CS5805 : Machine Learning I
Lecture Cross Validation

Reza Jafari

Associate Professor, Computer Science
Virginia Tech University

October 10, 2023

Generalization & Cross-Validation

- In machine learning (ML), generalization refers to the ability of an algorithm to be effective across various inputs.

Generalization & Cross-Validation

- In machine learning (ML), generalization refers to the ability of an algorithm to be effective across various inputs.
- It means that the ML model does not encounter performance degradation on the new inputs from the same distribution of the training data.

Generalization & Cross-Validation

- In machine learning (ML), generalization refers to the ability of an algorithm to be effective across various inputs.
- It means that the ML model does not encounter performance degradation on the new inputs from the same distribution of the training data.
- For human beings generalization is the most natural thing possible. We classify on the fly.

Generalization & Cross-Validation

- In machine learning (ML), generalization refers to the ability of an algorithm to be effective across various inputs.
- It means that the ML model does not encounter performance degradation on the new inputs from the same distribution of the training data.
- For human beings generalization is the most natural thing possible. We classify on the fly.
- For example, we would defiantly recognize a dog even if we did not see this breed before. Nevertheless, it is might be quit challenge for an ML model.

Generalization & Cross-Validation

- In machine learning (ML), generalization refers to the ability of an algorithm to be effective across various inputs.
- It means that the ML model does not encounter performance degradation on the new inputs from the same distribution of the training data.
- For human beings generalization is the most natural thing possible. We classify on the fly.
- For example, we would defiantly recognize a dog even if we did not see this breed before. Nevertheless, it is might be quit challenge for an ML model.
- That is why checking the algorithm's ability to generalize is an important task that requires a lot of attention when building the model.

Generalization & Cross-Validation

- In machine learning (ML), generalization refers to the ability of an algorithm to be effective across various inputs.
- It means that the ML model does not encounter performance degradation on the new inputs from the same distribution of the training data.
- For human beings generalization is the most natural thing possible. We classify on the fly.
- For example, we would defiantly recognize a dog even if we did not see this breed before. Nevertheless, it is might be quit challenge for an ML model.
- That is why checking the algorithm's ability to generalize is an important task that requires a lot of attention when building the model.
- To do that, we use Cross-Validation(CV)

What is Cross-Validation

- Cross-validation is a technique for evaluating a machine learning model and testing its performance.

What is Cross-Validation

- Cross-validation is a technique for evaluating a machine learning model and testing its performance.
- CV is commonly used in applied ML tasks and it helps to compare and select an appropriate model for the specific predictive modeling problem.

What is Cross-Validation

- Cross-validation is a technique for evaluating a machine learning model and testing its performance.
- CV is commonly used in applied ML tasks and it helps to compare and select an appropriate model for the specific predictive modeling problem.
- CV is a powerful tool to qualify the model and estimate generalization error on new unseen data.

Algorithm

1. Divide the dataset into two parts: train and test.
2. Train the model on the training set.
3. Validate the model on the test set.
4. Repeat 1-3 steps a couple of times.

Cross-Validation techniques

- There are plenty of CV techniques. Some of them are commonly used, others work only in theory.

CV Methods

1. Hold-out
2. K-folds
3. Leave-one-out
4. Leave-p-out
5. Repeated K-folds
6. Nested K-folds
7. Time series CV

Hold-out Cross Validation

- Hold-out-cross-validation is the simplest and the most common technique.

Hold-out Cross Validation

- Hold-out-cross-validation is the simplest and the most common technique.
- You might not know that it is a **hold-out** method but you certainly use it everyday.

Algorithm

- Divide the dataset into two parts: the train (usually 80%) and the test set (usually 20%)
- Train the model on the training set
- Validate on the test set
- Save the result of the validation

Hold-out Python Implementation

- We usually use the hold-out method on large datasets as it requires training the model only once.

