605.649 Programming Project 2: k-nearest neighbors algorithm

Christopher El-Khouri

1. Introduction

In the world of machine learning there are generally two types of modeling; Parametric modeling and Non-Parametric modeling. A parametric model is defined as a learning model that summarizes data with a set of parameters of fixed size (independent of the number of training examples) is called a parametric model (Russel & Norvig 2020).

Parametric algorithms generally involve selecting a form for the function, then learning the coefficients of the function based on training data. Linear regression is a very popular parametric model; the algorithm selects the form of the function which is generally as follows:

$$f(x) = b_0 + \sum_{i=1}^{d} b_i x_i$$

The algorithm proceeds to calculate the coefficients $b_0 : b_d$ based on the training data provided.

Non-parametric models, on the other hand, are designed under the assumption that similar inputs have similar outputs. The algorithms function by finding similar past instances from a training set then use a distance measure to calculate similarity and provide an output.

In this project, we will be assessing the performance of three non-parametric algorithms: K-Nearest Neighbors (K-NN), Edited K-Nearest Neighbors (E-NN), and Condensed K-Nearest Neighbors (C-NN).

K-NN is a non-parametric method initially proposed by Thomas Cover (1967) that can be used for classification or regression. When used for classification, the algorithm takes the input data and returns the most common class among its k nearest neighbors from the training data. In the case of regression, the algorithm returns a calculated value based on the output values of the input data's k nearest neighbors. The calculation could be the average of the output values of the neighbors or a calculation that considers the respective distance of the neighbors and weighs them accordingly.

E-NN is a variation of K-NN where the training process is modified. There are two ways to implement an E-NN algorithm, both occur during the training process. The first method is removing misclassified examples from the training set, and the second method is removing the correct examples. The modified training set will then proceed to be used to fit our test data. The idea behind implementing E-NN over K-NN is to tackle the possible noise, human error, or irrelevant attributes within the training data that may deceive the output.

C-NN is another variation of K-NN where the training process is modified. C-NN aims to decrease the size of the stored training data, therefore decreasing fitting computational time while maintaining performance. The smallest subset Ƶ of the initial training data X is selected such that when Ƶ is used as the training data instead of X, the performance does not decrease.

In this project, we will be comparing the performance of the three different algorithms on 6 different datasets tackling both classification and regression. We hypothesize that the accuracy of the outputs produced by the K-NN algorithms will be superior to the accuracy of the outputs produced by E-NN and C-NN. Although the computational time required to run the test data using E-NN and C-NN may be less than the computational time required by K-NN, K-NN has more training data and more neighbors for the test data to rely on, therefore producing more accurate results. The accuracy of our classification problems will be calculated using the percentage of correctly classified points. The accuracy of our regression problems will be calculated using the Mean Squared Error (MSE).

In the next section, we will describe the technical and theoretical basis of our algorithms, in section 3 we will present and compare the results, and in section 4 we will discuss the significance of our experiment.

2. Algorithms and Experimental Methods

In our project, the distance metric used for $D(x, y)$ is the Euclidean Distance:

$$D(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

K-NN Classifier

The K-NN Classifier functions as follows:

$$\text{Given } x_i \in X \text{ where } x_i = \left(\{x_i^1, \dots, x_i^d\}, c\right)$$
$$\text{Find } (r_1, \dots, r_k) \in X \text{ such that } \forall\, x \in X, x \neq (r_1, \dots, r_k),$$
$$D(x_q, r) < D(x_q, x)$$
$$\text{return most class c associated with } (r_1, \dots, r_k)$$

When fitting the model to a test set, we calculate the Euclidean distance from the test data to each of the points in the training data and return the k nearest training data points. The most common class associated with the k nearest training data points is returned as the output class of our test data.

