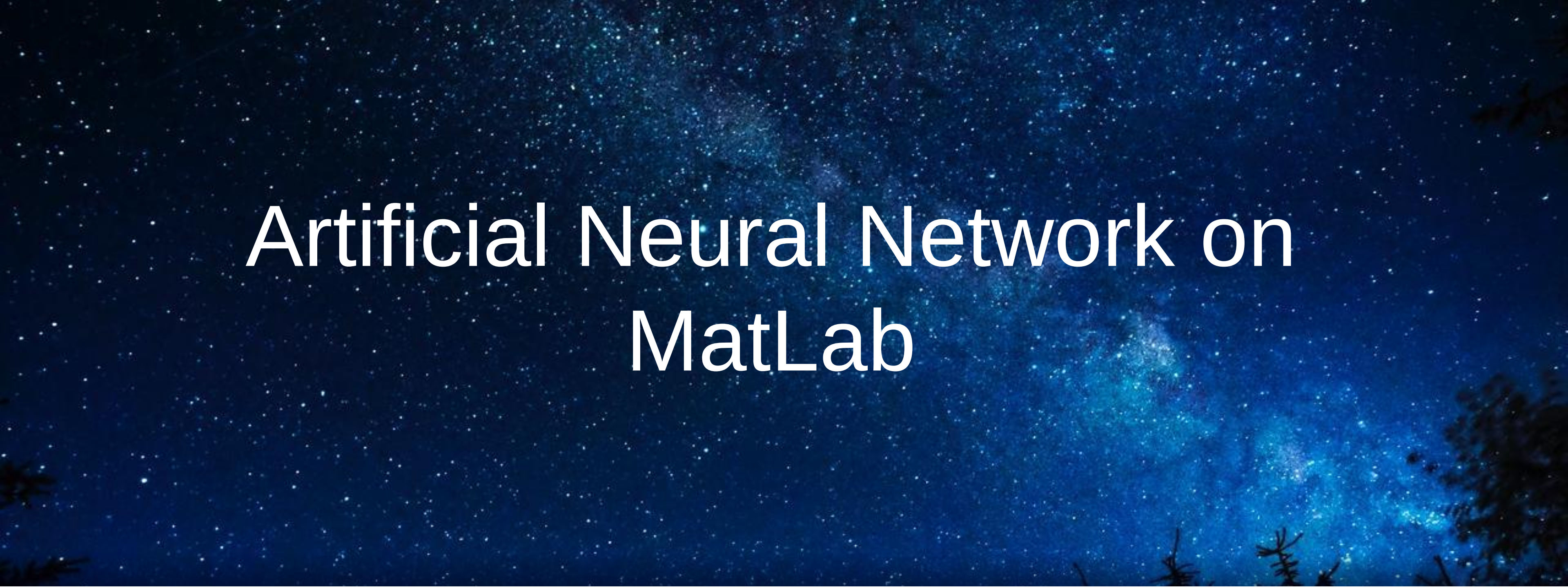
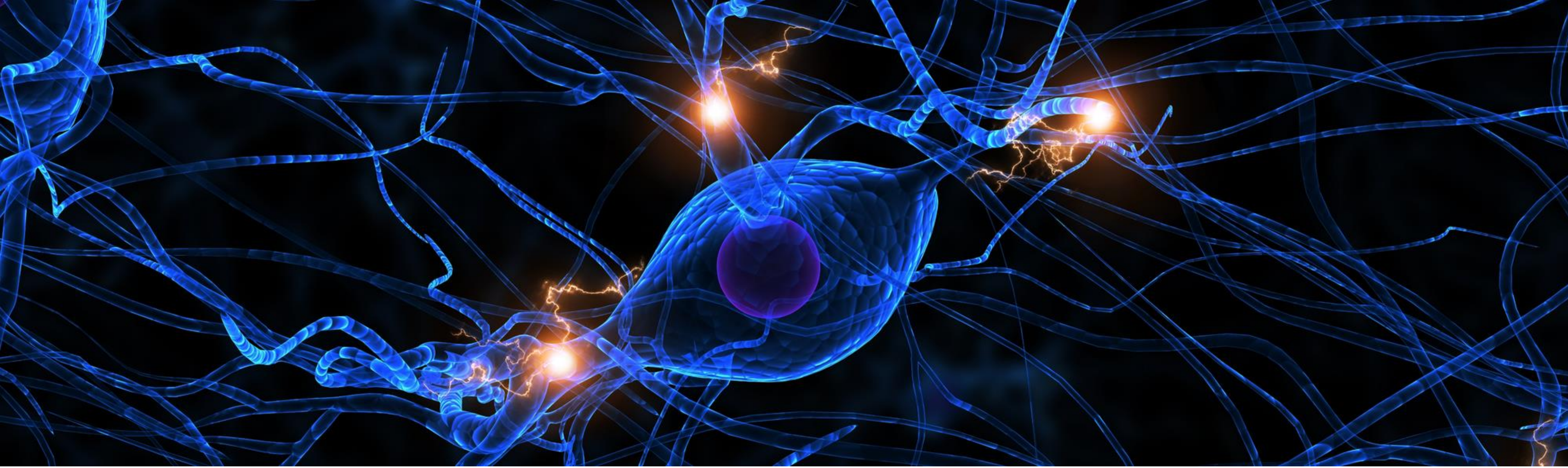


Artificial Neural Network on MatLab

Ankit Gupta
ITI, Madeira Technopolo, Floor -2, Room 3
Email: gupta.ankit894@gmail.com