Hold-out Python Implementation

- We usually use the hold-out method on large datasets as it requires training the model only once.
- It is easy to implement hold-out using python using skelearn.train_test_split package

```
from sklearn.model_selection import train_test_split
import numpy as np

X,y = np.arange(10).reshape((5,2)), range(5)
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=111)
```

Hold-out

Hold-out disadvantage

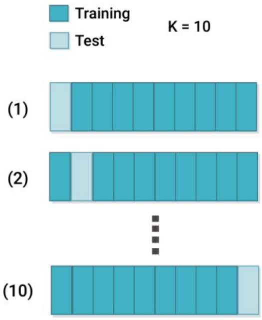
- A dataset that is not completely even distribution-wise.
- The train set will not represent the test set.
- Both training and test sets may differ a lot, one of them might be easier or harder.
- The fact that the model is tested only once, the result obtained by the hold-out may be considered *inaccurate*

K-fold Cross-validation

- k-fold cross-validation is a technique that minimizes the disadvantages of the hold-out method.

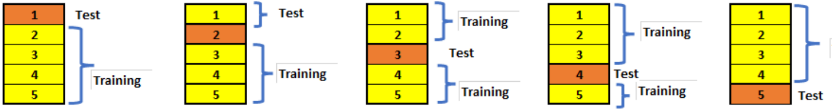
K-fold Cross-validation

- k-fold cross-validation is a technique that minimizes the disadvantages of the hold-out method.
- k-fold introduces a new way of splitting the dataset which helps to overcome the 'test only once bottleneck'.

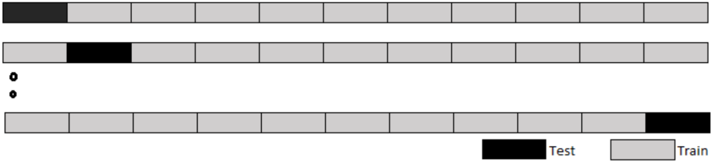


K-fold Pictorial Representation

- k=5

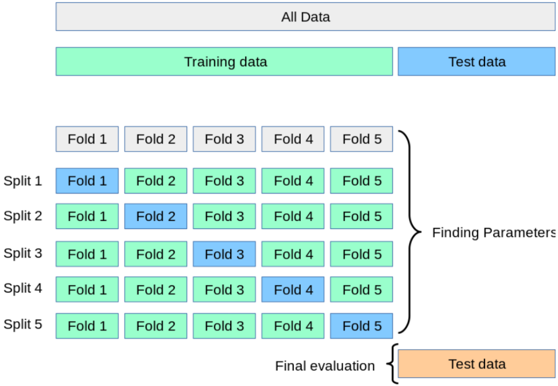


- Larger values of K eventually increase the running time of the CV process.



K-fold Pictorial Representation

- A model is trained $k-1$ of the folds as training data.
- the resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy).



K-fold Cross-validation

Algorithm

1. Pick a number of fold (k). Usually 5 or 10 but you can choose any number less than the dataset length.
2. Split the dataset into k equal(if possible) parts (called folds)
3. Choose $k-1$ folds as training set and the remaining test set.
4. Train the model on the training set.
5. Validate the result of the test.
6. Save the result of the validation.
7. Repeat steps 3-6 k times. Each time use the remaining fold as the test set. In the end, you should have validated the model on every fold that you have
8. To get the final score average the results that you got on step 6.

