NewProject

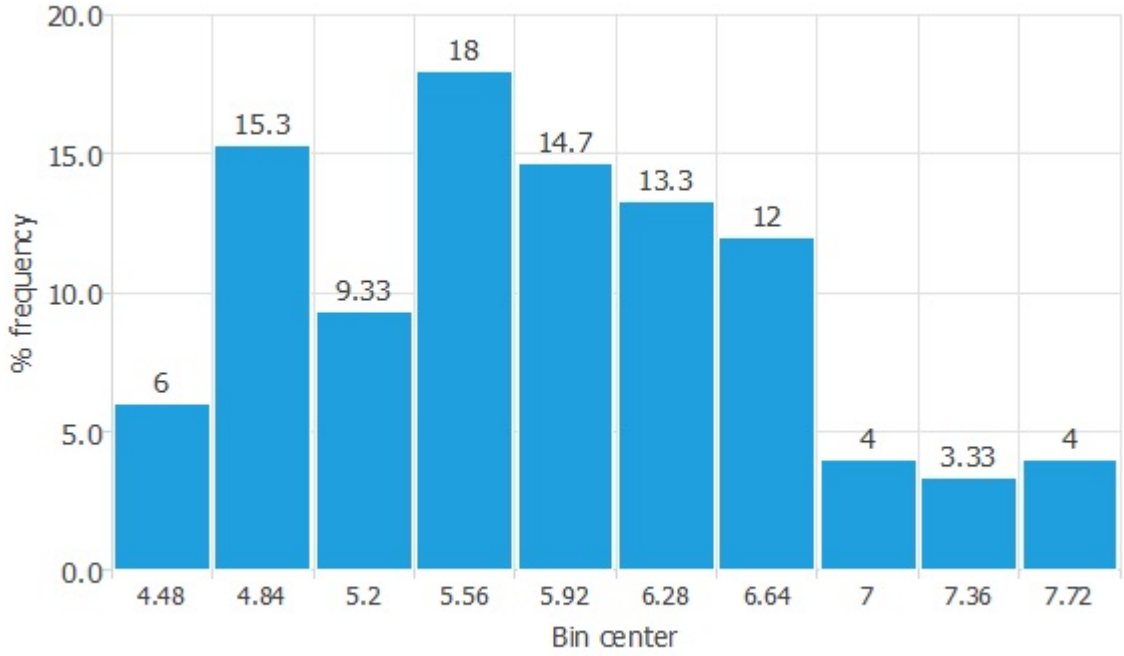
1 Data distribution

1.1 Task description

Histograms show how the data is distributed over its entire range.
In classification problems, a uniform distribution for all the variables is, in general, desirable.
If the data is very irregularly distributed, then the model will probably be of bad quality.

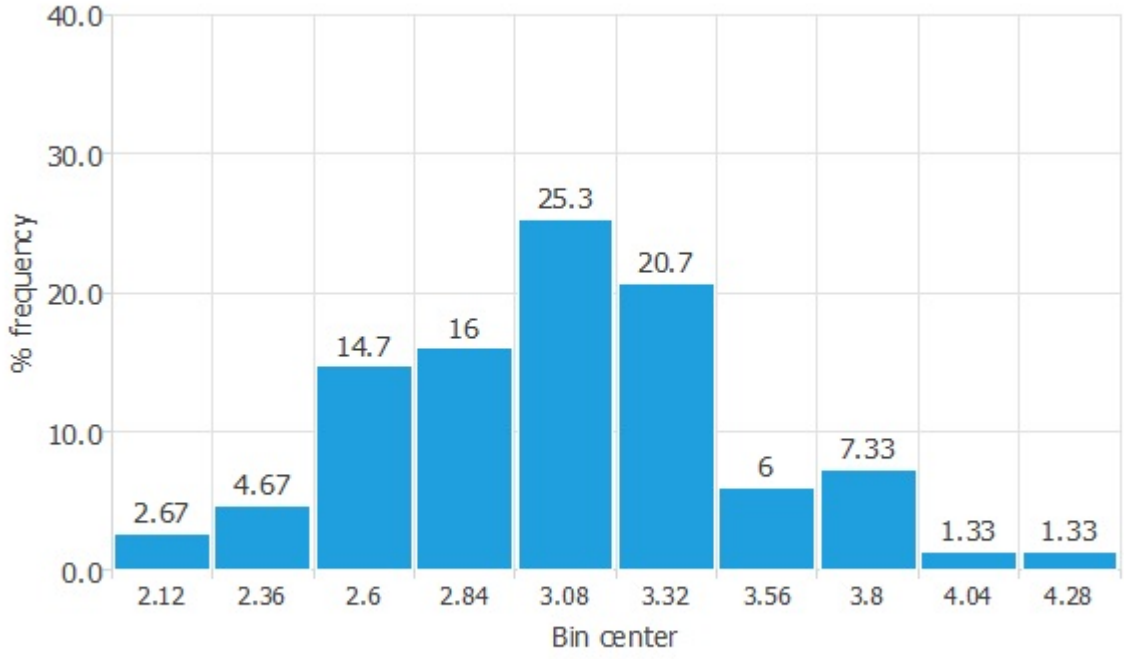
1.2 sepal_length distribution

The following chart shows the histogram for the variable sepal_length.
The abscissa represents the centers of the containers, and the ordinate their corresponding frequencies.
The maximum frequency is 18%, which corresponds to the bin with center 5.56.
The minimum frequency is 3.33333%, which corresponds to the bin with center 7.36.



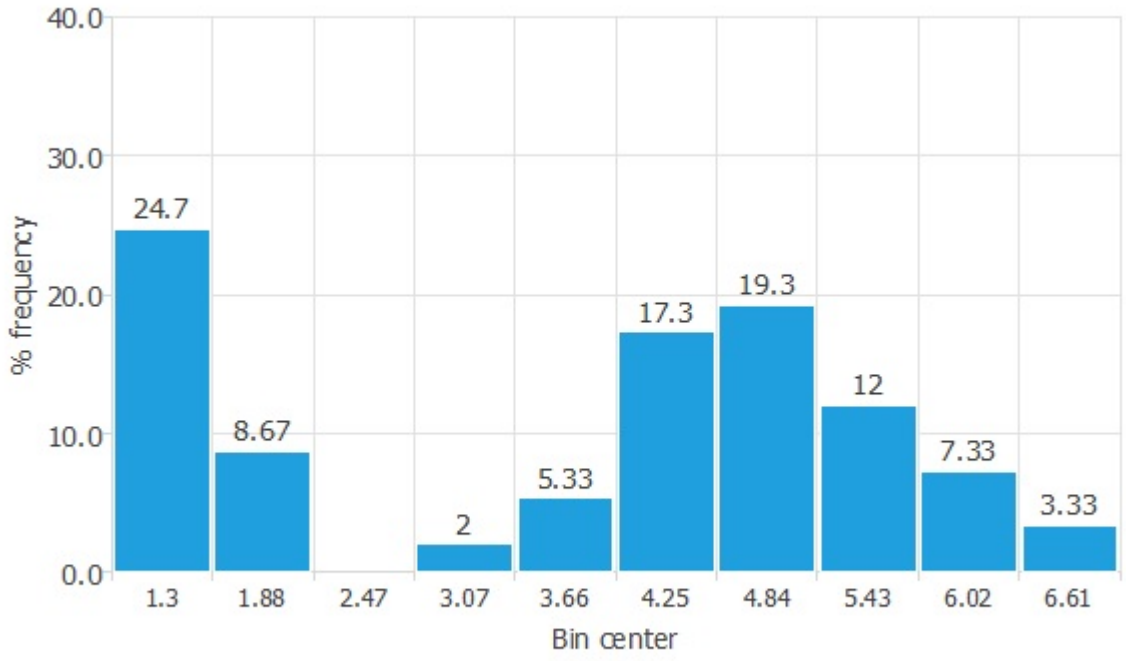
1.3 sepal_width distribution

The following chart shows the histogram for the variable sepal_width.
The abscissa represents the centers of the containers, and the ordinate their corresponding frequencies.
The maximum frequency is 25.3333%, which corresponds to the bin with center 3.08.
The minimum frequency is 1.33333%, which corresponds to the bins with centers 4.04, 4.28.



