CS 178: Machine Learning & Data Mining Project

Joselyne Guillen, Vini Mahendrakumar Patel, Thomas Nabil Issa

TOTAL POINTS

18 / 20

QUESTION 1

1 Letter Grade 18 / 20

- **- 0 pts** A+

✓ **- 2 pts** *A*

- **- 3 pts** A-/B+
- **- 4 pts** B
- **- 5 pts** B-/C+
- **- 6 pts** C
- **- 10 pts** D
- **- 20 pts** F (no submission)

1 Letter Grade **18 / 20**

- **0 pts** A+

✓ **- 2 pts** *A*

- **3 pts** A-/B+
- **4 pts** B
- **5 pts** B-/C+
- **6 pts** C
- **10 pts** D
- **20 pts** F (no submission)

Joselyne Guillen (Student ID: 65217789),
Thomas Nabil Issa (Student ID: 13229896),
Vini Mahendrakumar Patel (Student ID: 69608792)
Professor Kerrigan
CS178 - Spring 2023
June 12, 2023

CIFAR-10 Classification Methods Investigation

Team: Neural Ninjas

Summary

This investigation consists of applying various classification methods to the CIFAR-10 dataset and analyzing the results of each method. The classification methods we investigated were k-nearest neighbors (KNN), logistic regression, feedforward neural networks, and random forest. We found that KNN gave an accuracy of 30.01%, logistic regression gave an accuracy of 38.51%, random forest gave an accuracy of 44.44%, and feedforward neural network gave an accuracy of 53.58%. We concluded that the feedforward neural network is the best classifier to use for CIFAR-10 from the four classifiers we tested.

Data Description

The CIFAR-10 dataset is made up of 60,000 32x32 color images with 10 classes. Each class has 6,000 images sliced into 5,000 training images and 1,000 testing images. The dataset itself is divided into a training and test set, with the training set having 50,000 images and the test set having the remaining 10,000 images. The 10 mutually exclusive classes are: airplane, automobile (not truck or pickup truck), bird, cat, deer, dog, frog, horse, ship, and truck (but not pickup truck). Each image in this dataset is 32x32x3, where 32x32 is its height/width and the 3 in 32x32x3 is a reference to the RGB color channels. The labels are as follows: {0: Airplane, 1: Automobile, 2: Bird, 3: Cat, 4: Deer, 5: Dog, 6: Frog, 7: Horse, 8: Ship, 9: Truck}.

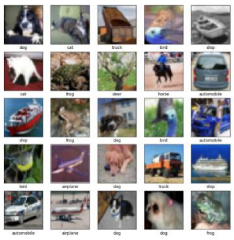


Figure 1.1: First 25 images in the training set with corresponding labels

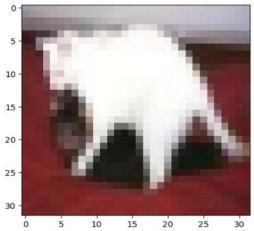


Figure 1.2: An image from the training set

Classifiers

KNN

K-nearest neighbors is a supervised learning classifier that looks at a datapoint's k nearest neighbors and bases its label prediction on the majority label of these closest neighbors. The most important hyperparameters that can be tuned are the number of neighbors k and the distance metric used to calculate nearest neighbors. We used the Euclidean distance and k values of: [1, 2, 5, 10, 50, 100, 110, and 500]. To aid our KNN investigation, we used scikit-learn's KNeighborsClassifier, model selection and metrics tools.

Logistic Classifier

The logistic classifier, also known as logistic regression, is a supervised learning algorithm that is commonly used for binary classification problems. It calculates the probability of an instance belonging to a particular class and makes predictions based on a predefined threshold. To investigate the performance of the logistic classifier, we can tune important hyperparameters such as using different penalties, or regularization functions, and different regularization strengths C. In this case, the C values used for running our logistic classifier are: [10, 1.0, 0.1, 0.01, 0.001]. These values span a range of magnitudes, allowing us to explore the effect of different regularization strengths on the model's performance.The penalties used are ["l1", "l2"]. We used scikit-learn's LogisticRegression and ConfusionMatrix to create a model and confusion matrix for the classifier.

