Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 1

Classification with a single-layer perceptron

Isabella Rositi and Gustav Thorén

25 January, 2023

1 Main objectives and scope of the assignment

Our major goals in the assignment were:

- to design single layer perceptrons and use them for a simple two-class classification tasks
- to set up and observe the performance of learning algorithms for single-layer perceptrons (perceptron learning and delta rule)

The scope of this Lab is to develop single-layer perceptrons with an emphasis on the classical perceptron and the Widrow-Hoff delta rule learning algorithms. We assume that the observations of the two classes are drawn from two different multivariate normal distributions, which need to be described by means and variances that allow the data points to be positioned in the 2D plane far enough one from the other, with little to none interaction in order to determine linearly separable data.

2 Methods

The implementation of perceptron learning and delta rule was done using Python in Google Colaboratory. Vectors and matrices were handled using the *Numpy* library, and *Matplotlib* was used to plot the results. The decision boundaries were plotted for both perceptron learning and delta rule to access the learning capabilities of both methods. For perceptron learning, the number of miss-classified samples was used as metric, while mean squared error was used for delta rule. For non linearly separable, subsampled data, sensitivity and specificity was used as a performance metric.

3 Results and discussion

3.1 Classification with a single-layer perceptron

The data set was generated from two Gaussian distributions with the parameters in Tab. 1. Each class was generated with 100 data points and was assigned a binary classification of 1 or -1. After adding a scalar value of 1 (in order to account for bias) to each point, the two classes were concatenated and shuffled.

class	A	B
mean (x1, x2)	(1.5, 2.5)	(-1, 0)
standard deviation	0.5	0.5

Table 1: Parameters for generating linearly separable data drawn from multivariate normal distribution

Classic perceptron learning suffers from poor generalization due to often placing the decision boundary too close to the training data. This happens because the training stops as soon as all the data have been correctly classified instead of trying to find an optimal boundary. This can be seen in Figure 1, where the misclassified data points are 0 from epoch 10. However, we can observe that the decision boundary is very close to the data, meaning that if we had new, unseen data points which distance themselves from the ones in the data sets generated, then these will probably be misclassified.

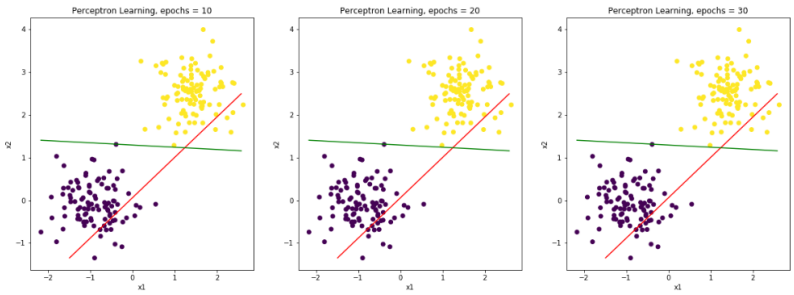

Figure 1: Classic perceptron learning with a learning rate of 0.01 and weight initialization drawn from a normal Gaussian distribution (red line).

Moreover, it is noticeable that as the learning rate decreases, the converge slows down, requiring more epochs before all data points are correctly classified. For $\eta = 0.01$ there's convergence after 10 epochs, whereas for $\eta = 0.001$ the convergence is reached only after 97 epochs.

We can see how the delta rule behaves differently when it comes to convergence in Figure 2. It can be observed that even if after 20 epochs every data point is correctly classified, the algorithm continues its training until a minimum in the MSE is reached. It can be seen that at epoch 30 the boundary has adjusted

slightly from epoch 20 to further separate the data in a way which intuitively allows for less error on the unseen data.

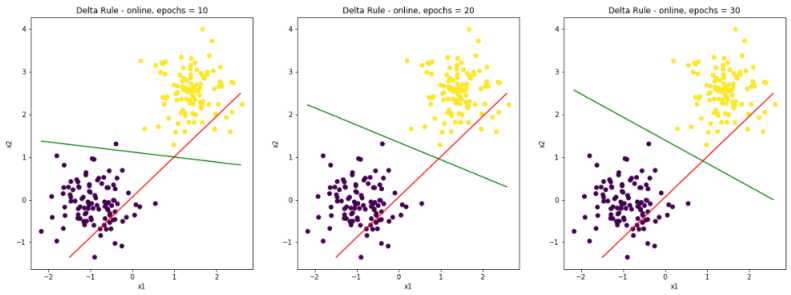

Figure 2: Delta rule with a learning rate of 0.001 and online mode. The weight are initialised by drawing from a normal Gaussian distribution (red line).

