SVC Classification Report

Papagrigoriou Vasileios Savvas
Auth

I. Introduction

In this study, I analyze their performance on the CIFAR-10 and Letter Recognition datasets, focusing on the impact of kernel choice, parameter tuning, model train/test accuracy and training duration. Also, I compare the performance of the SVM models with other classifiers, such as K-Nearest Neighbors (KNN) and Nearest Class Centroid (NCC), to understand the relative strengths and weaknesses of each classifier.

II. Methodology

The CIFAR-10 dataset consists of 50,000 32x32 color images in 10 classes, while the Letter Recognition dataset includes 20,000 instances of 26 capital letters in various fonts. I employed various SVM kernels: RBF, Polynomial (Poly), and Linear, with different settings for the hyperparameters C and gamma.

III. Notes

A. Code and Machine Specifications

- The code was written in Python 3.11.3 and executed on VSCode.
- The code was executed on a Windows 11 machine with 16GB of RAM and an Intel i7-7700 CPU.

B. Code Organization

- I used two different notebooks for the two datasets.
- One notebook was dedicated to the CIFAR-10 dataset, and the other to the Letter Recognition dataset.
- The notebook was utilized to keep track of the results.

C. Code Execution

- The CIFAR-10 notebook was hard to run on my machine so in order to have some results I truncated the dataset to 2000, 4000 and 8000 instances.
- The Letter Recognition dataset was easier to run so I used 3 different batch sizes: 2000, 8000, 12000. i.e. 12000 is the 60% of 20.000 from dataset.
- Additionally, I attempted to use GridSearchCV on the CIFAR-10 dataset, but it yielded no results due to excessive time consumption.

D. Code Output

- The code outputs the results in a table format.
- The results are either as an output of the code or commented as a markdown cell.

E. ChatGPT

- ChatGPT was utilized to enhance the grammar and syntax of some sentences in this document.

IV. Data Specifications

A. CIFAR-10 Dataset

I performed our analysis on different subsets of the CIFAR-10 dataset. In particular, I made use of the subsequent data batches:

- A collection of 2.000 pictures: Understanding the models' performance with little data is essential for comprehending their capacity to learn from a smaller sample size, and this smaller dataset offered insights into this capacity.
- A collection of 4.000 thousand pictures: I was able to assess how the model's performance scaled with an increase in the quantity of training data thanks to this medium-sized dataset.
- A collection of 8.000 thousand pictures: With a larger batch size, the models' performance was evaluated in situations where a greater quantity of data was available, more indicative of real-world circumstances.

These varying sizes helped in comprehensively understanding the scalability and adaptability of the SVM models with respect to different volumes of data.

B. Letter Recognition Dataset

In the case of the Letter Recognition dataset, I extended our analysis to cover a wider range of data batch sizes to evaluate the performance of SVM kernels more thoroughly. The data batches used were:

- A batch of 2.000 instances: This size served to test the models' performance in a constrained data environment, which is often a realistic scenario in many applications.
- A batch of 8.000 instances: This batch size was crucial to understand the model behavior with a moderately large dataset.
- A batch of 12.000 instances: This size, being closest to the full dataset, was critical for assessing the maximum performance capability of the SVM models.

These data batches were instrumental in providing a comprehensive view of how each SVM kernel performs across different volumes of data.

V. Implications of Data Size on Model Performance

The varied data sizes in both datasets allowed us to discern critical insights into the performance scalability of the SVM models. It became evident that as the size of the training data increases, the models' ability to generalize improves, but this also comes with the trade-off of increased training times, particularly for certain kernels like RBF. The smaller batches were useful in highlighting the models' efficiency and quick adaptability, whereas the larger batches emphasized the models' robustness and generalization capabilities.

VI. Detailed Analysis

A. CIFAR-10 Dataset Analysis

1) Kernel Performance: In the CIFAR-10 dataset, the Poly kernel outperformed the RBF and Linear kernels in terms of classification accuracy. This indicates that the Poly kernel's ability to model complex, non-linear relationships in image data is particularly effective for the diverse and intricate patterns present in CIFAR-10's image set. The RBF kernel, while generally robust across various datasets, may not capture the specificities of image data as effectively as the Poly kernel in this context.

2) Impact of Hyperparameters: The performance variations with different settings of C and gamma parameters indicate the sensitivity of SVMs to these hyperparameters. Optimizing these parameters is crucial for achieving the best possible model performance. This suggests the need for a more rigorous hyperparameter tuning process, possibly employing grid search or randomized search methods.

3) Training Times and Computational Efficiency: The CIFAR-10 dataset presented notable differences in training times across different models. Models with higher computational demands, although potentially more accurate, may not be feasible in time-sensitive or resource-constrained environments. This finding highlights the need for a balanced approach to model selection, considering both accuracy and computational efficiency.

B. Letter Recognition Dataset Analysis

1) Kernel Suitability: For the Letter Recognition dataset, the RBF kernel consistently delivered high performance, especially with larger data batches. This suggests that the RBF kernel's capability to handle non-linear patterns is well-suited to the task of recognizing different fonts and styles in letter images. The Poly kernel also showed promising results, particularly in larger datasets, indicating its potential applicability in similar text recognition tasks.