Feedforward Neural Network

A feedforward neural network is a simple example of a neural network, which are collections of layers of neurons that process the input they receive and decide whether the output should be passed on to the next layer as input. What differentiates feedforward neural networks is that data and inputs only move 'forward' through the network, unlike more advanced models that

use advanced techniques such as filters, pooling layers, cycles and loops. Some of the most impactful hyperparameters for a neural network are the learning rate, depth/width of the network, regularization strength, activation functions, and batch sizes. Our neural network also had a learning rate of 0.0001, a batch size of 100, and 10 epochs. Additionally, the layers used in order were: (3072, 512), (512, 256), (256, 256), (256, 128), and (128, 10) in the form (depth, width). 3072 was used because the CIFAR-10 image size is 32x32x3, in order to account for the pixels in the image. Pytorch was used to implement the classifier because of ease of use in installing the packages required as opposed to other packages for other software.

Random Forest

The Random Forest classifier is a powerful supervised learning algorithm commonly used for both classification and regression tasks. It is an ensemble model that combines multiple decision trees to make predictions. In our investigation of the Random Forest classifier's performance on the CIFAR-10 dataset, we focused on hyperparameter tuning to optimize its accuracy. The hyperparameters we specifically targeted were the number of trees (n_estimators) and the maximum depth of each tree (max_depth). We experimented with a range of values for the number of trees, including [10, 50, 100, 150, 200, 250, 300, 350, 400, 450]. This allowed us to assess the impact of the number of trees on the classifier's performance. Similarly, we explored various maximum depth values, such as [10, 20, 30, 40, 50, 60, 70, 80, 100, None], to determine the optimal depth for the trees. To evaluate the classifier's performance, we used the scikit-learn library to train the Random Forest model on the sampled training set. We then made predictions on the validation set to calculate the validation accuracy and error rate. Additionally, we measured the accuracy and error rate on the training set to assess potential overfitting.

Experimental Setup

Before using the data on our classifiers, we wanted to ensure we had our data partitioned into a training, validation, and an isolated test set. Since CIFAR-10 already provides a test set with 10,000 images, we decided to partition the remaining 50,000 images in the provided training set to give us a validation set by shuffling the data and using an 80/20 split with scikit-learn's train_test_split function. After splitting, we ended up with 40,000 images in our training set, 10,000 images in our validation set, and 10,000 images in our test set.

We decided to subsample the first 13,000 images from the training set for a faster runtime and to reduce memory requirements. The overall metrics we looked at were accuracy scores, learning curves, and confusion matrices.

KNN

When hyperparameter tuning for kNN, we consistently used the Euclidean distance as our distance metric in our configurations. For our k values, we decided to use 8 different hand-picked values ranging from 1 to 500. We wanted a k value somewhere in between this range since small values of k lead to overfitting and big values of k lead to underfitting. For each hand-picked value of k, we fit a kNN classifier on the training set using the k value.

```
k = [1, 2, 5, 10, 50, 100, 110, 500]
for k_val in k:
    k_neighbors = KNeighborsClassifier(n_neighbors= k_val)
    k_neighbors.fit(X_train_sampled, y_train_sampled)
```

k = 10 was chosen since it had the highest validation accuracy of 29.42%.

Logistic Classifier

For the logistic classifier on the CIFAR-10 dataset, we performed hyperparameter tuning using two parameters: penalty and C. We explored two penalty options, 'l1' and 'l2', and varied the values of C, including 10, 1.0, 0.1, 0.01, and 0.001. The code initializes variables to store the best hyperparameters and accuracy, as well as lists to track the error rates for both the training and validation sets. We iterate over the hyperparameter combinations using nested loops, creating a logistic regression model with the current hyperparameters for each combination. For each combination, we fit the model to the training data and evaluate its performance on both the train set and the validation set. We calculate the accuracy and error rate for each set and store them in the respective lists. Additionally, we keep track of the hyperparameter combinations for reference. Finally, we create a line plot to visualize the error rates for different hyperparameter combinations. For the tuning we got the best combination to be regularization strength of 0.01, penalty "l2" and solver "saga" with validation accuracy of 39.19%.

