OmniTech
Kolby Boyd
ITAI 1378
L04

Image Classification with k-Nearest Neighbors

Reflection on Learning:

1. I understand that the k-Nearest Neighbors (k-NN) algorithm is a simple yet effective method for classification tasks. The k-NN algorithm classifies a data point based on the majority class of its k nearest neighbors in the feature space (the n-dimensional points where the variables live).
2. Applying k-NN to the MNIST dataset involved measuring the distances between features that represent images of handwritten numerical digits. The algorithm then classified each image based on the labels of its nearest neighbors.
3. Preparing the MNIST dataset involved loading it, visualizing training samples, and splitting the data into training and testing sets. Training the k-NN classifier was straightforward using sci-kit-learn's implementation. The evaluation involved making predictions on the testing set and assessing its accuracy.
4. One challenge I encountered was choosing an appropriate value for k. I experimented with different values and observed how it affected the model's performance. Insights gained include the importance of feature scaling and the computational cost of making predictions with k-NN.

Responses to Lab Questions:

1. The k-NN algorithm is easy to understand and implement. This makes it a great choice for beginners. k-NN doesn't have a training phase so it can be used for both supervised and unsupervised learning tasks without the need for model training. Predictions made by the k-NN algo can be easily interpreted, because they are based directly on the data. Some cons of the k-NN algo are that as the dataset increases, the computational cost grows significantly, especially since the algorithm stores the entire dataset within its memory. This also increases the prediction time. k-NN is often referred to as "lazy," because it doesn't build a model during the training phase. Instead, it memorizes the training data and makes predictions based on the similarity of new instances to existing instances in the training data.
2. The Implications of this include:
    a. No upfront computational cost due to no training phase.
    b. Best for smaller size datasets for prediction time costs.
    c. Storage must be sufficient depending on the size of the dataset.

Advanced Questions:

1. Using feature selection algorithms or feature importance ranking may reduce the computation time in k-NN for larger datasets. Working with a representative sample of the dataset instead of using the entire dataset may lead to faster computation. This can result in a sample of results to be tested on a more robust model and can be done repeatedly on small sections of the dataset. Using parallel processing techniques in order to distribute the computation across multiple processors may reduce the overall computation time, as well.
2. In k-NN, overfitting can occur when the value of k is too small, leading to a noise sensitive model. On the other hand, underfitting can occur when the value of k is too large, resulting in a model that is too simple and fails to capture the patterns in the data.
3. Some strategies to address the balancing act between underfitting and overfitting include k-fold cross validation to find the optimal value of k that balances bias and variance. Also, experimenting with the combination of multiple k-NN models with different values of k or using different distance metrics to create an "ensemble" model can help to reduce the risk of over- or underfitting.

Inclusion of Visuals:

This block took longer with a larger dataset.

```
[*]: # Make predictions
     y_pred = knn_clf.predict(X_test)
```

The model's accuracy improved with a smaller dataset, the first image ([10]) with dataset size of 0.8 and the last image ([14]) with a dataset size of 0.1.

```
[10]: # Calculate accuracy
      accuracy = accuracy_score(y_test, y_pred)
      print(f"Model accuracy: {accuracy * 100:.2f}%")

      Model accuracy: 95.39%
```

```
[14]: # Calculate accuracy
      accuracy = accuracy_score(y_test, y_pred)
      print(f"Model accuracy: {accuracy * 100:.2f}%")

      Model accuracy: 97.17%
```

Critical Analysis:

1. I critically analyzed the results of the experiments, noting the strengths and limitations of the k-NN algorithm for image classification.

Reference:

1. Pedregosa et al. "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research 12 (2011): 2825-2830.