Outline

- ✓ REVISION: neural network
- ✓ CASE STUDY: wines classification
- ✓ CASE STUDY: Body Fat



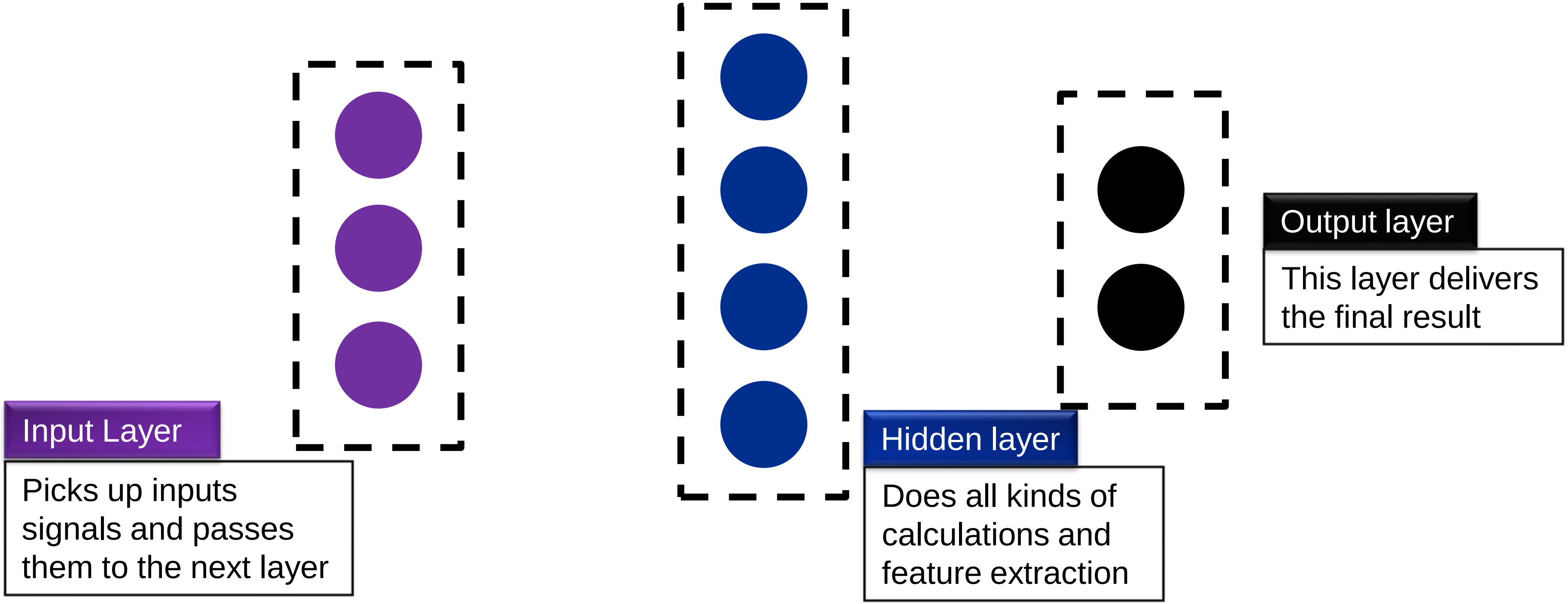
1.

REVISION

Let’s review some concepts

1. REVISION

How does a Neural Network work?



Input Layer

Picks up inputs signals and passes them to the next layer

Hidden layer

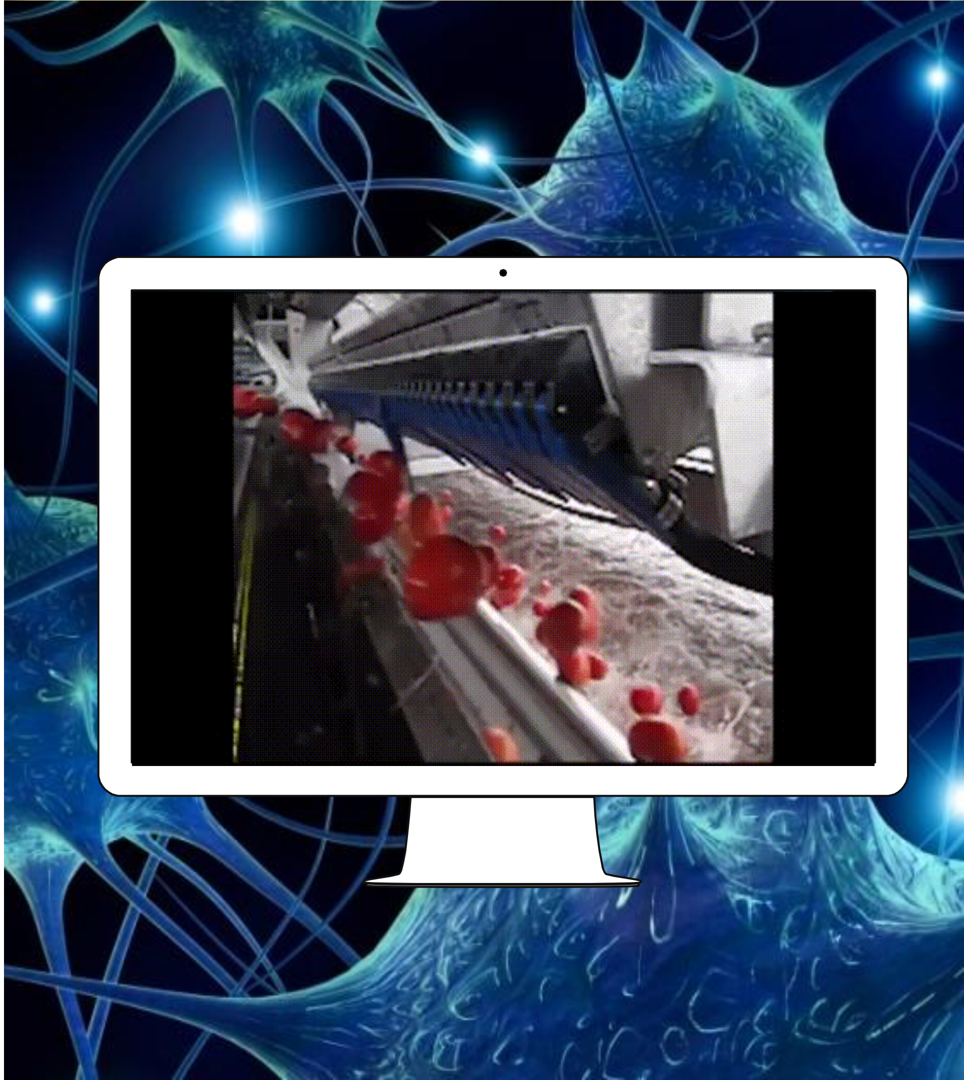
Does all kinds of calculations and feature extraction

