EXERCÍCIO 1

Implementar e treinar o modelo Adaline para reconhecer os símbolos A e A invertida

0.1 Dados

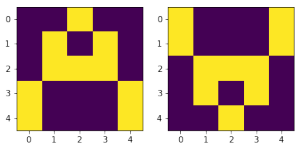

Figure 1: Patterns

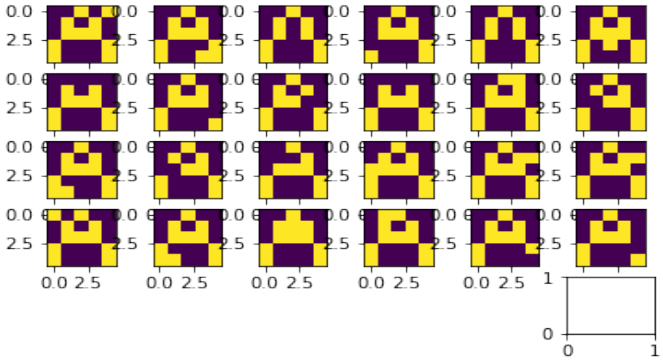

Figure 2: Samples , Normal A's

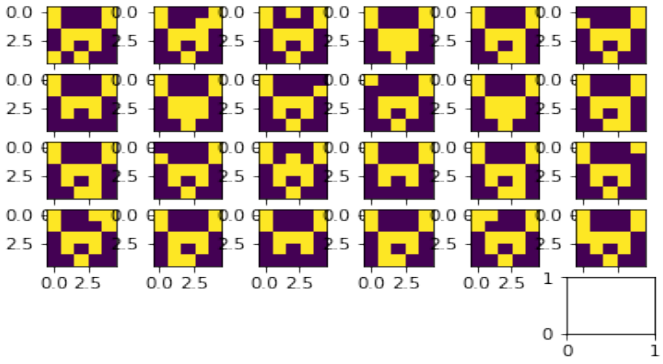

Figure 3: Samples , Inverted A's

All this data was created using **THIS** juypter notebook (Python Conda Kernel)

0.2 ADALINE Perceptron

0.2.1 Note

THIS juypter notebook (Python Conda Kernel) is the script use for training , test and construction of the perceptron.

0.2.2 Considerations

- Being the data binary, but written in a no common manner (using -1 and 1) could be transformed to in a regular representation (0 and 1), but by specifications of this exercise, every operation has to be operated using -1 and 1.
- When weights are generated, is used a random function; this random function creates values between 0 and 1, its possible to extend this range multiplying for some scalar, but, knowing the data set is linearly separable and also posses a binary data format, the right choice is to keep the 0 to 1 range

0.2.3 ADALINE

The ADALINE learning method uses the delta rule (this is not other than the gradient operator) as weight re-adjust technique; it's true in the ADALINE model, the adjustment ratio isn't dynamic but isn't necessary because the perceptron can only classify Linearly Separable Sets, Binary Class specifically

That static value who works as adjustment ratio is called "Teacher", choose the right value for this constant is important, because, if its choose wrong could create a no convergence stated (if exist a global minimum, the adjustment step for reach it is bigger than the necessary, bypassing it).

Like can be viewed in the code, not only is important choose correctly the teacher value, is necessary to use at least the right minimal amount of *epochs*, if it's lower than necessary the training process never converge, if it's bigger than necessary you have been wasted computer power, time and possibly generated an *over-fitting*

Finally, later of the training process, during the testing process (using 70% of the data for training and 30% for testing) is possible to verify that, even when the values are near to the expected, some values are very distant.

But, we have the certainty that the data is binary, when some value is bigger than 0, no matter the value, is 1, and when is less than 0, became -1.

Remembering, Remaining, that last "value correction" is made because, using values like [-1, 1] generates some erratics result, but keeping the sign.