CptS 570 Machine Learning

Homework - 1

Submitted by
Athul Jose P
11867566

School of Electrical Engineering and Computer Science
Washington State University

Contents

List of Figures

1 Analytical Part

1.1 Is the decision boundary of voted perceptron linear?

No. The decision boundary in a voted perceptron model is non-linear because it is determined by the combined output of several individual perceptrons. Each perceptron in the ensemble contributes a linear decision boundary, but when these are combined through voting, the resulting decision boundary is not simply linear. Instead, it's influenced by the aggregation of multiple non-linear sign functions. Since each perceptron may have a different perspective on the data (due to different weights and biases), the final decision boundary reflects the sum of these contributions. This blending of multiple linear boundaries creates a complex, non-linear boundary that better separates the classes in a more intricate decision space.

1.2 Is the decision boundary of averaged perceptron linear?

Yes, the decision boundary of an averaged perceptron is linear. This is because each individual perceptron within the model forms a linear decision boundary based on its learned weights and biases. When the averaged perceptron combines these individual perceptrons, it takes a weighted sum of their predictions. A weighted sum is essentially a linear combination of these individual linear decision boundaries. Since the sum of linear functions is also linear, the final decision boundary produced by the averaged perceptron remains linear. Therefore, regardless of how the individual perceptrons are combined, the overall decision boundary will still divide the feature space using a straight line (or hyperplane in higher dimensions).

2 PA Weight Update

The objective is to minimize the change to the weight vector while ensuring that the margin is corrected. Formally, we solve the following optimization problem:

$$\min_{w} \frac{1}{2}\|w - w_t\|^2$$

subject to the constraint:

$$y_t w^\top x_t \geq M$$

where y_t is the label, x_t is the feature vector and M is the desired margin
We define the Lagrangian as:

$$\mathcal{L}(w, \lambda) = \frac{1}{2}\|w - w_t\|^2 + \lambda \left(M - y_t w^\top x_t\right)$$

where $\lambda \geq 0$ is the Lagrange multiplier.
Taking the derivative of $\mathcal{L}$ with respect to w and setting it to zero gives:

$$\frac{\partial \mathcal{L}}{\partial w} = w - w_t - \lambda y_t x_t = 0$$

Solving for w, we get the update rule:

$$w = w_t + \lambda y_t x_t$$

Substitute the update rule into the margin constraint $y_t w^\top x_t \geq M$:

$$y_t \left(w_t^\top x_t + \lambda y_t x_t^\top x_t \right) \geq M$$

Simplifying:

$$y_t w_t^\top x_t + \lambda \|x_t\|^2 \geq M$$

Solving for λ:

$$\lambda = \frac{M - y_t w_t^\top x_t}{\|x_t\|^2}$$

The weight update for achieving margin M in the PA algorithm is:

$$w_{t+1} = w_t + \tau_t y_t x_t$$

where

$$\tau_t = \frac{M - y_t w_t^\top x_t}{\|x_t\|^2}$$

3

3.1 (a) Modified Perceptron Algorithm

To leverage the importance weights $h_i > 0$, we modify the update rule as follows:

Prediction:

$$\hat{y_i} = \text{sign}(w^\top x_i)$$

Update Rule (when misclassified, i.e., $y_i w^\top x_i \leq 0$):

$$w \leftarrow w + h_i y_i x_i$$

This modification scales the update by the importance weight h_i, giving more influence to examples with larger h_i and smaller influence to those with lower h_i.

3.2 (b) Reduction to Standard Perceptron

We can solve this learning problem using the standard perceptron algorithm by applying a reduction-based approach that adjusts the dataset to reflect the importance of each example.

The key idea is to reduce the problem to the standard perceptron algorithm by duplicating examples according to their importance weights h_i. For each example (x_i, y_i), we create $\lfloor h_i \rfloor$ copies, and if h_i has a fractional part, we add one more copy with a probability equal to that fractional value. This creates an expanded dataset where important examples appear more frequently, ensuring they have a larger influence on the model.

Once the dataset is expanded, we apply the standard perceptron algorithm as usual. The perceptron will treat each example equally but, because the important examples are duplicated, they will influence the decision boundary more strongly. This reduction ensures that the perceptron adapts its weights based on the importance of each example, without requiring any modifications to the core perceptron algorithm itself.

This approach effectively solves the learning problem by leveraging the standard perceptron algorithm on an expanded dataset that reflects the varying importance of the examples.

4

4.1 (a) Handling Imbalanced Data

1. Resampling: One approach to deal with imbalanced data is to resample the dataset, either by upsampling the minority class or downsampling the majority class, to balance the number of examples for each class.
2. Penalizing the Learning Rate: Alternatively, we can penalize the learning rate based on the class proportions. This ensures that less frequent classes have a larger impact on weight updates. The penalizing factor m_a for class a is defined as:

$$m_a = \frac{N}{N_a}$$

where N is the total number of training examples, and N_a is the number of examples in class a. This gives more importance to the minority class by increasing the learning rate for that class.