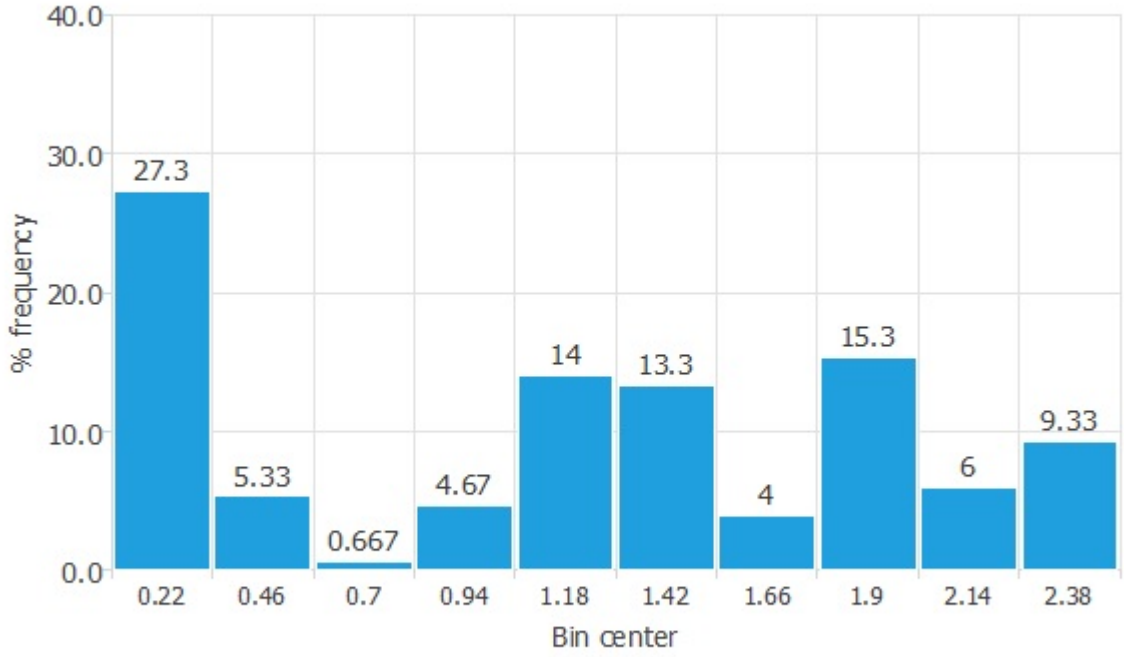
1.4 petal_length distribution

The following chart shows the histogram for the variable petal_length.
The abscissa represents the centers of the containers, and the ordinate their corresponding frequencies.
The maximum frequency is 24.6667%, which corresponds to the bin with center 1.3.
The minimum frequency is 0%, which corresponds to the bin with center 2.47.



1.5 petal_width distribution

The following chart shows the histogram for the variable petal_width.
The abscissa represents the centers of the containers, and the ordinate their corresponding frequencies.
The maximum frequency is 27.3333%, which corresponds to the bin with center 0.22.
The minimum frequency is 0.666667%, which corresponds to the bin with center 0.7.



1.6 class distribution pie chart

The following pie chart shows the distribution for the categorical variable class, which contains 3 variables: iris_setosa, iris_versicolor and iris_virginica.
The minimum frequency is 33.3333%, which corresponds to the categories iris_setosa, iris_setosa and iris_versicolor.
The maximum frequency is 33.3333%, which corresponds to the categories iris_setosa, iris_setosa and iris_versicolor.

2 Neural network

2.1 Task description

The neural network represents the predictive model. In Neural Designer neural networks allow deep architectures, which are a class of universal approximator.

2.2 Inputs

The number of inputs is 4.
The next table depicts the names of the inputs to the neural network.

	Name
1	sepal_length
2	sepal_width
3	petal_length
4	petal_width

2.3 Scaling layer

The size of the scaling layer is 4, the number of inputs. The following table shows the values which are used for scaling the inputs, which include the minimum, maximum, mean and standard deviation.

	Minimum	Maximum	Mean	Deviation	Scaler
sepal_length	4.3	7.9	5.84333	0.828066	MeanStandardDeviation
sepal_width	2	4.4	3.054	0.433594	MeanStandardDeviation
petal_length	1	6.9	3.75867	1.76442	MeanStandardDeviation
petal_width	0.1	2.5	1.19867	0.763161	MeanStandardDeviation

2.4 Perceptron layers

The number of perceptron layers in the neural network is 1. The following table depicts the size of each layer and its corresponding activation function.

	Inputs number	Neurons number	Activation function
1	4	3	HyperbolicTangent

2.5 Probabilistic layer

The following table depicts the size of the probabilistic layer the corresponding activation function.

	Inputs number	Neurons number	Activation function
1	3	3	Softmax

2.6 Outputs table

The number of outputs is 3. The next table depicts the name of the outputs from the neural network.

	Name
1	iris_setosa
2	iris_versicolor

3	iris_virginica

3 Training strategy

3.1 Training strategy

The procedure used to carry out the learning process is called training (or learning) strategy.
The training strategy is applied to the neural network in order to obtain the best possible loss.

3.2 Loss index

The loss index defines the task the neural network is requiered to do and provides a measure of the quality of the representation requiered to learn.
When setting a loss index, two different terms must be chosen: an error term and a regularization term.

The error term evaluates quantitatively how the neural network fits the data set. There are several error methods, and the choice of an appropriate one depends on the particular application. In this case, the Normalized Squared Error (MSE) is selected. The normalized squared error has a value of one when the outputs from the neural network are equal to the mean values of the target variables, while a value of zero means perfect prediction of the data.

The regularization term measures the values of the parameters in the neural network. Adding it to the error will cause the neural network to have smaller weights and biases, which will force its response to be smoother, i.e., to avoid overfitting. In this case, L2 regularization method is applied. It consists of the squared sum of all the parameters in the neural network.

3.3 Optimization algorithm

The quasi-Newton method is used here as optimization algorithm.
It is based on Newton's method, but does not require calculation of second derivatives.
Instead, the quasi-Newton method computes an approximation of the inverse Hessian at each iteration of the algorithm, by only using gradient information.

	Description	Value
Inverse hessian approximation method	Method used to obtain a suitable training rate.	BFGS
Learning rate method	Method used to calculate the step for the quasi-Newton training direction.	BrentMethod
Learning rate tolerance	Maximum interval length for the learning rate.	0.001000
Minimum loss decrease	Minimum loss improvement between two successive epochs.	0.000000
Loss goal	Goal value for the loss.	0.001000
Maximum selection error increases	Goal value for the norm of the objective function gradient.	100
Maximum epochs number	Maximum number of epochs at which the selection error increases.	1000

Maximum time	Maximum number of epochs to perform the training.	01:00:00

4 Training

4.1 Task description

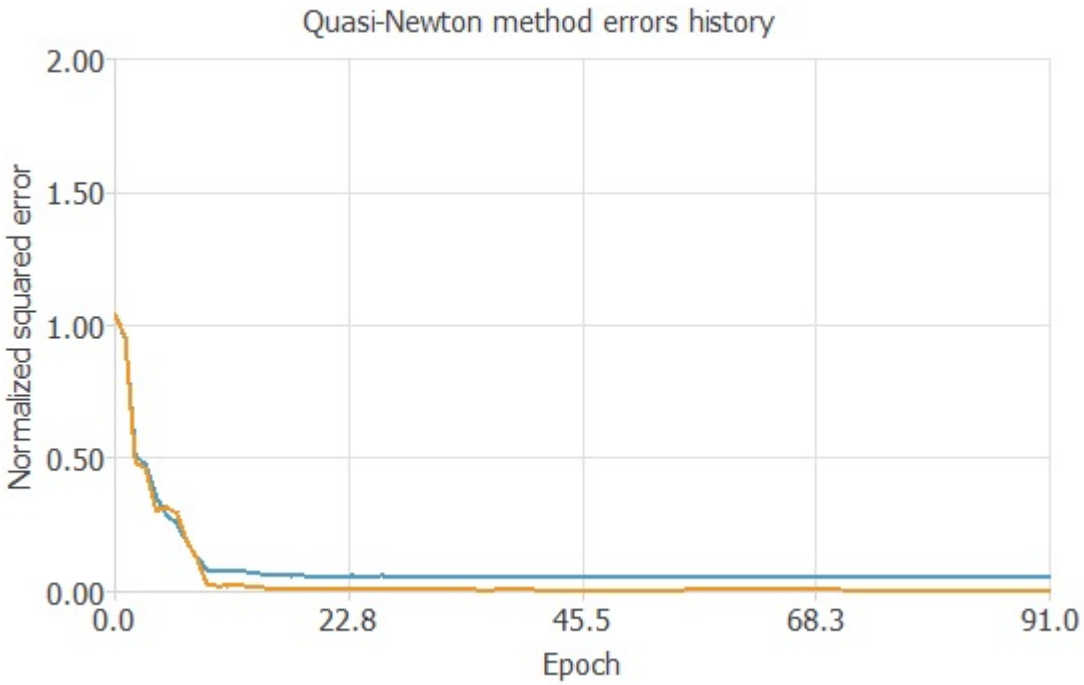
The procedure used to carry out the learning process is called training (or learning) strategy.
The training strategy is applied to the neural network in order to obtain the best possible loss.
The type of training is determined by the way in which the adjustment of the parameters in the neural network takes place.

