BLG 506E Computer Vision 2022-2023 Spring Assignment I

Resul Dagdanov - 511211135

Abstract— **This assignment focuses on developing proficiency with Numpy and PyTorch library and gaining experience using notebooks in Python language. The goal is to understand the primary image classification pipeline and the data-driven approach, including train/predict stages, train/test splits, and the use of validation data for hyperparameter tuning. The assignment includes implementing and applying the k-Nearest Neighbor (kNN) classifier. Additionally, the comparison of the prediction performance of different k values in kNN classifier is investigated in detail. CIFAR10 public dataset training samples (50000) and test image samples (10000) are used to evaluate the performance of the developed kNN classifier. Before making a final evaluation in the CIFAR10 dataset, the hyperparameter value of k in kNN classifier is tuned with a cross-validation approach. All code materials are attached to the final submission document along with detailed ReadMe and comment explanations.**

Index Terms— **Image Classifier, k-Nearest Neighbor, Python Language, Cross-Validation, PyTorch**

I. Introduction

The k-Nearest Neighbor (kNN) classifier [1] is a simple and widely used machine learning algorithm for classification problems. A method is a non-parametric approach that relies on the concept of similarity to classify new instances. The kNN classifier is based on the assumption that instances that are similar in feature space tend to belong to the same class. In other words, if a new instance has features that are similar to a set of instances in the training set, then it is likely to belong to the same class as those instances. The kNN algorithm works by finding the k-nearest neighbors to a new instance in the training set and assigning the class label that is most frequent among those neighbors to the new instance. One of the advantages of the kNN algorithm is that it can handle complex decision boundaries and can be applied to problems with large and high-dimensional datasets. However, the performance of the kNN algorithm can be sensitive to the choice of k and the distance metric used to measure similarity. In this context, selecting an appropriate k value and distance metric is crucial for achieving good classification performance.

A common benchmark dataset for image classification tasks in machine learning is the CIFAR10 dataset [2]. The dataset includes 60000 color photos divided into 10 classes, each with 6000 images. Compared to comparable datasets like ImageNet, the photos are quite modest at 32x32 pixels in size. The CIFAR10 dataset consists of 10 classes: truck, car, boat, cat, deer, dog, frog, horse, ship, and airplane. With an equal number of photos for each class in each set, the dataset is divided into a training set of 50000 images and a test set of 10000 images. Because of the relatively small image size and strong similarities between some of the classes, such as dogs and cats or ships and airplanes, the CIFAR10 dataset presents a challenge. The dataset has been used widely in studies to compare the effectiveness of different machine learning and deep learning models, and it has been crucial in the advancement of cutting-edge computer vision methods.

R. Dagdanov is with Department of Aeronautical and Astonautical Engineering, Istanbul Technical University, Turkey dagdanov21 at itu.edu.tr

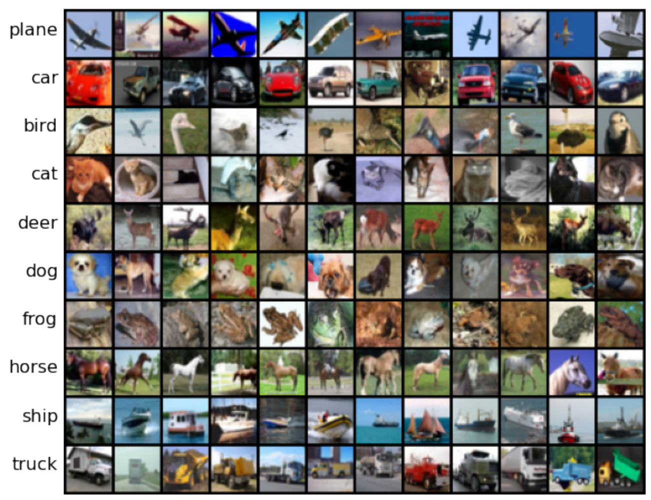

Fig. 1. CIFAR10 dataset example class label images [2].

When choosing the best hyperparameters for a model, cross-validation [3] is a popular machine-learning approach used to assess the model's performance. Finding the ideal value for the hyperparameter k, which denotes the number of neighbors to take into account when classifying a new image, can be done in the context of the k-Nearest Neighbor (kNN) image classifier using cross-validation. The process of cross-validation entails breaking the picture dataset into several subsets, or "folds," and training the kNN classifier on a subset of the images while evaluating it on the other fold. The steps are performed several times, using one fold as the evaluation set each time, and the results are averaged to provide an idea of how well the model is working. The kNN image classifier can be tuned for greater accuracy and generalization to new images by choosing the value of k that yields the greatest cross-validation performance.

