**MACHINE LEARNING FROM DATA**

**Report: Lab Session 7 – Neural Networks**

**Names: Abdullah Qureshi, S M Rakib Hasan and Niina Hietamäki**

**Instructions**

Getting the material:

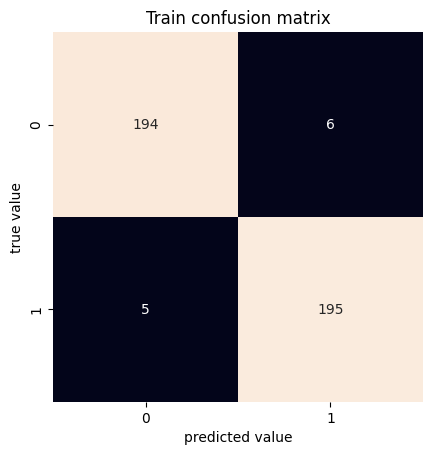
* Download and uncompress the file Mlearn\_Lab7\_soft.zip
* Answer the questions in the document Mlearn\_Lab7\_report\_surname.pdf

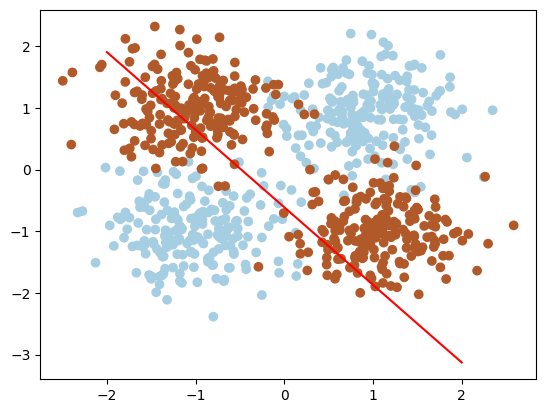
**Questions**

Q1. Which are the default parameters used by the Perceptron class? (check scikit-learn documentation). Compare the performance of the Perceptron on the two toy examples (linearly and non-linearly separable). Compare the performance of the Perceptron when using the original 2D features and features augmented by interaction

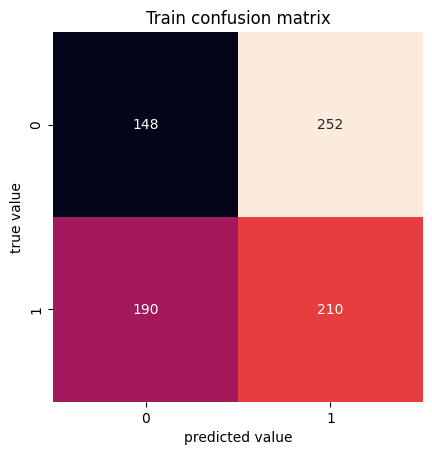
We train the Perceptron model using the default parameters provided by Scikit-learn: max\_iter=1000 (maximum iterations), tol=1e-3 (tolerance for stopping condition), eta0=1.0 (constant learning rate), shuffle=True (shuffling data at each epoch), fit\_intercept=True (fitting an intercept term).

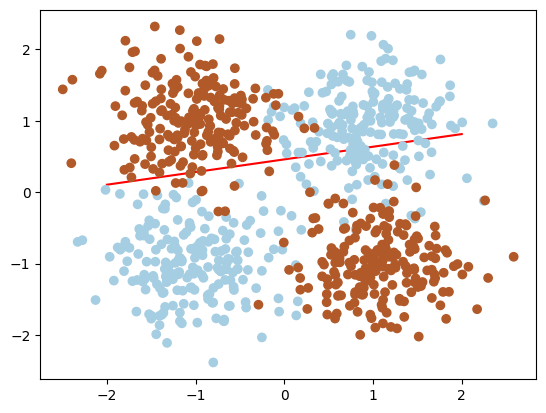
These parameters ensure the model uses a fixed learning rate of 1.0 and stops training when the improvement becomes smaller than 0.001.