Output layer

This layer delivers the final result

Example

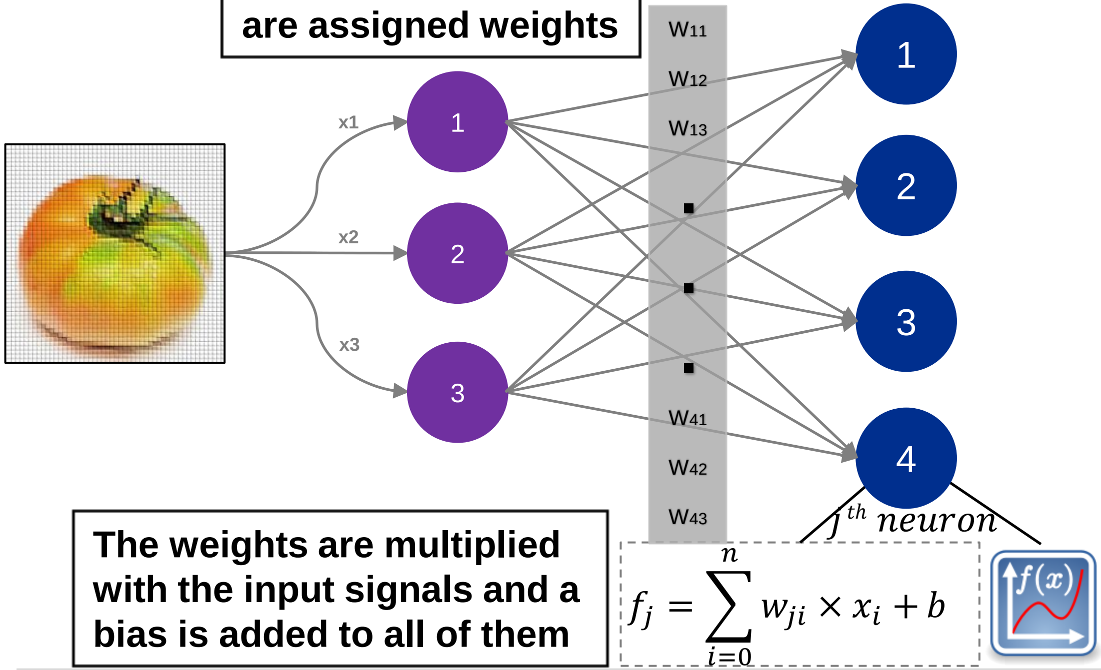
Neural network for ripe tomato classification



1. REVISION

How does a Neural Network work?

The interconnection are assigned weights



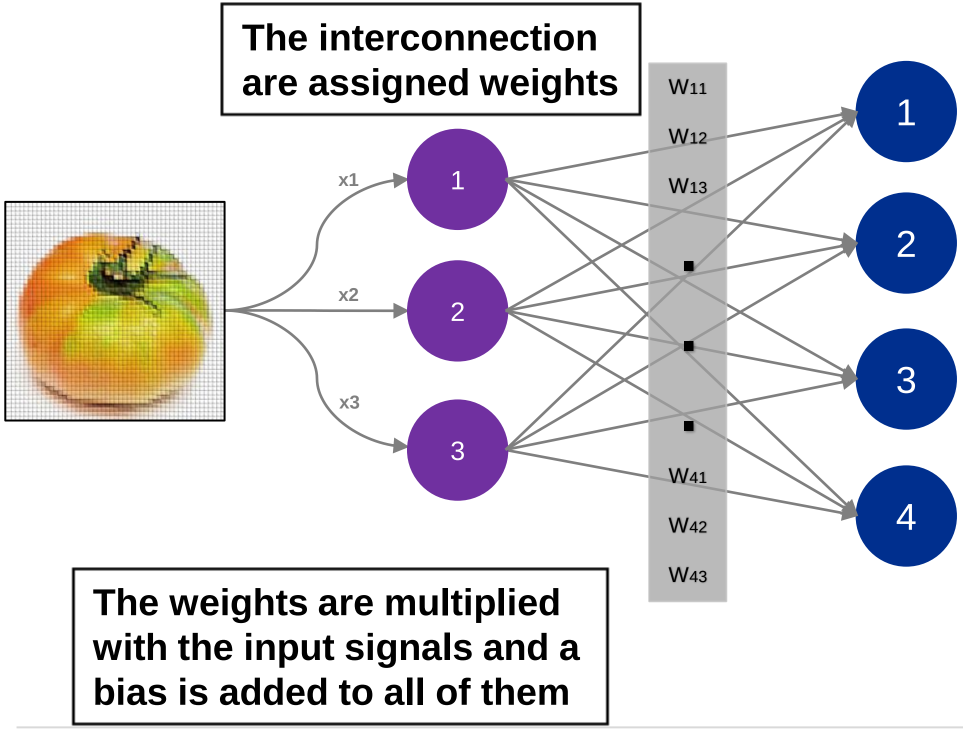
$$f_j = \sum_{i=0}^{n} w_{ji} \times x_i + b$$