4.2 Optimization algorithm

The quasi-Newton method is used here for training.
It is based on Newton's method, but does not require calculation of second derivatives.
Instead, the quasi-Newton method computes an approximation of the inverse Hessian at each iteration of the algorithm, by only using gradient information.

4.3 Quasi-Newton method errors history

The following plot shows the training and selection errors in each iteration.
The blue line represents the training error and the orange line represents the selection error.
The initial value of the training error is 1.04195, and the final value after 91 epochs is 0.0547881.
The initial value of the selection error is 1.03602, and the final value after 91 epochs is 0.00435.



4.4 Quasi-Newton method results

The next table shows the training results by the quasi-Newton method.
They include some final states from the neural network, the loss index and the optimization algorithm.

	Value
Training error	0.0548
Selection error	0.00435
Epochs number	91
Elapsed time	00:00:00
Stopping criterion	Minimum loss decrease

5 Order selection

5.1 Task description

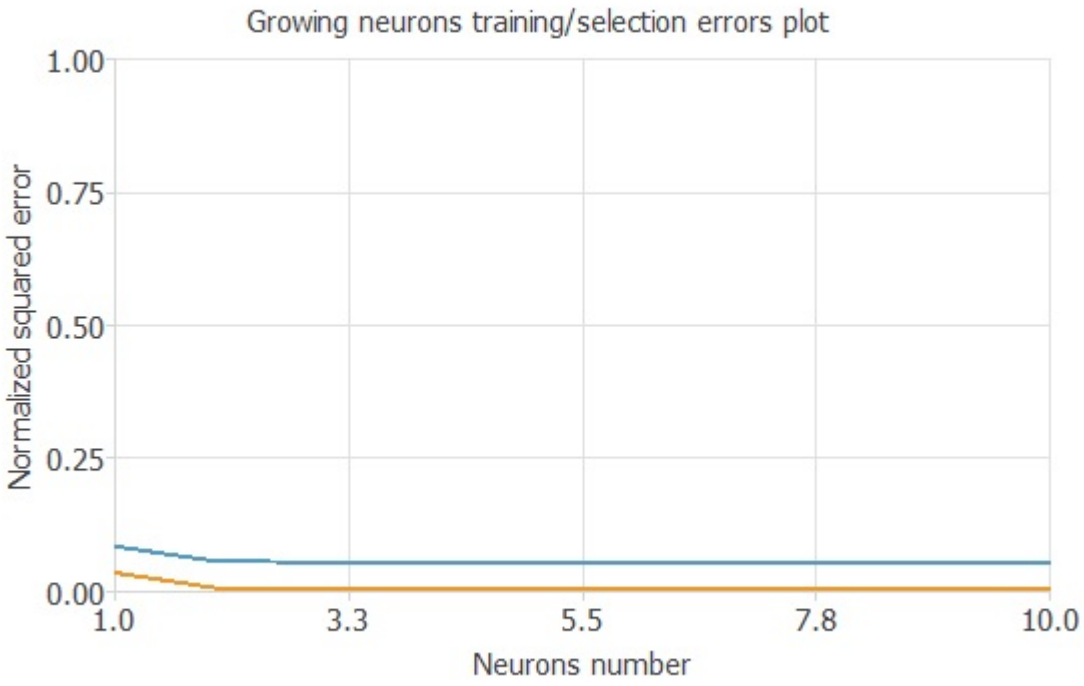
The best selection is achieved by using a model whose complexity is the most appropriate to produce an adequate fit of the data.
The neurons selection algorithm is responsible of finding the optimal number of neurons in the network.

5.2 Neurons selection algorithm

The growing neurons algorithm is used here to select the optimal order in the neural network.

5.3 Growing neurons training/selection errors plot

The next chart shows the error history for the different subsets during the growing neurons selection process.
The blue line represents the training error and the yellow line symbolizes the selection error.



5.4 Growing neurons results

The next table shows the neurons selection results by the growing neurons algorithm.
They include some final states from the neural network, the error functional and the neurons selection algorithm.

	Value
Optimal order	9
Optimum training error	0.0527869
Optimum selection error	0.00366208
Epochs number	10
Elapsed time	00:00:02

6 Confusion

6.1 Task description

In the confusion matrix the rows represent the target classes and the columns the output classes for the testing target data set. The diagonal cells in each table show the number of cases that were correctly classified, and the off-diagonal cells show the misclassified cases.

6.2 Confusion table

The following table contains the elements of the confusion matrix.
The total number of testing samples is 30.

The number of correctly classified samples is 28 (93.3%), and the number of misclassified samples is 2 (6.7%).

	Predicted iris_setosa	Predicted iris_versicolor	Predicted iris_virginica
Real iris_setosa	11 (36.7%)	0	0
Real iris_versicolor	0	8 (26.7%)	1 (3.3%)
Real iris_virginica	0	1 (3.3%)	9 (30.0%)

7 Training

7.1 Task description

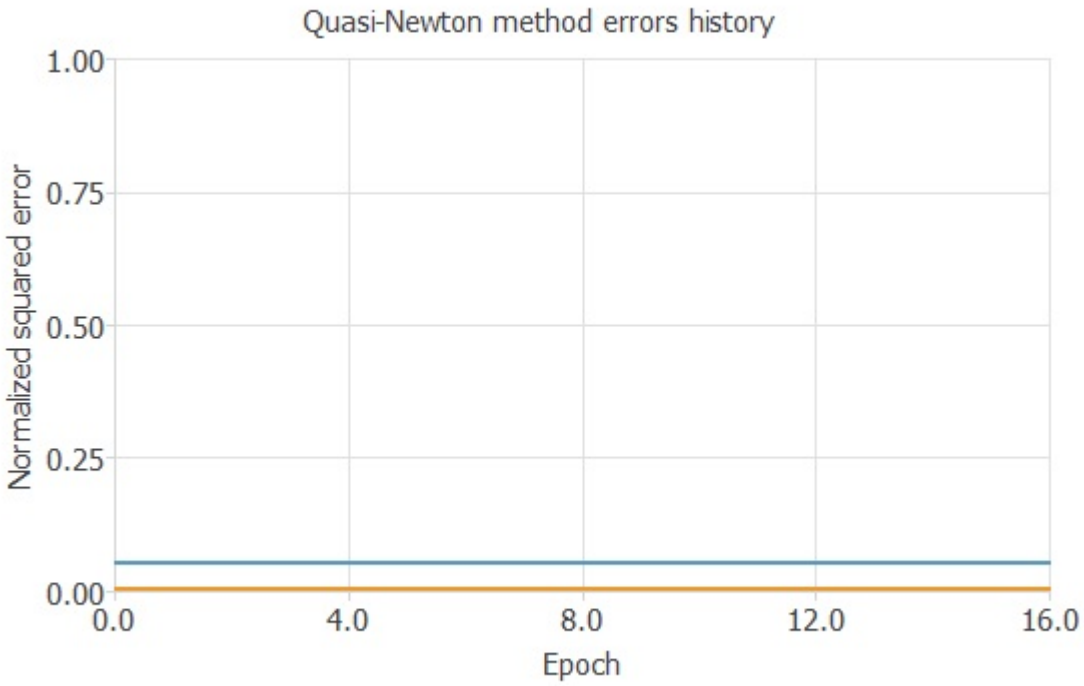
The procedure used to carry out the learning process is called training (or learning) strategy.
The training strategy is applied to the neural network in order to obtain the best possible loss.
The type of training is determined by the way in which the adjustment of the parameters in the neural network takes place.

7.2 Optimization algorithm

The quasi-Newton method is used here for training.
It is based on Newton's method, but does not require calculation of second derivatives.
Instead, the quasi-Newton method computes an approximation of the inverse Hessian at each iteration of the algorithm, by only using gradient information.

7.3 Quasi-Newton method errors history

The following plot shows the training and selection errors in each iteration.
The blue line represents the training error and the orange line represents the selection error.
The initial value of the training error is 0.0527869, and the final value after 16 epochs is 0.0528395.
The initial value of the selection error is 0.00366207, and the final value after 16 epochs is 0.00367739.



