Neural Network 1

Project 5 report

Team name: MicVanVan

Members: m5261108 Kazuki Fujita
m5261110 Mizuki Goto
m5261132 Haruki Maeda
m5261147 Kaito Ogino

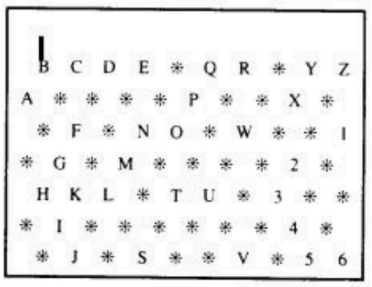
Table

1. Problem

- A program is given in the web for SOFM.
- Down –load this program, and check if it works for the example given in this lecture.
- Modify the program, and apply your program to IRIS dataset.
- Try to find a set of learning parameters that can result in good results (a 2-Dimensional map of the feature space).

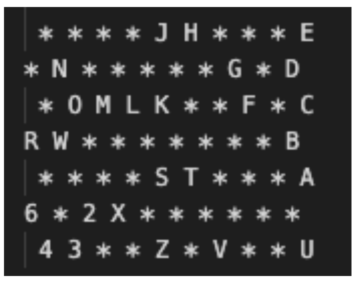
2. Checking SOFM program

We checked the SOFM example program. In lecture resume, the result for Example 1 was run with M1 = 7 and M2 = 10. Therefore, we changed M1 and M2 values to match the values for Example 1 and run the program. As a result, The output was different from the result of Example 1 and do not display some alphabets. The reason for the change is that it is initialized with a random number. A data value is close or not a random number value and may not fall within the M1*M2 range depending on it.



```
  B C D E * Q R * Y Z
A * * * * P * * X *
  * F * N O * W * * I
* G * M * * * * 2 *
  H K L * T U * 3 * *
* I * * * * * * 4 *
  * J * S * * V * 5 6
```

Result 1: SOFM result for Example 1



```
 * * * * J H * * * E
* N * * * * * G * D
 * O M L K * * F * C
R W * * * * * * * B
 * * * * S T * * * A
6 * 2 X * * * * * *
 4 3 * * Z * V * * U
```

Result 2: Result running the program on my PC
GitHub page: https://github.com/K-Fujita-onkyo/neural_network/blob/main/Project5/example_ans.txt

3. Implementation for Iris

We decided to create the example program with as few changes as possible, so we devised Iris data in this time. As a result, our program was successfully executed simply by changing P and I values. Our program is available on GitHub: https://github.com/K-Fujita-onkyo/neural_network/blob/main/Project5/p5_Iris.c

3.1. Iris data

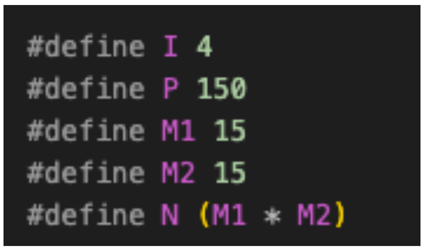
We described setosa as A, versicolor as B, and virginica as C in Iris data to take advantage of the good qualities of the example program. This program uses ASCII numbers to determine the output character. The program seeks characters relative to the ASCII number of “A”. Therefore, we have re-created the data as Table 1. The arranged Iris data is on GitHub: https://github.com/K-Fujita-onkyo/neural_network/blob/main/Project5/Iris_ans.txt

Attribute number	x_1	x_2	x_3	x_4	Item
Data value	sepal length	sepal width	petal length	petal width	Types of Iris

Table 1: Remade Iris data values

3.2. Program

We changed P and I values to to match Iris data. We set the values of M1 and M2 to 15 in this time. our program is exactly the same as the example program other than this part.

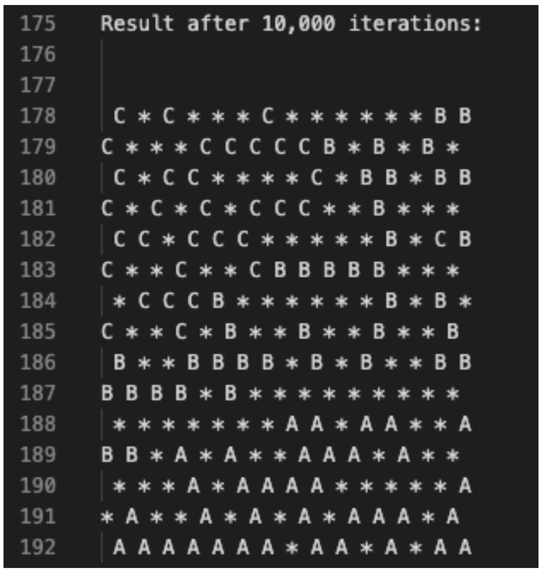


```
#define I 4
#define P 150
#define M1 15
#define M2 15
#define N (M1 * M2)
```