II. Methodology

In this section, a detailed inspection of kNN classifier algorithm is carried out with related equations. It is possible to formulate the kNN classifier as follows:

- The kNN classifier operates as follows given a set of labeled examples $D = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where $x_i \in \mathbb{R}^d$ represents the i^{th} instance with d features, and $y_i \in 1, 2, \ldots, C$ is the corresponding class label.
- To determine the distance between each instance x_i in the training set and a new instance $x_q \in \mathbb{R}^d$, use a distance metric like Euclidean distance or cosine similarity.
- Based on the estimated distances, choose the k training set neighbors that are closest to x_q.
- Provide x_q the class label that appears the most often among the k closest neighbors. If there is a tie, a class label chosen at random will be used.

An essential hyperparameter that must be adjusted for the given problem is the selection of k. A greater k number can produce a smoother decision border but can cause some instances to be misclassified, whereas a smaller k value can produce a more flexible decision boundary but can be sensitive to noise.

In this assignment, the Euclidean distance Eq. 1 measure is used as a kNN classifier metric in this assignment.

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{1}$$

where $\mathbf{p}$ and $\mathbf{q}$ are two n-dimensional vectors representing two points in Euclidean space, and $d(\mathbf{p}, \mathbf{q})$ is the distance between them.

The formula in Eq. 1 calculates the distance between two vectors by taking the square root of the sum of the squared differences of their corresponding indices. The index i in the summation runs from 1 to n, where n is the dimensionality of the space. The q_i and p_i terms represent the ith coordinate of the points $\mathbf{q}$ and $\mathbf{p}$, respectively. The formula is named after the ancient Greek mathematician Euclid, who is considered the father of geometry. The Euclidean distance is commonly used as a distance metric in machine learning and data analysis, including the k-Nearest Neighbor (kNN) algorithm. In our assignment on the kNN image classifier, the vectors q_i and p_i are flattened image pixels in a vector format.

III. Experiments

Initially, the Euclidean distance is calculated with different approaches. The first approach is to use two loops in the code which turns out to be the most inefficient method. The second approach in calculating Euclidean distance is to use only one loop in the code implementation. This method is conducted by taking the distance measurement between each train and test image features. The last approach is to not use any loops in the code implementation. This turns out to be the most efficient approach in terms of time complexity in Table I.

Naive hyperparameter tuning for the k value in kNN classifier is implemented with the brute force approach. The list of predefined testing k values is [1, 3, 5, 8, 10, 12, 15, 20, 50, 100]. As mentioned above, as the value of k increases, the decision boundary between different classes gets much smoother. This phenomenon is observed in the following Fig. 2. It is clearly observed that the value of k affects the shape of the decision boundary of the kNN classifier. When k is small (k=1), the decision boundary can be highly irregular, following the contours of the training data. In this case, the kNN classifier can be prone to overfitting the training data, which may result in poor generalization performance on unseen data. On the other hand, when k is large (k=5), the decision boundary tends to be smoother and less sensitive to small fluctuations in the training data. In this case, the kNN classifier may be less prone to overfitting, but may also be less able to capture the fine details of the data. In general, the choice of k depends on the specific dataset and the complexity of the underlying relationship between the input and output variables. As a rule of thumb, a small value of k is preferred when the data is highly variable or noisy, while a larger value of k may be more appropriate for smoother or more regular datasets.

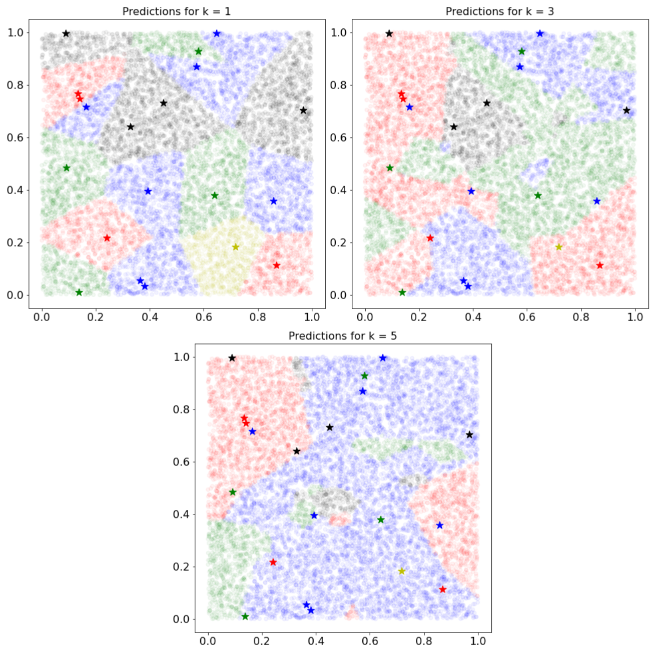

Fig. 2. Observation of how different k hyperparameter values affect decision boundary in kNN classifier.

When the value of k=1, the performance accuracy on the 500 samples of test image data in the CIFAR10 dataset is **27.40** percent. While the test accuracy for k=5 is **27.80** percent in the same dataset test samples.

