Universidade Federal do Ceará
Disciplina: Inteligência Computacional Aplicada
Professor(a): Guilherme Barreto and Ajalmar
Estudante: Rubem Vasceconcelos Pacelli Matrícula: 519024

Junho/2022

Neural network - Report

1 Work 01 - Rosenblatt's perceptron

This work considers a classification problem for a multivariate dataset. The Rosenblatt's perceptron is utilized to classify the iris flower dataset. The problem consist in classifying one class among four subspecies (Setosa, Virginica, and Versicolor).

The Rosenblatt's perceptron comprises a neuron mathematical model, introduced by McCulloch and Pitts in 1943, with a learning algorithm that adjusts the synaptic weights in a supervised fashion. The McCulloch and Pitts' activation function is a step function that triggers the output from 0 to 1 when the induced local field overpass the threshold. This method is effective for binary classification of linearly separable problems, where one can sketch a straight line that divides the classes without overlapping.

At the instant n, the induced local field is given by

$$v(n) = \mathbf{w}^{\mathsf{T}}(n)\,\mathbf{x}(n)\,, \tag{1}$$

where $\mathbf{w}^{\mathsf{T}}(n) = \begin{bmatrix} w_0(n) & w_1(n) & \cdots & w_{N_a}(n) \end{bmatrix}^{\mathsf{T}}$ and $\mathbf{x}^{\mathsf{T}}(n) = \begin{bmatrix} x_0(n) & x_1(n) & \cdots & x_{N_a}(n) \end{bmatrix}^{\mathsf{T}}$ are the synaptic weights of the perceptron and the input signal, respectively, and N_a indicates the number of attributes. The elements $w_0(n)$ and $x_0(n) \triangleq +1$[1] are, respectively, the bias and its input.

The machine learning algorithms settles on the well-established theory of adaptive filters. Particularly for the Rosenblatt's perceptron, it is utilized the Least-Mean-Square (LMS) algorithm, that aims to make a instantaneous approximation of the gradient vector. The optimization algorithm is given by

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta\mathbf{g}(n), \tag{2}$$

where η is the step-learning hyperparameter and $\mathbf{g}(n) \triangleq \nabla\mathscr{E}(\mathbf{w})$ is the stochastic approximation of the gradient vector, being $\mathscr{E}(\mathbf{w})$ the cost function. The Equation (1) passes through the step function, $\varphi(\cdot)$, generating the perceptron output, $y(n) = \varphi(v(n)) \in \{0, 1\}$. This signal is compared to the desired value, $d(n) \in \{0, 1\}$ and produces the error

[1] Depending on the author, it can the defined as -1.

signal $e(n) = d(n) - y(n) \in \{-1, 0, 1\}$, which indicates whether the perceptron misclassified or not. The LMS algorithm uses the instantaneous value of the MSE (Mean-Squared Error) cost function, that is,

$$\mathscr{E}(\mathbf{w}) = \frac{1}{2}e^2(n). \tag{3}$$

Differentiating this equation with respect to the synaptic weights, we get

$$\mathbf{g}(n) = \frac{\partial \mathscr{E}(\mathbf{w})}{\partial \mathbf{w}(n)} = -\mathbf{x}(n)e(n). \tag{4}$$

Substituting (5) into (2), it yields the learning equation, given by

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta \mathbf{x}(n)e(n). \tag{5}$$

The Algorithm 1 summarizes the procedure utilized for the Rosenblatt's perceptron, including data preparation techniques, such as hand-out and shuffling the dataset. The method utilizes $N_r = 20$ independent realizations, and passes through the training set $N_e = 100$ epochs. At the end of each realization, it is stored the accuracy reached by the test data, and the accuracy of all realizations are investigated in terms of mean and standard deviation. The iris dataset contains $N = 150$ instances with $N_a = 4$ attributes (petal length, petal width, sepal length, and sepal width) and $K = 3$ classes (Setosa, Versicolour, and Virginica). It was chosen a ratio of $80\% - 20\%$ for the training and testing dataset, respectively.

Algorithm 1: Rosenblatt's perceptron

Input: $\mathbf{X}, \mathbf{d}$ /* attributes and labels dataset */

forall $\{1, 2, \cdots, N_r\}$ **do**
- $\mathbf{w}(n) \leftarrow$ initialize
- $\mathbf{X}, \mathbf{d} \leftarrow$ shuffle
- $(\mathbf{X}_{trn}, \mathbf{d}_{trn}), (\mathbf{X}_{tst}, \mathbf{d}_{tst}) \leftarrow$ hold-out /* training and testing dataset */
- **forall** $\{1, 2, \cdots, N_e\}$ **do**
  - **forall** *Instancies in the training dataset* **do**
    - $v(n) \leftarrow \mathbf{w}^{\mathsf{T}}(n)\,\mathbf{x}(n)$
    - $y(n) \leftarrow \varphi(v(n))$
    - $e(n) \leftarrow d(n) - y(n)$
    - $\mathbf{w}(n+1) \leftarrow \mathbf{w}(n) + \eta \mathbf{x}(n)e(n)$
  - $\mathbf{X}_{trn}, \mathbf{d}_{trn} \leftarrow$ shuffle
- $accuracy \leftarrow \text{test}(\mathbf{X}_{tst}, \mathbf{d}_{tst})$