The k value is initially chosen by tuning the algorithm on our data set. Tuning is carried out by extracting 10% of our data while considering the distribution of the different classes among the entire dataset. Therefore, if 50% of our data consists of Class A and the other 50% consists of Class B, the 10% tuning dataset will consist of 5% Class A and 5% Class B. Next, we proceed to train the K-NN classifier on the data with k values ranging from 1 to 15 with increments of 2. Increments of 2 were chosen to keep the k values odd in order not to have a tie when deciding on the most common class c. The k value that results in the highest accuracy within the tuning data i.e. the highest amount of correct predictions will be selected for modeling the remaining data.

K-NN Regression

The K-NN Regression algorithm is designed as follows:

$$\text{Given } x_i \in X \text{ where } x_i = \left(\{x_i^1, \ldots, x_i^d\}, y_i\right)$$
$$\text{Use k} - \text{means clustering to find k centroids r where:}$$
$$r_i = \left(\{x_{r_i}^1, \ldots, x_{r_i}^d\}, y_{r_i}\right)$$
$$\text{Calculate Euclidean Distance from query point } x_q \text{ to each centroid r:}$$
$$D\left(x_q, r\right)$$
$$\text{Apply Gaussian Radial Basis Function Kernel on distances and centroids}$$
$$\text{The calculated value is returned as an output}$$

K-means clustering was used as an effort to reduce computational time while maintaining performance. The k-means clustering algorithm is designed as follows:

$$\text{Given } x_i \in X \text{ where } x_i = \left(\{x_i^1, \ldots, x_i^d\}\right)$$
$$\text{k points are selected at random to form the initial centroids } C = (C_1, \ldots, C_k)$$
$$\text{The distance for each } x_i \in X \text{ from each centroid C is calculated}$$
$$\text{The cluster C that results in the minimum distance from } x_i \text{ is designated to } x_i$$
$$\text{The clusters C are replaced with the mean values of the } x'\text{s that belong to them}$$
$$\text{This is repeated until C converges}$$

In our case, C converges when the percentage error between the previous C and the calculated C is less than 0.1%.

The Gaussian Radial Basis Function (RBF) is applied as follows:

$$\hat{g}(x) = \frac{\sum_k K\left(x_q, C_k\right) y^k}{\sum_k K\left(x_q, C_k\right)}$$

$$Where:$$

$$K\left(x_q, C_k\right) = \exp\left(-\frac{D\left(x_q, C_k\right)^2}{2\sigma^2}\right)$$

When fitting the model to a test set, we calculate the Euclidean distance from the test data to each of the centroids generated from k-means clustering then apply the Gaussian Radial Basis function as shown above to calculate the output.

The k and σ values are chosen by tuning the algorithm on our data set. Tuning is carried out by training the K-NN regression algorithm on 10% of the data. As shown in the algorithm design steps above, the first step is k-means clustering. K-means clustering is performed on 10% of our data with k values ranging from 1 to 14 with increments of 1. The k value was chosen by using a non-visual application of The Elbow Method.

The Elbow Method consists of calculating the Within-Cluster-Sum of Squared Errors (WSS) for different values of k. The percentage difference between consecutive WSS values is calculated, if the percentage difference is less than 15%, the k value of the previous WSS was chosen.

As for σ, once the k value is chosen, the K-NN regression algorithm is trained on 10% of the dataset with values of σ ranging from 1 to 10 with increments of 1. The value of σ that returns the lowest MSE is chosen.

E-NN Classifier

The E-NN Classifier functions as follows:

```
For each $x^t \in X$:
    Classify $x^t$ with KNN classifier using $X - \{x^t\}$
    Compare class returned to class of $x^t$
    If the class is different remove $x^t$ from X
Repeat until no more values are removed
```

The K-NN classifier used is the same one described earlier. When fitting the model to a test set, we calculate the Euclidean distance from the test data to each of the points in the edited training data and return the k nearest training data points. The most common class associated with the k nearest training data points is returned as the output class of our test data.