K-fold python Implementation

- It is easy to implement k-fold using python using skelearn.kFold package

```
from sklearn.model_selection import KFold
import numpy as np

X = np.array([[1,2],[3,4],[5,6],[7,8]])
y = np.array([1,2,3,4])
kf = KFold(n_splits = 2)


for train_index, test_index in kf.split(X):
    print('Train : ', train_index, 'Test : ', test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

K-fold

- In general, it is always better to use k-Fold technique instead of hold-out. In a head to head, comparison k-Fold gives a more stable and trustworthy result since training and testing is performed on several different parts of the dataset.

K-fold

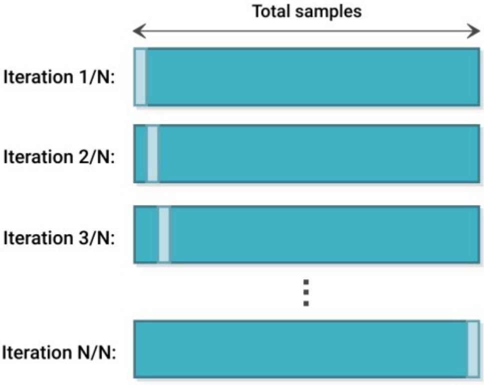
- In general, it is always better to use k-Fold technique instead of hold-out. In a head to head, comparison k-Fold gives a more stable and trustworthy result since training and testing is performed on several different parts of the dataset.
- We can make the overall score even more robust if we increase the number of folds to test the model on many different sub-datasets.

K-fold

- In general, it is always better to use k-Fold technique instead of hold-out. In a head to head, comparison k-Fold gives a more stable and trustworthy result since training and testing is performed on several different parts of the dataset.
- We can make the overall score even more robust if we increase the number of folds to test the model on many different sub-datasets.
- Still, k-Fold method has a disadvantage. Increasing k results in training more models and the training process might be really expensive and time-consuming.

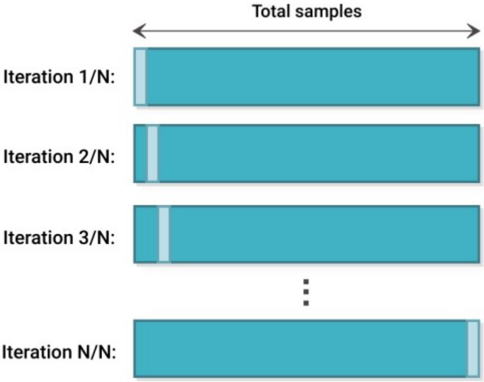
Leave-one-out Cross Validation

- **Leave-one-out CV** (LOOCV) is an extreme case of **k-fold CV**.



Leave-one-out Cross Validation

- **Leave-one-out CV** (LOOCV) is an extreme case of **k-fold CV**.
- If k is equal to N (number of samples) then such k-fold case is called Leave-one-out CV.



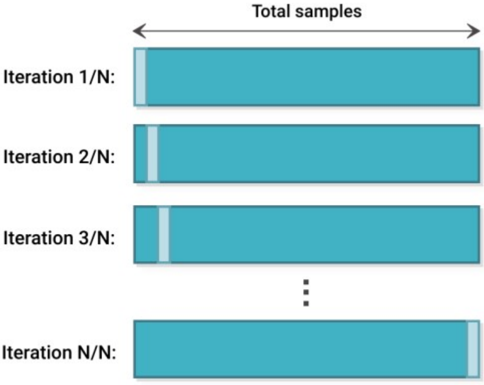
Leave-one-out Cross Validation Algorithm

Algorithm

1. Choose one sample from the dataset which will be the test set.
2. The remaining N–1 samples will be the training set.
3. Train the model on the training set. On each iteration, a new model must be trained.
4. Validate on the test set.
5. Save the result of the validation.
6. Repeat steps 1 – 5 n times as for n samples we have n different training and test sets.
7. To get the final score average the results that you got on step 5.

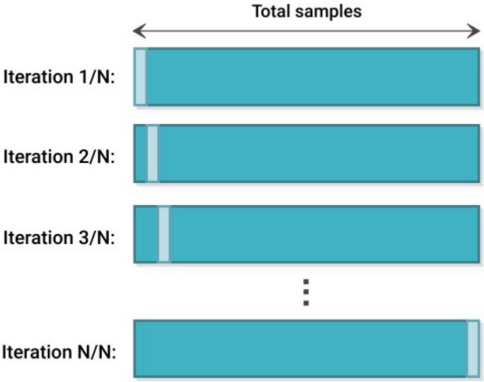
Leave-one-out Python Implementation

- **Leave-one-out CV** (LOOCV) is an extreme case of **k-fold CV**.



Leave-one-out Python Implementation

- **Leave-one-out CV** (LOOCV) is an extreme case of **k-fold CV**.
- If k is equal to N (number of samples) then such k-fold case is called Leave-one-out CV.



Leave-one-out python Implementation

- It is easy to implement leave-one-out using python using skelearn.LeaveOneOut package