7.4 Quasi-Newton method results

The next table shows the training results by the quasi-Newton method.
They include some final states from the neural network, the loss index and the optimization algorithm.

	Value
Training error	0.0528
Selection error	0.00368
Epochs number	16
Elapsed time	00:00:00
Stopping criterion	Minimum loss decrease

8 Order selection

8.1 Task description

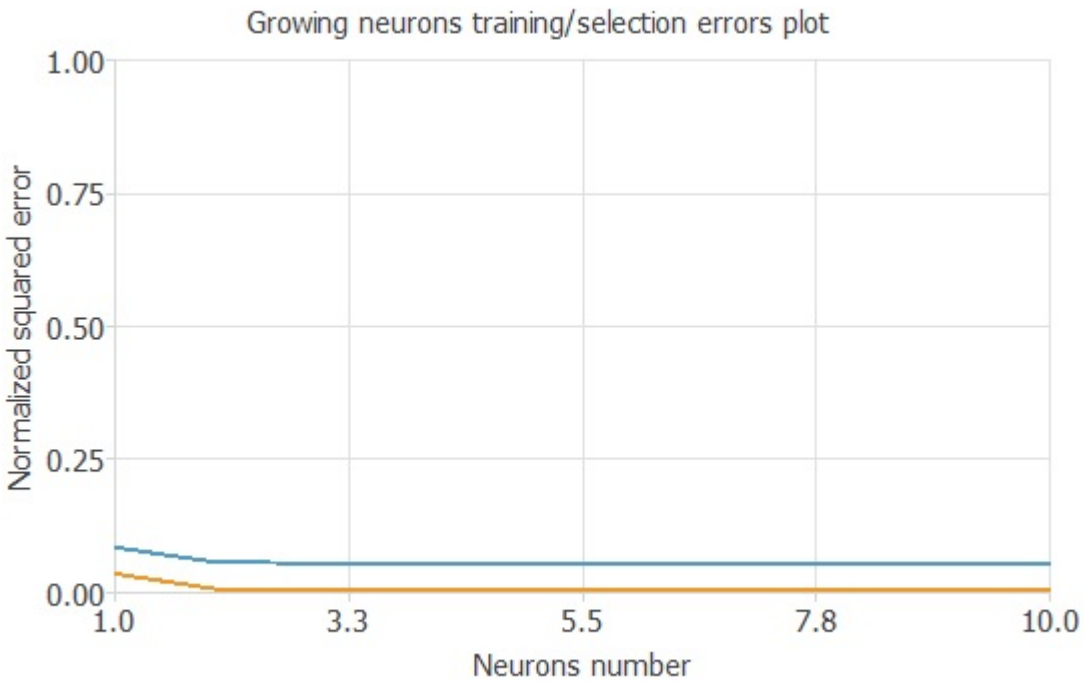
The best selection is achieved by using a model whose complexity is the most appropriate to produce an adequate fit of the data.
The neurons selection algorithm is responsible of finding the optimal number of neurons in the network.

8.2 Neurons selection algorithm

The growing neurons algorithm is used here to select the optimal order in the neural network.

8.3 Growing neurons training/selection errors plot

The next chart shows the error history for the different subsets during the growing neurons selection process.
The blue line represents the training error and the yellow line symbolizes the selection error.



8.4 Growing neurons results

The next table shows the neurons selection results by the growing neurons algorithm.
They include some final states from the neural network, the error functional and the neurons selection algorithm.

	Value
Optimal order	9
Optimum training error	0.0524035
Optimum selection error	0.00352931
Epochs number	10
Elapsed time	00:00:01

9 Confusion

9.1 Task description

In the confusion matrix the rows represent the target classes and the columns the output classes for the testing target data set. The diagonal cells in each table show the number of cases that were correctly classified, and the off-diagonal cells show the misclassified cases.

9.2 Confusion table

The following table contains the elements of the confusion matrix.
The total number of testing samples is 30.
The number of correctly classified samples is 28 (93.3%), and the number of misclassified samples is 2 (6.7%).

	Predicted iris_setosa	Predicted iris_versicolor	Predicted iris_virginica
Real iris_setosa	11 (36.7%)	0	0
Real iris_versicolor	0	8 (26.7%)	1 (3.3%)
Real iris_virginica	0	1 (3.3%)	9 (30.0%)

10 Neural network outputs

10.1 Task description

A neural network produces a set of outputs for each set of inputs applied. The outputs depend, in turn, on the values of the parameters.

10.2 Inputs-outputs table

The next table shows the input values and their corresponding output values. The input variables are sepal_length, sepal_width, petal_length and petal_width; and the output variables are iris_setosa, iris_versicolor and iris_virginica.

	Value
sepal_length	5.84332991
sepal_width	3.0539999
petal_length	3.75867009
petal_width	1.19867003
iris_setosa	0.0485592075
iris_versicolor	0.950197458
iris_virginica	0.00124332926

11 Python expression

11.1 Task description

The predictive model takes the form of a function of the outputs with respect to the inputs. The mathematical expression represented by the model can be exported to different programming languages, in the so called production mode. The Python programming language expression has been saved in the following file: F:/ML/Task1/model.py

12 Directional output

12.1 Task description

It is very useful to see the how the outputs vary as a function of a single input, when all the others are fixed.
This can be seen as the cut of the neural network model along some input direction and through some reference point.

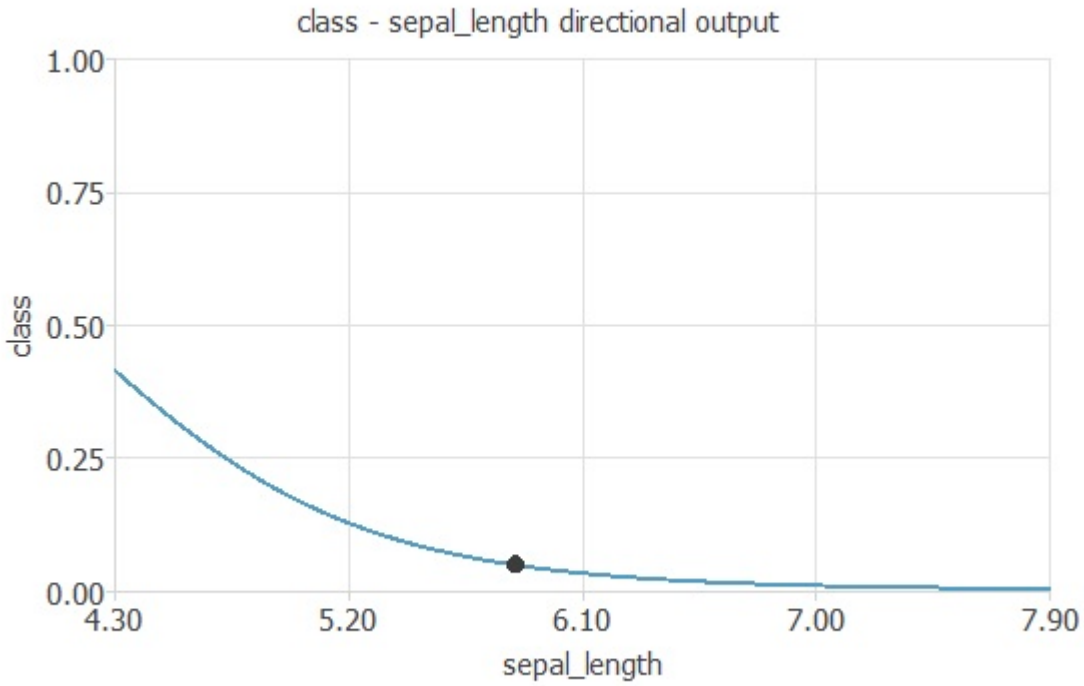
12.2 Reference point table

The next table shows the reference point for the plots.