The k value is initially chosen by tuning the algorithm on our data set. Just as we did in K-NN, tuning is carried out by extracting 10% of our data while considering the distribution of the different classes among the entire dataset. Next, we proceed to train the E-NN classifier on the data with k values ranging from 1 to 15 with increments of 2. Increments of 2 were chosen to keep the k values odd in order not to have a tie when deciding on the most common class c. The k value that results in the highest accuracy within the tuning data i.e. the highest amount of correct predictions will be selected for modeling the remaining data.

E-NN Regression

The E-NN Regression algorithm is designed as follows:

```
For each $(x^t, y^t) \in X$:
    Perform KNN regression on $x^t$
    Compare returned output y with $y^t$
    if $\left(\frac{|y - y^t|}{y^t} > \text{error threshold } \epsilon\right)$:
        Remove $x^t$ from X
Repeat until no more values are removed
```

The K-NN regression algorithm used is the same one described earlier that utilizes k-means clustering and the Gaussian RBF.

When fitting the model to a test set, we calculate the Euclidean distance from the test data to each of the centroids generated from k-means clustering on the edited training set then apply the Gaussian Radial Basis function described earlier to calculate the output.

The k, σ, and ϵ values are chosen by tuning the algorithm on our data set. Tuning is carried out by training the E-NN regression algorithm on 10% of the data. Similar to K-NN Regression, the first step is k-means clustering. K-means clustering is performed on 10% of our data with k values ranging from 1 to 14 with increments of 1. The k value was chosen by using a non-visual application of The Elbow Method.

As for the error threshold ϵ and σ, once the k value is chosen, the K-NN regression algorithm is trained on 10% of the dataset with values of ϵ ranging from 1% to 15%, and values of σ ranging from 1 to 10 with increments of 1. The values of σ and ϵ that return the lowest MSE are chosen.

C-NN Classifier

The C-NN Classifier functions as follows:

```
Let $\mathrm{Z} = \phi$
For all $\mathrm{x} \in \mathrm{X}$ (in random order)
    Find $\mathrm{x}' \in \mathrm{Z}$ such that $\mathrm{D}(\mathrm{x}, \mathrm{x}') = \min_{\mathrm{x}^{\mathrm{j}} \in \mathrm{Z}} \mathrm{D}(\mathrm{x}, \mathrm{x}^{\mathrm{j}})$
    If $\mathrm{class}(\mathrm{x}) \neq \mathrm{class}(\mathrm{x}')$ add x to Z
Until Z does not change
```

When fitting the model to a test set, we calculate the Euclidean distance from the test data to each of the points in Z and return the k nearest training data points. The most common class associated with the k nearest training data points is returned as the output class of our test data.

The k value is initially chosen by tuning the algorithm on our data set. Just as we did in K-NN, tuning is carried out by extracting 10% of our data while considering the distribution of the different classes among the entire dataset. Next, we proceed to train the C-NN classifier on the data with k values ranging from 1 to 15 with increments of 2. Increments of 2 were chosen to keep the k values odd in order not to have a tie when deciding on the most common class c. The k value that results in the highest accuracy within the tuning data i.e. the highest amount of correct predictions will be selected for modeling the remaining data.

C-NN Regression

The C-NN Regression algorithm is designed as follows:

```
Let $\mathrm{Z} = \phi$
For all $(\mathrm{x}, \mathrm{y}) \in \mathrm{X}$ (in random order)
    Find $(\mathrm{x}', \mathrm{y}') \in \mathrm{Z}$ such that $\mathrm{D}(\mathrm{x}, \mathrm{x}') = \min_{\mathrm{x}^{\mathrm{j}} \in \mathrm{Z}} \mathrm{D}(\mathrm{x}, \mathrm{x}^{\mathrm{j}})$
    If $\frac{|\mathrm{y} - \mathrm{y}'|}{\mathrm{y}'} > \epsilon$ add x to Z
Until Z does not change
```

When fitting the model to a test set, we calculate the Euclidean distance from the test data to each of the centroids generated from k-means clustering on the Z then apply the Gaussian Radial Basis function described earlier to calculate the output.

