Compute Intelligence PS s7 Lab 3

Joris Plaščinskas

November 5, 2024

Introduction

The goal of this laboratory work is to classify iris dataset using WEKA software. My student number is 2016020, which means that my variant is: "0. sepal length, sepal width, petal length". Iris dataset will be split into train and test sets at the start (80% & 20%). In figure-1 below you can see how the data is distributed (blue - iris setosa, red - iris versicolor, light blue - iris virginica).

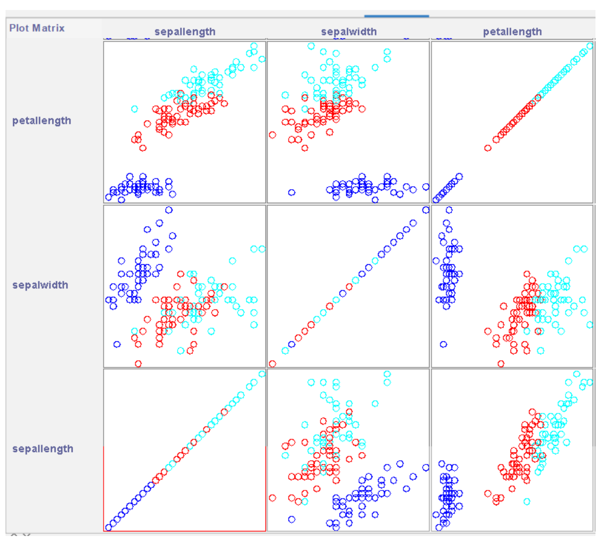

Figure 1: Iris Data Visualized

Schemes

Scheme 1

In figure-5 you can see the first scheme. This scheme takes the initial data, pre-processes it, trains a multilayer perceptron and evaluates it's performance. The learning rate and momentum were set to 0.2, 0.0002 and hidden layers were: 2,1. I tried changing the parameters, but the accuracy would always end up the same unless I set the learning rate to some extreme value. The performance of the model can be seen below in figure-2. In figure-3 you can see the metrics when normal parameters were used and in figure-4 you can see the results when a very low learning rate was used.

```
 a  b  c   <-- classified as
 0  0  0 |  a = Iris-setosa
 0 35  5 |  b = Iris-versicolor
 0  2 38 |  c = Iris-virginica
```

Figure 2: Confusion Matrix

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	Iris-setosa
0.875	0.025	0.946	0.875	0.909	0.868	0.989	0.978	Iris-versicolor
0.950	0.063	0.884	0.950	0.916	0.872	0.991	0.984	Iris-virginica
0.942	0.029	0.943	0.942	0.942	0.913	0.993	0.987	

Figure 3: Metrics

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.000	0.000	?	0.000	?	?	1.000	1.000	Iris-setosa
1.000	1.000	0.333	1.000	0.500	?	0.337	0.251	Iris-versicolor
0.000	0.000	?	0.000	?	?	0.780	0.661	Iris-virginica
0.333	0.333	?	0.333	?	?	0.706	0.637	