Similarly as with the perceptron rule we can observe that lowering the learning rate slows down the convergence. However, we can see that for the perceptron learning the convergence is reached faster for bigger values of η ($\eta = 0.01$) because convergence is always guaranteed. For the delta rule, the values of η needed for convergence are smaller ($\eta = 0.001$) in order to reach convergence.

The difference between sequential and batch mode can be observed in Figure 3, where the MSE is calculated over 100 epochs for both online and batch mode. The results show that the online mode is faster to converge then the batch mode for both values of the learning rate. However, we can notice that the batch mode provides smaller errors at the beginning of the training.

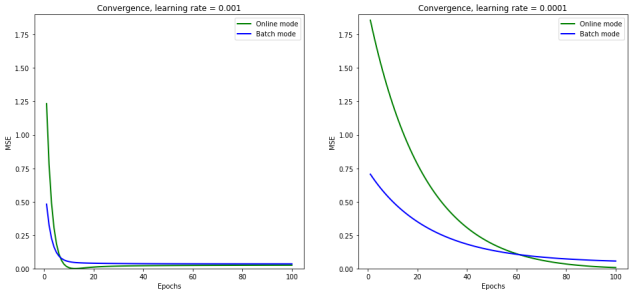

Figure 3: Delta rule with learning rates of 0.001 and 0.0001 comparing online and batch mode. The MSE is calculated over 100 epochs.

The weights initialization changes the behaviour of the delta rule for both methods. For the online mode, if the starting weights are far from the optimal solution, the algorithm may oscillate and never converge. For the batch mode, the random initialization can cause the optimization algorithm to converge to a suboptimal solution getting stuck in local minima.

When the bias in not included in the weight matrix, then the decision boundary can separate the two classes only if these are linearly separable with a line

through the origin. Our data set was not, hence the algorithm could not provide a perfect separation of the classes. We then created a new data set with means $mA = [1.5, 1.3]$ and $mB = [-1.0, -1.5]$, visible in Figure 4, in order to obtained data points separable with a line passing through the origin of the plane.

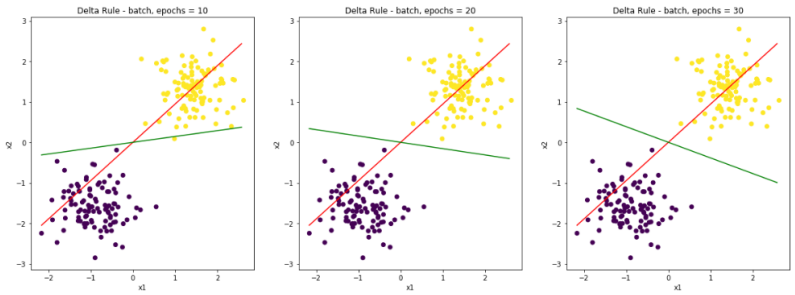

Figure 4: Delta rule with batch mode, learning rate of 0.001, weight initialization drawn from a normal Gaussian distribution (red line) and no bias.

3.2 Classification of data that are not linearly separable

3.2.1 Linearly non-separable data - 2 clusters

Two classes of data were generated from two Gaussian distributions with the parameters as given in Tab. 2. The parameters were chosen such that the two classes were slightly overlapped and not linearly separable. When applying the delta and perceptron rule to this data set, different behavior was observed as opposed to the linearly separable data set.

class	A	B
mean (x1, x2)	(1.0, 0.0)	(0.0, -0.1)
standard deviation	0.2	0.4

Table 2: Parameters for generating 2 clusters of non-linearly separable data with normal distribution

The perceptron rule never truly converges but instead oscillates in the 2D space. This behaviour is expected since the rule will update the boundary every time it encounters a misclassified sample. Therefore, if there is no way to separate the data with the linear boundary that the perceptron rule is adjusting, it will never stop oscillating.

The delta rule behaved essentially the same as with the linearly separable data. The largest difference was that it converged to a boundary which does not correctly separate all the data. This behaviour is expected since it is impossible by the nature of the data set to linearly separate it. Since the delta rule uses a heuristic for determining correctness instead of a binary system, the algorithm behaves in the same way regardless of whether the data can actually be linearly separated or not.

3.2.2 Linearly non-separable data - multi-clustered class A

For these simulations, class A has been split into 2 clusters surrounding the class B cluster. Class A was split by subtracting the x1-mean to half the data and adding it to the other half.

