, *validation set*, and *test set*
- The validation set is used to optimize the values of the system's hyper-parameters and the used ML algorithm's ones

  → For a hyper-parameter, **the optimal value** is the one that results in ***the best performance for the validation set***

Evaluation metrics (1)

- **Accuracy**
  - →The accuracy degree of the prediction of the trained system to the test examples
- Efficiency
  - →The costs in time and memory resources needed for the training and the test of the system
- Robustness
  - →The tolerance degree of the system to noise/error/missing-value examples

Evaluation metrics (2)

- Scalability
  - →How the system's performance (e.g., training/prediction speed) varies to the size of the dataset
- Interpretability
  - →How the system's results and operation are easy to understand for users
- Complexity
  - →The complexity of the model (i.e., the target function) learned by the system

Select a trained model

- The selection of a trained model should compromise (balance) between:
  - The complexity of the trained model
  - The prediction accuracy degree of the trained model
- *Occam's razor*. A good trained model is the one that **is simple** and **achieves high accuracy (in prediction)** for the used dataset
- For example:
  - A trained classifier *Sys1*: (Very) simple, and rather (to a certain degree) fit to the training set
  - A trained classifier *Sys2*: More complex, and perfectly fit to the training set

  → *Sys1* is preferred to *Sys2*