Indonesian Journal of Physics
Vol. 32. No.2, 29, 12, 2021
Journal homepage: http://ijphysics.com

MODIFIED CORRELATION WEIGHT K-NEAREST NEIGHBOR CLASSIFIER USING TRAINING DATASET CLEANING METHOD

Efraim Kurniawan D. Kette [1]
[1] *Faculty of Mathematics and Natural Science, Bandung, 40132, Indonesia*

(Received: December 18, 2021, Revised: December 29, 2021, Accepted: December 29, 2021)

Abstract

In pattern recognition, the k-Nearest Neighbor (kNN) algorithm is the simplest non-parametric algorithm. Due to its simplicity, the model cases and the quality of the training data itself usually influence kNN algorithm classification performance. Therefore, this article proposes a sparse correlation weight model, combined with the Training Data Set Cleaning (TDC) method by Classification Ability Ranking (CAR) called the CAR classification method based on Coefficient-Weighted kNN (CAR-CWKNN) to improve kNN classifier performance. Correlation weight in Sparse Representation (SR) has been proven can increase classification accuracy. The SR can show the 'neighborhood' structure of the data, which is why it is very suitable for classification based on the Nearest Neighbor. The Classification Ability (CA) function is applied to classify the best training sample data based on rank in the cleaning stage. The Leave One Out (LV1) concept in the CA works by cleaning data that is considered likely to have the wrong classification results from the original training data, thereby reducing the influence of the training sample data quality on the kNN classification performance. The results of experiments with four public UCI data sets related to classification problems show that the CAR-CWKNN method provides better performance in terms of accuracy.

Keywords: Sparse Representation, Correlation Weight, Dataset Cleaning, kNN

INTRODUCTION

The k-Nearest Neighbor (k-NN) algorithm is one of the most frequently used non-parametric algorithms in data mining. The kNN algorithm is often used in pattern recognition due to its simplicity and effectiveness. This algorithm is simple because it initially compares the distance between the test data and the training data. Labels of the test data class will be classified based on the majority labels from the closest neighbor training data. The kNN algorithm itself has several issues, where it's known that sensitive to the nearest neighbor size (k) and the quality of the training data used.

The first issue about determining the value of k, several alternatives have been proposed from several previous studies. Some of these alternatives include that optimal k value determined based on using results of 10-fold cross-validation [1]. Another alternative shows the k value can be determined using n if the number of training data n is greater than 100 [2]. As for these alternatives, predicting the label of each test data still depends on the majority of the closest label results. Therefore, modification by giving distance weight to label classification can improve classification performance on kNN.

In its development, the use of sparse representation in classification based on the distance of the nearest neighbor has often been applied [3], [4]. The sparse representation coefficient has several advantages such as providing good similarity, can reflect adjacent data structures, and capable of reflecting the relationship between data samples. Therefore, the coefficient-weighted k-nearest neighbor classifier (CW-KNN) method is used, which provides a distance weight sparse representation model that can make solid results for the classification process [5].

The next issue is related to the processing stages that affected the quality of the training data used. In the KNN classifier, each training sample is assumed to have the same probability of being selected as one of the closest neighbors of a given test sample. In

* Corresponding author..
E-mail address: 20920301@mahasiswa.itb.ac.id

general, some training samples may contain noise or be incorrect. Because of this, some of these training samples are likely to be unnecessary or even dangerous for classification tasks. To achieve better classification accuracy, some of these training samples should be safely removed from the training data set as their usefulness for correcting classification decisions is somewhat limited. Therefore, the Training Dataset Cleaning method is used in the task of preparing data on the KNN classification to increase its effectiveness [6], [7].

The Training Dataset Cleaning method selected training data samples based on the classification ability ranking. Initially, the classification capability that measures the contribution of each training sample to the leave-one-out (LV1) strategy is defined as a function. The data set of the training sample with the correct classification will be obtained while acting as the nearest neighbor by utilizing other training samples by using the defined function. Next, the process of cleaning the training sample data with low classification ability from the training data set. Regarded that other samples from the same feature space can use a higher quality training data set. Therefore, when the classification process occurs, the expected result by using the cleaned data set can reduce the error rate generated.