Feedforward Neural Network

The seeds for numpy and torchvision were set to 1234, in order to achieve consistent results. In the hyperparameter tuning process each parameter was tested with different options, then the option that gave the highest accuracy rate was kept for the next category. First, the training, validation, testing split of 80%, 10% and 10%, and 70%, 15%, 15%, were compared, the first split was kept. Next, RELU was the most successful activation function, out of the functions: RELU and Sigmoid. Then, because batch size and epochs affect each other they were tested together. The options were batch size of 50 with epochs of 20, batch size of 100 with epochs of 10, and batch size of 200 with epochs of 5. Batch size of 100 with epochs of 10 was the most

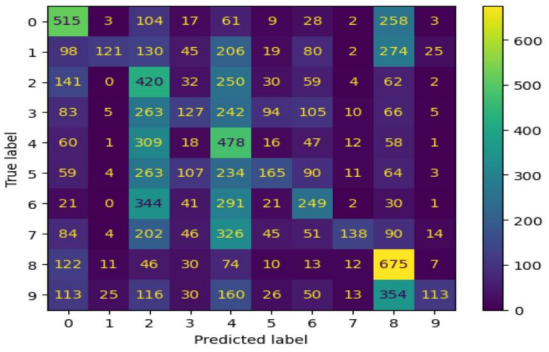
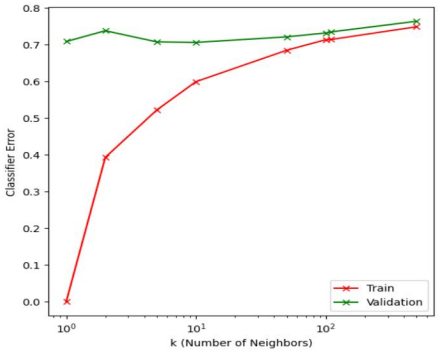
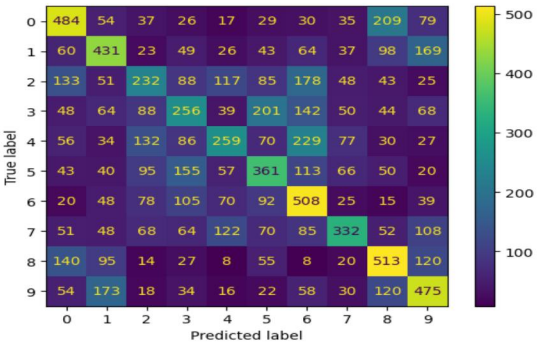
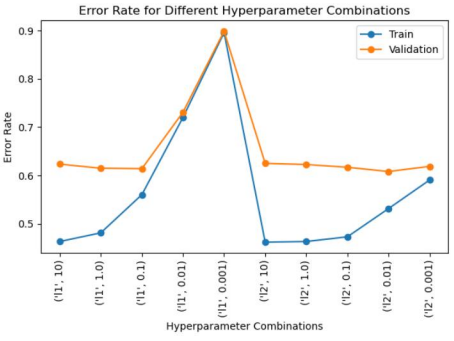
successful. Next, 0.0001 was the most optimal learning rate out of the options of: 0.01, 0.001 ,0.0001, and 0.00001. Finally, hidden layers were added until the accuracy decreased, the width of the layers was repetitively changed with the bias that width would generally decrease the further the layers were from the first input. And the best accuracy on the validation set was 48%.

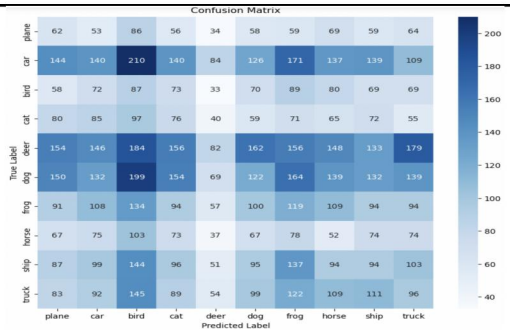
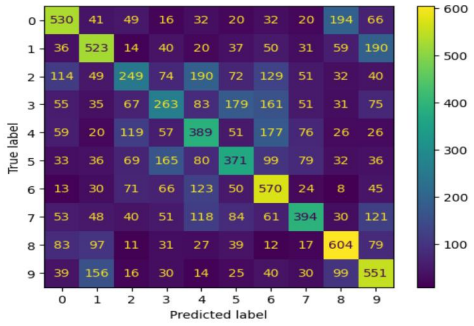
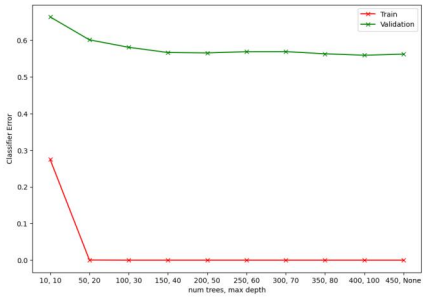
Random Forest