TABLE I
Comparison of time complexity efficiency of different code implementations of Euclidean distance calculations.

Loops in Code	Duration	Speed Comparison
One Loop	7.36 seconds	1X Speedup
Two Loops	0.75 seconds	9.8X Speedup
Three Loops	0.01 seconds	834.3X Speedup

As briefly discussed in the Introduction section of this report, the cross-validation technique could be used to accomplish higher accuracies. Cross-validation application is implemented to get higher accuracy in kNN image classification by tuning and selecting k hyperparameter value according to best average accuracy in 5 folds. Table II shows the accuracy of each fold in the 5-fold cross-validation method for different k values. The tested k values are predefined as k = [1, 3, 5, 8, 10, 12, 15, 20, 50, 100].

TABLE II
5-FOLD CROSS-VALIDATION PERCENTAGE ACCURACY FOR DIFFERENT K HYPERPARAMETER VALUES IN CIFAR10 DATASET.

k Value	1st Fold	2nd Fold	3rd Fold	4th Fold	5th Fold
k=1	26.3	25.7	26.4	27.8	26.6
k=3	23.9	24.9	24.0	26.6	25.4
k=5	24.8	26.6	28.0	29.2	28.0
k=8	26.2	28.2	27.3	29.0	27.3
k=10	26.5	29.6	27.6	28.4	28.0
k=12	26.0	29.5	27.9	28.3	28.0
k=15	25.2	28.9	27.8	28.2	27.4
k=20	27.0	27.9	27.9	28.2	28.5
k=50	27.1	28.8	27.8	26.9	26.6
k=100	25.6	27.0	26.3	25.6	26.3

The visualization plot of the 5-fold cross-validation accuracy performance is shown in Fig. 3.

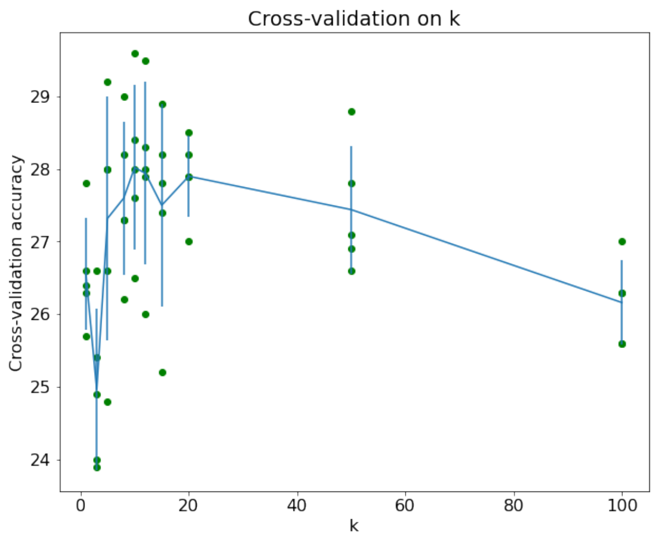

Fig. 3. 5-Fold Cross-Validation Accuracy Percentages on k.

As a result, the best hyperparameter value for k after cross-validation with 5-folds is **k=10**. By testing k=10 on all CIFAR training and test set, the test accuracy reached **33.86** percent. All of the results could be observed in the .ipynb file and code materials.

IV. ACCOMPLISHMENTS

The general list of accomplished tasks is enumerated below:

1) Implemented a K-Nearest Neighbors classifier on the CIFAR10 dataset.
2) Successfully trained the classifier by memorizing the training data.
3) Made accurate predictions on test images by comparing them to the training images and using the majority vote among the K nearest training examples.
4) Conducted cross-validation to find the best value of K for the classifier.
5) Practiced writing efficient, vectorized code in PyTorch.
6) Gained experience with the data-driven image classification pipeline.

V. CONCLUSION

n conclusion, a deeper understanding of the data-driven image classification pipeline was made possible by the application of a K-Nearest Neighbors classifier on the CIFAR10 dataset. By comparing test photos to training images and using the majority vote among the K nearest training examples, precise predictions on test images were achieved using effective, vectorized code in PyTorch. Performance was enhanced by using cross-validation to choose the classifier's optimal K value. Overall, this exercise helped with the implementation and optimization of a straightforward picture categorization method.

ACKNOWLEDGEMENT

This work is conducted as an assignment of the ”BLG 506E Computer Vision” graduate course lecture given by Prof. Hazım Kemal Ekenel.

REFERENCES

[1] L. E. Peterson, “K-nearest neighbor,” *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.
[2] Y. Abouelnaga, O. S. Ali, H. Rady, and M. Moustafa, “Cifar-10: Knn-based ensemble of classifiers,” in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2016, pp. 1192–1195.
[3] M. W. Browne, “Cross-validation methods,” *Journal of mathematical psychology*, vol. 44, no. 1, pp. 108–132, 2000.