4.2 (b) Modified Perceptron Algorithm with Penalizing Factors

Let x_i and y_i represent the input and label for example i. Use m_a and m_b as penalizing factors for class a (where $y = 1$) and class b (where $y = -1$), respectively. The perceptron update rule is modified as follows:

No Mistake: If $y_i(w_i \cdot x_i) > 0$, no update is performed:

$$w_{i+1} \leftarrow w_i$$

Mistake: If $y_i(w_i \cdot x_i) \leq 0$, update the weights using the penalizing factor:

$$w_{i+1} \leftarrow w_i + m_{y_i} y_i x_i$$

Here, m_{y_i} is the penalizing factor corresponding to the class label y_i, ensuring that the weight update is scaled by the class proportion.

5 Programming and Empirical Analysis Part

5.1 Binary Classification

5.1.1 a

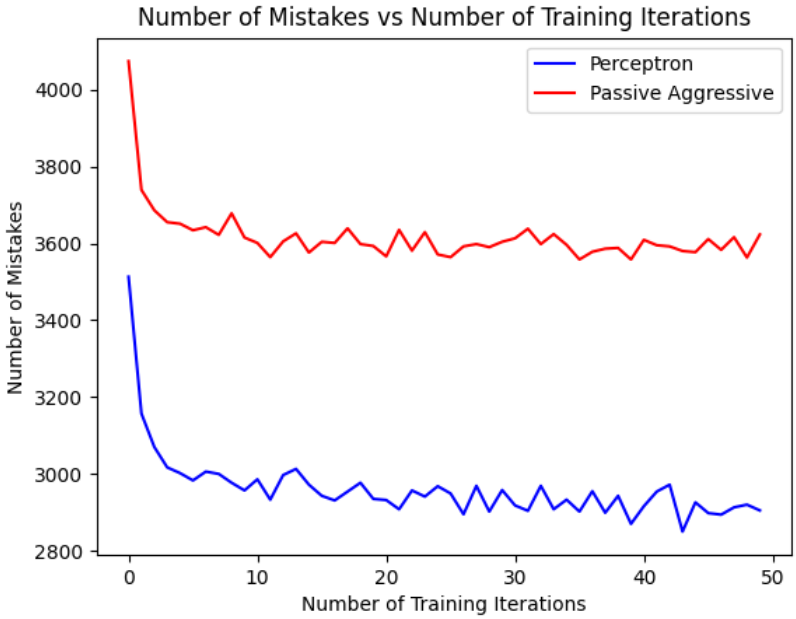

Figure 1: Mistakes vs Number of training iterations

Figure 1 illustrates the number of mistakes made by the Perceptron and Passive-Aggressive algorithms over 50 training iterations, where the Perceptron consistently outperforms the Passive-Aggressive algorithm from the beginning of training until the end. The Perceptron starts with around 3200 mistakes and steadily reduces them to about 2900, whereas the Passive-Aggressive algorithm begins with approximately 4000 mistakes and reduces to around 3500. This difference in performance can be attributed to the learning rates used by each algorithm: the Perceptron employs a fixed learning rate of 1, while the Passive-Aggressive algorithm adjusts its learning rate for each correction. Although both algorithms show a decrease in the number of mistakes as training progresses, the Perceptron converges faster and maintains a lower error rate throughout the training process. Furthermore, the graph indicates that increasing the number of training iterations beyond a certain point does not significantly improve the performance of either algorithm, highlighting the diminishing returns in terms of reducing mistakes as training iterations continue.