Linearly: 



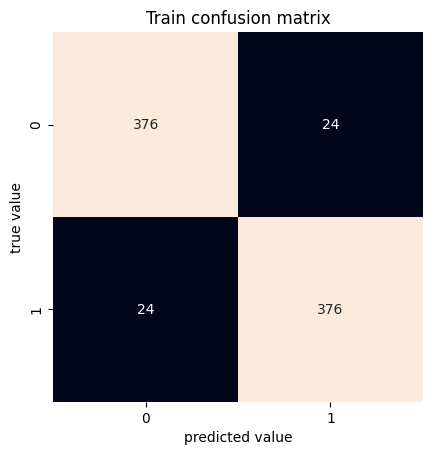
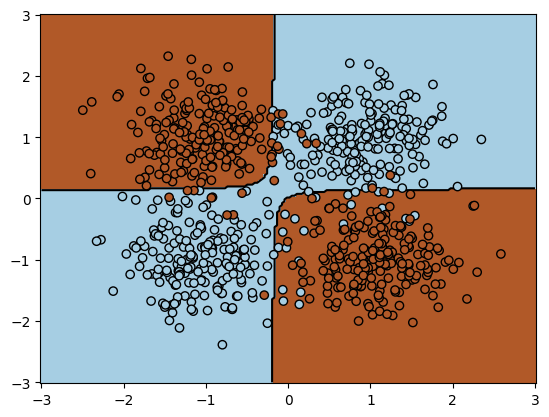
Linearly: The confusion matrix and the decision boundary indicate that the model performs well, correctly predicting most of the classes. The confusion matrix shows that most predictions are accurate, with only a few misclassifications, suggesting that the Perceptron is effective in separating linear classes.

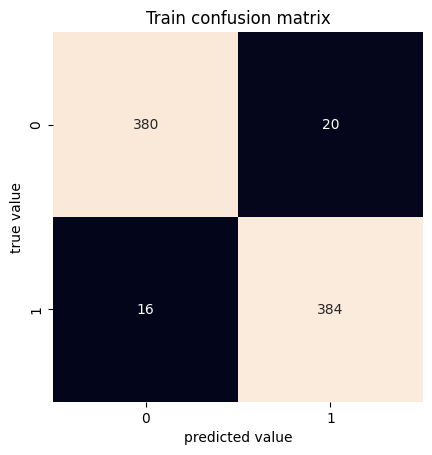
Non-linearly: 

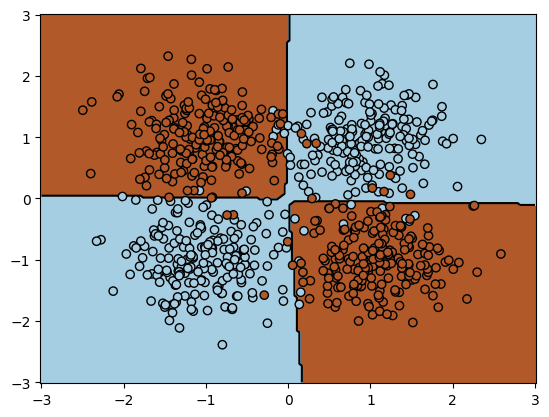


Non-linearly: In this case, the Perceptron does not perform as well. The confusion matrix reveals a higher number of errors, showing that the Perceptron cannot effectively separate classes that are not linearly separable. The image of the decision boundary shows that the classes overlap more, and the model's ability to distinguish them is weaker.

Q2. Compare the performance of the Perceptron with 3D features (with interaction) and the Multi-Layer perceptron with 2D features (you can try to improve the performance by varying some hyperparameters).

3D features with Interaction 

Multi-Layer perceptron with 2D features



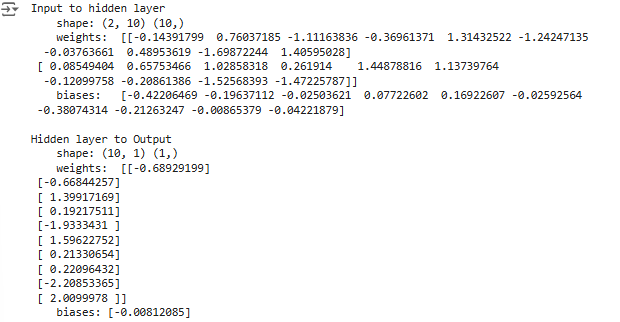
**Perceptron with interaction features:**

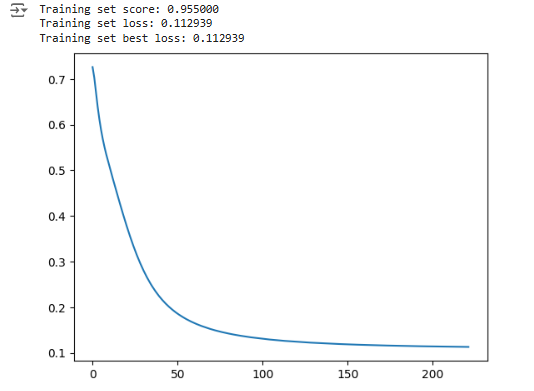
The Perceptron improved its performance when we added an interaction term (3D features). The confusion matrix shows a very small error of only 6%.