This article aims to combine the two algorithms CW-KNN and TDC, in the research topics "Modified Correlation Weight k-Nearest Neighbor Classifier Using Training Dataset Cleaning Method" to solve several classification cases obtained from the UCI repository dataset. The modified method result then will be compared to each of the related methods with the expectation that the modified method can improve the classification performance of the nearest neighbor-based methods.

EXPERIMENTAL METHOD

This article uses a literature study method on k-Nearest Neighbor (kNN), the correlation weight of kNN (CW-KNN), and the classification ability ranking for kNN (CAR-KNN).

KNN

kNN is a simple and effective non-parametric classification algorithm for pattern recognition problems [8]. This algorithm performs a classification based on the nearest neighbor distance of each data sample that has the same features. In general, the Euclidean distance function is used to calculate the distance between each sample of the data.

$$d(\boldsymbol{y}, \boldsymbol{x_i}) = \sum_{i=1}^{n} |\boldsymbol{y} - \boldsymbol{x_i}|^2 \quad (1)$$

The Euclidean distance function in Equation (1) is often used to determine the shortest distance between the new sample data $\boldsymbol{y}$ and the training data $\boldsymbol{x_i}$ [9], [10]. Every training data included in the *k* closest distances will be marked as the query of nearest neighbors. The query denoted as $\boldsymbol{T_k} = \{(\boldsymbol{x_i^{NN}}, \boldsymbol{l_i^{NN}})\}_{i=1}^{k}$ with the notation $\boldsymbol{l_i}$ is the class label for each of the nearest distances training data ($\boldsymbol{x_i^{NN}}$).

$$l_y = \arg\max_c \left(\sum_{(x_i^{NN}, l_i^{NN}) \in T_k} \delta(c = l_i^{NN})\right) \quad (2)$$

Equation (2) is a calculation to determine the label based on sample data ($\boldsymbol{y}$), where $\delta(c = l_i^{NN})$ is the Kronecker delta function that takes a value of 1 if $\boldsymbol{c} = \boldsymbol{l_i^{NN}}$ and 0 otherwise. In addition, a distance-weighted rule for k - nearest neighbors is given to determine the predicted labels [11]. The determination of distances weight ($\boldsymbol{w_i}$) ranges on a scale of 0 to 1. The weight value will be greater (approaching value 1) if the training data sample is near to the new sample, while it is smaller for the opposite. So, based on Equation (2), the label prediction based on sample data ($\boldsymbol{y}$) with distance weights ($\boldsymbol{w}$) can be seen in Equation (3).

$$l_y = \arg\max_c \left(\sum_{(x_i^{NN}, l_i^{NN}) \in T_k} w_i \cdot \delta(c = l_i^{NN})\right) \quad (3)$$

CW - KNN

CWKNN method provides a sparse weight distance representation model for each query sample ($\boldsymbol{y}$) [5]. The sparse weight distance is denoted by $y = X\beta$, where $\beta = [\beta_1, \beta_2, \ldots, \beta_n]^T$ is the sparse coefficient vector of all training samples towards $\boldsymbol{y}$. This model uses the sparse representation concept for distance weights. Therefore, it can reflect the relationship between the new data samples for each training dataset [12], [13].

$$\bar{\beta} = \arg\min_{\beta} \{\|y - X\beta\|_2^2 + \rho\|\beta\|_1\} \quad (4)$$

The Least Absolute Shrinkage and Selection Operator (LASSO) function in Equation (4) is the least square with L_1 norm regularization. This function is used to reconstruct the sparse coefficient vector. After obtaining the set of sparse coefficients, each value will be used to select the k closest neighbors from each query sample. A total of k - Coefficients with the largest values will be selected in the set $\bar{\beta} = \{\bar{\beta}_1, \bar{\beta}_2, \ldots, \bar{\beta}_n\}$, and the corresponding training data included in the k closest distances will be marked as the nearest neighbor query denoted as $\boldsymbol{T_k} = \{(\boldsymbol{x_i^{NN}}, \boldsymbol{l_i^{NN}})\}_{i=1}^{k}$. So, based on Equation (3),

the label prediction based on sample data (***y***) with sparse coefficient vector (***w***) can be seen in Equation (5).

$$l_y = \arg\max_c \left(\sum_{(x_i^{NN}, l_i^{NN}) \in Tr_k} \bar{\beta}_i \cdot \delta(c = l_i^{NN})\right) \quad (5)$$

CAR - KNN

The TDC used two approaches to process training data before the classification process. These approaches are Classification Ability Ranking (CAR) and Leave One Out (LV 1). In CAR, the classification ability between each training data sample using kNN is measured in 5 cases. Each case can be seen in Table 1.