5.1.2 b

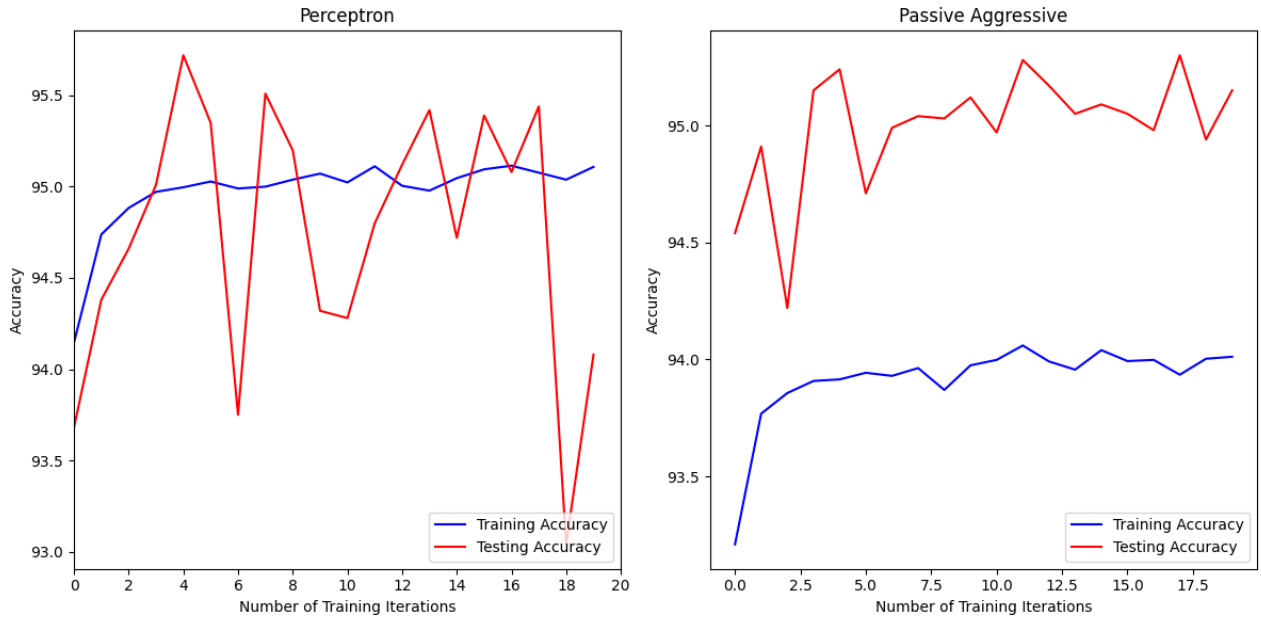

Figure 2: Comparison between perceptron and passive aggressive algorithm

Figure 2 display the accuracy curves for the perceptron and passive-aggressive algorithms, on both the training and test datasets. The perceptron shows higher accuracy than the passive-aggressive algorithm on the training data. However, the perceptron also exhibits a larger variance between training and test accuracy, indicating a tendency to overfit the data. Interestingly, the passive-aggressive algorithm performs better on the training data than on the test data, which could be due to calculating the number of mistakes after each weight update for the training data, while the accuracy for the test data is computed only after completing all iterations. These curves highlight that training accuracy alone is not sufficient to evaluate the performance of machine learning algorithms, as high accuracy on training data may not translate to good performance on test data. For a learning algorithm to be considered effective, its performance should be consistent across both training and test data.