Figure 1: Changed P and I values

4. Evaluation and Result

We evaluated our program using our remade Iris data. As a result, Iris data were classified by type in a two-dimensional plane. Result 3 expressed setosa as A, versicolor as B, and virginica as C. We looked at Result 3 and found that setosa had a completely isolated distribution, while versicolor and virginica had a slightly mixed distribution.



```
Result after 10,000 iterations:


 C * C * * * C * * * * * * B B
C * * * C C C C C B * B * B *
 C * C C * * * * C * B B * B B
C * C * C * C C C * * B * * *
 C C * C C C * * * * * B * C B
C * * C * * C B B B B B * * *
 * C C C B * * * * * * B * B *
C * * C * B * * B * * B * * B
 B * * B B B B * B * B * * B B
B B B B * B * * * * * * * * *
 * * * * * * * A A * A A * * A
B B * A * A * * A A A * A * *
 * * * A * A A A A * * * * * A
* A * * A * A * A * A A A * A
 A A A A A A A * A A * A * A A
```

Result 3: Our Iris program result after 10000 iterations
GitHub page: https://github.com/K-Fujita-onkyo/neural_network/blob/main/Project5/Iris_ans.txt

5. Discussion

We will discuss the following things:

- Why was it possible to classify 4-dimensional data in 2 dimensions?
- How does SOFM update neurons?

5.1. Why was it possible to classify 4-dimensional data in 2 dimensions?

The reason is that SOFM can reduce the dimensionality and may also preserve the neighborhood relation structure of the problem space. Self-organizing feature maps (SOFM) learn to classify input vectors according to how they are grouped in the input space [1]. SOFM uses Kohonen network to reduce the dimensionality of the pattern space. Therefore, it is possible to reduce the dimensionality of multi-dimensional data for output.

5.2. How does SOFM update neurons?

SOFM is updated in the following steps.

1. Select a sample *p* at random from the training set.
2. Find the neuron closest to the input.
3. Update the weight of the neuron if it is in the neighborhood of the winner neuron.

In Step2, SOFM find a winner neuron using Mean Squared Error (MSE):

$$MSE = \sum_{i=1}^{I} (w_i - x_i)^2$$

SOFM finds the minimum difference between the weights w and Iris data x. In Step 3, SOFM updates not only the winner neuron but also its surrounding neurons using Kohonen rules. The weights are updated as follows:

$$W_k = \alpha(X - W_k) \text{ for } k \in N_m$$

N_m is the neighborhood of the m-th neuron. In our program, α is updated by the following:

$$\alpha = \frac{q * (r_1 - r_2)}{U} + r_1$$

q is the number of updates. U is the max number of updates given by a function. r_1 and r_2 are learning rates given by a function. In this time during the first 1,000 learning cycle, the learning rate decreases linearly with time from 0.5 to 0.04. In this time during the first 10,000 learning cycle, the learning rate decreases linearly with time from 0.04 to 0.0. Thus, α depends on the number of updates and the learning rate.

This step is repeated to update the neurons.

6. Conclusion

In Project 5, we classified Iris data using SOFM. Self-organizing feature maps (SOFM) learn to classify input vectors according to how they are grouped in the input space. SOFM can output patterns with reduced dimensionality. As a result, we were able to reduce the dimension of our four-dimensional Iris data to two dimensions for classification.

Bonus report title

Compare results from Self-Organizing Feature Map, winner-take-all learning, and Gaussian Mixture Model

Team: MicVanVan Author: m5261108 Kazuki Fujita

1. Introduction

We learned distance-based neural networks in Lectures 4 and 5. In Project 4, we created a program using winner-take-all learning and a program with Gaussian Mixture Model (GMM). In Project 5, we created a program based on Self-Organizing Feature Map (SOFM). We classified Iris in each of their evaluations.