Table 1. Five different cases of the classification ability of y_s [7]

Cases	Conditions		Classification result	Corresponding classification ability of y_s
1	$y_s \in NN_k^{Tr_{y_j}}(y_j)$ and	$c_s = l_y \cap c_s = c_j$	Correct	$ca_{yj}(y_s) = a_1$
2		$c_s \neq l_y \cap c_s = c_j$	Wrong	$ca_{yj}(y_s) = a_2$
3		$c_s \neq l_y \cap c_s \neq c_j$	Uncertain	$ca_{yj}(y_s) = a_3$
4		$c_s = l_y \cap c_s = c_j$	Wrong	$ca_{yj}(y_s) = a_4$
5	$y_s \notin NN_k^{Tr_{y_j}}(y_j)$		/	$ca_{yj}(y_s) = a_5$

From Table 1, a_i is the corresponding classification ability of training sample (y_s) in each case where $a_i \in [0,1], a_1 = 1$ and $a_5 = 0$. The first two cases illustrate the events where the y_s has a positive contribution to the classification of testing sample (y_j), since the sample y_s not only exists in the KNN of y_j but also has the same class label of y_j. More specifically, in Case 1, y_s exists in the majority class of $NN_k^{Tr_{y_j}}(y_j)$ and has a more direct impact on the final correct classification of y_j, whereas in Case 2, y_s has the potential to predict the true class of y_j due to $c_s = c_j$ even though it holds a different class label with the majority class. On the other hand, Case 3 and Case 4 illustrate that the training sample y_s has a negative contribution to the classification of y_j. In both cases, y_s may lead to a misclassification of y_j due to $c_s \neq c_j$. In particular, y_s in Case 4 is more harmful to the classification of y_j since y_s belongs to the majority class of $NN_k^{Tr_{y_j}}$. In addition, Case 5 illustrates that y_s is useless for classifying y_j as it is not selected as one of the kNN of y_j.

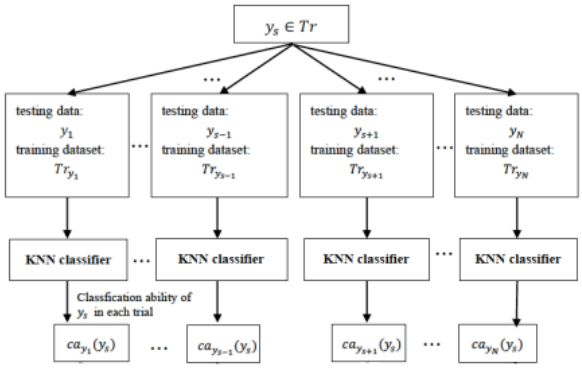

Fig 1. Framework to calculate classification ability of each selected training sample y_j based on the LV 1 strategy in CAR-based TDC method [7].

The LV 1 generally used strategy is the cross-validation method. Each sample in the training data set will be assumed to be a test sample. After that, the classification ability is measured against the remaining samples. Figure 1 shows the final model of CAR and LV 1.

Pseudo code 1 The Algorithm of the *CAR-CWKNN* Classifier

Input:

The original training dataset (*x_train*)

The corresponding classification ability for five different cases ($a_1 \ldots a_5$; $\bar{a} = \frac{a_1+a_2+a_3+a_4}{4}$)

The parameter *k_cl* and *k_tr* for KNN classification in the cleaning stage and classification stage

Output:

The class label (l_y) of the testing sample (*x_test*)

Procedure:

Step 1. Cleaning Stages

Step 1.1. Determine the suitability of each training data to the five categories of classification ability using the LV 1 strategy and *k_cl* variable.

Step 1.2. Obtain the ranked sequence $Tr_{ranked} = \{(x_i^{NN}, l_i^{NN}) \in Tr\}_{i=1}^{N}$ in descending order according to their classification abilities, where $ca_{yj}(y_1) \geq \cdots \geq ca_{yj}(y_i) \geq \cdots \geq ca_{yj}(y_N)$ holds.

Step 1.3. Assign each sample of training data to $Tr_{cleaned}$ set if the corresponding sample has greater ability than the average classification abilities in Tr_{ranked}

Step 2. Classification Stages

Step 2.1. Solve the sparse representation of y using $Tr_{cleaned}$.

