CS557 - Project One

Andrew Struthers, Austin Laverdure, Bridget Smith

April 2023

What to do

1. Use a 3×3 binary matrix representation of the characters.
2. Create a train set (vectors representing versions of characters L and I) and train your classifier. Start with two input vectors, one for L and one for I. Then experiment by adding to the train set more vectors representing graphical variations of the two characters. Compute each time the classification accuracy for the used train set:

$$\text{accuracy} = ((\#\text{ correctly predicted characters})/(\#\text{ train characters})) \times 100\%$$

3. Your neural network should have one neuron.
4. Create a test set using vectors not used for training and test your classifier.
5. Experiment with different learning rates to observe how it influences the learning phase.
6. Repeat Steps 1-5 using a 5×5 binary representation.

Hint: Do not forget that perceptron learning requires fixing of one component of the input vector, usually at the -1 level. If you plan to use your own implementation, the following link may be useful: Useful Python Link

Questions

1. Does the training phase always converge? If not, explain why. **Hint:** According to the Perceptron Convergence Theorem, convergence means achieving 100% classification accuracy for the train set.

   - The training phase does always converge. This means that the two different types of inputs, L and I, are represented by linearly separable vectors. This also implies, because of the Perceptron Convergence Theorem, that the classification of the training set will always reach 100% accuracy. This can be seen in the output of our code.

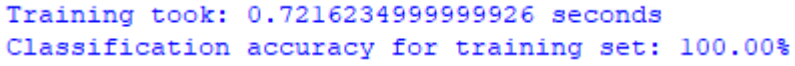

Figure 1: Convergence and accuracy of training

   This means that our training data (consisting of 9 L vectors and 10 I vectors) is able to be linearly separated and then classified with 100% accuracy.

2. When you increase the resolution of the matrix representation, will this improve the classification accuracy? Explain why.

   - Increasing the resolution of the matrix should increase the classification accuracy. With a higher resolution, the training set can contain more variation, more noisy data, and more cases. This will improve the calculated weights by requiring them to handle more situations. One problem that could arise with more data and higher variation in the data is we could generate so much data that not all of it is linearly separable, which would result in the training phase to not converge. The training data needs to be clean and representative of the possible inputs, so while higher resolution can certainly lead to higher accuracy, the quantity and quality of training data must be high. When we experimented with a 5×5 representation, our classification accuracy actually went down. This is most likely due to us not including enough training examples.

3. Does the learning rate influence the learning phase? How?

   - The learning rate certainly impacts the training phase. We usually consider learning rates between $[0.001, 1]$. Larger training rates (i.e. closer to 1) result in much larger changes in the weights from iteration to iteration. This can be a good thing if we are trying to test data without wanting a very long execution time, or if we have very linearly separable data. If the data is very separable, a higher training rate will get a sufficient training weight vector faster. But, in opposition to this, smaller training rates will help to generate much more precise weights in the cases where separation isn't as easy. Smaller learning rates will also help to reduce the likelihood that the training set gets caught in a "local max" instead of a "global max" when determining the appropriate weights.

1 Implementation and Results

1. Numerical results of our experiment and conclusion

   - In the code that produced the below Figure, we had 9 3×3 training vectors for L, and 10 training vectors for I. The training converged and gave us a 100% accuracy with the training set. We then tested this classification on 6 different 3×3 test vectors, 3 noisy L and 3 noisy I examples. The classification was able to hit an 83.3333% accuracy. The Figure below shows which test vectors were classified as what input, and the calculations outlined in step 2 of the "What To Do" section.

```
Correctly Classified:
1 0 0
1 0 0
1 1 0
as: L
=========================
Incorrectly Classified:
1 0 0
1 0 1
1 0 0
as: L
=========================
Correctly Classified:
1 0 1
1 0 0
1 1 0
as: L
=========================
Correctly Classified:
1 0 1
0 0 1
0 0 1
as: I
=========================
Correctly Classified:
1 0 0
1 0 0
0 0 1
as: I
=========================
Correctly Classified:
1 1 0
0 1 0
0 1 1
as: L
=========================
Classification accuracy for test set: 83.33%
```

Figure 2: Training and classification of 3×3 vectors

We also ran the same code with some 5×5 training and test vectors. In the code below, we only had 3 5×5 training vectors for L and 3 for I. We could have added far more training vectors to get a better classification, but with the 6 training vectors we gave the Perceptron model, we got 100% convergence. The classification did much worse than thee 3×3 version, which is contrary to our answer for question 2 above. This is most likely due to our lack of many more training vectors that would have allowed for a better classification output.

```
Incorrectly Classified:
0 0 1 0 0
0 0 1 0 0
0 0 1 0 0
0 0 1 1 1
0 0 0 0 0
as: I
=========================
Incorrectly Classified:
0 1 0 0 1
0 1 0 0 0
0 1 0 1 0
0 1 0 0 0
0 1 1 1 1
as: I
=========================
Incorrectly Classified:
1 0 0 1 0
0 0 0 1 0
0 1 0 1 0
0 0 0 1 0
0 0 0 1 0
as: L
=========================
Correctly Classified:
0 1 0 0 0
0 1 0 0 0
0 0 0 0 0
0 1 0 0 0
0 1 0 0 0
as: I
=========================
Classification accuracy for test set: 25.00%
```

Figure 3: Training and classification of 5×5 vectors

2. Describe which implementation you used

   - We used Python and the Numpy package to implement this code. We made functions for the training and classification of the data, as well as a few other helper functions that helped with data input and visualization. Our implementation came from a programmatic solution to the first homework assignment, and did not rely on information from the book. We wanted to challenge ourselves to implement this learning model ourselves to solidify the basics before moving on to more complicated models in future labs.

3. Code:

   - We uploaded the code we created for this project along with the Canvas submission for this PDF.