University of Thessaly

Neuro-Fuzzy Computing

ECE447

3rd Problem Set

Alexandra Gianni
ID: 3382

Nikos Stylianou
ID: 2917

February 26, 2024

Problem 1

Problem 2

We are asked to write a Python program that implements steepest descent algorithm for the 1-S^1-1 RBF network. The input function that we want to approximate is

$$g(p) = 1 + \sin(p\pi/8), \quad \text{for } p \in [-4, 4]$$

We select 30 data randomly from that interval and all parameters are initialized as small numbers using `numpy.random.randn` function. It returns a number at the exact specification as needed.

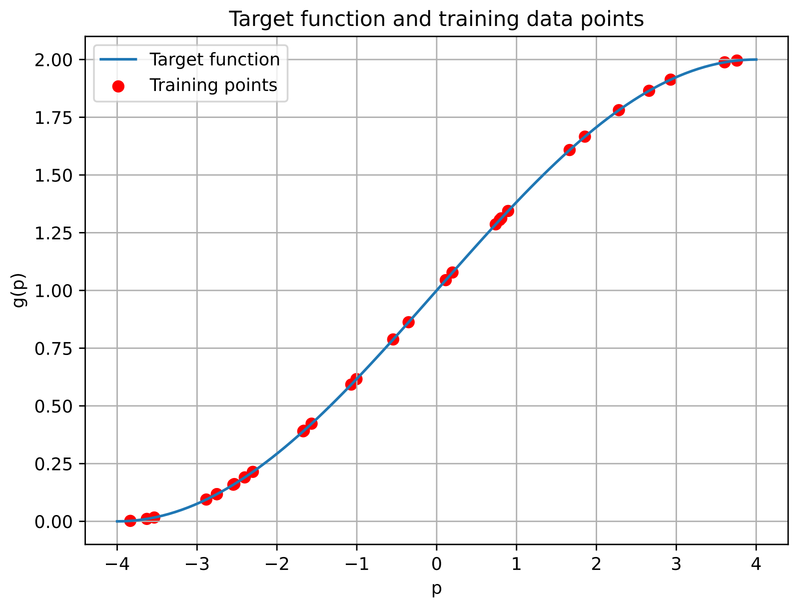
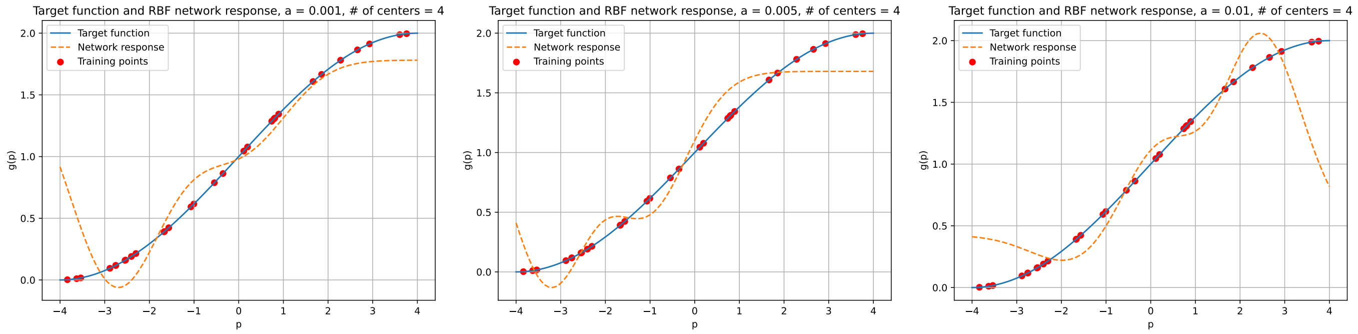