**MLP with 2D features:**The MLP performed better than the Perceptron using only 2D features. This is because the MLP can learn complex, non-linear decision boundaries.

The MLP had higher accuracy compared to the Perceptron, even though the Perceptron improved with interaction features. This shows that more advanced models like MLP are better for non-linear problems.

We calculated the accuracies and observed that the MLP achieved a higher accuracy of 97%, while the Perceptron with 3D features achieved 94%. This confirms that MLP is better for non-linear problems





Q3: For the MNIST task, copy the global accuracy and the confusion matrix for the training and test set and analyze the results.

train accuracy: 0.8184

train error: 0.18159999999999998

test accuracy: 0.7728

test error: 0.22719999999999996

|  |  |
| --- | --- |

We used a MLP classifier with a single hidden layer (10 neurons) using SGD and no momentum. We achieve an average performance on the MNIST dataset with a training accuracy of 81.8% and test accuracy of 77.3%. This shows there might be some overfitting but not a lot. The classifier performs well on some digits like 0, 1, and 6 but fails to perform with some digit 5 (low recall).

By analyzing the confusion matrix for both training and testing data we get some info on the classifier's performance. The model performs well on digits 0, 1, and 6, with minimal misclassifications. Other digits like 5 and 9 show significant confusion, particularly with neighboring digits (e.g., 5 misclassified as 3 or 8). This shows a lack of distinct features for learning these digits in the training phase. In the test data digits 0, 1, and 6 maintain high accuracy, while digits 5, 8 and 9 exhibit higher misclassification rates. Again we see the same problem of the digit 5 often misclassified as 3 or 8

Q4. For the MNIST task, analyze and compare the methods based on the loss curves.

In this section, we train a Multilayer Perceptron (MLP) with the following configurations:

1. SGD, constant

2. SGD with momentum, 'constant'

3. SGD with nesterov, 'constant'

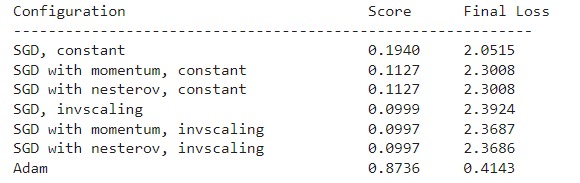
4. SGD, 'invscaling'

5. SGD with momentum, `invscaling`

6. SGD with nesterov, `invscaling`

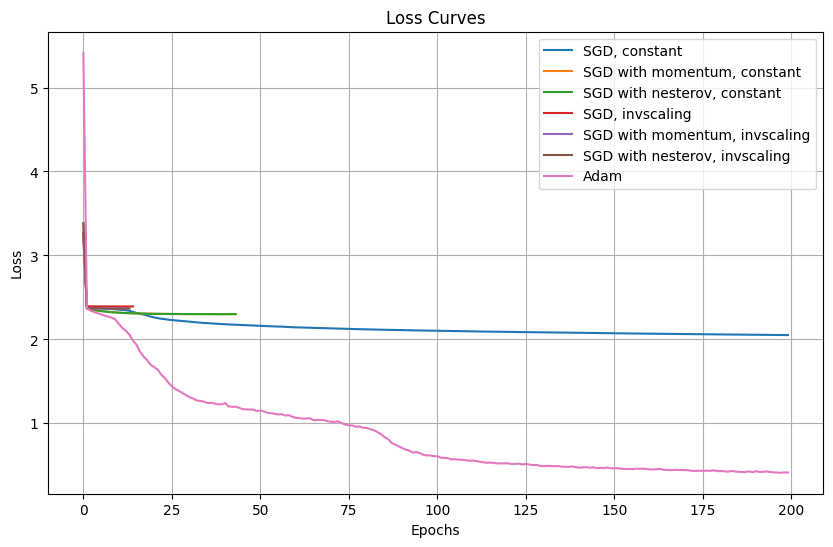
7. Adam

We evaluate the models based on their training set scores and plot all loss curves.