The weights are multiplied with the input signals and a bias is added to all of them

The weighted sum of inputs is a fed as input to the activation function to decide which nodes to fire for feacture extraction

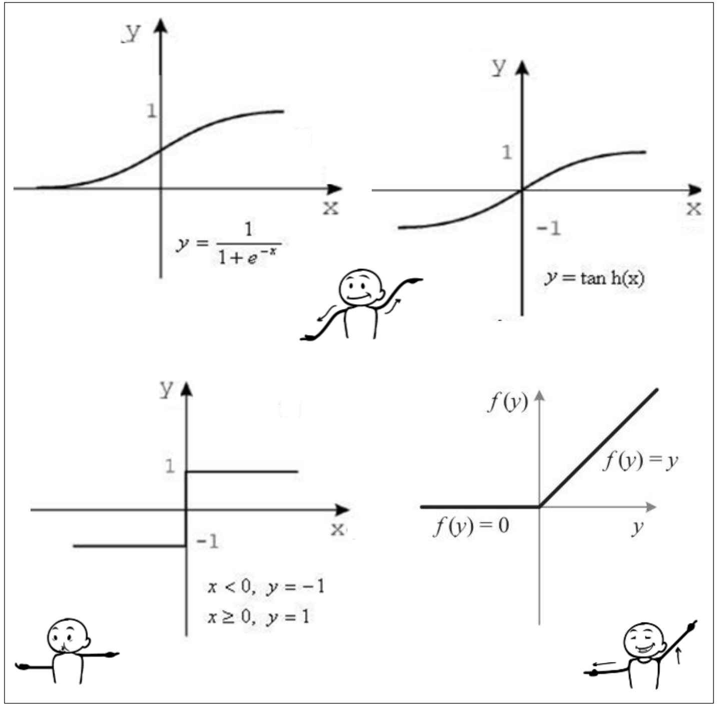
1. REVISION

How does a Neural Network work?



The interconnection are assigned weights

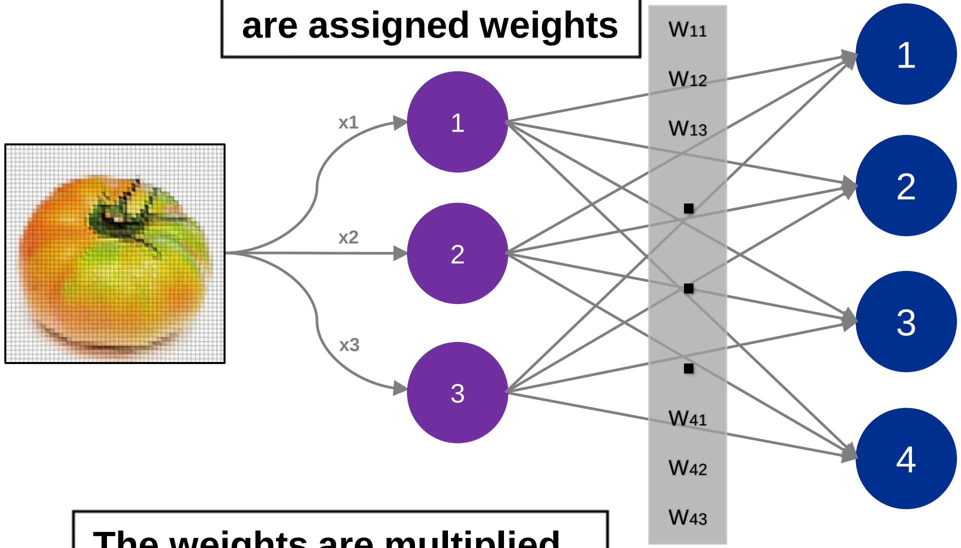
The weights are multiplied with the input signals and a bias is added to all of them



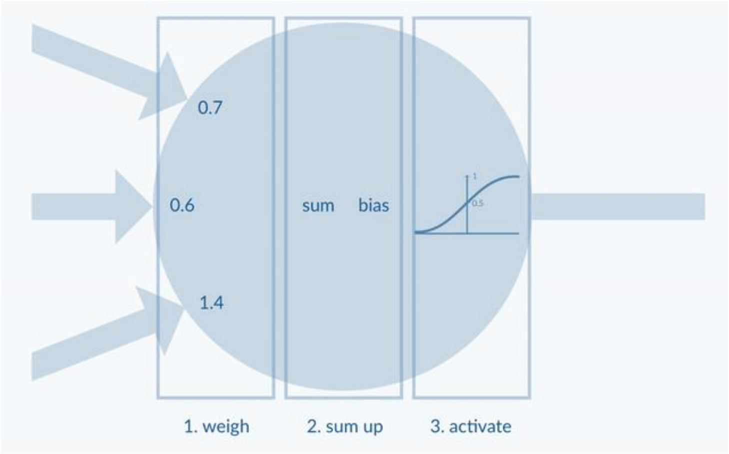
1. REVISION

How does a Neural Network work?