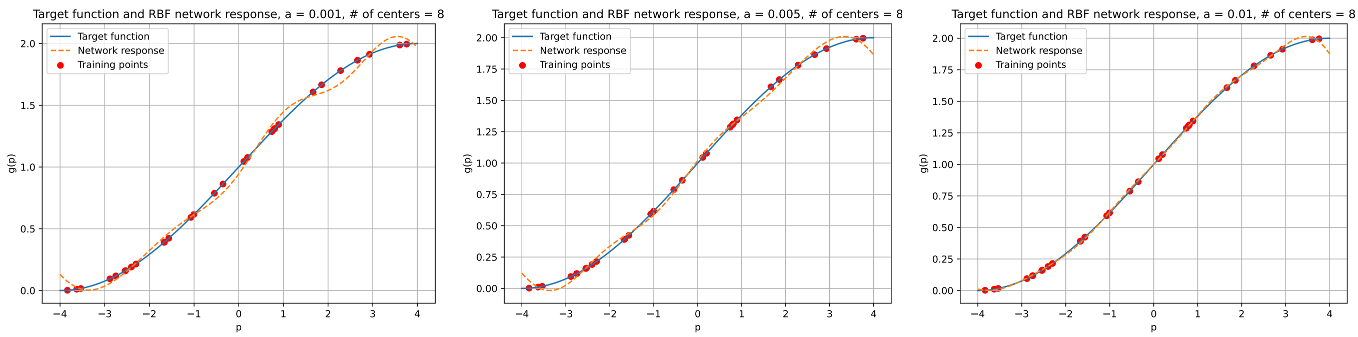
Figure 1: Input function in the specified area and the randomly assigned train data points.

For the randomly assigned data points, we added a custom seed number using in order for the results to be comparable but still use randomness. The number of centers define the total number of hidden layers in the network. So, in order to train it, thus impacting its accuracy.
Every arithmetic operation is done in matrix mode using `np.array` object type for better scalability and automation. For learning rate values α we used $[0.001, 0.005, 0.01]$ and for center the values $[4, 8, 12, 20]$ as suggested. As for the iterations number, we opted for $20 \cdot 10^3$ as we didn't want any possible bottleneck in the epoch number.
In figures 2, 3, 4 and 5 we can see the output of the neural network for each α and number of center (*hidden layers*).
Firstly, let's consider the impact of the number of centers. As we increase hat number from 4 to 8, 12, and then 20, the output dynamics undergo a notable transformation. With a smaller number of centers, the network struggles to capture intricate patterns in the data, leading to underfitting. This can be seen particularly well in figure 2, where number of centers is 4. The network fails to predict the input signal across all data points. Conversely, increasing the number of centers enables the network to model more complex relationships, reducing the underfitting and improving generalization. Again, these observations are very clear from the first change in the number of centers, from 4 to 8. The latter network has a smaller error across all data points when compared to the input signal. This increase of quality is not observed only in this increase of center numbers, but to all increases.



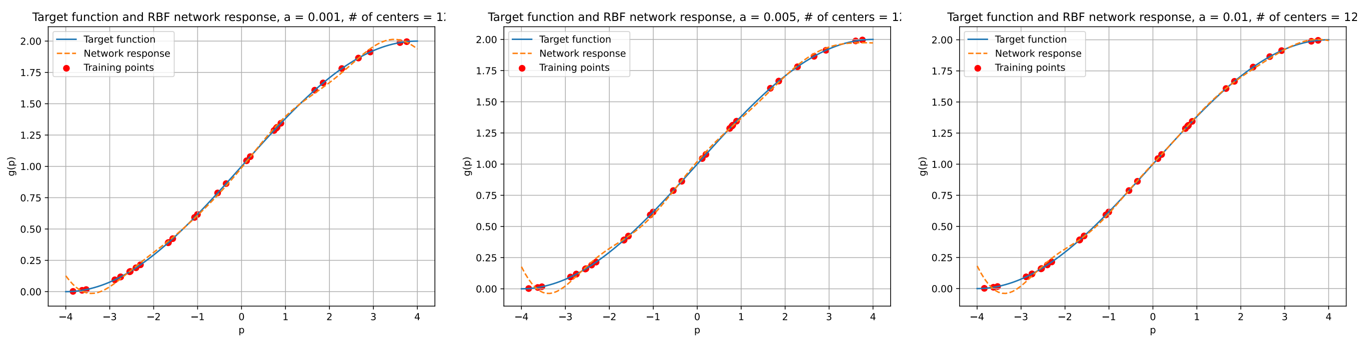
(a) $a = 0.001$ (b) $a = 0.005$ (c) $a = 0.01$

Figure 2: Input function approximation with number of centers $= 4$.



