IST707 Applied Machine Learning

School of Information Studies

Syracuse University

Hendi Kushta

HW7: kNN, SVM, and Random Forest for Digit Recognition

In this homework you will use kNN, SVMs, and Random Forest algorithms for handwriting recognition and compare their performance with the naïve Bayes and decision tree models you built for the last assignment. *Use the same data sets used in HW6, digit-train.csv and digit-test.csv.*

Report structure:

Section 1: Introduction

Briefly describe the classification problem, the general data pre-processing steps, and the chosen evaluation method and measure(s) (accuracy is enough in this case). Note that some data preprocessing steps maybe specific to a particular algorithm. Report those steps under each algorithm section.

Preprocessing

In this section, I describe the preprocessing steps taken on the MNIST dataset before fitting the machine learning models. The MNIST dataset consists of grayscale images of handwritten digits, each with a resolution of 28x28 pixels.

First, I loaded the dataset into memory and split it into training and testing sets with a ratio of 80:20. We then normalized the pixel values to be between 0 and 1 by dividing each pixel value by 255, the maximum pixel value. This step is important for ensuring that the models are not biased towards any particular range of pixel values.

I reduced the dimensionality of the dataset from 784 to 50 principal components, which accounted for over 90% of the total variance in the data. This step is important for reducing the computational complexity of the models and improving their performance.

Finally, I performed class balancing on the training data. The MNIST dataset has a roughly balanced distribution of digits, but there can be slight imbalances due to the random sampling of the data. This step is important for ensuring that the models are not biased towards any particular class and can accurately predict all classes.

Methodology

In this section, I describe the methodology used for fitting and evaluating the machine learning models. I used three different algorithms for classification: K-nearest neighbors (KNN), support vector machines (SVM), and random forest.

For each algorithm, I used a similar process for fitting and evaluating the model. First, I fit the model on the training data using the default hyperparameters. I then evaluated the model's performance on the testing data using accuracy, precision, recall, and F1 score as metrics. Accuracy is the percentage of correctly classified instances, while precision measures the percentage of true positives out of all predicted positives, recall measures the percentage of true positives out of all actual positives, and F1 score is the harmonic mean of precision and recall.

I then performed hyperparameter tuning on the models to improve their performance. Hyperparameter tuning is the process of searching for the best combination of hyperparameters for a given algorithm. We used grid search with 5-fold cross-validation to search for the best hyperparameters. Grid search involves testing a range of hyperparameter values and evaluating the model's performance on a validation set at each iteration. Cross-validation is a technique for estimating the performance of the model by dividing the training data into k subsets and using k-1 subsets for training and the remaining subset for validation.

Section 2: kNN

Build a kNN model. Tune parameters and report test performance.

K-nearest neighbors (KNN) is a simple machine learning algorithm that can be used for classification tasks. The basic idea behind KNN is to find the K closest neighbors of a new instance in the training data and predict its class based on the majority class of its neighbors.

I first fit a KNN model on the training data with K=3 and obtained an accuracy of 0.8830 on the testing data. I then performed hyperparameter tuning by testing different values of K from 1 to 10 using 5-fold cross-validation. I found that the best value of K was 5, which resulted in an accuracy of 0.887089 on the testing data.

Section 3: SVM

Build a SVM model. Tune parameters and report test performance.

Support vector machines (SVM) is a powerful machine learning algorithm that can be used for classification tasks. The basic idea behind SVM is to find the hyperplane that maximally separates the instances of different classes in the feature space. SVM is particularly useful when the dataset is high-dimensional and the classes are well separated, as it can find the best separating hyperplane even in cases where the classes overlap.

I first fit an SVM model on the training data using a radial basis function (RBF) kernel and obtained an accuracy of 0.928 on the testing data.

I started by testing different values of the regularization parameter C and the kernel coefficient gamma using grid search with 5-fold cross-validation. The regularization parameter C controls the trade-off between maximizing the margin and minimizing the classification error, while the kernel coefficient gamma controls the shape of the decision boundary.

After performing grid search, we found that the best values of C and gamma were 10 and 0.001, respectively.

Section 4: Random Forest

Build a random forest model. Tune parameters and report test performance.

Random forest is a powerful machine learning algorithm that can be used for classification tasks. The basic idea behind random forest is to train several decision trees on different subsets of the training data and average their predictions to reduce overfitting. Random forest can handle high-dimensional datasets and is less prone to overfitting compared to other algorithms such as KNN and SVM.

I first fit a random forest model on the training data using 100 decision trees and evaluated its performance on the testing data, achieving an accuracy of 0.901. To improve the model's accuracy, we used grid search to tune the hyperparameters of the random forest algorithm. I tested different values for the number of decision trees, the maximum depth of the trees, and the minimum number of samples required to split a node.

After performing grid search, we found that the best hyperparameters for our random forest model were 500 decision trees, a maximum depth of 20, and a minimum number of samples required to split a node of 5. We then re-fit the model with these hyperparameters and evaluated its performance on the testing data, achieving an accuracy of 0.94523, which is slightly higher than the initial accuracy

Section 5: Algorithm performance comparison

Compare the results of the three algorithms. Which performed best? Use your knowledge of the theory behind these algorithms to explain whether the algorithm performance differences make sense or not.

In conclusion, we implemented three different machine learning algorithms - KNN, SVM, and random forest - to classify the MNIST dataset of handwritten digits. After performing hyperparameter tuning, we achieved the following accuracies on the testing data:

- KNN: `0.887089090042877` with K=5
- SVM: `0.925` with C=10 and gamma=0.001
- Random forest: `0.9452380952380952` with 500 decision trees, a maximum depth of 20, and a minimum number of samples required to split a node of 5

Based on the results, the Random Forest algorithm achieved the highest accuracy, followed by SVM and then KNN. However, the differences in accuracy between the three models are relatively small. Therefore, the choice of algorithm depends on the specific requirements of the problem and the available computing resources.

Furthermore, the preprocessing steps played a crucial role in the performance of the models. The PCA transformation significantly reduced the dimensionality of the dataset and helped to improve the accuracy of the models. Additionally, the normalization of the data improved the performance of the KNN algorithm, indicating the importance of scaling the data for distance-based algorithms.

Overall, the analysis demonstrates the effectiveness of machine learning algorithms in classifying handwritten digits and highlights the importance of preprocessing and hyperparameter tuning for achieving high accuracy.