The interconnection are assigned weights



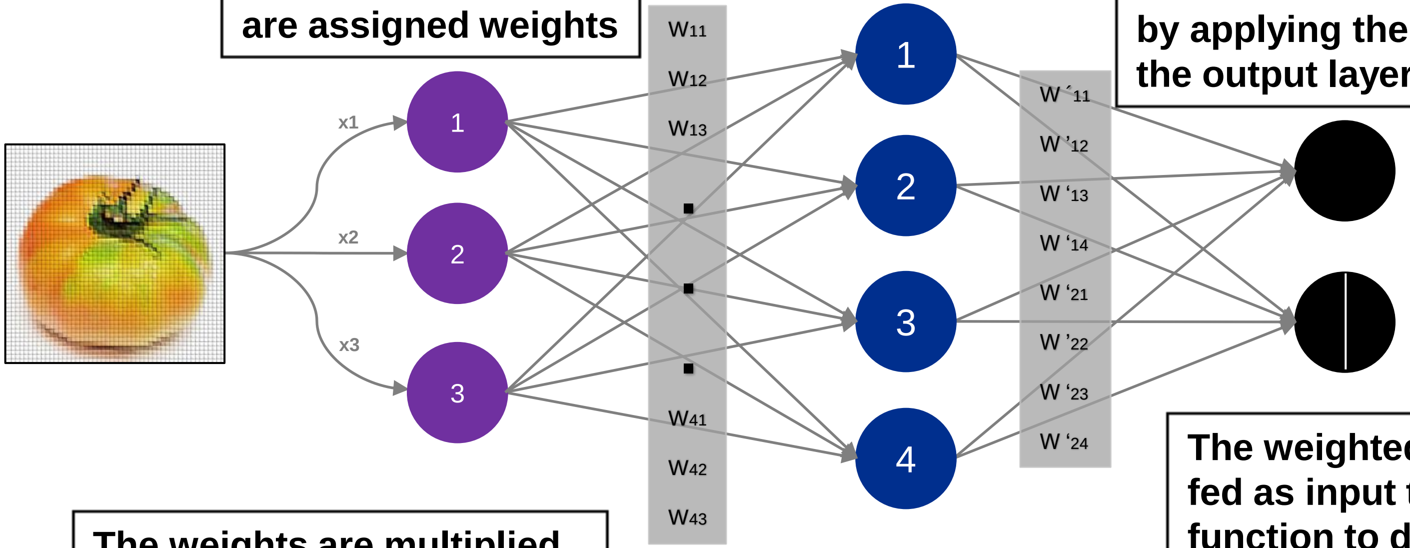
The weights are multiplied with the input signals and a bias is added to all of them



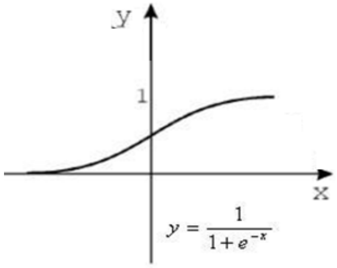
1. REVISION

How does a Neural Network work?

The interconnections are assigned weights



The model would predict the outcome by applying the activation function to the output layer.



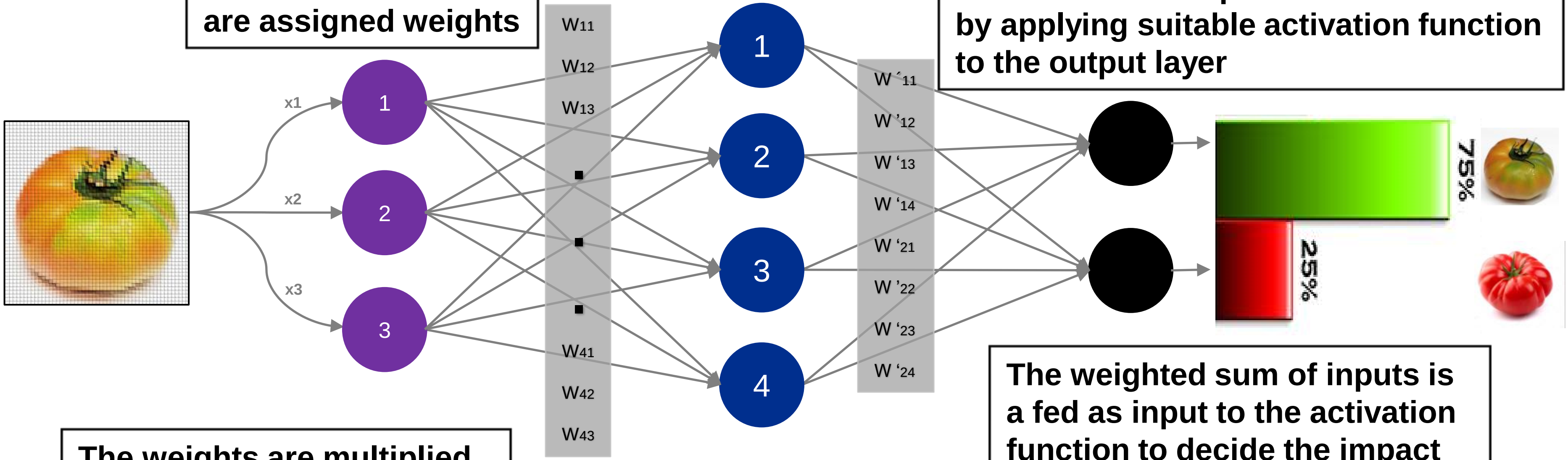
The weights are multiplied with the input signals and a bias is added to all of them

The weighted sum of inputs is fed as input to the activation function to decide the impact of neuron units in the succeeding layer.