5.1.3 c

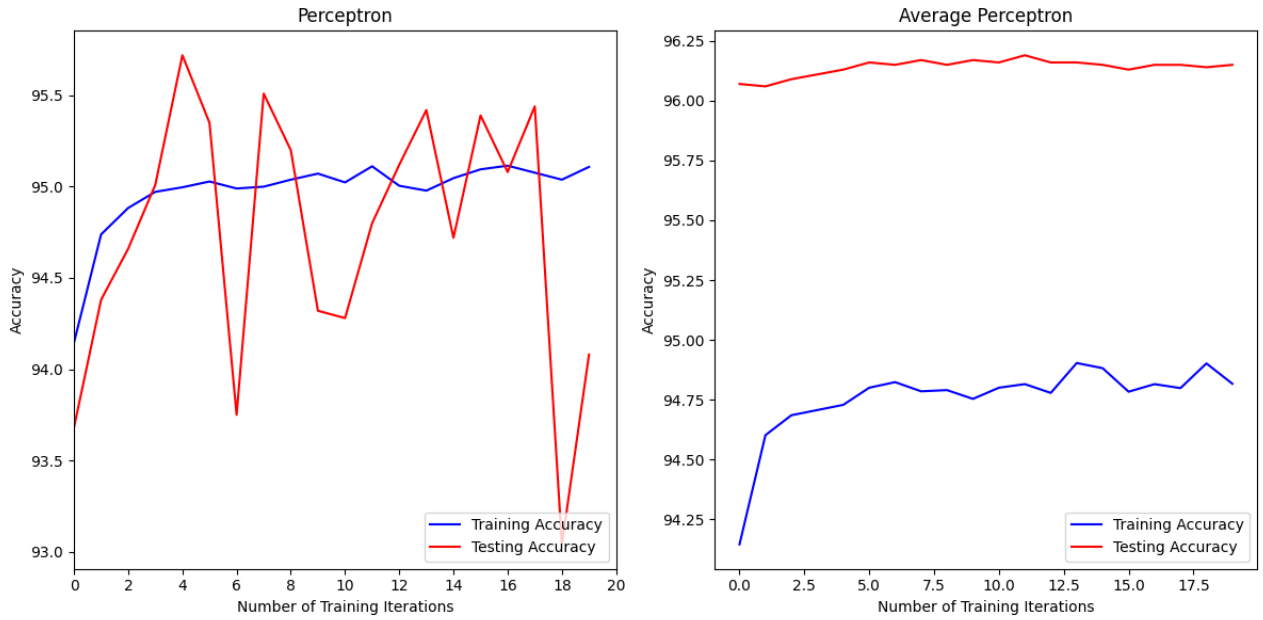

Figure 3: Comparison between perceptron and averaged perceptron

The figure 3 presents two graphs comparing the performance of the Perceptron algorithm (left) and the Averaged Perceptron algorithm (right) over multiple training iterations, showing both training and testing accuracy. In the left graph, the training accuracy of the perceptron gradually increases and stabilizes around 95%, but the testing accuracy fluctuates significantly between 93.5% and 95.5%, suggesting potential overfitting. This instability in testing accuracy implies that the perceptron does not generalize well to unseen data, as its performance on the test set varies considerably during different iterations. In contrast, the right graph demonstrates that the averaged perceptron algorithm provides a smoother increase in training accuracy, which also stabilizes around 95%, while the testing accuracy remains consistently higher and more stable around 96%. This indicates that the averaged perceptron handles both training and testing data more effectively, with less variance and better generalization. Overall, the averaged perceptron appears to outperform the standard perceptron by maintaining more consistent performance across both datasets, reducing the risk of overfitting.

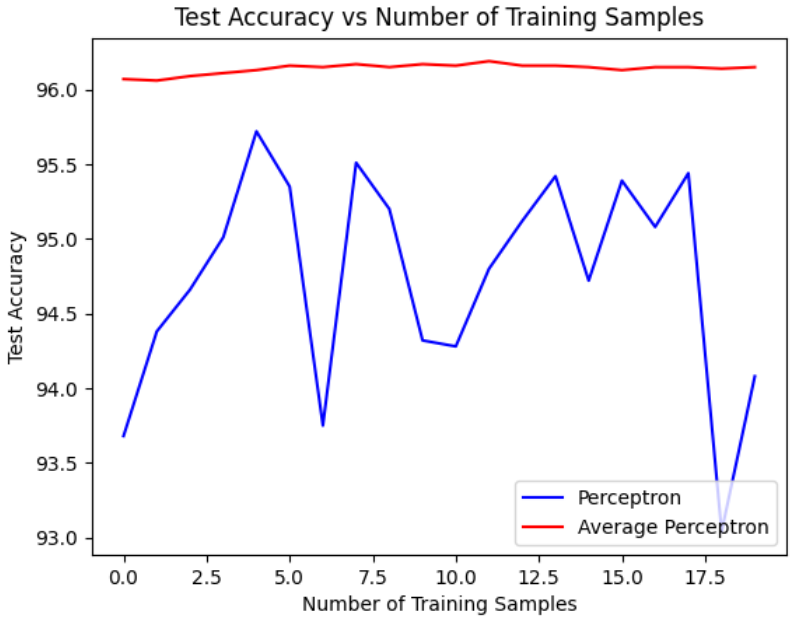

Figure 4: Testing accuracy of perceptron and averaged perceptron

The figure 4 illustrates the test accuracy of the Perceptron and Averaged Perceptron algorithms as the number of training samples increases. The blue line represents the Perceptron algorithm, while the red line shows the performance of the Averaged Perceptron. The Perceptron's accuracy fluctuates significantly throughout the training process, with noticeable peaks and valleys, ranging between 93.5% and 95.5%. These sharp variations indicate that the Perceptron algorithm struggles to maintain stable accuracy on the test data, suggesting sensitivity to training data changes and a potential lack of generalization. In contrast, the Averaged Perceptron demonstrates a much smoother and more consistent performance, with its test accuracy staying steadily around 96% throughout the training process. This consistency highlights that the Averaged Perceptron is better at generalizing to the test data, making it a more reliable algorithm compared to the Perceptron, which suffers from variability in its accuracy. The clear distinction in stability and performance between the two algorithms emphasizes the advantage of averaging in improving model robustness and reducing overfitting.