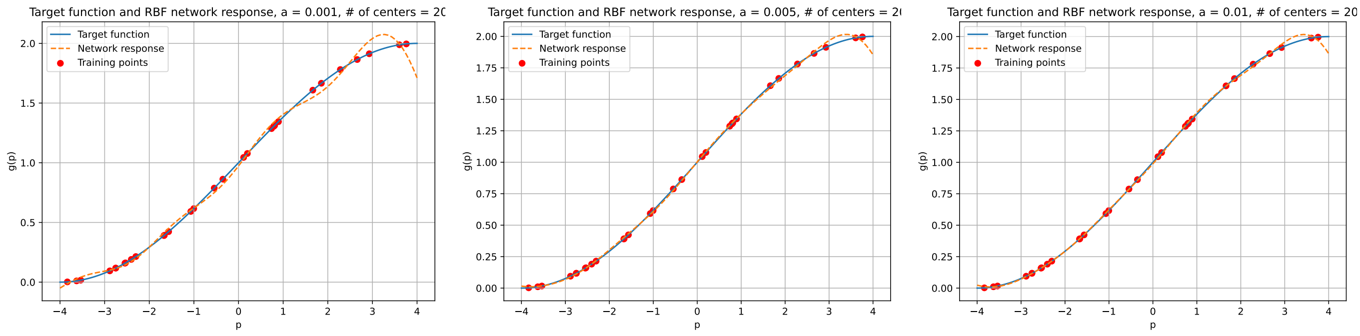
(a) $a = 0.001$ (b) $a = 0.005$ (c) $a = 0.01$

Figure 3: Input function approximation with number of centers $= 8$.



(a) $a = 0.001$ (b) $a = 0.005$ (c) $a = 0.01$

Figure 4: Input function approximation with number of centers $= 12$.



(a) $a = 0.001$ (b) $a = 0.005$ (c) $a = 0.01$

Figure 5: Input function approximation with number of centers $= 20$.

Next, let's delve into the impact of the learning rate on the RBF layer's output. A higher learning rate accelerates the convergence of the training process, enabling the model to reach a satisfactory solution faster. However, a learning rate that is too high might leads to oscillations or divergence, hindering the convergence process. From our calculations, when α is increased above 0.05, the system is diverging and by a large factor. On the other hand, a lower learning rate ensures more cautious updates to the model parameters, reducing the risk of divergence but potentially prolonging the convergence process. Thus, the output dynamics with different learning rates exhibit variations in convergence speed and possibly final performance, with an optimal learning rate striking a balance between convergence efficiency and stability.

Problem 3

LVQ (Learning Vector Quantization) is a type of artificial neural network algorithm used for supervised learning. It belongs to the category of competitive learning algorithms and is particularly effective for classification tasks.

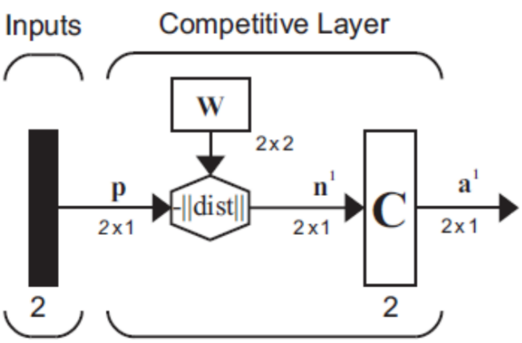


Figure 6: Given neural network.