	Value
sepal_length	5.84
sepal_width	3.05
petal_length	3.76
petal_width	1.2

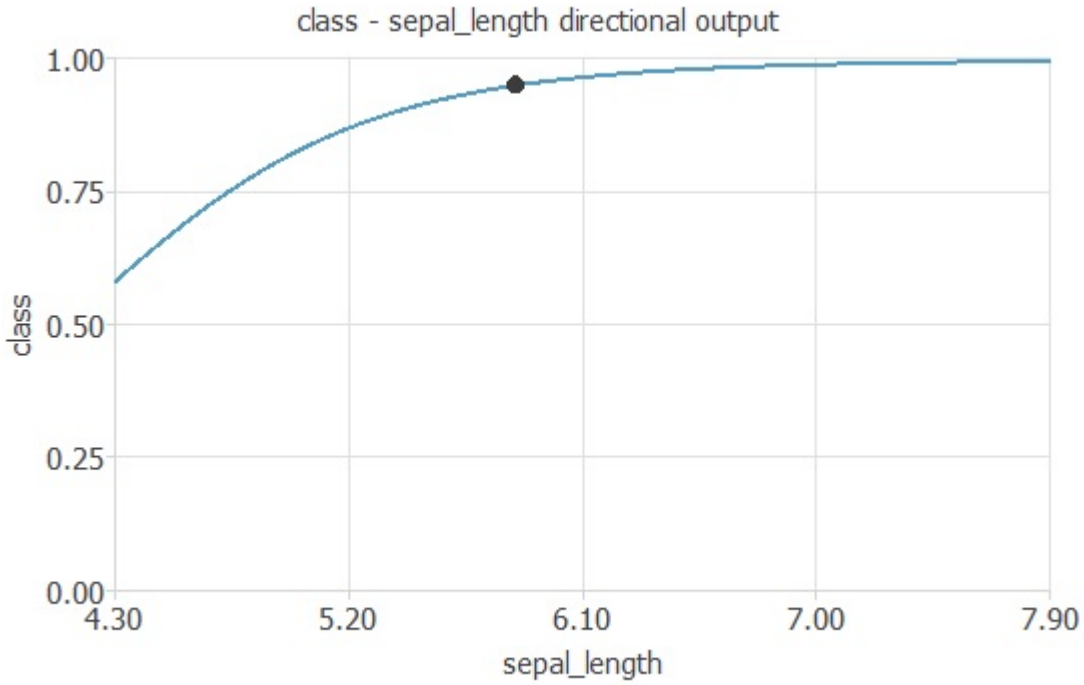
12.3 class - sepal_length directional output

The next plot shows the output class as a function of the input sepal_length. The x and y axes are defined by the range of the variables sepal_length and class, respectively. The gray point represents the reference point. Note that some directional outputs fall outside the range [0, 1], and therefore they are not plotted.



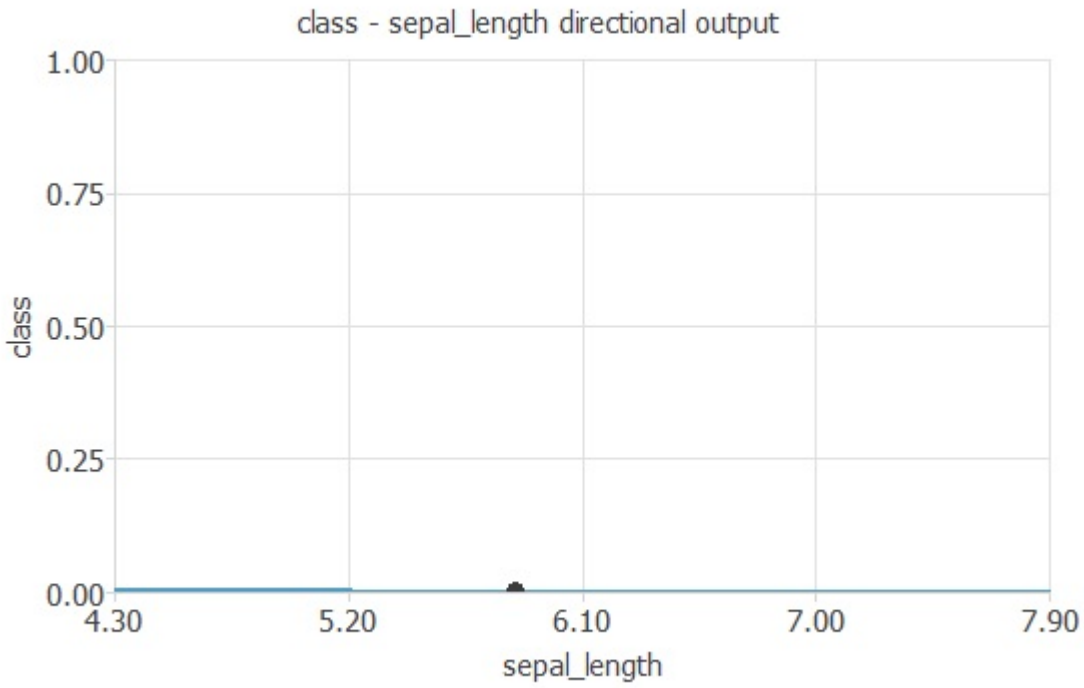
12.4 class - sepal_length directional output

The next plot shows the output class as a function of the input sepal_length. The x and y axes are defined by the range of the variables sepal_length and class, respectively. The gray point represents the reference point. Note that some directional outputs fall outside the range [0, 1], and therefore they are not plotted.



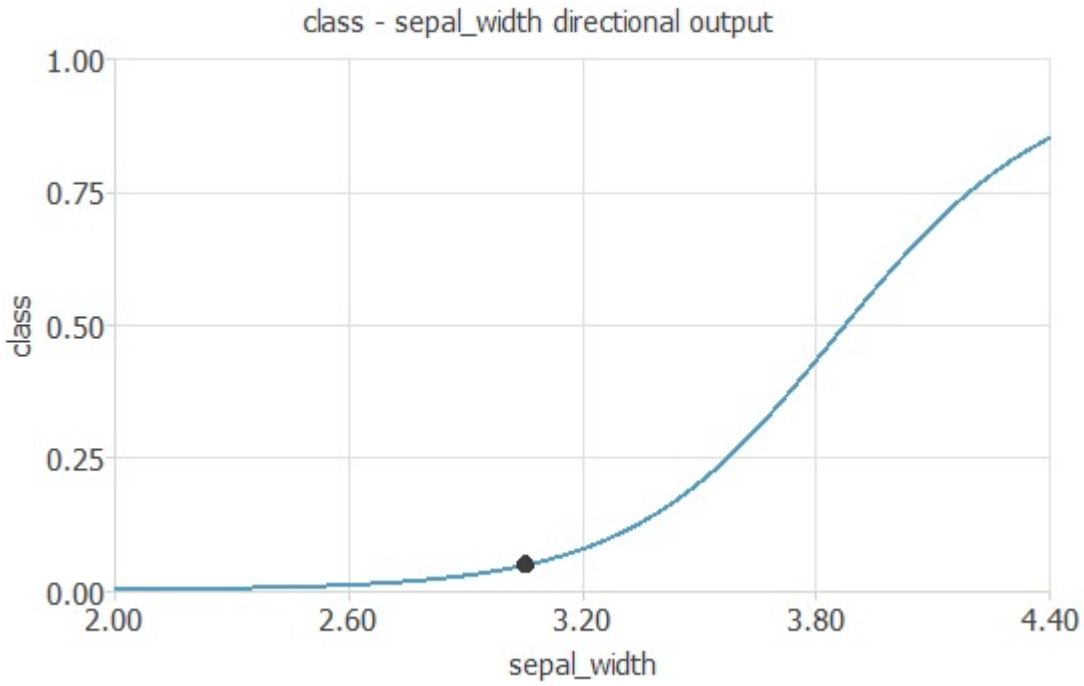
12.5 class - sepal_length directional output

The next plot shows the output class as a function of the input sepal_length. The x and y axes are defined by the range of the variables sepal_length and class, respectively. The gray point represents the reference point. Note that some directional outputs fall outside the range [0, 1], and therefore they are not plotted.