Figure 4: Extreme LR Metrics

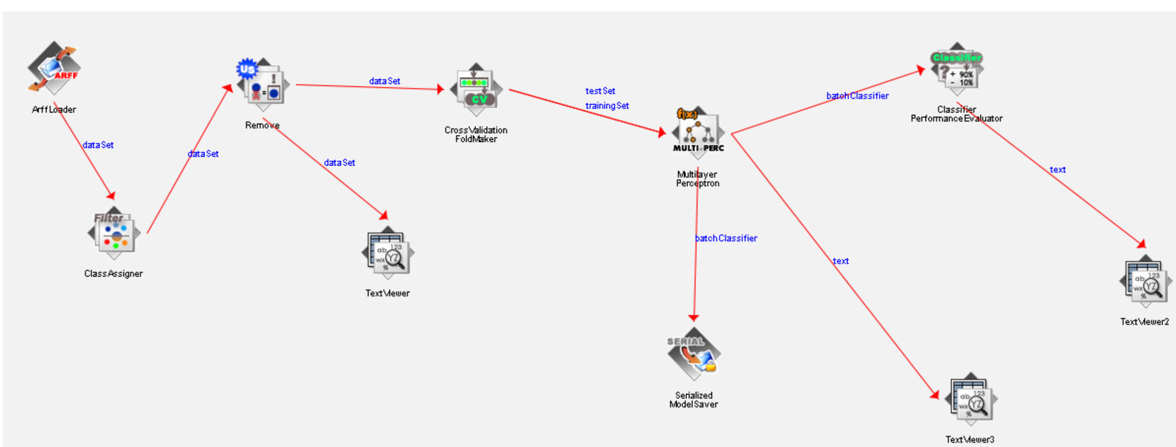

Figure 5: Scheme 1

Scheme 2

Scheme 2 can be seen in figure-8 below. This scheme now adds a prediction appender, that outputs the whole dataset with model predictions attached. Model predictions can be seen in figure-6. The model performs flawlessly on test data as seen in figure-7

```
@data
5,3.5,0.3,Iris-setosa,0.964727,0.033524,0.001749
4.5,2.3,0.3,Iris-setosa,0.892982,0.104564,0.002455
4.4,3.2,0.2,Iris-setosa,0.964562,0.033687,0.001751
5,3.5,0.6,Iris-setosa,0.962776,0.035447,0.001777
5.1,3.8,0.4,Iris-setosa,0.96524,0.033019,0.001741
4.8,3,0.3,Iris-setosa,0.960803,0.037392,0.001805
5.1,3.8,0.2,Iris-setosa,0.965577,0.032687,0.001736
4.6,3.2,0.2,Iris-setosa,0.96431,0.033935,0.001755
5.3,3.7,0.2,Iris-setosa,0.965339,0.032922,0.001739
5,3.3,0.2,Iris-setosa,0.964278,0.033967,0.001755
5.5,2.6,1.2,Iris-versicolor,0.027937,0.958677,0.013385
6.1,3,1.4,Iris-versicolor,0.026666,0.95941,0.013924
5.8,2.6,1.2,Iris-versicolor,0.024224,0.96154,0.014236
5,2.3,1,Iris-versicolor,0.032287,0.955152,0.012561
5.6,2.7,1.3,Iris-versicolor,0.024769,0.960977,0.014254
5.7,3,1.2,Iris-versicolor,0.119952,0.871965,0.008083
5.7,2.9,1.3,Iris-versicolor,0.039828,0.948395,0.011777
6.2,2.9,1.3,Iris-versicolor,0.027526,0.958925,0.013549
5.1,2.5,1.1,Iris-versicolor,0.038434,0.949731,0.011835
5.7,2.8,1.3,Iris-versicolor,0.029307,0.957468,0.013225
6.7,3.1,2.4,Iris-virginica,0.000015,0.001339,0.998646
6.9,3.1,2.3,Iris-virginica,0.000015,0.001385,0.9986
5.8,2.7,1.9,Iris-virginica,0.000027,0.003235,0.996739
6.8,3.2,2.3,Iris-virginica,0.000016,0.001443,0.998541
6.7,3.3,2.5,Iris-virginica,0.000015,0.001349,0.998636
6.7,3,2.3,Iris-virginica,0.000015,0.001365,0.99862
6.3,2.5,1.9,Iris-virginica,0.000019,0.001888,0.998093
6.5,3,2,Iris-virginica,0.000023,0.002509,0.997468
6.2,3.4,2.3,Iris-virginica,0.000018,0.001836,0.998145
5.9,3,1.8,Iris-virginica,0.000422,0.178163,0.821414
```

Figure 6: Predictions

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	Iris-setosa
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	Iris-versicolor
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	Iris-virginica
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	