The following $-||w_i - p||$ applies to every n_i^1. Also, we can also state that $a^1 = compet\left(n^1\right)$, where $compet$ is competitive learning layer.
During training, the distance between a and w is calculated using $dist = norm\left(p - w_{1,2}\right)$ and judging by whose norm is grater, the winning neutron gets updated. Also, the presentation order of the vectors during training is: $p_1, p_2, p_3, p_2, p_3, p_1$.

All initial values are presented below:

$$p_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad p_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, \quad p_3 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \quad w_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad w_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

In order to reduce unnecessary computations, we implemented a convergence control that stops training when weights have very small differences with each other.
After letting the network converge, we have as final weights the following:
Training converged at epoch 10.

$$w_1 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \quad w_2 = \begin{bmatrix} 0.2 \\ 1.4 \end{bmatrix}$$

Finally, in figure 7 we can see the weights over epochs during training.

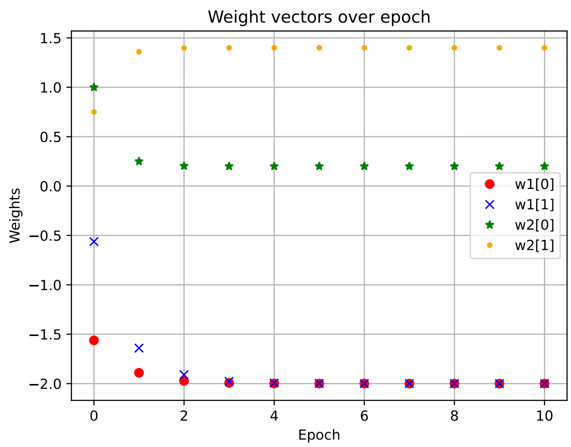


Figure 7: Weights over epoch during training.

Problem 4

Problem 5

Problem 6

Problem 7

Modeling expressions as fuzzy subsets in the context of fuzzy logic allows us to handle the uncertainty and imprecision inherent in certain descriptions like ”large,” ”very small,” or ”medium-weight.” In fuzzy logic, each element of the subset has a degree of membership ranging from 0 to 1, where 0 means ”not a member at all” and 1 means ”fully a member”.
Below are examples of how you might model the given expressions as fuzzy subsets using MATLAB code snippets, assuming some reasonable definitions for each term.

LARGE INTEGERS

For ”Large integers,” we need a membership function that gradually increases as the integers become larger.

A simple way to model this is to use a function that increases the membership score as the number exceeds a certain threshold. However, defining "large" is subjective and can vary depending on the context. For simplicity, let's assume integers greater than 100 are considered large, but the transition starts at 50, becoming more "large" as the number increases.

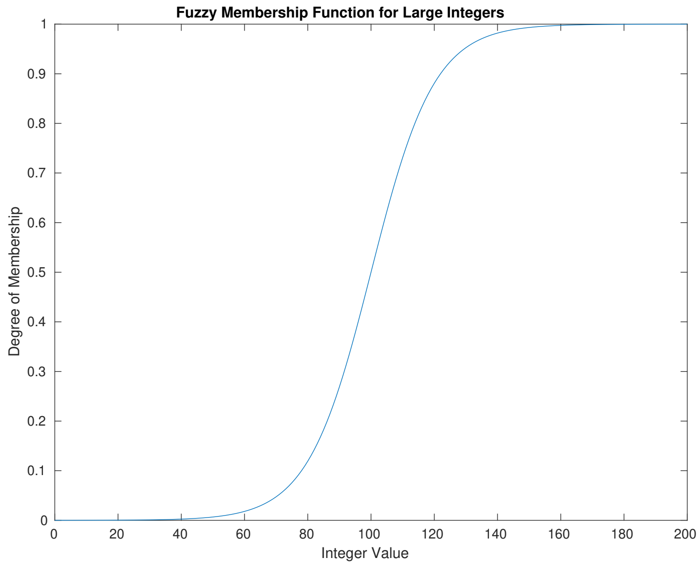


Figure 8: Fuzzy membership function for Large Integers

we are using the sigmoid function for a smooth transition.