2) Balancing Training Time and Model Performance: The trade-off between training time and model performance was evident in this dataset as well. While the RBF kernel achieved higher accuracy, its longer training times may not be practical for all applications. Conversely, the Poly kernel, with its quicker training times, could be a more feasible option for applications with limited computational resources.

VII. Letter Recognition Dataset Performance Analysis

A. Impact of Scaler and Encoder on Classifier Performance

Tables detailing SVM performance on the Letter Recognition dataset illustrate these impacts:

Without Standard Scaler and Label Encoder: The first table{TABLE 1} shows SVM performance without these preprocessing steps. The 'rbf' kernel with C=10 and gamma=auto achieved high training and testing accuracy (1 and 0.971, respectively), with an F1 score of 0.970534.

With Standard Scaler and Label Encoder: The second table{TABLE 2} reveals the impact of preprocessing. For instance, the 'rbf' kernel with C=10 and gamma=0.1, post-preprocessing, showed a training accuracy of 0.99725 and a testing accuracy of 0.9675, with an F1 score of 0.967001.

While preprocessing steps like scaling and encoding have a marginal impact on accuracy and F1 score, their significant contribution to reducing computational time underscores their importance in practical machine learning applications

B. Performance

TABLE I: SVM Performance on Letter Recognition (12000 Data Batch)

Kernel	C	gamma	Train Accuracy	Test Accuracy	F1 Score	Time Taken
rbf	10	auto	1	0.971	0.970534	0.189119
poly	0.1	auto	0.999417	0.942875	0.942055	0.165183
linear	1	auto	0.879333	0.853625	0.851967	0.176536

TABLE II: SVM Performance on Letter Recognition (12000 Data Batch) using Label Encoder and StandarScaler

Kernel	C	gamma	Train Acc	Test Acc	F1 Score	Time Taken
rbf	10	0.1	0.99725	0.9675	0.967001	0.175049
poly	10	0.1	0.9935	0.9475	0.947067	0.0753085
linear	1	auto	0.8715	0.853	0.851172	0.0778167

TABLE III: SVM Performance on Letter Recognition (8000 of 12000 Data Batch)

Kernel	C	gamma	Train Acc	Test Acc	F1 Score	Time Taken
rbf	100	auto	1	0.96275	0.962294	0.11026
poly	0.1	auto	0.999875	0.92475	0.924075	0.0860672
linear	1	auto	0.889	0.845875	0.84401	0.0931167

TABLE IV: SVM Performance on Letter Recognition (2000 of 12000 Data Batch)

Kernel	C	gamma	Train Acc	Test Acc	F1 Score	Time Taken
rbf	10	auto	1	0.88575	0.885288	0.0223
poly	0.1	0.1	1	0.854375	0.852639	0.0237332
linear	0.1	auto	0.8775	0.80375	0.802889	0.0206833

TABLE V: Optimal Performance of Different Kernels on CIFAR-10 Dataset

Kernel	C	Gamma	Train Acc	Test Acc	F1Score	Time (s)	Batch
Poly	10	auto	1	0.407	0.4091	537.393	8000
Poly	1	0.0001	1	0.3757	0.3772	315.936	4000
RBF	100	1.00E-10	0.681	0.3303	0.3269	978.653	2000
Linear	10	0.0001	1	0.3073	0.3084	414.259	4000

VIII. Classifier Performance Summary

A. CIFAR-10 Dataset

I analyze the performance of K-Nearest Neighbors (KNN), Nearest Class Centroid (NCC), and Support Vector Machine (SVM) classifiers on a CIFAR-10 dataset subset of 4,000 data points. The SVM, with a polynomial kernel, achieved high training accuracy (1.0) and testing accuracy of 0.3757 across varying C (1, 10, 100) and gamma (auto, 0.0001, 0.001) values. Its F1 score was approximately 0.3772, with processing times ranging from 288 to 389 seconds.

The KNN, trained on 50,000 data points, had a testing accuracy of 0.3398 and an F1 score of 0.326017, but a longer processing time of 279.211 seconds. The NCC showed a best F1 score of 0.255551 with a rapid processing time of 0.605985 seconds. These results indicate SVM's superiority in accuracy and F1 score, while KNN and NCC offer benefits in speed or larger dataset handling.

B. Letter Recognition Dataset

For the Letter Recognition dataset with 12,000 training data points, the performance analysis of SVM, KNN, and NCC classifiers shows distinct outcomes. The SVM models excelled, with the top-performing model achieving a testing accuracy of 0.95025 and an F1 score around 0.95. In contrast, KNN's performance varied more broadly, with an F1 score difference ranging from -0.067031 to 0.058034 compared to SVMs. The NCC classifier, while significantly faster, had its best F1 score lower than SVMs by up to -0.417116.

C. Overall

In summary, while SVMs excel in accuracy and F1 score, KNN and NCC are more suited for scenarios prioritizing speed or handling larger datasets, despite their lower accuracy metrics.