Figure 7: Metrics

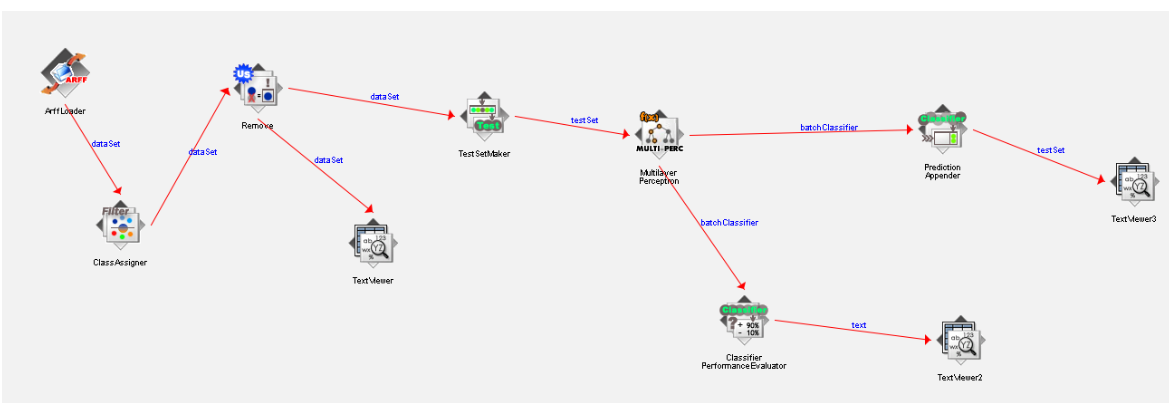

Figure 8: Scheme 2

Scheme 3

Scheme 3 can be seen in figure-10 below. This scheme implements two datasets: train and test. It also outputs model parameters, which will be used later to implement the model in python code.