1. REVISION

How does a Neural Network work?

The interconnections are assigned weights



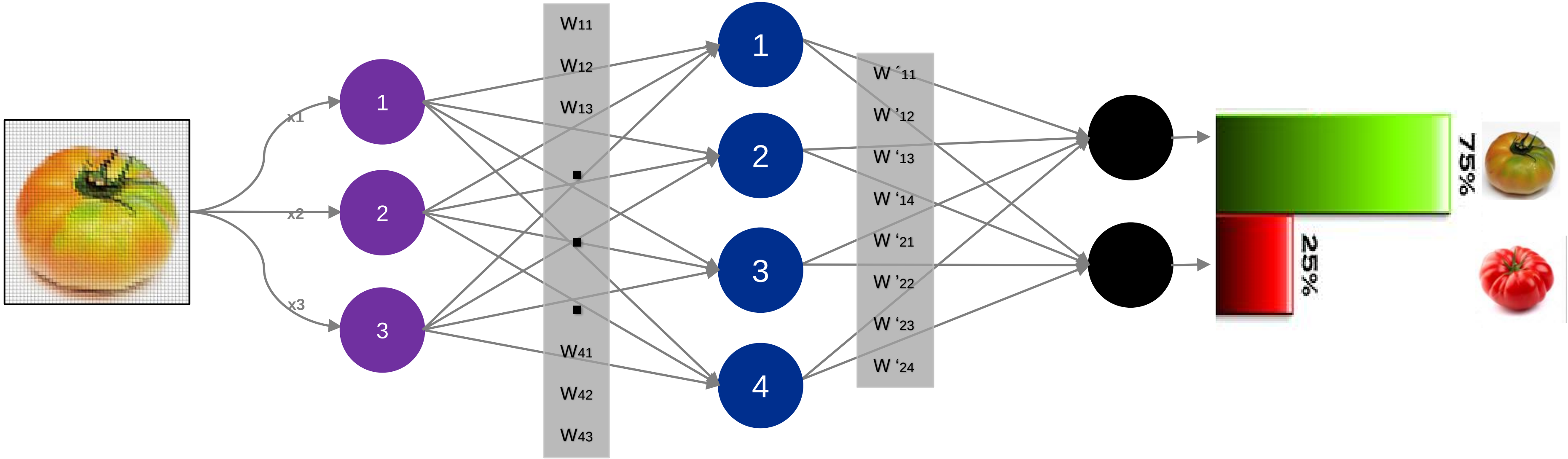
The model would predict the outcome by applying suitable activation function to the output layer

The weights are multiplied with the input signals and a bias is added to all of them

The weighted sum of inputs is a fed as input to the activation function to decide the impact of neuron units in the succeeding layer.

1. REVISION

How does a Neural Network work?

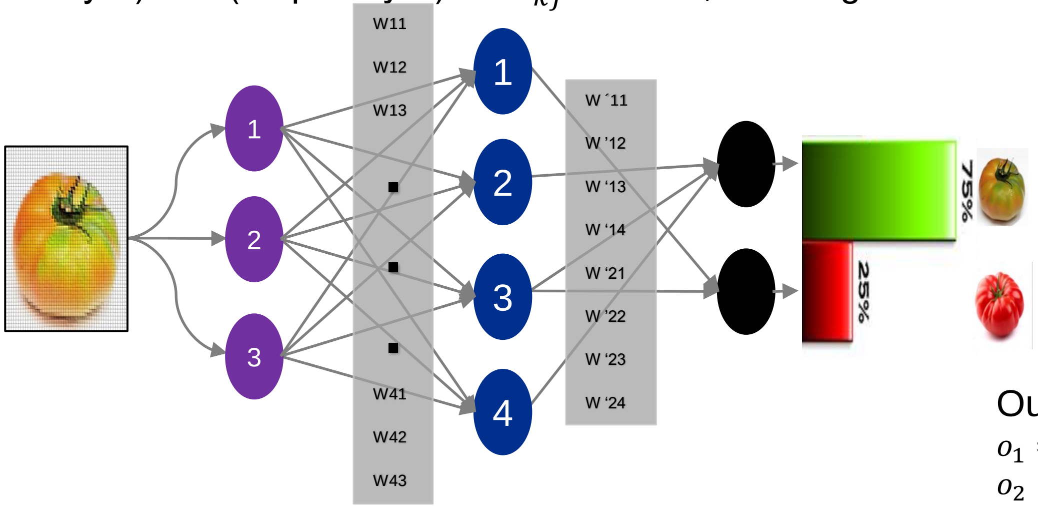


Input features are mapped to corresponding class labels using interative process called epoch.The respective weights of interconnections are optimised based on the estimated error (Backpropagation).

1. REVISION

Mathematical Operations behind ANNs

Let the input unit, hidden unit and output unit be represented as $i_1, h_1, and\ o_1$.
Let the weight pointing from node i (input layer) to j (hidden layer) be w_{ji} , and from node j (hidden layer) to k (output layer) be w'_{kj}. Further, the weights associated with bias terms is equal to 1.