5.1.4 d

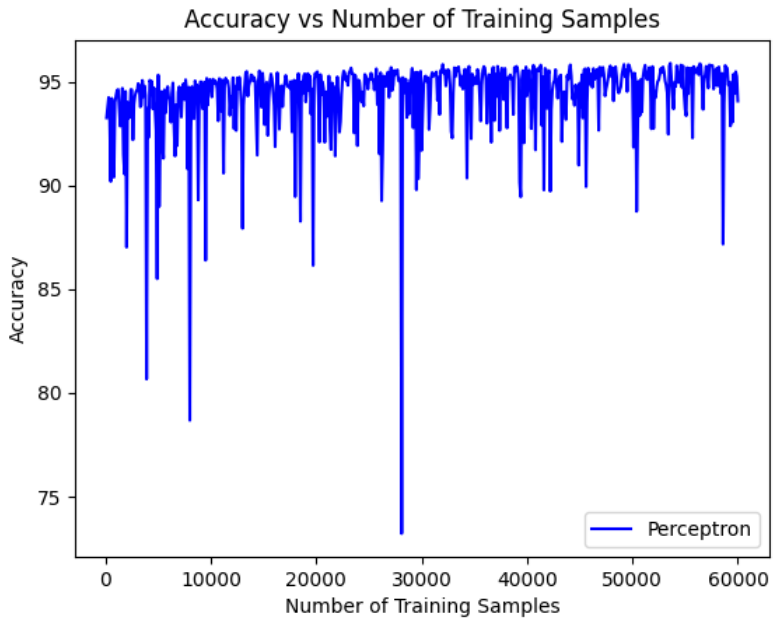

Figure 5: Generalized curves for perceptron

The graph in Figure 5 shows the accuracy of the Perceptron algorithm as a function of the number of training samples. As the number of samples increases, the accuracy generally remains high, fluctuating around 95%. However, there are noticeable dips in the accuracy, with some extreme outliers where the accuracy drops sharply to values as low as 75%. These sudden drops suggest that the Perceptron algorithm experiences instability during training, potentially due to certain challenging data points or a lack of robustness in handling larger datasets. Despite these occasional dips, the overall accuracy trend remains consistent, indicating that the algorithm performs reasonably well across most training samples but struggles with specific cases. This variability could point to overfitting in certain areas, leading to less consistent performance across different parts of the dataset.

5.2 Multi-Class Classification

5.2.1 a

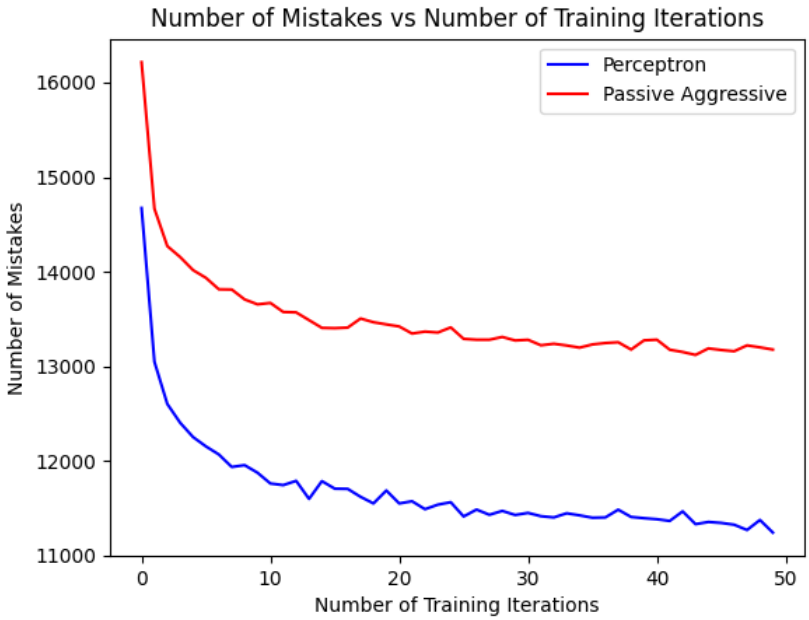

Figure 6: Mistakes vs Number of training iterations

Figure 6 illustrates the number of mistakes made by the Perceptron and Passive-Aggressive algorithms over 50 training iterations. The blue line represents the Perceptron algorithm, and the red line represents the Passive-Aggressive algorithm. Both algorithms show a significant decrease in the number of mistakes as training iterations increase, but the Perceptron consistently makes fewer mistakes compared to the Passive-Aggressive algorithm. Initially, both algorithms start with a high number of mistakes, with the Passive-Aggressive algorithm starting at around 16,000 and the Perceptron slightly lower. However, as training progresses, the Perceptron's mistake count decreases more rapidly, stabilizing around 11,000 mistakes, while the Passive-Aggressive algorithm shows a slower decline, stabilizing around 13,000 mistakes. This indicates that the Perceptron learns more effectively during the training process, making fewer mistakes and converging faster than the Passive-Aggressive algorithm.