The k, σ, and ϵ values are chosen by tuning the algorithm on our data set. Tuning is carried out by training the C-NN regression algorithm on 10% of the data. Similar to K-NN Regression, the first step is k-means clustering. K-means clustering is performed on 10% of our data with k values ranging from 1 to 14 with increments of 1. The k value was chosen by using a non-visual application of The Elbow Method.

As for the error threshold ϵ and σ, once the k value is chosen, the K-NN regression algorithm is trained on 10% of the dataset with values of ϵ ranging from 1% to 15%, and values of σ ranging from 1 to 10 with increments of 1. The values of σ and ϵ that return the lowest MSE are chosen.

5-fold cross-validation

Our algorithms were fitted and validated using 5-fold cross-validation. After tuning our algorithms using 10% of the data, 90% of the data was used in the 5-fold cross-validation process. The data is split into 5 parts (or folds) and the classifiers are trained on each of the folds accordingly. In the case of our classification algorithms, stratified cross-validation was applied where the same proportion of classes of the total dataset was applied to each of the folds. The performance of the classification algorithms was determined by calculating the accuracy of each of the folds by dividing the number of correct predictions over the total length of the folds, then calculating the mean of the accuracies. The performance of the regression algorithms was determined by calculating the MSE of each of the folds, then calculating the mean of the MSEs.

3. Datasets

The following datasets were used in this project:

1. Glass
2. Image Segmentation
3. Vote
4. Abalone
5. Computer Hardware
6. Forest Fires

Dataset 1: Glass (Classification)
This study of the classification of types of glass was motivated by a criminological investigation. The dataset contains 214 instances, 6 classes, and 11 columns. The values were normalized before entering them into the classifier algorithms.

Dataset 2: Image Segmentation (Classification)
The instances were drawn randomly from a database of 7 outdoor images. The images were hand segmented to create a classification for every pixel. The dataset contains 210 instances, 7 classes, and 20 columns.

Dataset 3: Vote (Classification)
This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the Congressional Quarterly Almanac. The dataset contains 435 instances, 2 classes, and 17 columns. The attribute values were discretized by giving ‘y’ values a value of 1, ‘n’ values a value of 0, and ‘?’ values which represent ‘abstain’ were given a value of -1.

Dataset 4: Abalone (Regression)
Predicting the age of abalone from physical measurements. The dataset contains 4177 instances and 9 columns. The values were normalized before entering them into the regression algorithms.

Dataset 5: Computer Hardware (Regression)
This dataset consists of relative CPU Performance Data. The estimated relative performance values were estimated by the authors using a linear regression method. The results are indicated in the data set as "ERP." The dataset contains 209 instances and 10 columns. The values were normalized before entering them into the regression algorithms.

Dataset 6: Forest Fires (Regression)
This is a difficult regression task, where the aim is to predict the burned area of forest fires, in the northeast region of Portugal, by using meteorological and other data. The dataset contains 517 instances and 13 columns. The values were normalized before entering them into the regression algorithms.

4. Results

Dataset 1: Glass (Classification)
Modeling the Glass dataset with our classification algorithms we get the following:

Table 1: Classification results for Dataset 1

Dataset 1: Glass				
Classifier		K-NN	E-NN	C-NN
k		1	1	1
Accuracy (%)	Fold 1	84.62	61.54	79.49
	Fold 2	61.54	66.67	71.79
	Fold 3	76.32	68.42	65.79
	Fold 4	66.67	74.36	66.67
	Fold 5	73.68	65.79	71.05
	Average	**72.56**	**67.35**	**70.96**

Dataset 2: Image Segmentation (Classification)
Modeling the Image Segmentation dataset with our classification algorithms we get the following:

Table 2: Classification results for Dataset 2

Dataset 2: Image Segmentation				
Classifier		K-NN	E-NN	C-NN
k		1	1	1
Accuracy (%)	Fold 1	86.84	68.42	81.58
	Fold 2	89.47	81.58	78.95
	Fold 3	84.21	78.95	78.95
	Fold 4	92.11	78.95	86.84
	Fold 5	89.19	86.49	86.49
	Average	**88.36**	**78.88**	**82.56**

