Collaborative and Adaptive Classifier Systems

*Machine Learning HW1

Liang-Jen Huang
Institute Of Data Science, NCKU
November 17, 2024

***Abstract*—This assignment focuses on two key objectives: Collaborative Multi-Classifier System Design and Adaptive Classifier Adjustment System. Using the Mini-ImageNet dataset, the goal is to develop a system that integrates multiple classifiers (e.g., k-NN, SVM, Decision Tree) to improve classification accuracy. The system should also dynamically select the most suitable classifier and tune its parameters based on dataset characteristics, such as class imbalance or sample density, to achieve optimal performance in varied scenarios. The code is available at: https://github.com/edogawa-liang/ML-2024Fall/tree/main/HW1**

I. Introduction

Choosing the right classifier and fine-tuning its parameters is crucial for getting good results in machine learning, especially when dealing with diverse or tricky datasets. In this project, we're building a collaborative multi-classifier system that brings together the strengths of classifiers like k-NN, SVM, and Decision Tree to tackle complex classification tasks more effectively.

We're using the Mini-ImageNet dataset, which is pretty challenging because it has a lot of classes but not many samples. To handle these difficulties, our system can dynamically pick the best classifier and adjust its parameters based on key features of the data, like how imbalanced the classes are or how densely packed the samples are. This flexibility lets the system adapt to different kinds of data distributions, making it more robust and efficient in a variety of scenarios.

The primary objectives of this project are two main goals:

1) Collaborative Multi-Classifier System Design: Develop a system that integrates multiple classifiers, such as k-NN, SVM, and Decision Tree, to leverage their complementary strengths and improve classification accuracy.

2) Adaptive Classifier Adjustment System: Create a dynamic system that selects the most suitable classifier and fine-tunes its parameters based on dataset characteristics, such as class imbalance and sample density, ensuring optimal performance across diverse data distributions.

II. Method

A. Classifier Selection Logic

In my code, the *auto* mode incorporates a classifier selection logic to automatically determine the most suitable model—k-NN, Decision Tree, or SVM—based on the dataset's characteristics, such as noise levels and feature relationships. DBSCAN is used to assess the noise ratio: if the ratio is low, k-NN is selected for its reliability in clean datasets where proximity relationships are meaningful.

When noise levels are higher, the system examines feature correlations. Decision Trees are preferred when correlations are high, as they efficiently handle potential interaction effects between features. In cases where noise is high and correlations are low, SVM is chosen for its robustness in handling complex decision boundaries, though it comes with higher computational cost.

B. Combination Strategies

The weighted voting mechanism in my system uses the model performance on the training set to allocate the weight for each model. Each classifier's weight is determined by its accuracy relative to the total accuracy:

$$\text{weight}_i = \frac{\text{accuracy}_i}{\text{total\_acc}}$$

This ensures that more accurate classifiers have a larger influence on the final prediction.

C. Adaptive Selection Logic

The model selection process in my system is designed to adapt to the dataset's level of imbalance. For imbalanced data, skewness is calculated to measure class imbalance, leading to the selection of SVM, which can effectively manage uneven distributions. k-NN, on the other hand, is less suitable for imbalanced data due to challenges in determining an appropriate k when class sizes differ significantly. For balanced datasets, k-NN is favored for its strength in leveraging proximity relationships. However, SVM is sensitive to imbalanced data as it tends to maximize the margin, potentially leading to biased decision boundaries favoring the majority class. To address this, dynamically adjusting the C-parameter mitigates this issue by balancing the trade-off between margin size and misclassification errors, enabling SVM to perform better on imbalanced datasets.

D. Dynamic Parameter Adjustment

The system dynamically determines k for k-NN and C for SVM based on dataset characteristics, optimizing performance under varying conditions.

1) Deciding k for k-NN: k is mapped from density (the ratio of sample count to average pairwise distance) using:

$$k = k_{\text{min}} + \left(\frac{\log(1+\text{density})}{\log(1+\text{max\_density})}\right) \times (k_{\text{max}} - k_{\text{min}})$$

K_{max} and K_{min} set the range for the k-NN parameter k, and density is calculated below:

$$density = \frac{Number\ of\ Samples}{Average\ Pairwise\ Distance}$$

To save time, I randomly pick 1,000 samples to estimate density instead of calculating pairwise distances for the entire dataset. For low-density data, where the samples are sparsely distributed, smaller k are more effective for k-NN as they focus on closer, more relevant neighbors. In contrast, high-density data benefits from larger k, which provide smoother predictions by averaging over a greater number of neighbors, capturing the overall structure of the dataset more effectively.