5.2.2 b

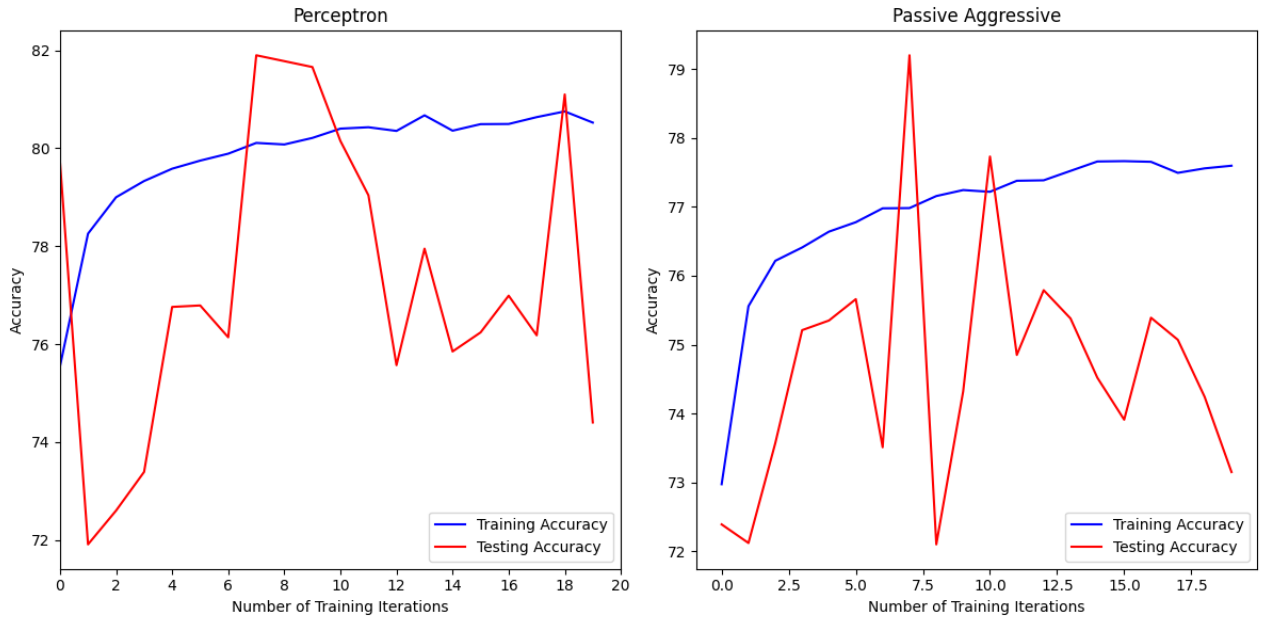

Figure 7: Comparison between perceptron and passive aggressive algorithm

Figure 7 compares the training and testing accuracy of the Perceptron (left) and Passive-Aggressive (right) algorithms over multiple training iterations. In both graphs, the blue line represents training accuracy, and the red line represents testing accuracy.

For the Perceptron algorithm (left), the training accuracy improves steadily, stabilizing around 80%. However, the testing accuracy fluctuates significantly, ranging between 72% and 80%, showing high variance and inconsistency across different iterations. This indicates that while the Perceptron performs well on the training data, it struggles to generalize to the test data, likely due to overfitting.

In the case of the Passive-Aggressive algorithm (right), the training accuracy also improves gradually, stabilizing around 77%. The testing accuracy, however, exhibits even more pronounced fluctuations than the Perceptron, with a sharp spike around the 7th iteration followed by steep drops. The high variance in the Passive-Aggressive algorithm's testing accuracy suggests that it has difficulty maintaining stable performance on unseen data, despite reasonable training accuracy.

Overall, both algorithms show good performance on the training data, but their testing accuracy curves reveal inconsistent generalization, with the Passive-Aggressive algorithm experiencing more pronounced instability compared to the Perceptron. This suggests that while both models can learn from the training set, they are prone to overfitting and may not perform reliably on new data.