Dataset 3: Vote (Classification)
Modeling the Vote dataset with our classification algorithms we get the following:

Table 3: Classification results for Dataset 3

Dataset 3: Vote				
Classifier		K-NN	E-NN	C-NN
k		3	5	1
Accuracy (%)	Fold 1	94.94	91.14	91.14
	Fold 2	96.15	87.18	83.33
	Fold 3	93.67	92.41	91.14
	Fold 4	91.03	88.46	88.46
	Fold 5	85.90	93.59	93.59
	Average	**92.33**	**90.56**	**89.53**

Dataset 4: Abalone (Regression)
Modeling the Abalone dataset with our regression algorithms we get the following:

Table 4: Regression results for Dataset 4

Dataset 4: Abalone			
Regression		K-NN	E-NN
k		5	5
σ		3	10
ϵ			0.14
MSE	Fold 1	0.77	0.74
	Fold 2	0.64	0.74
	Fold 3	0.64	0.71
	Fold 4	0.67	0.74
	Fold 5	0.62	0.82
	Average	**0.67**	**0.75**

Dataset 5: Computer Hardware (Regression)
Modeling the Computer Hardware dataset with our regression algorithms we get the following:

Table 5: Regression results for Dataset 5

Dataset 5: Computer Hardware				
Regression		K-NN	E-NN	C-NN
k		3	4	4
σ		5	10	1
ϵ			0.13	0.07
MSE	Fold 1	1.04	0.18	1.42
	Fold 2	0.38	0.49	0.09
	Fold 3	0.19	0.08	0.18
	Fold 4	0.17	0.06	0.28
	Fold 5	0.3	1.47	0.1
	Average	**0.42**	**0.46**	**0.42**

Dataset 6: Forest Fires (Regression)
Modeling the Computer Hardware dataset with our regression algorithms we get the following:

Table 6: Regression results for Dataset 6

Dataset 6: Forest Fires		
Regression		K-NN
k		3
σ		7
MSE	Fold 1	1.48
	Fold 2	0.08
	Fold 3	0.14
	Fold 4	3.6
	Fold 5	0.26
	Average	**1.11**

5. Discussion

Looking at our classification results above on tables 1-3, our hypothesis appears to stand with K-NN outperforming both the E-NN and C-NN classifiers. The C-NN classifier outperformed E-NN in datasets 1 and 2, however, E-NN outperformed C-NN in dataset 3.

As for the regression results, looking at Table 4 regarding the Abalone dataset, due to the large size of the dataset, the C-NN regression algorithm designed in this project would take too much processing time to return the results. Although C-NN would reduce the processing time when testing a new dataset based on a training set, it requires a much higher processing time to train. However, K-NN outperformed E-NN confirming our hypothesis.

Looking at Table 5, all the regression algorithms designed were able to run on the dataset, the results show a tie between K-NN and C-NN. Although the average MSE values were equal, C-NN had a lower minimum MSE among the folds, defying our hypothesis. As for Table 6, the Forest Fires Dataset, regression was not successful therefore not allowing E-NN nor C-NN algorithms to be able to run as the error thresholds were not able to be tuned. Perhaps a thorough EDA on Dataset 6 could uncover a successful regression method.

6. Conclusion

In conclusion, it seems that our hypothesis is not necessarily confirmed to be true as C-NN seems to challenge the performance of K-NN. Further research into the combination of EDA methods along with C-NN could prove that C-NN may be the superior algorithm.

7. References

Peter Norvig and Stuart Russell.
Artificial Intelligence: A Modern Approach.
Prentice Hall, Upper Saddle River, New Jersey, 2020.

Thomas Cover and Peter Hart.
Nearest neighbor pattern classification
IEEE Transactions on Information Theory, 13(1):21-27, 1967.