The **Adam optimizer** clearly outperforms all SGD-based methods and achieves the lowest final loss and smooth convergence.

1. **Adam** Its loss decreases sharply at the start and continues to decline smoothly over the epochs. Achieves the lowest **final loss** (~0.24) compared to other methods.
2. **SGD, constant**: The loss curve flats out early and shows minimal improvement after the initial epochs.
3. **SGD with momentum, constant**: Loss behaves similarly to basic SGD but with higher instability in the initial epochs.
4. **SGD with Nesterov, constant**: Very similar to standard momentum-based SGD. Final loss and performance remain poor
5. **SGD, invscaling**: Learning rate decays too quickly. Loss stabilizes early but remains high (~2.39). Training accuracy is close to **10%**
6. **SGD with momentum, invscaling**: High loss and no significant learning.
7. **SGD with Nesterov, invscaling**: Same as momentum showing high losses and poor convergence



Q5. Analyze the results provided by grid\_search.cv\_results\_

Mention if you find significant differences in performances for some of the hyperparameters.

Answer:

Performance Across Hyperparameters:

The mean\_test\_score ranges from 0.952 to 0.956, showing slight variations across hyperparameter combinations.

param\_clf\_\_learning\_rate and param\_clf\_\_learning\_rate\_init seem to influence results marginally, as seen with differences in mean\_test\_score when switching between "constant" and "adaptive" learning rates.

For example:

(100,) with constant learning rate and learning\_rate\_init=0.001 achieved 0.95467.

(100,) with adaptive learning rate and learning\_rate\_init=0.010 performed slightly better at 0.9552.

Effect of hidden\_layer\_sizes:

Larger hidden layers like (100, 100) appear in the dataset, but further examination is required to identify their impact relative to single-layer structures like (100,).

Hyperparameter Variability:

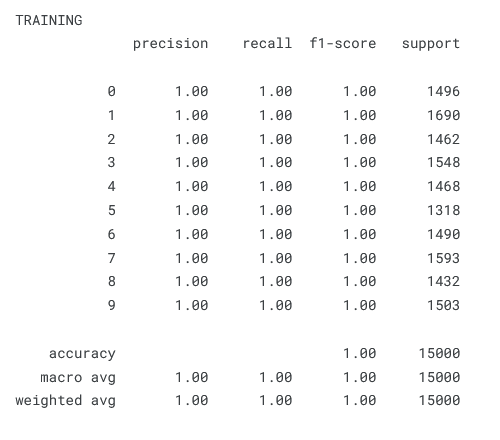
The rank\_test\_score column reflects slight changes in ranking due to minimal score differences.

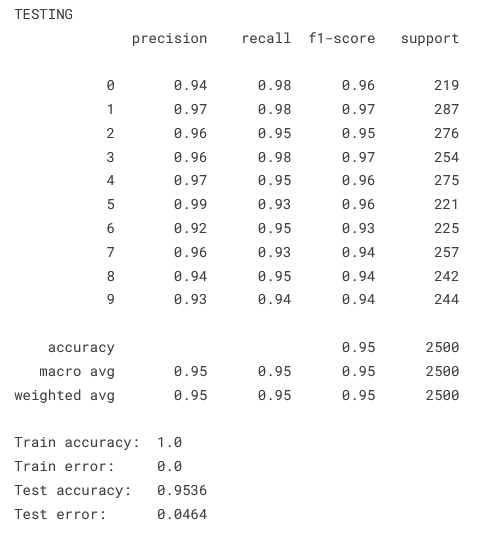
std\_test\_score is 0.0 for all combinations, implying consistent performance across splits.

Q6. Classify the training and test sets with the best hyperparameters. Compute the classification reports, accuracy, error and confusion matrices for the training and test sets. Discuss the results.

Answer: Best hyperparameters-

{'clf\_\_activation': 'relu', 'clf\_\_alpha': 0.0001, 'clf\_\_batch\_size': 16, 'clf\_\_hidden\_layer\_sizes': (100,), 'clf\_\_learning\_rate': 'adaptive', 'clf\_\_learning\_rate\_init': 0.001}

Classification report-

  
Accuracy- Train accuracy- 100%, Test Accuracy 95.36%

Error-Train error- 00%, Test error 04.64%  
Confusion matrix-