Training the batch delta rule algorithm with the entire data set, and the training data with 25% of each class removed (Figure 5), both resulted in very similar decision boundaries. Both seemed to converge toward this solution after 20 epochs.

When 50% of samples from class A were removed (Figure 5), the boundary climbed in the x2-direction. A likely explanation for this is that the overrepresentation of samples from class B has lead to the weights being more biased towards that class. Similarly, when 50% of class B was removed, the boundary showed an increased bias towards the class A samples, and placed itself lower to correctly classify more of those samples.

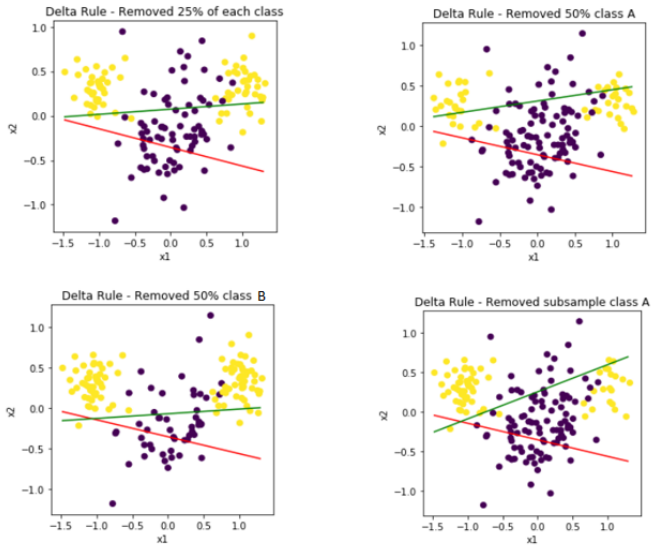

Figure 5: Decision boundaries for multiple subsamples using the batch mode delta rule and 0.001 learning rate.

For the last subsample, where 20% of class A samples with x1 < 0 and 80% with x1 > 0 were removed (Figure 5), the decision boundary heavily favored the left class A cluster, as most of the other cluster was removed. This is to be expected, as the left cluster is mostly intact and represents a clear majority of class A samples and it was able to draw a decent boundary between class A and class B. However, the generalisation capability is severely diminished as this subsample is not representative of class A.

The simulations using perceptron learning showed similar behavior as previously seen with linearly non-separable data, it finds a general area and keeps oscillating until max epochs have passed. As with delta rule, the subsample will introduce

some bias when the samples of either class is uneven or non-representative. Removing data from one class would make the decision boundary favor the other, and representing only one of the class A clusters would make the boundary more biased towards the other cluster. The final boundary is heavily dependent on initial values and training time since there will always be a large amount of misclassifications.

Removed data	**Sensitivity**	**Specificity**
No data removed	0.39	0.39
25% class A,B	0.55	0.55
50% class A	0.38	0.57
50% class B	0.6	0.56
20% class A x1<0, 80% of class A x1>0	0.46	0.43

Table 3: Sensitivity and specificity of subsamples using the delta rule

Regarding uneven class representations and non-representative sample distribution, the result showed how this can affect the decision boundary of a linear model quite drastically. When 50% of samples from one class was removed, the boundary started to favor the other class more. This will likely hurt the generalisation capacity of the perceptron, because while it can more achieve a higher rate of true positives or true negatives (depending on which class is underrepresented), it will not be able to do both (assuming that new, unseen data is distributed similarly), see Table 3. For perceptron learning, the sensitivity and specificity of the subsample would swing in one direction or the other depending on how prevalent one class was in the subsample. When one class is mis-represented, it can heavily skew the decision boundary in one direction as seen with the last subsample.

4 Final remarks

The one layered perceptron is a good introduction into artificial neural networks but is ultimately a very weak model. It gives no nature into the distribution of the classes, it has no way of classifying more than two classes and if the classes are not linearly separable it will never correctly classify all data. Despite all its flaws it is still has a few powerful traits. It is computationally very inexpensive, it can be easily extended to an arbitrary input dimension and unlike most trained networks it is very intuitive to understand. These traits make it a very good model to base the first lab on.

While the Lab in general was interesting and was a good introduction to some of the basic concepts behind artificial neural networks, the last section about subsampling should be revised. It was incredibly confusing as to what is actually asked to be presented. It hints at the standard train-test split procedure but not really, and it is very unclear what results are asked for.