The process described in Algorithm 1 was repeated for each class and results are shown in Table 1. The setosa class clearly outperforms other classes since it is linearly separable

Table 1: Rosenblatt's perceptron performance for classification problem

Classes	mean accuracy	standard deviation
Setosa	98.33	0.01972
Virginica	54.16	0.1251
Versicolor	53.66	0.1591

for some attributes, as shown in the decision surface in Figure 1[2].

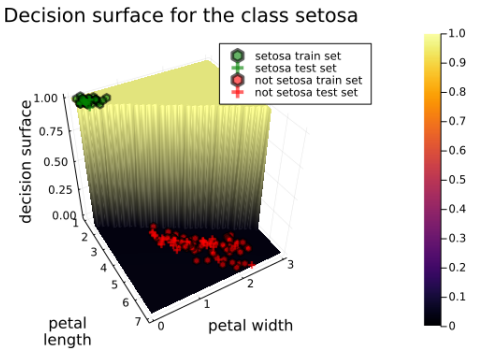


Figure 1: Decision surface of setosa class.

The confusion matrix of the setosa class is shown in Figure 2 for the first realization. The main diagonal indicates that there were neither false negative nor false positive.

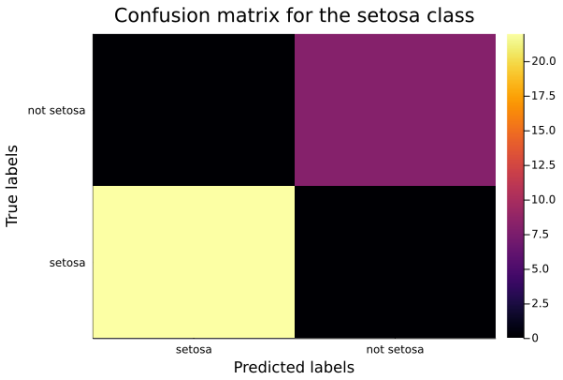


Figure 2: confusion matrix for setosa class.

The Figure 3 shows the evolution of the training dataset accuracy throughout the epochs. One can notice the fast convergence to accuracy of 100%.

[2]Since the problem has four attributes, this plot would be impossible as the would get 2 degree of freedom. Therefore, for this result, we considered only the two attributes showed in the figure.

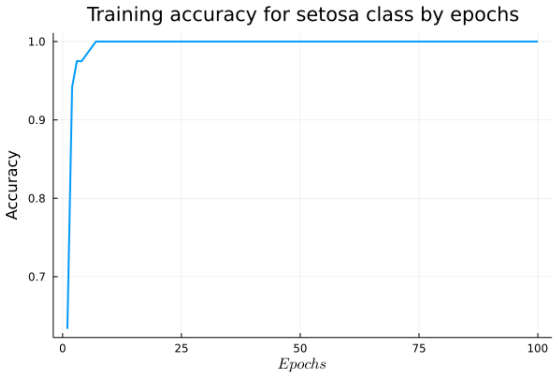


Figure 3: Training dataset evolution for the setosa classification.

For a dummy dataset with $K = 4$ classes, the Rosenblatt's perceptron achieved a mean accuracy of 97.5% and a standard deviation of 0.05. The Figure 4 shows the decision surface of the desired class for the realization whose accuracy the closest to the mean accuracy. All instances of all classes are samples drawn from a Gaussian distribution with a given mean and variance.

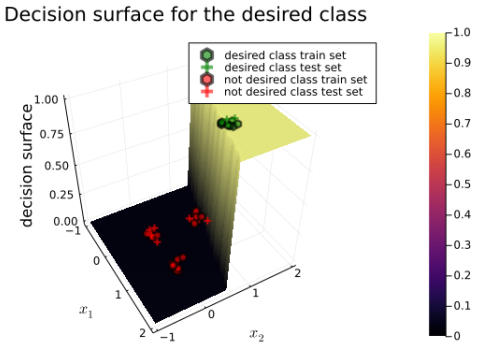


Figure 4: Decision surface for the desired class.

2 ADALINE

The Adaptive Linear Element (or ADALINE) is a variation of the Rosenblatt's perceptron, where the step function is replaced by a linear function, that is, $y(n) = \varphi(u(n)) = u(u)$. One can combine a tapped delay line with an ADALINE, thus creating an adaptive filter, widely used in statistical signal processing.