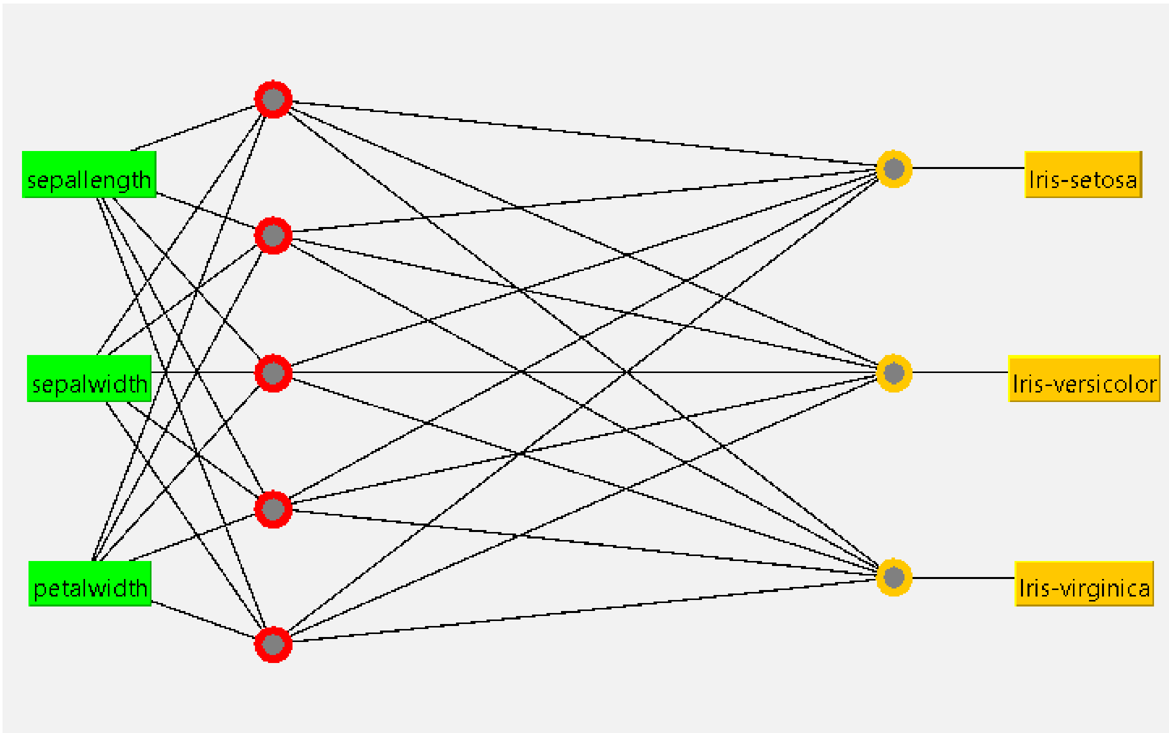

Figure 9: Model

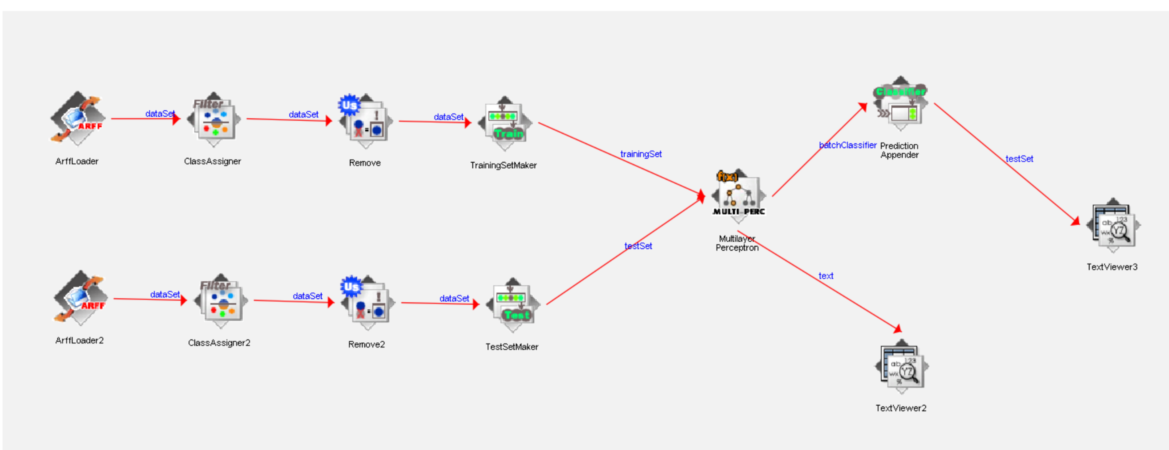

Figure 10: Scheme 3

Excel - Or Python Code In My Case

I chose to implement the model in python instaed of Excel. I copy pasted model weights and biases directly into numpy matrices. I implemented sigmoid and softmax functions, then using these functions and matrix multiplication I calculated the predictions, which I then processed into labels, the whole code can be seen below. Model parameters can be seen in figure-10.

```
# Each row represents a neuron (each column a specific features weights)
W0 = numpy.array([[1.0094239000969418, -2.8861984211733054, 3.524109890797802],
                  [-0.6033113487383684, 0.25333881908187433, -3.6220367976542143],
                  [-1.8152549194443377, 3.1348671687767653, -5.417135204096832],
                  [0.25354873129701394, 1.690814522158558, -8.296843560887408],
                  [1.600551815234887, -2.841080599997625, 6.556426248370086]])
b0 = numpy.array([[1.3195263166421949],
                   [1.1195114721466286],
                   [-2.8805678200060667],
                   [3.9195462572571373],
                   [-2.4707970148546785]])

W1 = numpy.array([[-5.155589087569882, 0.25303286914960293, 4.09097683452935, 1.5018299785124933, -3.201927916423887],
                  [1.8726064573393189, 1.0937839520407748, -7.831382567951255, 5.169787408661677, -5.3936820851078355],
                  [3.7599418891368646, -2.341082920843232, -3.7328746654661598, -5.198485427952334, 3.970058776658249]])
b1 = numpy.array([[-1.0107020526599064],
                   [-2.942710079399252],
                   [-0.9416790197108846]])
```

Figure 11: Model Parameters

```
import numpy



def softmax(x):
    x_shifted = x - numpy.max(x, axis=1, keepdims=True)
    e_x = numpy.exp(x_shifted)
    return e_x / e_x.sum(axis=1, keepdims=True)



def sigmoid(X):
    return 1 / (1 + numpy.exp(-X))



# Each column represents a data point (each row a feature)
X = numpy.genfromtxt('iris-new-test.csv', delimiter=",")[:, :-1].T
min_vals = X.min(axis=0)
max_vals = X.max(axis=0)
```

```
X = (2 * X - min_vals - max_vals) / (max_vals - min_vals)


# Each row represents a neuron (each column a specific features weights)
W0 = numpy.array([[1.0094239000969418, -2.8861984211733054, 3.524109890797802],
                 [-0.6033113487383684, 0.25333881908187433, -3.6220367976542143],
                 [-1.8152549194443377, 3.1348671687767653, -5.417135204096832],
                 [0.25354873129701394, 1.690814522158558, -8.296843560887408],
                 [1.600551815234887, -2.841080599997625, 6.556426248370086]])
b0 = numpy.array([[1.3195263166421949],
                  [1.1195114721466286],
                  [-2.8805678200060667],
                  [3.9195462572571373],
                  [-2.4707970148546785]])


W1 = numpy.array([[-5.155589087569882, 0.25303286914960293, 4.09097683452935, 1.5018299785124933, -3.
                 [1.8726064573393189, 1.0937839520407748, -7.831382567951255, 5.169787408661677, -5.3
                 [3.7599418891368646, -2.341082920843232, -3.7328746654661598, -5.198485427952334, 3.
b1 = numpy.array([[-1.0107020526599064],
                  [-2.942710079399252],
                  [-0.9416790197108846]])



print(X.shape)
print(W0.shape)
Y = softmax(W1 @ sigmoid(W0 @ X + b0) +b1)
print(Y)
Y = numpy.argmax(Y, axis=0).T
print(Y)
class_names = numpy.array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'])
Y = class_names[Y]
print(Y)
```