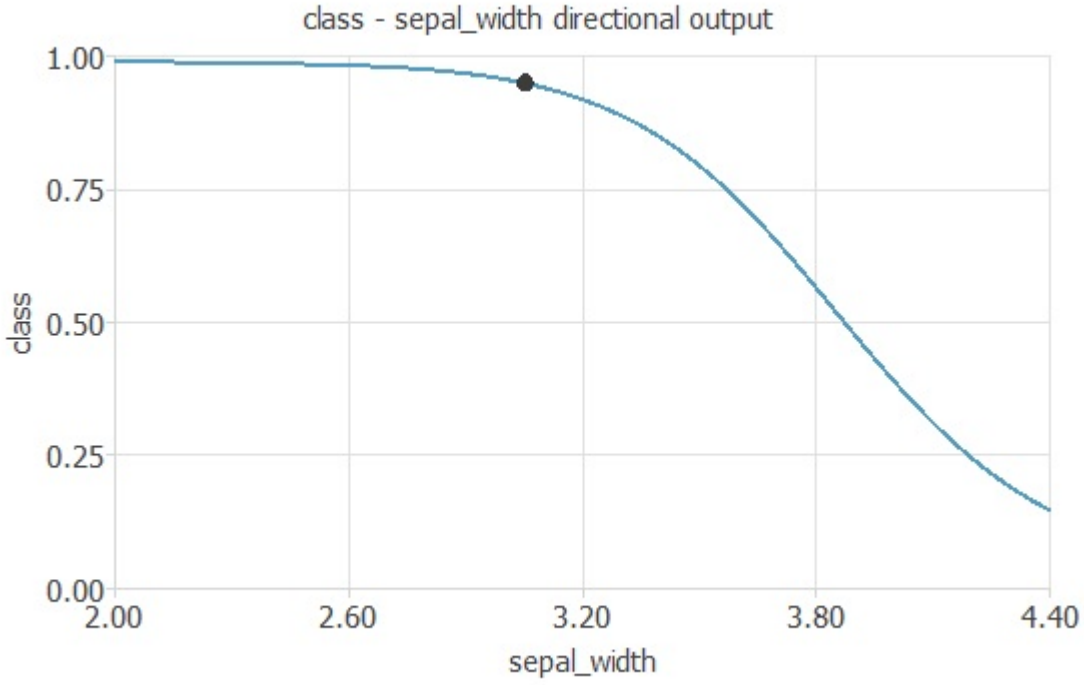
12.6 class - sepal_width directional output

The next plot shows the output class as a function of the input sepal_width. The x and y axes are defined by the range of the variables sepal_width and class, respectively. The gray point represents the reference point. Note that some directional outputs fall outside the range [0, 1], and therefore they are not plotted.



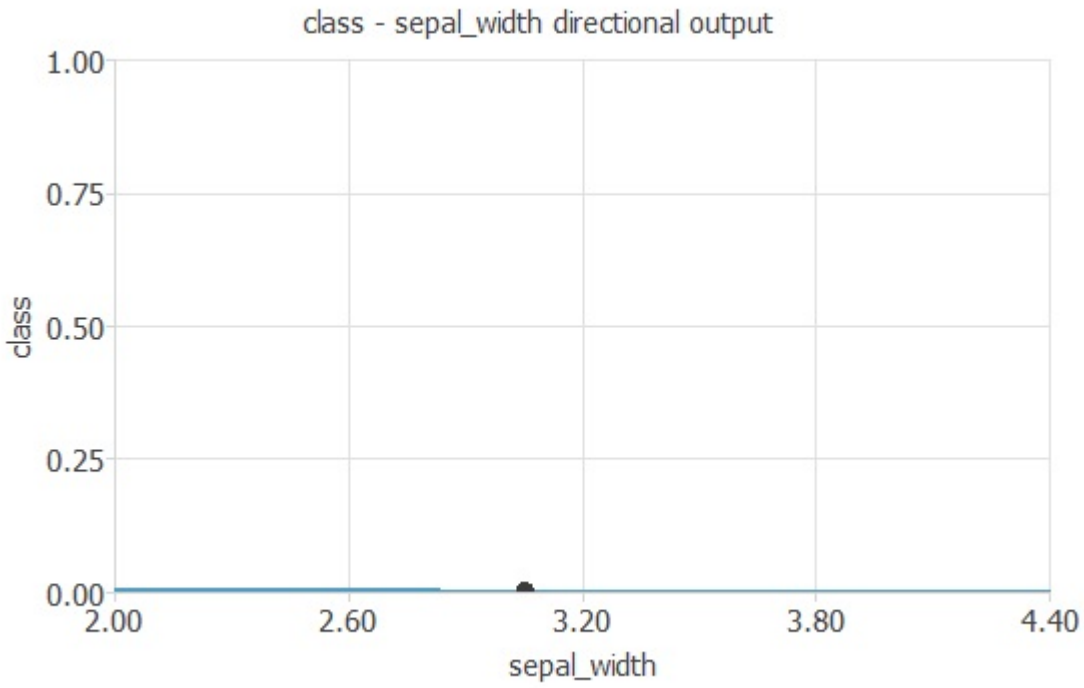
12.7 class - sepal_width directional output

The next plot shows the output class as a function of the input sepal_width. The x and y axes are defined by the range of the variables sepal_width and class, respectively. The gray point represents the reference point. Note that some directional outputs fall outside the range [0, 1], and therefore they are not plotted.



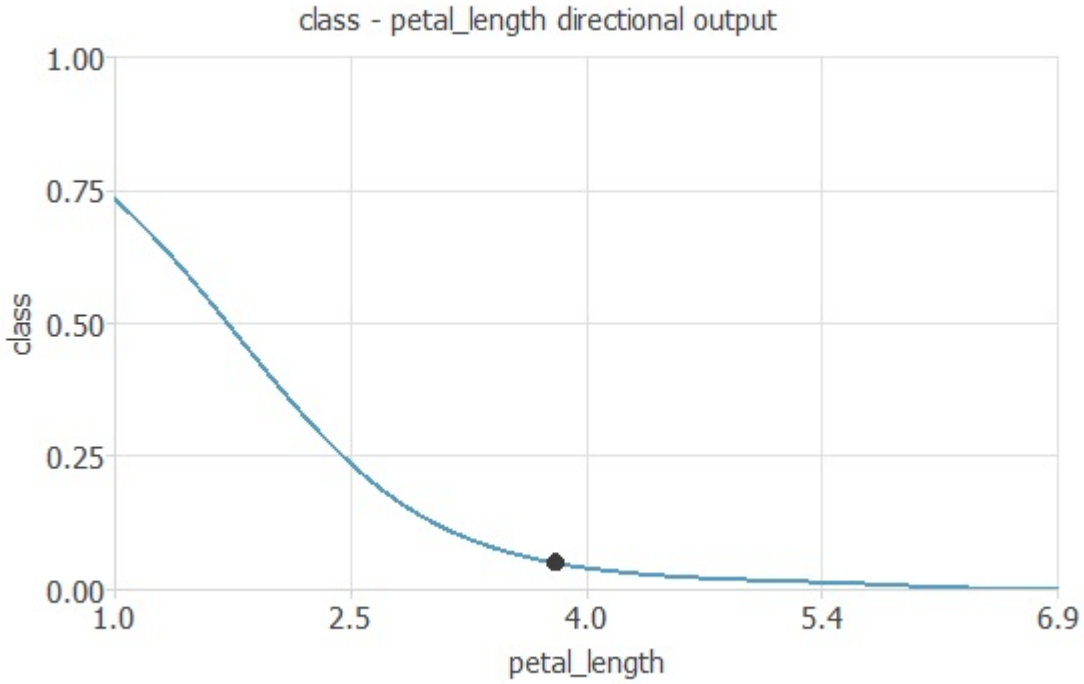
12.8 class - sepal_width directional output

The next plot shows the output class as a function of the input sepal_width. The x and y axes are defined by the range of the variables sepal_width and class, respectively. The gray point represents the reference point. Note that some directional outputs fall outside the range [0, 1], and therefore they are not plotted.



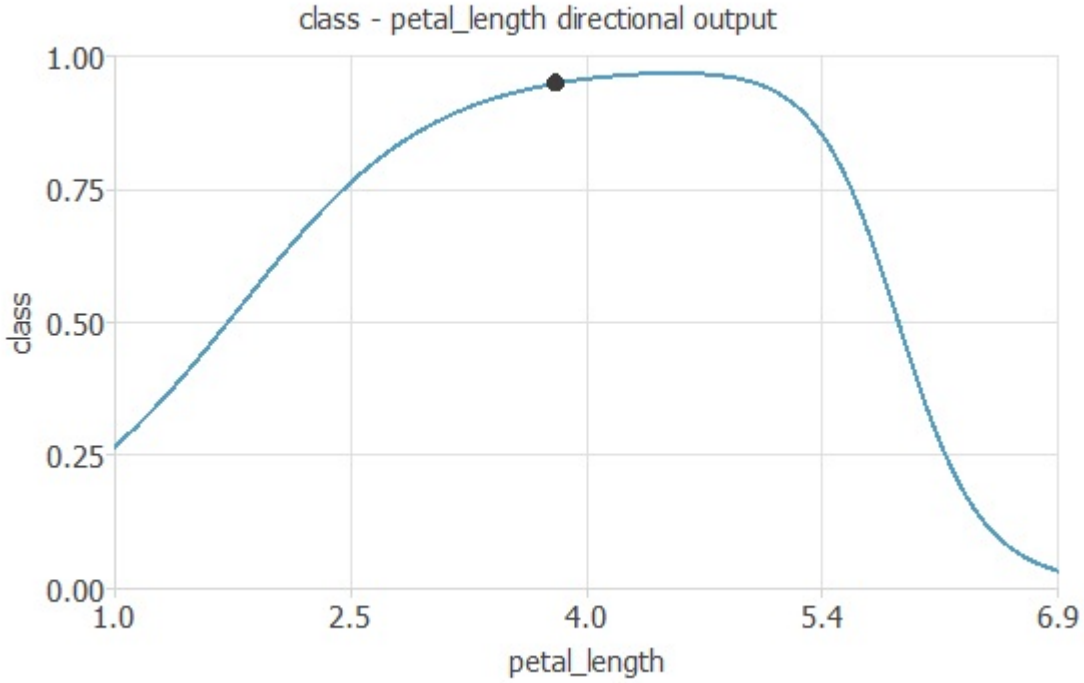
12.9 class - petal_length directional output

The next plot shows the output class as a function of the input petal_length. The x and y axes are defined by the range of the variables petal_length and class, respectively. The gray point represents the reference point. Note that some directional outputs fall outside the range [0, 1], and therefore they are not plotted.