5.2.3 c

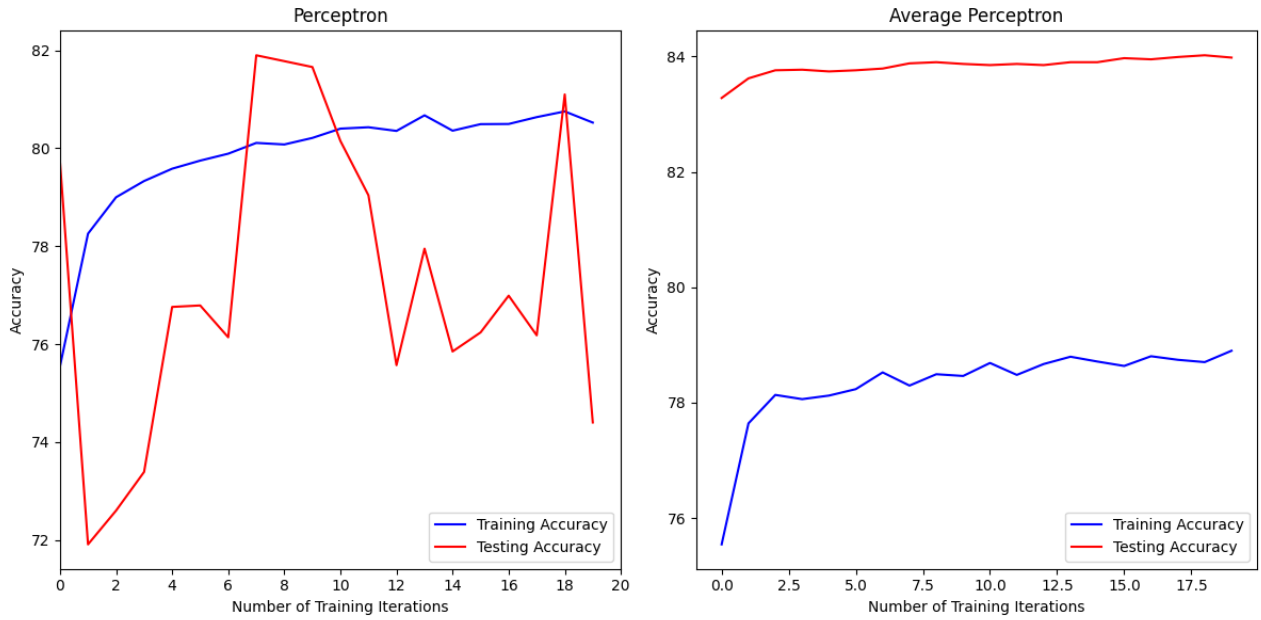

Figure 8: Comparison between perceptron and averaged perceptron

Figure 8 presents a comparison between the Perceptron (left) and Averaged Perceptron (right) algorithms, showing their training and testing accuracy over multiple training iterations. In both graphs, the blue line represents the training accuracy, while the red line represents the testing accuracy.

For the Perceptron (left), the training accuracy increases steadily, stabilizing around 80% after several iterations. However, the testing accuracy is highly unstable, fluctuating between 72% and 82%. These sharp fluctuations indicate that the Perceptron algorithm struggles with overfitting, as it performs well on the training data but inconsistently on the test data.

On the other hand, the Averaged Perceptron (right) demonstrates a much more stable performance. The training accuracy gradually increases and stabilizes around 79%, while the testing accuracy remains consistently high, around 84%, with only minimal fluctuations. This consistent performance shows that the Averaged Perceptron is better at generalizing to unseen data, as evidenced by the small gap between training and testing accuracy.

In conclusion, the Averaged Perceptron outperforms the Perceptron in terms of stability and generalization, making it a more reliable model for both training and testing datasets.