2) Deciding C for SVM: This regularization parameter C in SVM is based on the skewness of the dataset. The goal is to dynamically adjust C to handle imbalanced data distributions better.

The process begins by calculating a weight that reflects the level of skewness in the data. The formula for the weight is:

$$\text{Weight} = \frac{\log(1 + (\text{max\_skewness} - \min(\text{skewness}, \text{max\_skewness})))}{\log(1 + \text{max\_skewness})}$$

The skewness value is capped at max_skewness to avoid extreme values. The difference between max_skewness and the actual skewness highlights higher levels of imbalance. The $\log(1 + x)$ function smooths this effect, making the adjustment less sensitive to large changes. Finally, dividing by $\log(1 + \text{max\_skewness})$ scales the weight to the range $[0, 1]$, ensuring it fits within the defined limits.

The weight is then used to determine C as follows:

$$C = C_{\text{min}} + \text{Weight} \cdot (C_{\text{max}} - C_{\text{min}})$$

The weight is mapped linearly between C_{min} and C_{max}. For highly imbalanced datasets, the weight decreases, making C closer to C_{min}, which helps SVM generalize with a larger margin. For balanced datasets, the weight increases, bringing C closer to C_{max}, allowing SVM to fit the data more tightly.

III. Experiments

A. Effect on Combination Strategies

TABLE I
Multi-Classifier System Design Result

	ACC%	PR%	RE%	F1%
KNN	6.04	7.76	6.04	4.96
SVM	3.42	4.04	3.52	3.13
Decision Tree	5.81	6.22	5.81	4.28
Combination	8.42	8.96	8.42	7.64

Looking at the table I, the results demonstrate the advantage of combining classifiers through a collaborative system. While individual classifiers may perform well in specific scenarios, they often struggle with complex datasets, as reflected in the lower scores for SVM and Decision Tree. Additionally, since the models are not extensively fine-tuned due to a limited parameter range, their performance is constrained. However, the combination approach helps to slightly mitigate these issues by leveraging the strengths of each model, leading to a modest improvement in overall performance. This highlights the collaborative system's ability to provide more balanced and robust results despite the parameter limitations.

B. Result of using Dynamic Parameter Adjustment

TABLE II
Adaptive Classifier Adjustment Result

	Skew	Model	ACC%	PR%	RE%	F1%
Dataset 1	Imbalance	SVM, C=292	3.44	4.20	3.44	3.17
Dataset 2	Balance	KNN, k=8	6.26	8.59	6.26	4.88
Dataset 3	Imbalance	SVM, C=287	3.73	4.18	3.73	3.54
Dataset 4	Balance	KNN, k=10	6.58	8.00	6.58	5.06

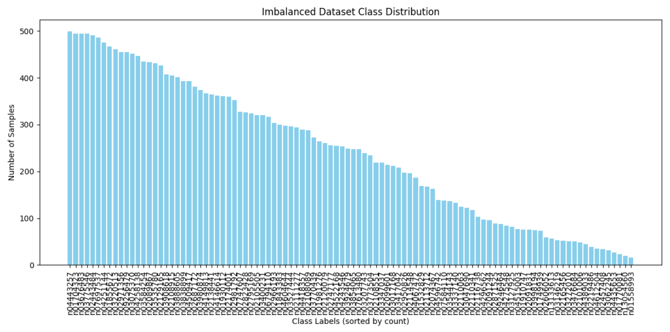

Fig. 1. Imbalance: Dataset 1

The table II shows the performance of the Adaptive Classifier Adjustment system on datasets with different skewness levels. For imbalanced datasets (Dataset 1 and 3), SVM was chosen with optimized C-values, achieving moderate recall but low overall metrics due to the dataset challenges. For balanced datasets (Dataset 2 and 4), k-NN was selected with k-values of 8 and 10, leading to slightly better accuracy and F1-scores. The system effectively adapts classifiers based on skewness, but overall low scores highlight the need for further tuning and preprocessing improvements.

IV. Conclusions

This project tackled two tasks: Collaborative Multi-Classifier System and Adaptive Classifier Adjustment using the Mini-ImageNet dataset.

In Task 1, combining classifiers like k-NN, SVM, and Decision Tree with weighted voting showed slight improvements but was limited by simple parameter tuning and dataset challenges.

Task 2 adapted classifiers based on dataset characteristics. SVM performed better on imbalanced data with tuned C-values, while k-NN worked slightly better on balanced data with adjusted k-values.

Overall, the systems demonstrated flexibility but need further optimization for complex datasets.