For the Random Forest Classifier, we chose a range of values for the number of trees ranging from 10 to 450, and for the maximum depth, ranging from 10 to 100 including 'None' (indicating no maximum depth). Using a loop, we iterated over the ten different hyperparameter combinations and trained a Random Forest Classifier with each combination on the sampled training set. We used the 'sqrt' option for the max_features parameter, which randomly selects the square root of the total number of features at each node, providing a good balance between randomness and diversity. To visualize the results, we plotted the training and validation errors against the hyperparameter combinations. After tuning, we got the best hyperparameter combination to be 400 numbers of trees, 100 as max depth, and "sqrt" as max features with validation accuracy of 44.07%.

Experimental Results

Our results show accuracy scores for all 4 classifiers obtained on the isolated test dataset and the tuned hyperparameters. Based on these scores, we can conclude that feed forward neural networks are the most accurate out of the 4 in classifying images using the CIFAR-10 dataset with an accuracy of 53.58%. Random forests were second best with an accuracy of 44.44%. The classifier with the poorest accuracy was KNN with 30.01%. Logistic regression had an 8.5% difference from KNN with an accuracy of 38.51%. All 4 classifiers had some similar errors such as wrongly classifying cars as planes, deer as birds, and dogs as cats. Some of the toughest classes for our classifiers were deer, frogs, and ships. The simpler classifiers such as KNN, logistic regression, and random forest often misclassified cats.

Classification Method	Accuracy score%	Confusion matrix	Error rate graph
KNN	30.01%		
Logistic Regression	38.51%		

Feed Forward neural network	53.58%		
Random forest	44.44%		

Insights

KNN seems to have the best accuracy when classifying planes and ships. This could be due to the fact that planes are found in the sky and ships are found in the sea, which means they have distinct background noise KNN can use as a discriminator. The animal classes are often misclassified, likely due to the fact that animals such as dogs, cats, horses, and deer can have similar shades of fur while frogs and birds either have colors found in fur or vibrant colors found in backgrounds.

The logistic classifier struggled to accurately predict labels for classes 2, 3, and 4. This indicates a difficulty in distinguishing between bird, cat, and deer images. This could be happening because as logistic regression is only able to assume linearity for features, these labels being animals with similar color could have similarity which make it difficult to predict. The precision (38.26%) and recall (38.51%) scores suggest that the model struggled to identify positive instances for these classes. Logistic regression is efficient and interpretable, but it may not capture complex relationships between features.

We expected our random forest classifier to give us a higher accuracy score since it is a more sophisticated model. However upon reflection, random forests wouldn't be good for perceptual data such as images because they don't do feature extraction. We also could have improved the classifier's accuracy if we used a larger number of trees (n_estimators) when training, but increasing the number of trees would lead to an increase in computational training, which isn't optimal. It is likely our results wouldn't differ by a large margin if we increased the number of trees in the forest.

According to the confusion matrix the feedforward neural network shows a bias against classifying pictures as deer and a bias towards classifying pictures as birds. The error rate for both validation and training error reached their lowest point around 9 epochs. Additionally, The classifier could have been more successful if the parameters of the feedforward neural network were tuned according to a logical basis, instead of successively testing then implementing batches of parameters because the current approach failed to consider numerous hyperparameter combinations that may have been more effective.

Overall, the feedforward neural network was most effective in terms of accuracy in comparison with the other 3 classifiers. Further research should be done with other evaluation metrics outside of accuracy score, such as training speed and learning curves to further differentiate the differences between using the discussed classifiers on CIFAR-10.

Contributions

Joselyne: KNN analysis/tuning, summary, data description, experimental setup, random forest analysis/tuning, insights, and experimental results

Vini: Logistic regression analysis/tuning, experimental setup, random forest analysis/tuning

Thomas: Feedforward Neural Network analysis/tuning, summary, experimental results, random forest analysis/tuning