Implement Nonparametric Algorithm From Scratch

Noboru Hayashi NHAYASH3@JHU.EDU

1. Introduction

This paper describes the process to build a nonparametric algorithm from scratch using Python to perform classification and regression. Specifically K-nearest neighbor classifier and regressor, and the variants: Edited KNN and Condense KNN are built. Along with the implementation of these machine learning algorithms, the demonstration of classification and regression with the actual datasets will be shown.

Also the hyperparameter tuning is demonstrated with a 5-fold approach. As the measurement of the performance, accuracy and MSE are picked for classification and regression tasks to tune the parameter K, epsilon and the bandwidth sigma.

2. Implementation

2.1. K-nearest neighbor (knn.py)

2.1.1 Eucidian_distance()

As the measurement of the distance between each data point, the euclidean distance is chosen. This function calculates the square of the difference of the dimension values in each dimension and returns the square root of the sum.

2.1.2 Plurality_vote()

As a helper function for classification tasks, plurality_vote() takes a list of neighbors with the class labelings, then returns the major class.

2.1.3 RBF_kernel()

Similar to the plurality_vote() function, rbf_kernel() returns the value of the target variable to which a Gaussian kernel is applied.

2.1.4 Get_neighbors()

Given the pool of points, the target point, and K, this function calculates the distances between the target point and each point from the pool, then returns the K nearest neighbors with the euclidean distances.

2.1.5 Predict_classification() / Predict_regression()

With the use of get_neighbors() and plurality_vote() or rbf_kernel(), this function returns the prediction of the classification/regression.

2.1.6 Knn()

Given the pool of training points, test points and K(, and A bandwidth sigma for RBF kernel if regression), this function operates a series of KNN predictions and returns the predictions of test_data.

2.2 Edited K-nearest neighbor (edited_knn.py)

2.2.1 Remove_false_classification() / Remove_false_regression()

As a part of the edited KNN algorithm, this function operates the prediction of a sample point from the training dataset, and if the prediction is misclassified for classification, or predicted value exceeds the threshold of the original for regression, the removals of the false points from the training dataset are operated.

2.2.2 edited_knn()

Similar to knn() for K-nearest neighbor algorithm, this function operates a series of KNN predictions and returns the result. Before the predictions, removals of false training data are operated with remove_false_classification() / remove_false_regression.

2.3 Condensed K-nearest neighbor (condensed_knn.py)

2.3.1 Sfs_classificaiton / Sfs_regression()

This function operates a stepwise forward selection from the training dataset. Before the selection, the training dataset is shuffled (seeds can be specified as the function argument).

2.3.2 Condensed_knn()

This function calls a series of operations: SFS on the training dataset, KNN prediction based on the processed dataset, and returns the results.

2.4 Utils

2.4.1 K_fold()

A function for operating 5-fold cross validation, which samples uniformly from the dataset and returns the list of folds with the length of K. For example, if there are 1000 examples, this function generates a list of 5 data lists, each containing 200 examples.

2.4.2 Read_file, Evaluation, Data_handler

Helper functions for data loading, preprocessing and the model evaluation. Implemented in project 1.

3. Results

In the following section, the results of hyperparameter tunings and the prediction will be demonstrated. The demonstrations are based on the 6 dataset below:

- Breast Cancer [Classification]
- Car Evaluation [Classification]
- Congressional Vote [Classification]
- Abalone [Regression]
- Computer Hardware [Regression]
- Forest Fires [Regression]

3.1 Hyperparameter Tuning

3.1.1 KNN

For tuning the hyperparameter of KNN, k and sigma for regression tasks, the search process has been done with the below code snippet (using breast cancer dataset for example) :

```
    # breast-cancer

    breast_cancer = datasets['breast-cancer']

    data_handler.handle_missing(breast_cancer)

    breast_cancer_folds = k_fold.k_fold(breast_cancer)

    tuning_set = breast_cancer_folds[0]

    tuning_fold = k_fold.k_fold(tuning_set, 4)

    tuning_train = tuning_fold[0] + tuning_fold[1] + tuning_fold[2]

    tuning_test = tuning_fold[3]

    # tuning k-vals

    for k in range(1, 30, 2):

        knn_predictions = knn.knn(tuning_train, tuning_test, k, 1)

        met = evaluation_metric('accuracy', [data[-1] for data in tuning_test],
knn_predictions)

        print('KNN: k =', k, ", Accuracy =", met)
```

Output:

KNN: k = 1 , Accuracy = 0.4

KNN: k = 3 , Accuracy = 0.4857142857142857

...