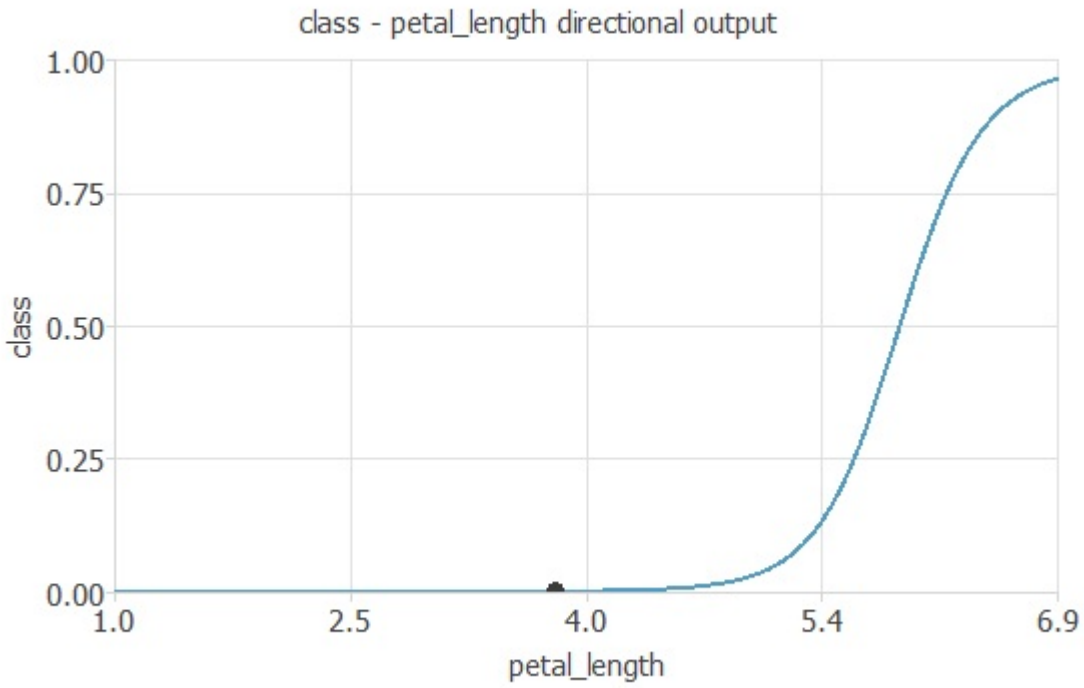
12.10 class - petal_length directional output

The next plot shows the output class as a function of the input petal_length. The x and y axes are defined by the range of the variables petal_length and class, respectively. The gray point represents the reference point. Note that some directional outputs fall outside the range [0, 1], and therefore they are not plotted.



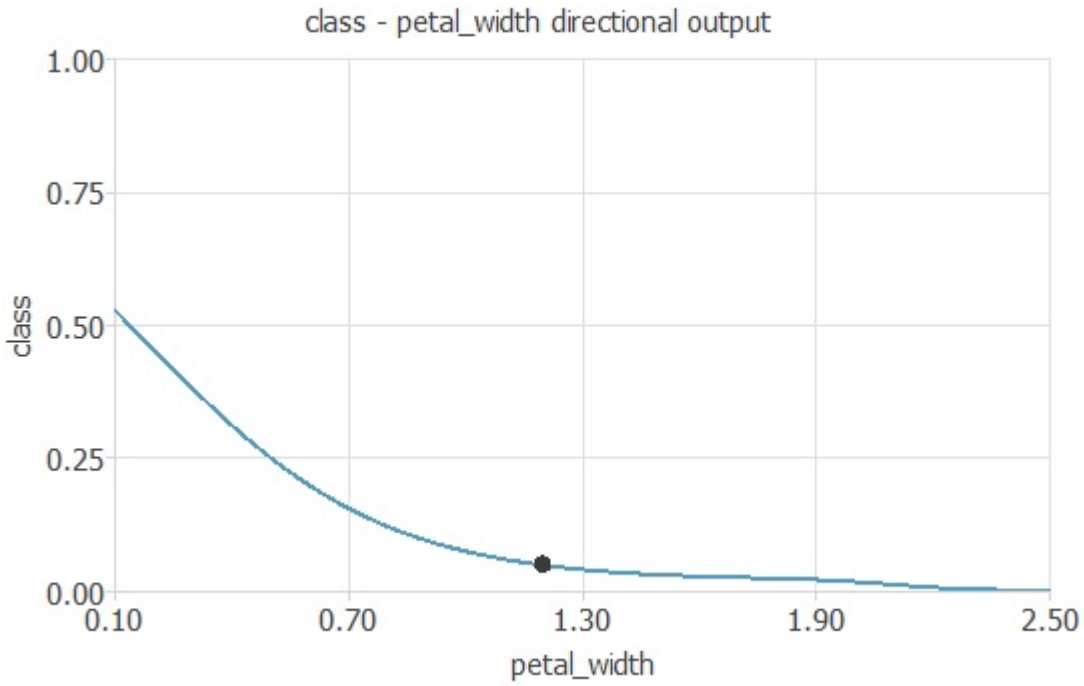
12.11 class - petal_length directional output

The next plot shows the output class as a function of the input petal_length. The x and y axes are defined by the range of the variables petal_length and class, respectively. The gray point represents the reference point. Note that some directional outputs fall outside the range [0, 1], and therefore they are not plotted.



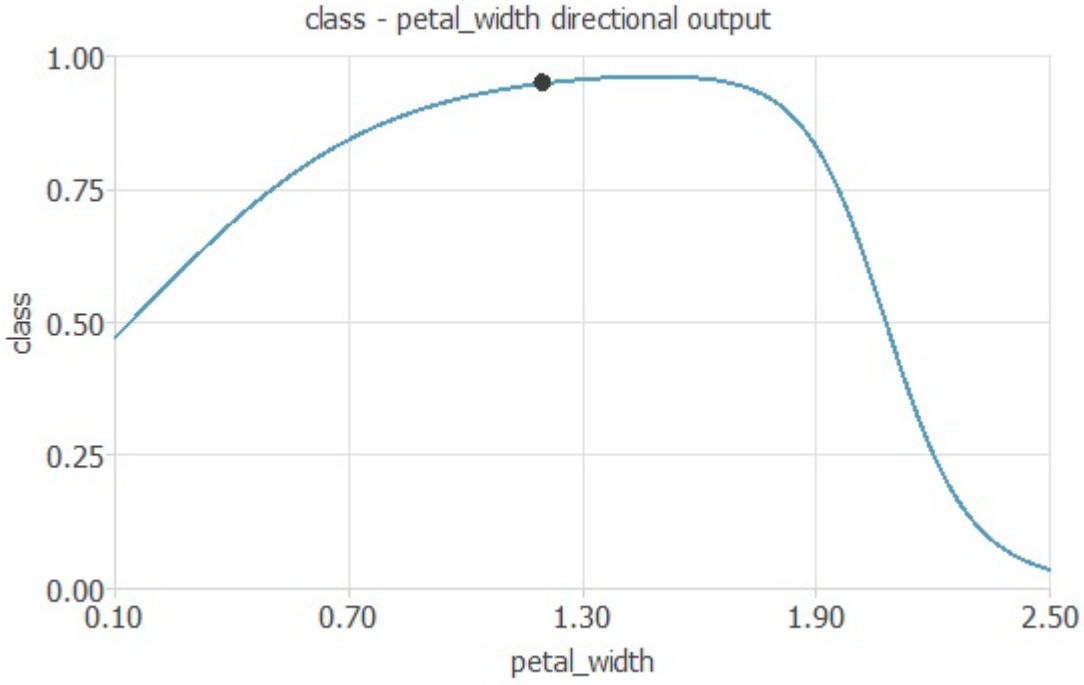
12.12 class - petal_width directional output

The next plot shows the output class as a function of the input petal_width. The x and y axes are defined by the range of the variables petal_width and class, respectively. The gray point represents the reference point. Note that some directional outputs fall outside the range [0, 1], and therefore they are not plotted.