Hidden Layer Units calculation

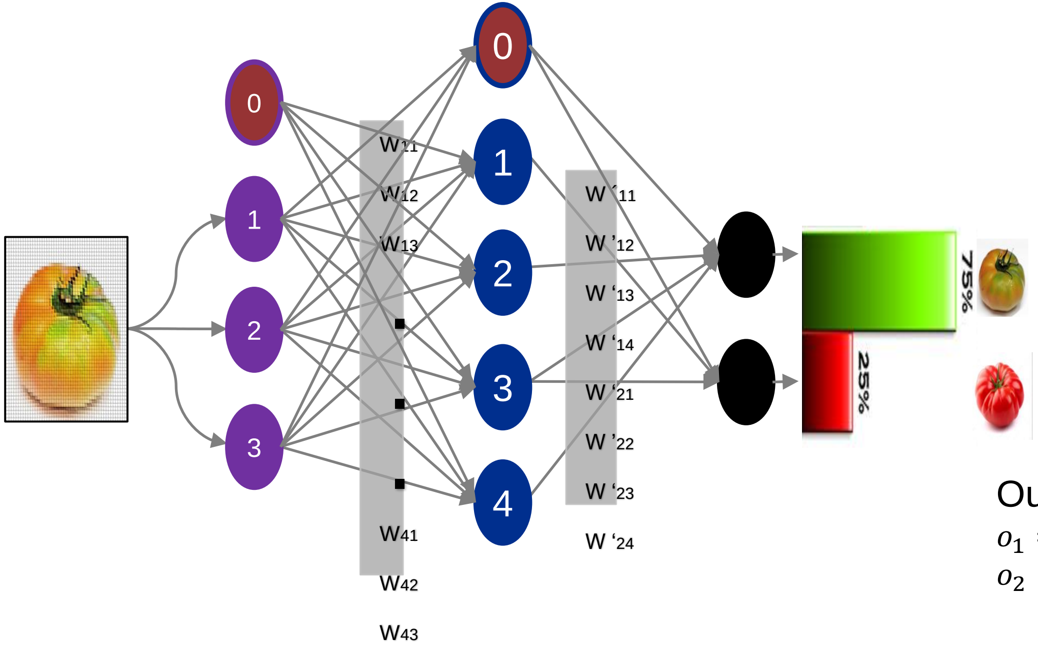
$$h_1 = \sigma(w_{10} * i_0 + w_{11} * i_1 + w_{12} * i_2 + w_{13} * i_3)$$
$$h_2 = \sigma(w_{20} * i_0 + w_{21} * i_1 + w_{22} * i_2 + w_{23} * i_3)$$
$$h_3 = \sigma(w_{30} * i_0 + w_{31} * i_1 + w_{32} * i_2 + w_{33} * i_3)$$
$$h_4 = \sigma(w_{40} * i_0 + w_{41} * i_1 + w_{42} * i_2 + w_{43} * i_3)$$

Output Layer Units calculation

$$o_1 = \sigma(w'_{10} * h_0 + w'_{11} * h_1 + w'_{12} * h_2 + w'_{13} * h_3 + w'_{14} * h_4)$$
$$o_2 = \sigma(w'_{20} * h_0 + w'_{21} * h_1 + w'_{22} * h_2 + w'_{23} * h_3 + w'_{24} * h_4)$$

1. REVISION

Mathematical Operations behind ANNs



Hidden Layer Units calculation

$$h_1 = \sigma(w_{10} * i_0 + w_{11} * i_1 + w_{12} * i_2 + w_{13} * i_3)$$
$$h_2 = \sigma(w_{20} * i_0 + w_{21} * i_1 + w_{22} * i_2 + w_{23} * i_3)$$
$$h_3 = \sigma(w_{30} * i_0 + w_{31} * i_1 + w_{32} * i_2 + w_{33} * i_3)$$
$$h_4 = \sigma(w_{40} * i_0 + w_{41} * i_1 + w_{42} * i_2 + w_{43} * i_3)$$

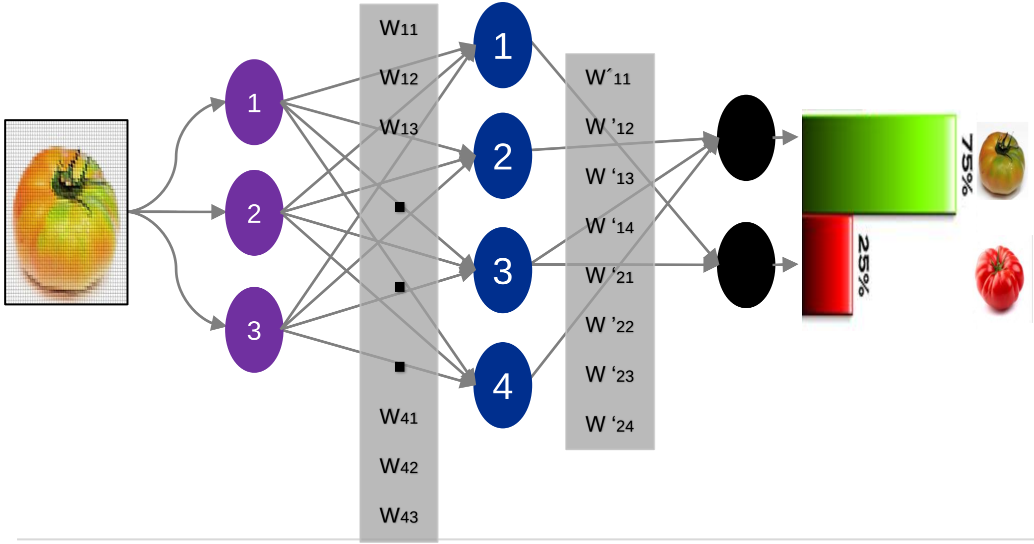
Output Layer Units calculation

$$o_1 = \sigma(w'_{10} * h_0 + w'_{11} * h_1 + w'_{12} * h_2 + w'_{13} * h_3 + w'_{14} * h_4)$$
$$o_2 = \sigma(w'_{20} * h_0 + w'_{21} * h_1 + w'_{22} * h_2 + w'_{23} * h_3 + w'_{24} * h_4)$$

