Multi-Class AdaBoost with Perceptron

Introduction

In project 3, we implemented the AdaBoost algorithm, specifically focusing on the algorithm Stagewise Additive Modeling Multi-class Exponential loss function (SAMME). The work involved improving the perceptron model from the previous lab to handle multi-class problems, training it in conjunction with AdaBoost, and evaluating its performance. The report will describe the methods used, enhancements made to the perceptron, and results obtained. It will also explain the parameter selection strategy, and summarize core ideas from the provided AdaBoost paper.

Understanding Multi-Class AdaBoost

AdaBoost, short for Adaptive Boosting, is an ensemble learning method that combines multiple weak learners, such as the perceptron, to form a strong classifier. While the original AdaBoost algorithm handles binary classification, the SAMME extension generalizes it to multi-class classification problems.

SAMME adjusts the boosting framework by using a different calculation for the importance of weak learners, called alpha (α_m):

$$\alpha_m = ln(\frac{1-e_m}{e_m}) + ln(K - 1)$$

where e_m is the weighted error of the weak learner, and K is the number of classes. This modification ensures that the ensemble focuses on misclassified samples across multiple classes by updating sample weights based on their classification difficulty.

The final prediction of the SAMME classifier aggregates the predictions of all weak learners, weighted by their respective alphas:

$$F(x) = arg\,max_k \sum_{n=1}^{M} \alpha_m \cdot 1[h_m(x) = k]$$

where M is the number of boosting rounds, hm(x) is the prediction of the M-th weak learner, and k is a class label.

Improving the Perceptron

In the previous lab, the perceptron model was limited to binary classification, with labels $y \in \{-1,1\}$. To adapt it for multi-class classification, we implemented the following improvements:

1. **Multi-class Output**:
    - Instead of a single weight vector, the updated perceptron maintains one weight vector per class.
    - During training, the perceptron computes scores for all classes and predicts the class with the highest score.
2. **Training Rule**:
    - If the predicted class does not match the true class, weights are updated for both the predicted class (penalized) and the correct class (rewarded).
    - This ensures the model learns to distinguish among all available classes.

These changes enabled the perceptron to serve as a weak learner in the SAMME algorithm.

Training the Model

The training process for our AdaBoost implementation involved the following steps:

1. **Dataset**:
    - We used the digits dataset from scikit-learn, consisting of 8x8 grayscale images of digits (0–9), totaling 64 features per sample.
    - The dataset was split into 80% training and 20% testing data.
2. **AdaBoost Implementation**:
    - For each boosting round, a perceptron was trained on the dataset with weights adjusted according to the errors of the previous round.
    - The importance (α) of each weak learner was calculated using the SAMME formula.
    - The final predictions were determined by aggregating predictions from all weak learners, weighted by their respective alphas.
3. **Parameter Testing**:
    - We tested 12 different values for the number of estimators in the AdaBoost model: [1, 2, 4, 5, 10, 12, 13, 14, 15, 16, 18, 50].
    - For the perceptron, we evaluated different learning rates and epochs to identify the best-performing configuration.
    - The best model achieved an accuracy of **96.67%** on the validation set with $\alpha=0.0001$, epochs=10, and boosting rounds=14.

Results and Model Selection

The table below summarizes the results for varying the number of estimators:

Number of Estimators	Accuracy (%)
1	95.83%
2	95.83%
4	95.56%
5	95.28%
10	96.39%
12	96.39%
13	96.39%
14	96.67%
15	96.67%
16	96.11%
18	96.11%
50	96.11%

From these results, we observed that increasing the number of estimators improved the model's accuracy, but only up to 14 estimators. Beyond this, additional boosting rounds might have led to overfitting, which we demonstrated with 16, 18 and 50 estimators.

The best model was selected based on the accuracy of the validation set. The optimal parameters for the perceptron and adaboost algorithm were:

- Learning rate (α) = 0.0001
- Epochs = 10
- Boosting rounds (estimators) = 14

Conclusion

This project gave us insights into multi-class AdaBoost through using the SAMME algorithm. We extended our perceptron model to handle multi-class classification and integrated it as the weak learner in AdaBoost. Through testing different combinations of parameters, we arrived at a model which demonstrated the effectiveness of boosting

and improving performance. Our best model achieved a validation accuracy of 96.67%, proving the power of ensemble methods in multi-class problems.

ChatGPT Usage

- Proofread report
- Discuss coding choices and help with documentation

Contributions

- Herman Østengen:
    - Implemented the updated Perceptron model.
    - Designed the strategy for parameter testing, including combinations of alpha, epochs, and estimators.
    - Facilitated and contributed to the other member's work.
    - Wrote the report.
- Isabel Wolden:
    - Researched and implemented the SAMME algorithm for multi-class AdaBoost.
    - Analyzed the results and contributed to writing the section on parameter tuning in the report.
    - Facilitated and contributed to the other member's work.
    - Wrote the report.