Evaluation and validation of Artificial Neural Networks

Brandon Marquez Salazar

I. Introduction

Currently, for a further usage of ANNs it's needed to measure the performance of the model, this implies the need of defining a way of computing the error and set a filter for those ANNs which can't satisfy the minimum threshold of performance. That's the reason why there are several methods to evaluate the error of any model. Some of those methods and its applications will be discussed in this document.

II. Core Concepts and Methods

Evaluation and validation of an Artificial Neural Network is made through different performance measurements, which can differ it's final values.

A. Resubstitution

Here the testing process is made with the same exact data that the model has been trained with. Here, the model measure can be highly biased because the objective data is a;ready known.

B. Holdout

This method consists on dividing the data in two parts, one for training and one for testing. The division proportion might affect the final estimation, and this method is allows quite inconsistent behaviour for small datasets.

C. Leave-One-Out

The data is divided in N parts, one for testing and N-1 for training. Then the process of training is repeated one per part. The resultant models are quite good but its training and testing processes increases the computational cost.

D. K-fold and Repeated K-fold

Similar to LOO, k-fold divides the data in k parts, one for testing and k-1 for training. The process of training is repeated k times, leaving the last one for testing. The resultant estimation is the average of the k estimations. This method is mostly known as cross-validation.

For repeated k-fold, the process is repeated several times. The final estimation is the average of each repeated estimation.

E. Montecarlo

Also known as Shuffle Split or Random Subsampling; this method repeats training multiple times, but, the data partitioning is mades randomly. As in k-fold, the final estimation is the average of all the estimations.

III. Applications and Discussion

Those methods are used for training validation, answering the question ¿Does my model perform well? ¿Is the model generalizable to new data? ¿Do I need t tune my model?, etc.

A. Testing validation methods for ANNs

To demonstrate the performance of a model's validations method, the following experiment is made. For this experiment it'll be used sklearn.

```
!pip install numpy
!pip install scikit-learn
```

1) Importing modules and libraries: After installing the libraries, it's needed to import the required modules and libraries.

```
from sklearn.datasets         import load_iris
from sklearn.neural_network   import MLPClassifier
from sklearn.metrics          import accuracy_score
from sklearn.model_selection  import KFold
from sklearn.model_selection  import LeaveOneOut
from sklearn.model_selection  import ShuffleSplit
from sklearn.model_selection  import RepeatedKFold
from sklearn.model_selection  import train_test_split
import random
```

2) Data preparation: The dataset for this experiment is the Iris dataset from `sklearn.datasets`.

```
x, y = load_iris(return_X_y=True)
x_train, x_test, y_train, y_test = train_test_split(
  x, y,
  random_state=random.randint(0, 20000)
)
```

3) ANN definition: The multilayer perceptron implemented will be a 3-layer ANN with 10 nodes in each layer.

```
mlp = MLPClassifier(hidden_layer_sizes=(10, 10, 10))
```

4) ANN evaluation with resubstitution:

```
def resubstitution(x_train, y_train, mlp):
  # Train the model
  mlp.fit(x_train, y_train)
  # Evaluate on training data
  y_train_pred = mlp.predict(x_train)
  resubstitution_accuracy = \
    accuracy_score(y_train, y_train_pred)

  print(f"Resubstitution Accuracy: " + \
        f"{resubstitution_accuracy}")

resubstitution(x_train, y_train, mlp)
```

```
Resubstitution Accuracy: 0.9732142857142857
```

5) ANN evaluation with holdout:

```
def holdout(x_train, y_train, x_test, y_test, mlp):
  # Train the model
  mlp.fit(x_train, y_train)

  # Evaluate on test data
  y_test_pred = mlp.predict(x_test)
  holdout_accuracy = \
    accuracy_score(y_test, y_test_pred)

  print(f"Holdout Accuracy: {holdout_accuracy}")

holdout(x_train, y_train, x_test, y_test, mlp)
```

```
Holdout Accuracy: 0.9473684210526315
```

6) ANN evaluation with leave-one-out:

```
def leave_one_out(x, y, mlp):
  loo = LeaveOneOut()
  loo_accuracies = []

  for train_index, test_index in loo.split(x):
      x_train_fold, x_test_fold = \
        x[train_index], x[test_index]
      y_train_fold, y_test_fold = \
        y[train_index], y[test_index]

      mlp.fit(x_train_fold, y_train_fold)
      y_pred = mlp.predict(x_test_fold)
      loo_accuracies.append(
        accuracy_score(y_test_fold, y_pred))

  loo_accuracy = sum(loo_accuracies) / len(loo_accuracies)
  print(f"Leave-One-Out Accuracy: {loo_accuracy}")

leave_one_out(x, y, mlp)
```

```
Leave-One-Out Accuracy: 0.86
```