<u>VERY SMALL NUMBERS</u>

For "very small numbers," we can consider numbers close to zero as having a higher degree of membership. "Very small numbers" can include both positive numbers close to zero and negative numbers. For this, a membership function that assigns higher scores to numbers closer to zero can be used. We'll focus on positive numbers for simplicity, but this can be adjusted to include negatives.

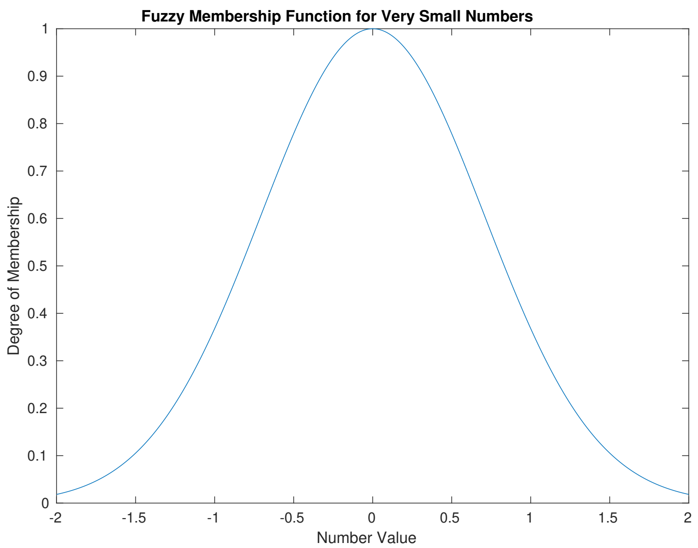


Figure 9: Fuzzy membership function for Very Small Numbers

we are using the Gaussian function centered at 0.

Problem 8

The term ”ordinary subset of level α in fuzzy set theory refers to a non-fuzzy subset that is created from a fuzzy subset by taking into consideration only those elements whose degrees of membership are at least α. In essence, it’s a threshold-based selection from a fuzzy set in which each element is only included if its membership score is greater than or equal to α.
Conversely, this idea is extended to pairs of components for a fuzzy relation by the ”ordinary relation of level α”. A set of ordered pairs with corresponding degrees of membership that indicate the strength of the relationship between the elements form a fuzzy relation. Thus, the set of all pairs whose membership in the fuzzy relation is at least α is the ordinary relation of level α.

Analytically, the ordinary relation of level α for a fuzzy relation is defined as the set of all pairs (x, y) for which the membership function $\mu_{\tilde{R}}(x, y)$ is greater than or equal to α. Alternatively, it is a crisp set that is derived from the fuzzy relation by incorporating all element pairs with a degree of membership greater than the specified level α

For a fuzzy relation with a membership function $\mu_{\tilde{R}}(x, y) = 1 - \frac{1}{1+x^2+y^2}$, the ordinary relation of level 0.3 is the set:
$R_{0.3} = (x, y)|\mu_{\tilde{R}}(x, y) \geq 0.3$
This means you are looking for all the pairs (x, y) where the membership value is at least 0.3.

To determine analytically the ordinary relation of level 0.3, we will solve the inequality

$$
\begin{aligned}
\mu_{\tilde{R}}(x, y) &\geq 0.3 \\
1 - \frac{1}{1+x^2+y^2} &\geq 0.3 \\
\frac{1}{1+x^2+y^2} &\leq 0.7 \\
1+x^2+y^2 &\geq \frac{1}{0.7} \\
x^2+y^2 &\geq \frac{1}{0.7} - 1
\end{aligned}
$$

From basic maths we know that $x^2 + y^2$ is the equation of a circle. This represents the region outside a circle centered at the origin with a radius squared of $\frac{1}{0.7} - 1$. By calculating the radius we can plot the region that satisfies the ordinary relation of level 0.3.

Problem 9

Problem 10