Step 2.2. Find first *k_tr* largest sparse coefficients from $\bar{\beta}$

Step 2.3. Choose the *k_tr* nearest neighbors $Tr_{k\_tr} = \{(x_i^{NN}, l_i^{NN})\}_{i=1}^{k\_tr}$ corresponding to *k_tr* largest sparse coefficients $\bar{\beta}$.

Step 2.4. Classify ***y*** to the class l_i^{NN} with the largest sum of weights among all the class using equation (5).

After knowing the classification ability of each sample in training data, the sample which rate is

below the average classification ability will be removed from the original training dataset.

CAR-CWKNN Classifier

This article proposes the CAR-CWKNN method, particularly the CAR method is used when preparing training data and CWKNN when conducting the final classification. The algorithm of the proposed method can be seen in Pseudocode 1. Pseudocode 1 shows the initial stages of cleaning training data. The next stage is measuring the classification ability ranking of each training data sample using the defined CAR method. After that, each classification ability of the training data sample is assigned in the set. The set is then sorted by descending rules according to the ranked sequence of classification ability. Then each ability value that is higher than the average ability value in the set will be selected and assigned into the cleaned data set.

The next stage is to obtain sparse representation y towards cleaned training data. After that, determine the k value of the largest coefficient of the vector. Then, compute the k nearest neighbors of the new data sample against each cleaned training data sample. Finally, classify the new data sample using Equation (5).

RESULTS AND DISCUSSION

In this experiment, four public data sets are used from the UCI Repository of Machine Learning Datasets [14]. These data sets have been widely used for academic research to evaluate the proposed methods and the competing methods on the classification task in terms of classification accuracy. The main characteristics of four public data sets are listed in Table 2, including names, numbers of samples, attributes, and classes.

To evaluate classification performance in terms of accuracy, the classification error rate (Err.) is used for evaluation metrics of our proposed method [7], [15]. Err is the proportion of instances misclassified over the whole set of instances. It is analytically defined as the number of all incorrect predictions divided by the total number of the dataset.

Table 2. The four public UCI data sets used in the experiments

Dataset	Samples	Attributes	Classes
Heart	303	14	2
Climate	540	18	2
Blood	748	4	2
Segmentation	2310	19	7

In the proposed method, there were two k-nearest neighbor parameters used. Parameters k_{cl} and k_{tr} were used respectively in the LV 1 strategy during the cleaning stages, and at the final classification stage. Therefore, to understand the relationship between the two-parameters values on the classification performance of the CAR-CWKNN method, an experiment was carried out by assigning several values randomly to the two parameters. This experiment's results are shown in Table 3. As shown in Table 3, the CAR-CWKNN classifier performs best on all ten real-world data sets when $k_{cl} = 1.5\ k_{tr}$, particularly in conditions when each parameter's value is larger. So based on these experiments, the remaining experiments in this article use a larger k_{cl} value than the k_{tr} value.

Table 3. Average error rate of four public data sets under different relationships between k_{tr} and k_{cl} of CAR-CWKNN classifier

Value of k_{tr}	$k_{cl} = k_{tr}$	$k_{cl} = 1.5k_{tr}$	$k_{cl} = 2.0k_{tr}$
1	0.0650	**0.0475**	0.0479
3	0.0530	**0.0300**	0.0775
5	0.0475	**0.0450**	0.0575
7	0.0300	**0.0225**	0.0250

To further evaluate the proposed methods, the experiment is occurred to find the accuracy rates of each method towards all the data sets.

Table 4. The average of classification accuracy rates (%) of each method on all the data sets.

Dataset	kNN	CAR-KNN	CWKNN	CAR-CWKNN
Heart	82.95	78.6	80.2	**93.41**
Climate	86.29	91.89	90.45	**98.89**
Blood	74.4	76.92	76.51	**94.35**
Segmentation	81.86	80.94	80.04	**93.44**

Table 4 indicates the result with corresponding data sets shown in Table 2. Note that the average classification accuracy rates of CAR-CWKNN, CAR-KNN, CWKNN, and kNN are obtained among the range of the neighborhood size k from 1 to 10 in terms of the classification results in Figure 2. Table 4 indicates the proposed CAR-CWKNN outperforms the other methods in most cases.