KNN: k = 11 , Accuracy = 0.6571428571428571

…

In this case, k=11 will be the best K value for breast cancer tuning dataset. The result of the K and sigma value search as below:

Dataset	Breast Cancer	Car Evaluation	Congressional Vote
K	11	5	1

Dataset	Abalone	Computer Hardware	Forest Fires
K	1	1	1
Sigma	100	100	10

3.1.2 Edited KNN

Similar to the regular KNN, the tuinings of the hyperparameters K (, epsilon and sigma for regression) are operated by looping the predictions and evaluations with different parameter values.

The results of the tunings for Edited KNN are as below:

Dataset	Breast Cancer	Car Evaluation	Congressional Vote
K	9	3	1

Dataset	Abalone	Computer Hardware	Forest Fires
K	1	1	9
Epsilon	100	100	10
Sigma	100	100	100

3.1.3 Condensed KNN

The results of the hyperparameter tunings for condensed KNN algorithm are as below:

Dataset	Breast Cancer	Car Evaluation	Congressional Vote
K	9	1	1

Dataset	Abalone	Computer Hardware	Forest Fires
K	1	1	5
Epsilon	0.001	1	10
Sigma	100	100	100

3.2 Model Performances

Given the hyperparameters values tuned by searching, the remaining folds used to evaluate the model performances.

3.2.1 KNN

The remainder of dataset folds are used to evaluate the KNN algorithm with the best hyperparameter values, with the code snippet as below (breast cancer dataset as an example):

```
# breast-cancer

breast_cancer = datasets['breast-cancer']

data_handler.handle_missing(breast_cancer)

breast_cancer_folds = k_fold.k_fold(breast_cancer)

tuning_set = breast_cancer_folds[0]

tuning_fold = k_fold.k_fold(tuning_set, 4)
```

```
    tuning_train = tuning_fold[0] + tuning_fold[1] + tuning_fold[2]

    tuning_test = tuning_fold[3]

    rem = breast_cancer_folds[1] + breast_cancer_folds[2] +
breast_cancer_folds[3] + breast_cancer_folds[4]

    eval_fold = k_fold.k_fold(rem, 4)

    eval_train = eval_fold[0] + eval_fold[1] + eval_fold[2]

    eval_test = eval_fold[3]

    # Evaluate Performances

    knn_predictions = knn.knn(eval_train, eval_test, 9, 1)

    met = evaluation_metric('accuracy', [data[-1] for data in eval_test],
knn_predictions)

    print("Accuracy =", met)
```

For classification, the performances for Car Evaluation and Congressional Vote achieve high accuracy (>90%). Regarding the performance for breast cancer dataset, even though the accuracy is around 60%, it is expected that the number is close to thethe performance during the parameter tuning (around 60%). The results of the evaluations as below:

Dataset	Breast Cancer	Car Evaluation	Congressional Vote
K	11	5	1
Accuracy	64.03%	90.43%	95.40%

For the regression problem, the loss measure (MSE) for each dataset is as below.

Dataset	Abalone	Computer Hardware	Forest Fires

K	1	1	1
Sigma	100	100	10
MSE	8.56	6669.63	13969.94

3.2.2 Edited KNN

Compared to the KNN algorithm, the values of the measurements are quite similar to ones from regular KNNs.

Dataset	Breast Cancer	Car Evaluation	Congressional Vote
K	9	3	1
Accuracy	66.19%	87.83%	93.10%

Dataset	Abalone	Computer Hardware	Forest Fires
K	1	1	9
Epsilon	100	100	10
Sigma	100	100	100
MSE	8.56	10562.93	2687.76

3.2.3 Condensed KNN

In most cases as below, from the experiment the performances of condensed KNNs are worse than the ones of edited KNNs. However, the runtime is much faster since this approach is meant to reduce the search pool of get_neighbors, and speed up the prediction.

The results are as below:

Dataset	Breast Cancer	Car Evaluation	Congressional Vote
K	9	1	1
Accuracy	55.40%	82.03%	95.40%

Dataset	Abalone	Computer Hardware	Forest Fires
K	1	1	9
Epsilon	100	100	10
Sigma	100	100	100
MSE	73.33	19242.29	14260.89

4. Conclusion

In this paper, I described my approach to implement the K-nearest neighbor algorithm and its variants: Edited K-nearest neighbor and Condensed K-nearest neighbor. Also the results of hyperparameter values search and model evaluation are shown, and with these data the characteristic of each nonparametric algorithm is illustrated.