Consider a regression problem where the desired signal comes from a function $f(x)$ corrupted with a Gaussian noise. The ADALINE model tries to retrieved the original data using the same process described in Algorithm 1. However, the performance analysis is around the MSE error instead the accuracy since it is now a regression problem.

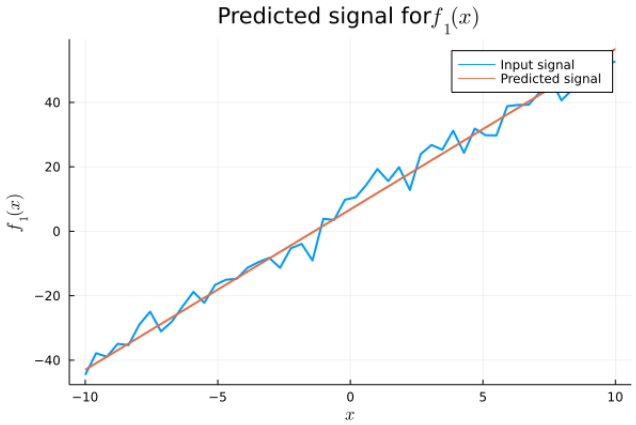
The Table 2 shows the performance of the mean MSE and its standard obtained over the independent realizations, in addition the root mean squared error (RMSE). We

considers two scenarios for the $f(x)$: linear and quadratic function.

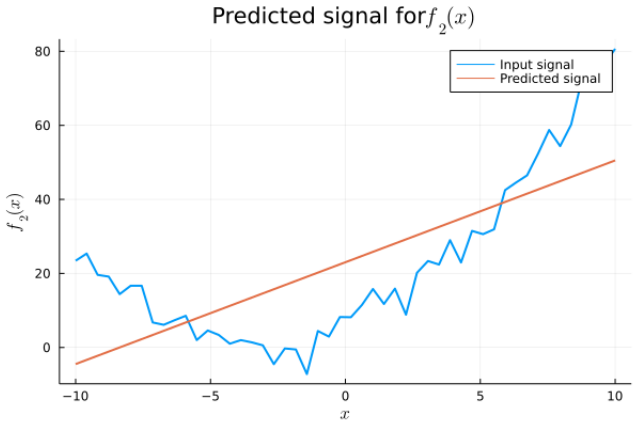
Table 2: ADALINE performance for regression problem

$d(n)$	MSE mean	MSE standard deviation	RMSE mean	RMSE standard deviation
$5x + 8$	13.17	4.69	3.57	0.65
$0.5x^2 + 3x + 6$	281.85	131.4	16.41	3.5

Naturally, the linear function achieved the best result since ADALINE can effectively solve a linear problem. Indeed, the synaptic weights of the ADALINE model is close to the coefficients of the linear function, $\mathbf{w} = \begin{bmatrix} -6.8 & 4.977 \end{bmatrix}$[3]. The Figure 5 shows the regression for the ADALINE model.



(a) ADALINE regression for $5x + 8$



(b) ADALINE regression for $0.5x^2 + 3x + 6$

Figure 5: ADALINE regression

3 Single Layer Perceptron

Although Rosenblatt's perceptron can solve linear problems, it has only one output variable. A more reasonable model for a multivariate class problem is a single-layer

[3]The negative value it is because it was chosen an input bias of $x_0(n) = -1$.

perceptron consisting of K neurons, where each neuron receives the same input signal, $\mathbf{x}(n)$.

The matrix of all coefficients is given by

$$\mathbf{W}(n) = \begin{bmatrix} \mathbf{w}_1(n) & \mathbf{w}_2(n) & \cdots & \mathbf{w}_K(n) \end{bmatrix} \in \mathbb{R}^{K \times N_a + 1}, \tag{6}$$

where K is the number of classes (a neuron for each class) and N_a is the number of attributes. The learning algorithm is given by

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \eta \boldsymbol{\delta}(n) \mathbf{x}(n)^{\mathsf{T}}, \tag{7}$$

where $\boldsymbol{\delta}(n) = \begin{bmatrix} \delta_1(n) & \delta_2(n) & \delta_K(n) \end{bmatrix}^{\mathsf{T}}$ is the vector of the local gradients, being $\delta_k(n) = e_k(n)\varphi'(v_k(n))$ the local gradient of the kth perceptron, and $\varphi'(v_k(n))$ is the derivative of output activation function, $\varphi(v_k(n))$, with respect to $v_k(n)$. For the step function (MacCulloch and Pitts' activation function), its derivative does not exist, and the local gradient of the kth neuron is simply $\delta_k(n) = e_k(n)$. For this classification problem, the labels was encoded using the one-hot method.