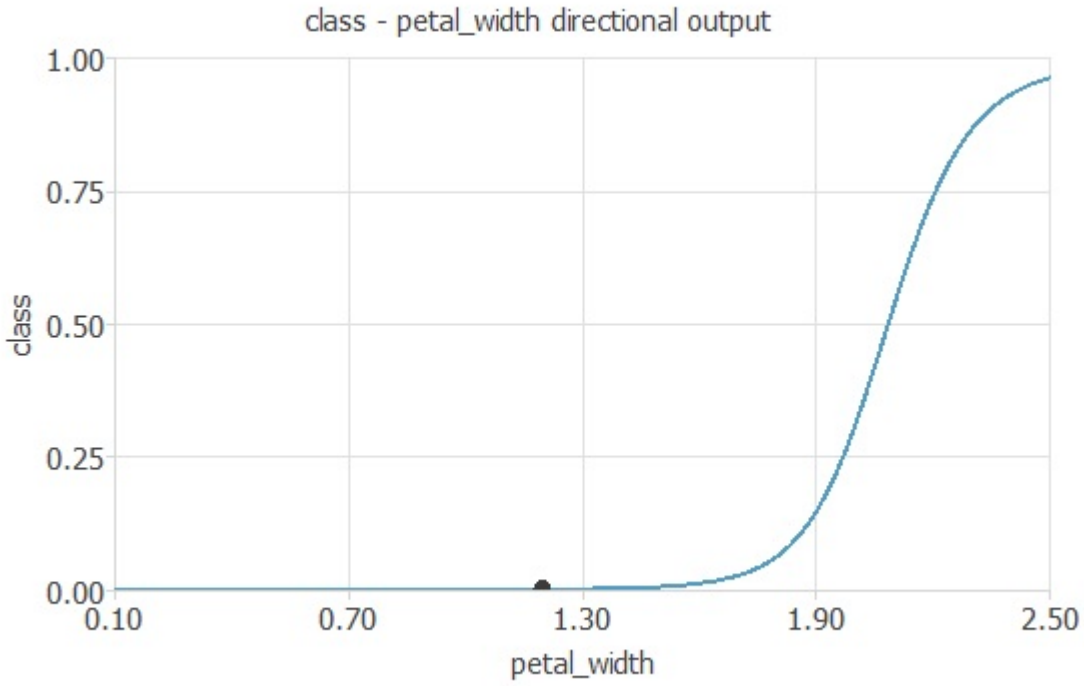
12.13 class - petal_width directional output

The next plot shows the output class as a function of the input petal_width. The x and y axes are defined by the range of the variables petal_width and class, respectively. The gray point represents the reference point. Note that some directional outputs fall outside the range [0, 1], and therefore they are not plotted.



12.14 class - petal_width directional output

The next plot shows the output class as a function of the input petal_width. The x and y axes are defined by the range of the variables petal_width and class, respectively. The gray point represents the reference point. Note that some directional outputs fall outside the range [0, 1], and therefore they are not plotted.



13 Mathematical expression

13.1 Task description

The predictive model takes the form of a function of the outputs with respect to the inputs. The mathematical expression represented by the model can be used to embed it into another software, in the so called production mode.

13.2 Expression

The mathematical expression represented by the neural network is written below. It takes the inputs sepal_length, sepal_width, petal_length, petal_width, to produce the outputs iris_setosa, iris_versicolor, iris_virginica,
In classification models, the information is propagated in a feed-forward fashion through the scaling layer, the perceptron layers and the probabilistic layer.

```
scaled_sepal_length = (sepal_length-5.843329906)/0.8280659914;
scaled_sepal_width = (sepal_width-3.053999901)/0.4335939884;
scaled_petal_length = (petal_length-3.758670092)/1.764420033;
scaled_ petal_width = ( petal_width-1.19867003)/0.7631610036;
```

```
perceptron_layer_1_output_00 = tanh( 0.311149 +
(scaled_sepal_length*0.212714) + (scaled_sepal_width*-0.
246406) + (scaled_petal_length*0.451339) + (scaled_
petal_width*0.383649) );
perceptron_layer_1_output_11 = tanh( 1.04875 +
(scaled_sepal_length*0.187369) + (scaled_sepal_width*-0.
0618301) + (scaled_petal_length*-0.887339) + (scaled_
petal_width*-1.04888) );
perceptron_layer_1_output_22 = tanh( 0.311996 +
(scaled_sepal_length*0.20603) + (scaled_sepal_width*-0.244952)
+ (scaled_petal_length*0.453892) + (scaled_
petal_width*0.393238) );
perceptron_layer_1_output_33 = tanh( -0.531124 +
(scaled_sepal_length*-0.0526154) + (scaled_sepal_width*-1.
04609) + (scaled_petal_length*0.854773) + (scaled_
petal_width*0.351947) );
perceptron_layer_1_output_44 = tanh( -0.32101 +
(scaled_sepal_length*-0.22425) + (scaled_sepal_width*0.242037)
+ (scaled_petal_length*-0.469606) + (scaled_
petal_width*-0.382011) );
perceptron_layer_1_output_55 = tanh( 0.294097 +
(scaled_sepal_length*0.193946) + (scaled_sepal_width*-0.25412)
+ (scaled_petal_length*0.448971) + (scaled_
petal_width*0.400635) );
perceptron_layer_1_output_66 = tanh( 0.324435 +
(scaled_sepal_length*0.208327) + (scaled_sepal_width*-0.
245706) + (scaled_petal_length*0.472093) + (scaled_
petal_width*0.392537) );
perceptron_layer_1_output_77 = tanh( 0.965306 +
(scaled_sepal_length*0.170135) + (scaled_sepal_width*-0.
071614) + (scaled_petal_length*-0.830352) + (scaled_
petal_width*-0.986609) );
perceptron_layer_1_output_88 = tanh( 1.00378 +
(scaled_sepal_length*0.178492) + (scaled_sepal_width*-0.
0677249) + (scaled_petal_length*-0.857134) + (scaled_
petal_width*-1.01536) );

probabilistic_layer_combinations_0 = 0.0939058
-0.739234*perceptron_layer_1_output_0
+0.411133*perceptron_layer_1_output_1
-0.744149*perceptron_layer_1_output_2
-0.624206*perceptron_layer_1_output_3
+0.739931*perceptron_layer_1_output_4
-0.748885*perceptron_layer_1_output_5
-0.752674*perceptron_layer_1_output_6
+0.427613*perceptron_layer_1_output_7
+0.419404*perceptron_layer_1_output_8
probabilistic_layer_combinations_1 = 0.0168138
+0.498375*perceptron_layer_1_output_0
+0.998832*perceptron_layer_1_output_1
+0.496825*perceptron_layer_1_output_2
-0.519348*perceptron_layer_1_output_3
```

```
-0.507961*perceptron_layer_1_output_4
+0.483064*perceptron_layer_1_output_5
+0.519898*perceptron_layer_1_output_6
+0.89796*perceptron_layer_1_output_7
+0.949883*perceptron_layer_1_output_8
probabilistic_layer_combinations_2 = -0.110662
+0.240458*perceptron_layer_1_output_0
-1.41055*perceptron_layer_1_output_1
+0.246733*perceptron_layer_1_output_2
+1.14343*perceptron_layer_1_output_3
-0.231997*perceptron_layer_1_output_4
+0.264322*perceptron_layer_1_output_5
+0.231996*perceptron_layer_1_output_6
-1.32438*perceptron_layer_1_output_7
-1.36846*perceptron_layer_1_output_8

        sum_ = exp(probabilistic_layer_combinations_0 +
exp(probabilistic_layer_combinations_1 +
exp(probabilistic_layer_combinations_2;

        iris_setosa = exp(probabilistic_layer_combinations_0)/
sum_;
        iris_versicolor =
exp(probabilistic_layer_combinations_1)/sum_;
        iris_virginica =
exp(probabilistic_layer_combinations_2)/sum_;
```