```
import numpy as np
from sklearn.model_selection import LeaveOneOut

X = np.array([[1,2],[3,4],[5,6],[7,8]])
y = np.array([1,2,3,4])
loo = LeaveOneOut()

for train_index, test_index in loo.split(X):

    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    print(f'X_train : {X_train}, y_train: {y_train}')
    print(f'X_test : {X_test}, y_test: {y_test}')
```

Leave-p-out cross-validation

- Leave-p-out cross-validation (LpOC) is similar to **Leave-one-out CV** as it creates all possible training and test sets by using **p** samples as the test set.

Algorithm

1. Choose p samples from the dataset which will be the test set.
2. The remaining $N - p$ samples will be the training set.
3. Train the model on the training set. On each iteration, a new model must be trained.
4. Validate the test set.
5. Save the result of the validation.
6. Repeat steps 2-5 C_p^N times.
7. To get the final score average the results that you got on step 5.

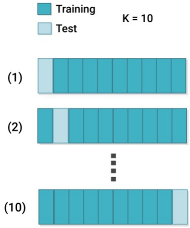
Leave-p-out python Implementation

- It is easy to implement leave-p-out using python using skelearn.model_selection package and LeavePOut

```
import numpy as np
from sklearn.model_selection import LeavePOut

X = np.array([[1,2],[3,4],[5,6],[7,8]])
y = np.array([1,2,3,4])
lpo = LeavePOut(2)

for train_index, test_index in lpo.split(X):

    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    print(f'X_train : {X_train}, y_train: {y_train}')
    print(f'X_test : {X_test}, y_test: {y_test}')
```

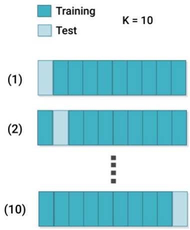
Stratified k-fold cross-validation

- Stratified k-fold CV is used for a large imbalance of target value in the dataset.



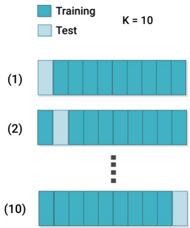
Stratified k-fold cross-validation

- Stratified k-fold CV is used for a large imbalance of target value in the dataset.
- For example: in cats and dog dataset, there might be a large shift towards the dog class.



Stratified k-fold cross-validation

- Stratified k-fold CV is used for a large imbalance of target value in the dataset.
- For example: in cats and dog dataset, there might be a large shift towards the dog class.
- *Stratified k-fold* is the improved version of the standard **k-fold CV** that ensures each fold of dataset has the same proportion of observations with a given label.



Stratified k-fold cross-validation-python

- sklearn.model_selction.StratifiedKFold can be used to implement Stratified CV.

Stratified k-fold cross-validation-python

- sklearn.model_selction.StratifiedKFold can be used to implement Stratified CV.
- Provides train/test indices to split data in train/test sets.

Stratified k-fold cross-validation-python

- sklearn.model_selction.StratifiedKFold can be used to implement Stratified CV.
- Provides train/test indices to split data in train/test sets.
- This CV object returns stratified folds. The folds are made by preserving the % of samples for each class.

Parameters

1. n_splits: at least 2. Default value 5
2. shuffle: Preserve order dependencies in the dataset when **False**
3. random_state: When shuffle = *True* random_state control randomness of each fold for each class. Otherwise leave *None*.