Table 1: Final Results

	Predicted In Python			Predicted In WEKA			Actual
	Iris-setosa	Iris-versicolor	Iris-virginica	Iris-setosa	Iris-versicolor	Iris-virginica	
1	**0.9969**	0.0031	0.0000	**0.8276**	0.1724	0.0000	Iris-setosa
2	0.2236	**0.7763**	0.0001	0.3344	**0.6575**	0.0081	Iris-setosa
3	**0.9978**	0.0022	0.0000	**0.8265**	0.1735	0.0000	Iris-setosa
4	**0.9940**	0.0060	0.0000	**0.8264**	0.1736	0.0000	Iris-setosa
5	**0.9969**	0.0031	0.0000	**0.8283**	0.1717	0.0000	Iris-setosa
6	**0.9572**	0.0428	0.0000	0.8187	**0.1813**	0.0000	Iris-setosa
7	**0.9983**	0.0017	0.0000	**0.8284**	0.1716	0.0000	Iris-setosa
8	**0.9944**	0.0056	0.0000	**0.8262**	0.1738	0.0000	Iris-setosa
9	**0.9959**	0.0041	0.0000	**0.8282**	0.1718	0.0000	Iris-setosa
10	**0.9886**	0.0114	0.0000	**0.8267**	0.1733	0.0000	Iris-setosa
11	0.0000	0.0046	**0.9954**	0.0595	**0.7754**	0.1651	Iris-versicolor
12	0.0003	0.2487	**0.7510**	0.0656	**0.7854**	0.1490	Iris-versicolor
13	0.0004	**0.7998**	0.1998	0.0527	**0.7613**	0.1859	Iris-versicolor
14	0.0003	**0.9762**	0.0235	0.0597	**0.7757**	0.1646	Iris-versicolor
15	0.0003	0.2420	**0.7578**	0.0508	**0.7566**	0.1926	Iris-versicolor
16	0.0004	**0.7612**	0.2384	**0.2807**	0.7062	0.0130	Iris-versicolor
17	0.0004	**0.6290**	0.3705	0.1026	**0.8115**	0.0859	Iris-versicolor
18	0.0004	**0.8611**	0.1385	0.0727	**0.7944**	0.1329	Iris-versicolor
19	0.0001	**0.9997**	0.0002	0.0725	**0.7942**	0.1334	Iris-versicolor
20	0.0004	**0.7380**	0.2615	0.0672	**0.7876**	0.1452	Iris-versicolor
21	0.0000	0.0001	**0.9999**	0.0015	0.3527	**0.6458**	Iris-virginica
22	0.0002	0.1910	**0.8088**	0.0015	0.3529	**0.6456**	Iris-virginica
23	0.0000	0.0000	**0.9999**	0.0021	0.3620	**0.6359**	Iris-virginica
24	0.0000	0.0000	**0.9999**	0.0015	0.3530	**0.6455**	Iris-virginica
25	0.0000	0.0001	**0.9999**	0.0015	0.3526	**0.6459**	Iris-virginica
26	0.0000	0.0159	**0.9841**	0.0015	0.3529	**0.6456**	Iris-virginica
27	0.0000	0.0007	**0.9993**	0.0019	0.3599	**0.6382**	Iris-virginica
28	0.0000	0.0031	**0.9969**	0.0017	0.3566	**0.6417**	Iris-virginica
29	0.0000	0.0001	**0.9999**	0.0016	0.3534	**0.6451**	Iris-virginica
30	0.0000	0.0000	**0.9999**	0.0032	0.3843	**0.6125**	Iris-virginica

In the table above python predictions don't match the WEKA predictions because I removed the wrong column in WEKA (index 3 - which is the 4th column). Theoretically predicted probabilities should match fully, because it doesn't matter in what environment the calculations are executed.