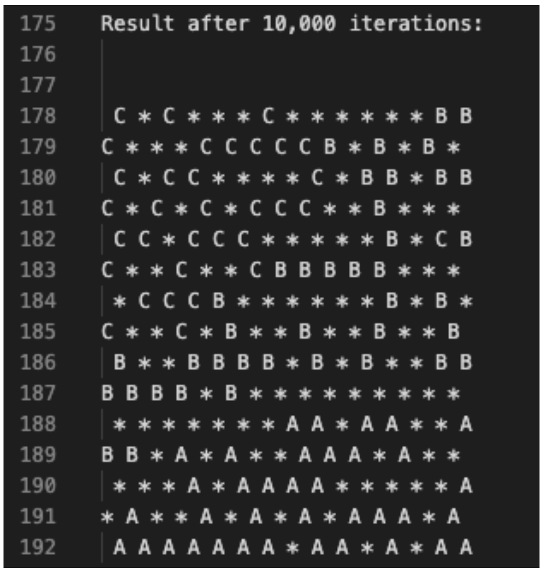
In this time, we compared results from SOFM, winner-take-all learning, and GMM. Additionally, we discussed what can be read from those results.

2. Comparison

We created a table summarizing the results of the winner-take-all learning and Gaussian Mixture Model in the Bonus report for Project 4. In Project 5, SOFM also output a two-dimensional distribution map (Result 3). We compared and discussed these results. We show again those results.

Algorithm	Species	Number of data	Corrects	Incorrects		
				Iris-setosa	Iris-versicolor	Iris-virginica
Winner-take-all learning	Iris-setosa	50	50	-	0	0
	Iris-versicolor	50	48	0	-	2
	Iris-virginica	50	34	0	16	-
Gaussian Mixture Model	Iris-setosa	50	50	-	0	0
	Iris-versicolor	50	45	0	-	5
	Iris-virginica	50	50	0	0	-

Table1: Comparison of Winner-take-all learning and GMM



```
Result after 10,000 iterations:


 C ∗ C ∗ ∗ ∗ C ∗ ∗ ∗ ∗ ∗ ∗ B B
C ∗ ∗ ∗ C C C C C B ∗ B ∗ B ∗
 C ∗ C C ∗ ∗ ∗ ∗ C ∗ B B ∗ B B
C ∗ C ∗ C ∗ C C C ∗ ∗ B ∗ ∗ ∗
 C C ∗ C C C ∗ ∗ ∗ ∗ ∗ B ∗ C B
C ∗ ∗ C ∗ ∗ C B B B B B ∗ ∗ ∗
 ∗ C C C B ∗ ∗ ∗ ∗ ∗ ∗ B ∗ B ∗
C ∗ ∗ C ∗ B ∗ ∗ B ∗ ∗ B ∗ ∗ B
 B ∗ ∗ B B B B ∗ B ∗ B ∗ ∗ B B
B B B B ∗ B ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗ ∗
 ∗ ∗ ∗ ∗ ∗ ∗ ∗ A A ∗ A A ∗ ∗ A
B B ∗ A ∗ A ∗ ∗ A A A ∗ A ∗ ∗
 ∗ ∗ ∗ A ∗ A A A A ∗ ∗ ∗ ∗ ∗ A
∗ A ∗ ∗ A ∗ A ∗ A ∗ A A A ∗ A
 A A A A A A A ∗ A A ∗ A ∗ A A
```

(Re-show) Result 3: Our Iris program result after 10000 iterations, expressed setosa as A, versicolor as B, and virginica as C

3. Discussion

We discussed the following:

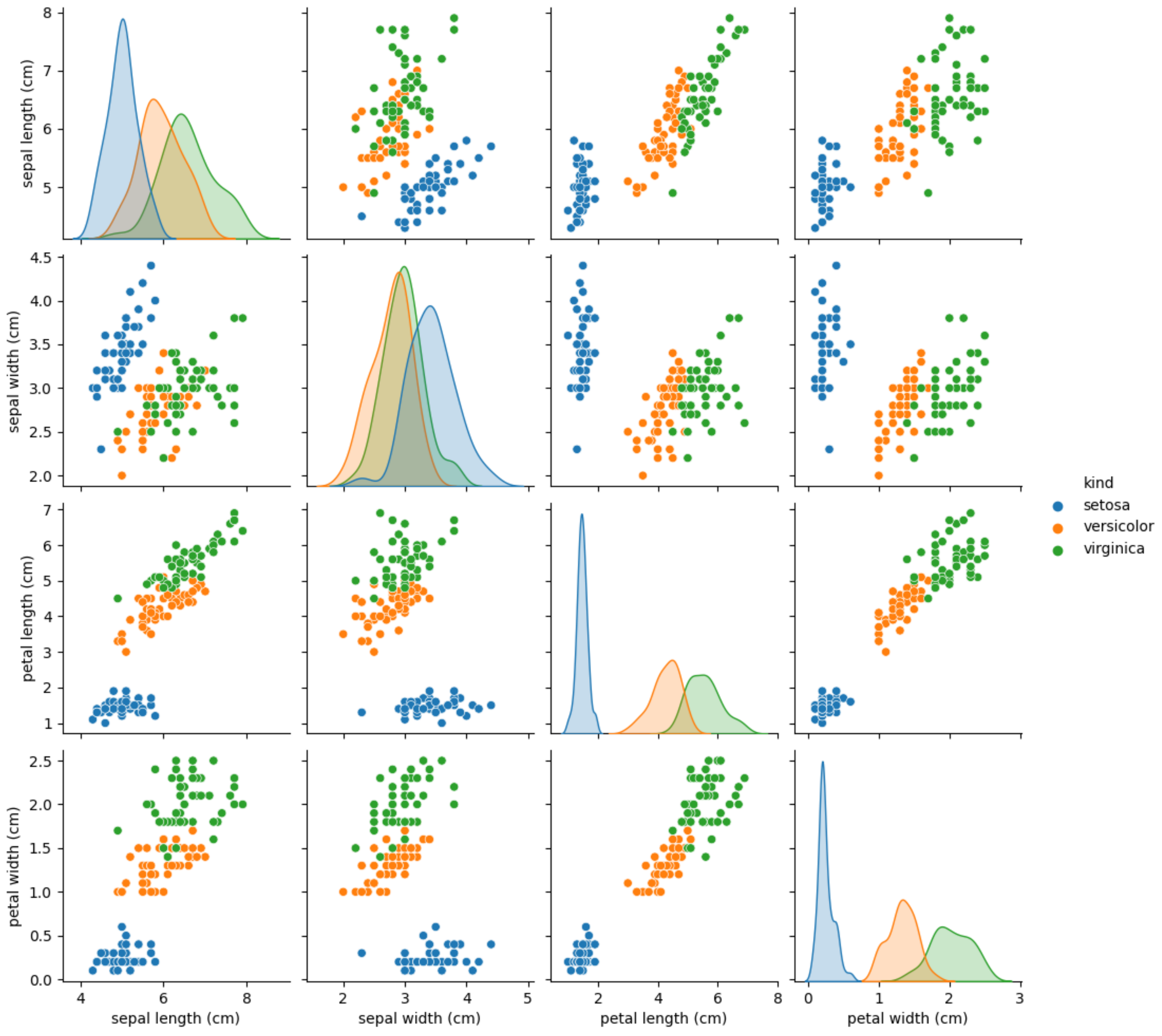
- Is there a relationship between each of the results?
- What do these results tell us about Iris-data?

3.1. Is there a relationship between each of the results?

There is a relationship between winner-take-all learning and GMM and SOFM because They classified setosa perfectly, but their results mixed versicolor and virginica. Result 3 shows that virginica is mixed in versicolor group. Also, Table 1 shows that GMM and winner-take-all learning mistook versicolor for virginica. In other words, the reason why winner-take-all learning and GMM misunderstood the classification of versicolor and virginica can be read from the SOFM diagram. Therefore, we found there is a relationship between the results of each algorithm.

3.2. What do these results tell us about Iris-data?

From these results, we found that setosa has an independent shape compared to other types, but versicolor and virginica have several similar shapes because winner-take-all learning and GMM misunderstood multiple classifications of versicolor and virginica, and they were mixed in the SOFM output. Then we investigated whether our interpretation really matched. We have created a program to output a graph that statistics the shape of Iris: https://github.com/K-Fujita-onkyo/neural_network/blob/main/Project5/Iris_relationships.py The output result is described below.



(Re-show) Graph 1: Iris statistical results by shape

From Graph 1, we found that setosa is independent. Also, it can be seen that versicolor and virginica are mixed for each parameter. Especially in the graph where sepal width and sepal length are parameters, versicolor and virginica overlap most. Therefore, we found that versicolor and virginica have similar shapes. In other words, it turns out that our interpretation is correct.

4. Conclusion

We compared the results of Project 4 and Project 5. As a result, it was found that each result has relationships. In addition, we found that there are similar types independent types in Iris data from these result.

At the end, we learned Neural Network. At first, we had a lot of things we didn't know. but Prof. Liu's easy-to-understand lessons, We were able to understand the basics of Neural Networks. Thank you Prof. Liu.

Reference

[1] Kaski, S.; Kangas, J.; and Kohonen, T. (1998). Bibliography of self-organizing map (SOM) papers: 1981-1997. *Neural Computing Surveys*, 1: 102-350.