In detail, each classification accuracy with corresponding k neighbor size is shown in Figure 2. Figure 2 (a) indicates the highest result of CAR-CWKNN is 98.38 percent with 2-k neighbor size, and the lowest result of CWKNN is 84.41 percent with 4-k neighbor size. Figure 2 (b) indicates the highest result of CAR-CWKNN is 98.81 percent with 2-k neighbor size, and the lowest result of kNN is 87.69 percent with 5-k neighbor size. Figure 2 (c) indicates the highest result of CAR-CWKNN is 95.83 percent with 1-k neighbor size, and the lowest result of CWKNN is 74.12 percent with 7-k neighbor size.

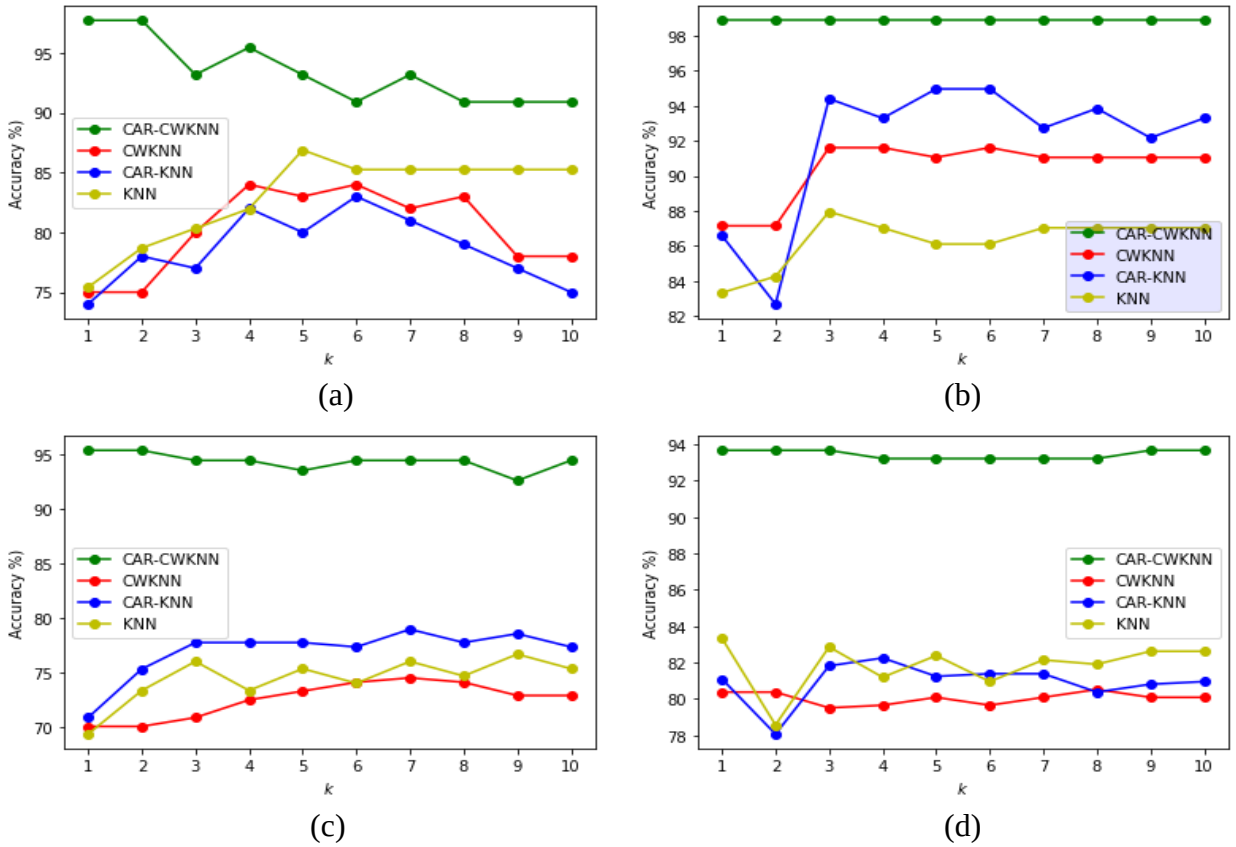

Figure 2. The classification accuracy rates of each method with corresponding neighborhood size k on four public UCI data sets. (a) Heart. (b) Climate. (c) Blood. (d) Segmentation.