The Figure 6 shows the heatmap for a dummy dataset consisting of $K = 3$ classes, $N_a = 2$ attributes, and $N = 150$ instances. The classifier used the MacCulloch and Pitts' activation function and achieved a mean accuracy of 99.49% and a standard deviation of 0.0218. The same classifier was used for the iris dataset ($K = 3$ classe, $N_a = 4$ attributes, and $N = 150$ attributes) and the column dataset ($K = 3$ classes, $N_a = 6$ attributes, and $N = 310$ instances). For the iris dataset, the classifier achieved a mean accuracy of 88% with a standard deviation of 0.14, while for the column dataset the model achieved mean accuracy of 77.66% with a standard deviation of 0.06.

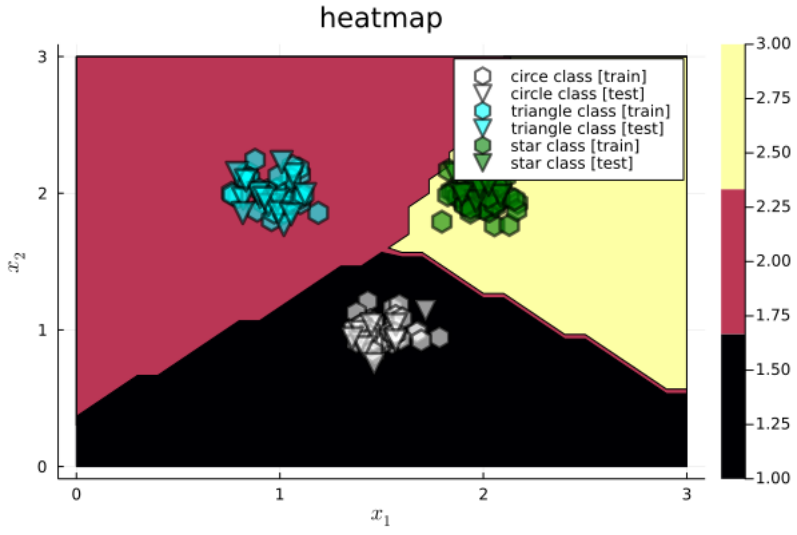
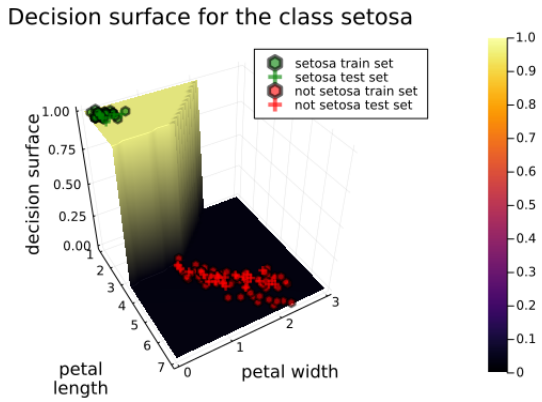
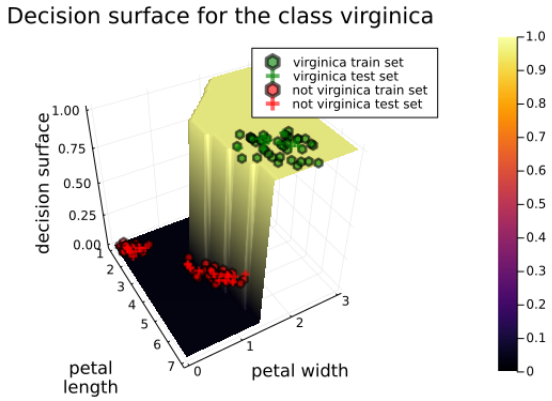


Figure 6: Heatmap of the dummy dataset.

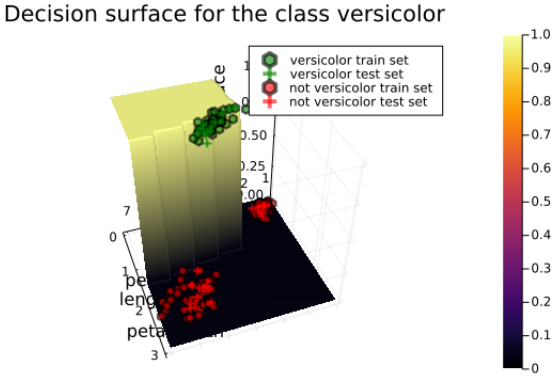
Using the signal function, the model achieved a mean accuracy of 100% for the dummy data. The surface of decision for each class of iris data is shown in Figure 7. It is possible to notice that the classifier can successively solve the problem for the setosa class as it is linearly separable from the other classes for the attributes considered (petal length and petal width).



(a) Setosa class



(b) Verginica class



(c) Versicolor class

Figure 7: main caption