7) ANN evaluation with k-fold:

```
def k_fold(x, y, mlp):
  k = 5
  kf = KFold(n_splits=k)
  kf_accuracies = []

  for train_index, test_index in kf.split(x):
      x_train_fold, x_test_fold = \
        x[train_index], x[test_index]
      y_train_fold, y_test_fold = \
        y[train_index], y[test_index]

      mlp.fit(x_train_fold, y_train_fold)
      y_pred = mlp.predict(x_test_fold)
      kf_accuracies.append(
        accuracy_score(y_test_fold, y_pred))

  kf_accuracy = sum(kf_accuracies) / len(kf_accuracies)
  print(f"K-fold Cross-Validation Accuracy:"+
        f" {kf_accuracy}")

k_fold(x, y, mlp)
```

```
K-fold Cross-Validation Accuracy: 0.7266666666666667
```

8) ANN evaluation with repeated k-fold:

```
def repeated_k_fold(x, y, mlp):
  k = 5
  n_repeats = 3
  rkf = RepeatedKFold(
    n_splits=k,
    n_repeats=n_repeats,
    random_state=random.randint(0, 20000)
  )
  rkf_accuracies = []

  for train_index, test_index in rkf.split(x):
      x_train_fold, x_test_fold = \
        x[train_index], x[test_index]
      y_train_fold, y_test_fold = \
        y[train_index], y[test_index]

      mlp.fit(x_train_fold, y_train_fold)
      y_pred = mlp.predict(x_test_fold)
      rkf_accuracies.append(
        accuracy_score(y_test_fold, y_pred))

  rkf_accuracy = sum(rkf_accuracies) / len(rkf_accuracies)
  print(f"Repeated K-fold Cross-Validation Accuracy: "+
        f"{rkf_accuracy}")

repeated_k_fold(x, y, mlp)
```

```
Repeated K-fold Cross-Validation Accuracy: 0.88
```

9) ANN evaluation with montecarlo:

```
def montecarlo(x, y, mlp):
  n_iter = 10
  test_size = 0.2
  mc_cv = ShuffleSplit(
    n_splits=n_iter,
    test_size=test_size,
    random_state=random.randint(0, 20000)
  )
  mc_accuracies = []

  for train_index, test_index in mc_cv.split(x):
      x_train_fold, x_test_fold = \
        x[train_index], x[test_index]
      y_train_fold, y_test_fold = \
        y[train_index], y[test_index]

      mlp.fit(x_train_fold, y_train_fold)
      y_pred = mlp.predict(x_test_fold)
      mc_accuracies.append(
        accuracy_score(y_test_fold, y_pred))

  mc_accuracy = sum(mc_accuracies) / len(mc_accuracies)
  print(f"Monte Carlo Cross-Validation Accuracy: "+
        f"{mc_accuracy}")

montecarlo(x, y, mlp)
```

```
Monte Carlo Cross-Validation Accuracy: 0.89
```

IV. Conclusion

As shown in this experiment, the ANN evaluation methods have different performance depending on it's definition and the model. The majority of the models are focused on data partitioning and performing different evaluations to achieve a more precise estimation.

References

[1] M. Ekman. *Learning Deep Learning: Theory and Practice of Neural Networks, Computer Vision, Natural Language Processing, and Transformers Using TensorFlow*. Pearson Education, 2021. ISBN: 9780137470297. URL: https://books.google.com.mx/books?id=wNnPEAAAQBAJ.

[2] S.S. Haykin. *Neural Networks and Learning Machines*. Neural networks and learning machines v. 10. Prentice Hall, 2009. ISBN: 9780131471399. URL: https://books.google.com.mx/books?id=K7P36lKzI_QC.

[3] Christopher M. *Pattern Recognition and Machine Learning*. en. 1st ed. Information Science and Statistics. New York, NY: Springer, Aug. 2006.

[4] Gireen Naidu, Tranos Zuva, and Elias Mmbongeni Sibanda. "A Review of Evaluation Metrics in Machine Learning Algorithms". In: *Artificial Intelligence Application in Networks and Systems*. Springer International Publishing, 2023, pp. 15–25. ISBN: 9783031353147. DOI: 10.1007/978-3-031-35314-7_2. URL: http://dx.doi.org/10.1007/978-3-031-35314-7_2.

[5] quarto. *Guide*. URL: https://quarto.org/docs/guide/.

[6] quarto. *PDF Options*. URL: https://quarto.org/docs/reference/formats/pdf.html.

[7] quarto. *Welcome Page*. URL: https://quarto.org.

[8] scikit-learn. *MLPClassifier*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html.

[9] Parul Singh. *A Must-Read History of Artificial Intelligence*. 2020. URL: https://cyfuture.com/blog/history-of-artificial-intelligence/.