Figure 2 (d) indicates the highest result of CAR-CWKNN is 93.67 percent with 2-k neighbor size, and the lowest result of CWKNN is 80.38 percent with 2-k neighbor size.

CONCLUSION

This article introduces a combined method called CAR-CWKNN. The proposed CAR-CWKNN method is inspired by both the CAR-KNN and CWKNN rules. The purpose of our CAR-KNN is mainly to improve the classification performance with the nearest neighbor-based classifier. For the investigation of the classification performance of the proposed method, classification experiments on four public UCI data sets are conducted. Comparative experimental results obtained using several KNN-based classifiers illustrate that the proposed CAR-CWKNN classifier has better performance in terms of accuracy. In further research work is how to implement the proposed methods for some practical applications in pattern recognition.

ACKNOWLEDGMENT

This article was supported by Bandung Institute of Technology regarding the requirement for lectures on computational science research methodology.

REFERENCES

[1] X. Zhu, S. Zhang, Z. Jin, Z. Zhang, and Z. Xu, "Missing Value Estimation for Mixed-Attribute Data Sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 1, pp. 110–121, Jan. 2011, doi: 10.1109/TKDE.2010.99.

[2] U. Lall and A. Sharma, "A Nearest Neighbor Bootstrap For Resampling Hydrologic Time Series," *Water Resources Research*, vol. 32, no. 3, pp. 679–693, 1996, doi: 10.1029/95WR02966.

[3] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006, doi: 10.1109/TIT.2006.871582.

[4] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang, "A Survey of Sparse Representation: Algorithms and Applications," *IEEE Access*, vol. 3, pp. 490–530, 2015, doi: 10.1109/ACCESS.2015.2430359.

[5] H. Ma, J. Gou, X. Wang, J. Ke, and S. Zeng, "Sparse Coefficient-Based k -Nearest

Neighbor Classification," *IEEE Access*, vol. 5, pp. 16618–16634, 2017, doi: 10.1109/ACCESS.2017.2739807.

[6] A. Esuli and F. Sebastiani, "Training Data Cleaning for Text Classification," in *Advances in Information Retrieval Theory*, Berlin, Heidelberg, 2009, pp. 29–41. doi: 10.1007/978-3-642-04417-5_4.

[7] Y. Wang, Z. Pan, and Y. Pan, "A Training Data Set Cleaning Method by Classification Ability Ranking for the k -Nearest Neighbor Classifier," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 5, pp. 1544–1556, May 2020, doi: 10.1109/TNNLS.2019.2920864.

[8] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967, doi: 10.1109/TIT.1967.1053964.

[9] J. Gou, Y. Zhan, Y. Rao, X. Shen, X. Wang, and W. He, "Improved pseudo nearest neighbor classification," *Knowledge-Based Systems*, vol. 70, pp. 361–375, Nov. 2014, doi: 10.1016/j.knosys.2014.07.020.

[10] S. Zhang, X. Li, M. Zong, X. Zhu, and D. Cheng, "Learning *k* for kNN Classification," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 3, p. 43:1-43:19, Jan. 2017, doi: 10.1145/2990508.

[11] S. A. Dudani, "The Distance-Weighted k-Nearest-Neighbor Rule," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, no. 4, pp. 325–327, Apr. 1976, doi: 10.1109/TSMC.1976.5408784.

[12] S. Gao, I. W.-H. Tsang, and L.-T. Chia, "Laplacian sparse coding, Hypergraph Laplacian sparse coding, and applications," *IEEE Trans Pattern Anal Mach Intell*, vol. 35, no. 1, pp. 92–104, Jan. 2013, doi: 10.1109/TPAMI.2012.63.

[13] J. Zhang and J. Yang, "Linear reconstruction measure steered nearest neighbor classification framework," *Pattern Recogn.*, vol. 47, no. 4, pp. 1709–1720, Apr. 2014, doi: 10.1016/j.patcog.2013.10.018.

[14] D. Dua and C. Graff, "{UCI} Machine Learning Repository," 2017. http://archive.ics.uci.edu/ml (accessed Oct. 11, 2021).

[15] G. Toussaint, "Bibliography on estimation of misclassification," *IEEE Transactions on Information Theory*, vol. 20, no. 4, pp. 472–479, Jul. 1974, doi: 10.1109/TIT.1974.1055260.