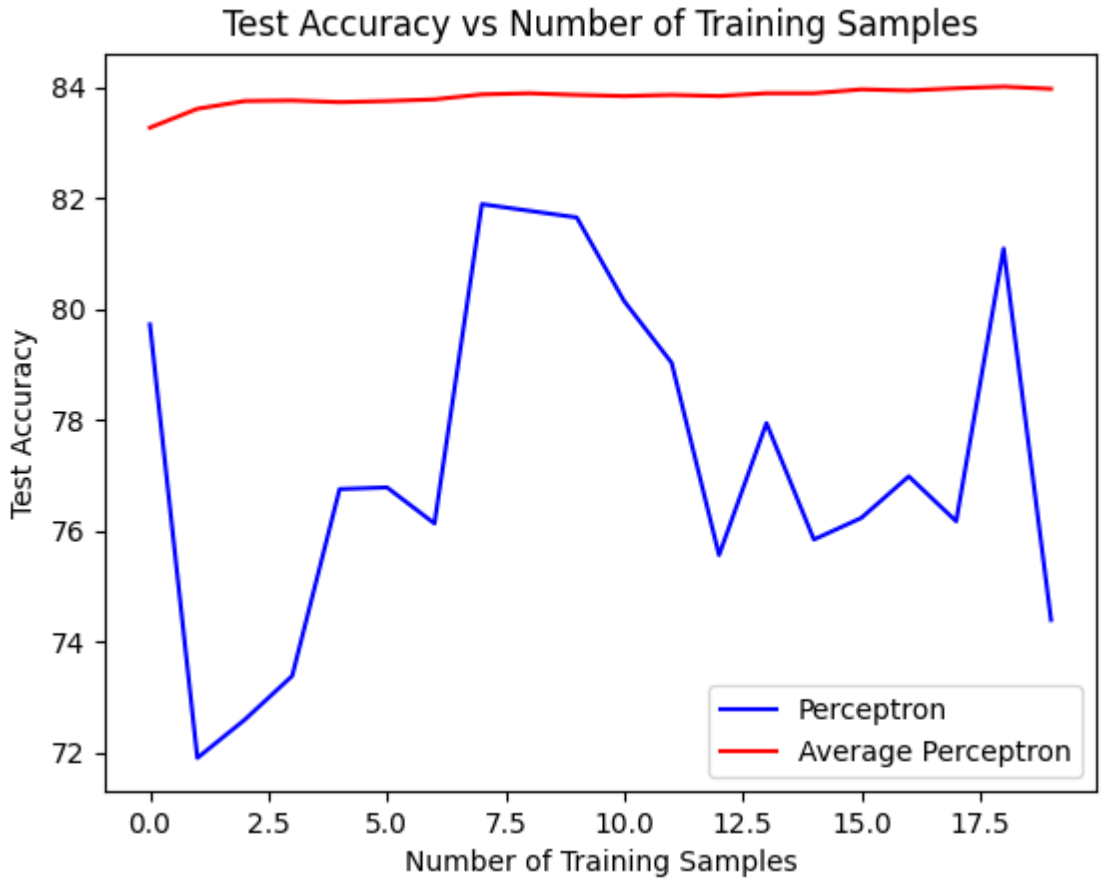

Figure 9: Testing accuracy of perceptron and averaged perceptron

Figure 9 compares the test accuracy of the Perceptron and Averaged Perceptron algorithms as the number of training samples increases. The blue line represents the Perceptron algorithm, while the red line represents the Averaged Perceptron algorithm.

The Perceptron shows significant fluctuations in test accuracy, ranging from 72% to around 82%, with no clear stabilization even as the number of training samples increases. This variability suggests that the Perceptron struggles to generalize well, showing inconsistent performance on the test data across different training samples.

In contrast, the Averaged Perceptron demonstrates much more stable performance, with its test accuracy consistently hovering around 84%. The lack of significant fluctuations in the Averaged Perceptron's accuracy curve indicates that it generalizes more effectively and maintains reliable performance across varying training sample sizes.

Overall, the figure shows that while the Perceptron experiences instability and inconsistency in its test accuracy, the Averaged Perceptron performs more consistently and robustly, making it a more reliable choice for generalization to unseen data.

5.2.4 d

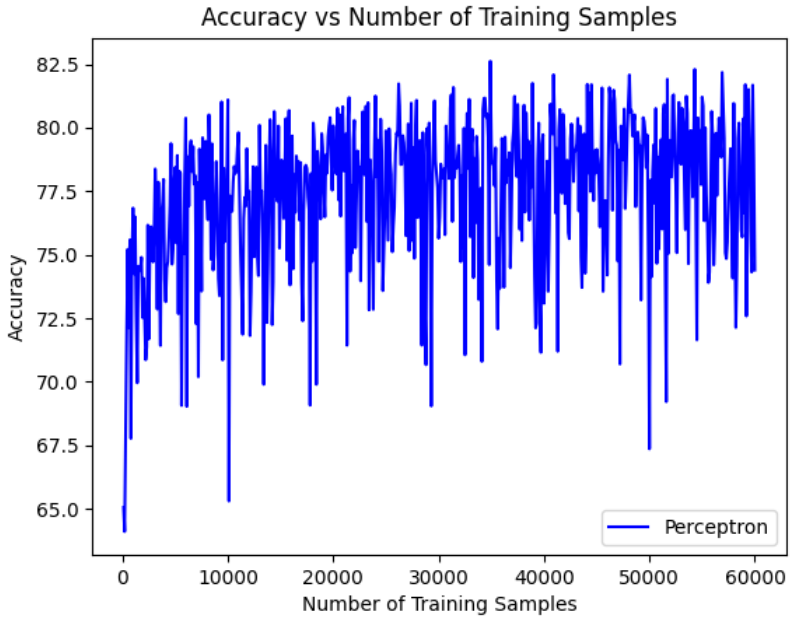

Figure 10: Generalized curves for multi-class perceptron

Figure 10 illustrates the accuracy of the Perceptron algorithm as the number of training samples increases. The accuracy curve exhibits significant fluctuations, ranging from approximately 65% to just over 82%. As more training samples are added, the accuracy generally trends upward, but the large number of sharp spikes and drops indicates high variability in the model's performance. This suggests that the Perceptron algorithm struggles with maintaining consistent accuracy across the training process, particularly as the dataset grows larger.

While there is a noticeable improvement in overall accuracy compared to the initial samples, the persistent oscillations reflect the Perceptron's sensitivity to certain training data and its potential difficulty in generalizing effectively. This figure highlights the instability of the Perceptron when applied to larger datasets, as it fails to produce a consistently reliable accuracy despite the increasing number of training samples.