1. REVISION

Mathematical Operations behind ANNs

Let the input unit, hidden unit and output unit be represented as i_1, $h_1, and\ o_1$.
Let the weight pointing from node i (input layer) to j (hidden layer) be w_{ji} , and from node j (hidden layer) to k (output layer) be w'_{kj}. Fu



Hidden Layer Units calculation (Matrix Form)

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} = \begin{bmatrix} w_{10} & w_{11} & w_{12} & w_{13} \\ w_{20} & w_{21} & w_{22} & w_{23} \\ w_{30} & w_{31} & w_{32} & w_{33} \\ w_{40} & w_{41} & w_{42} & w_{43} \end{bmatrix} \cdot \begin{bmatrix} i_0 \\ i_1 \\ i_2 \\ i_3 \end{bmatrix}$$

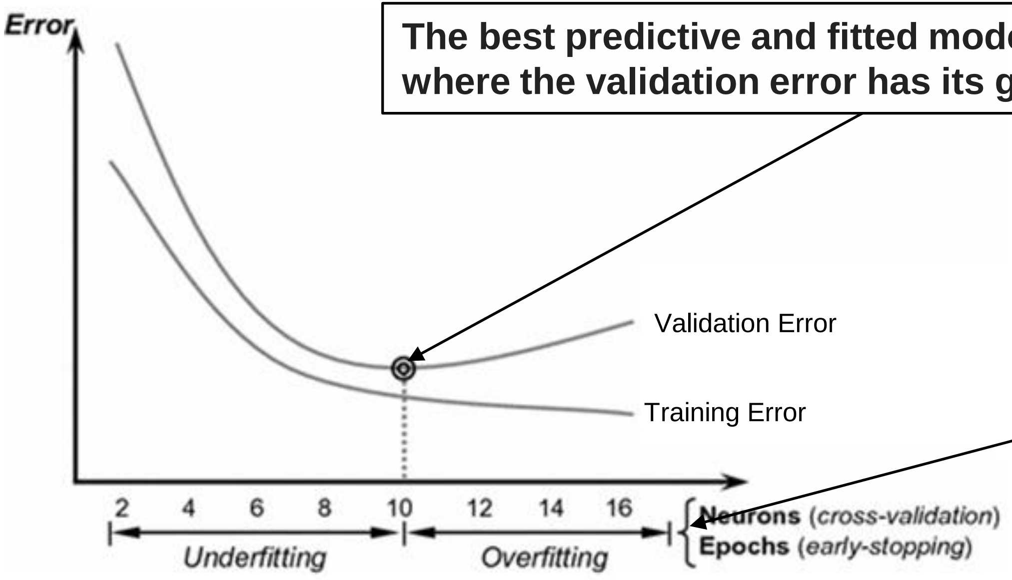
Output Layer Units calculation (Matrix Form)

$$\begin{bmatrix} o_1 \\ o_1 \end{bmatrix} = \begin{bmatrix} W'_{10} & W'_{11} & W'_{12} & W'_{13} & W'_{14} \\ W'_{20} & W'_{21} & W'_{22} & W'_{23} & W'_{24} \end{bmatrix} \cdot \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix}$$

1. REVISION

How does a Neural Network work?

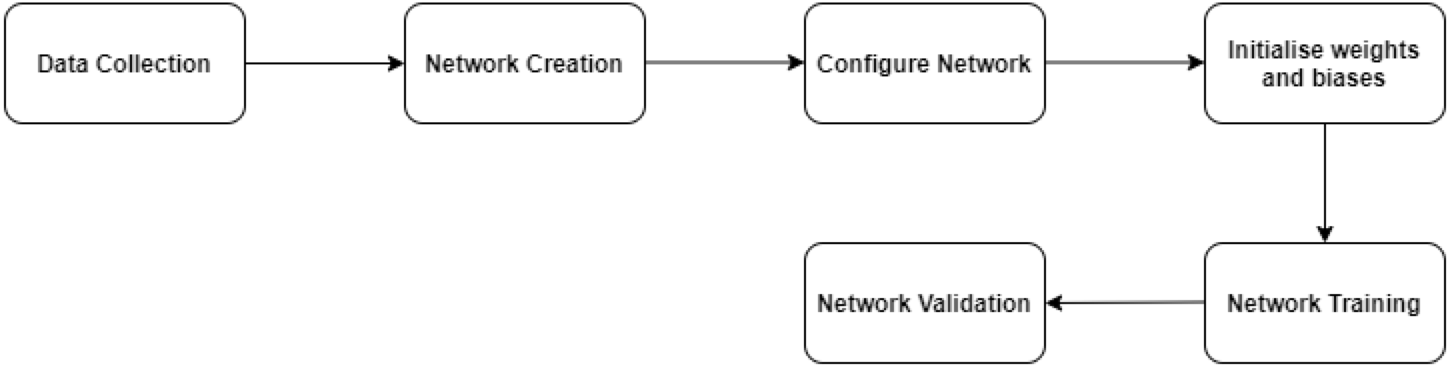
Training error and validation error are functions of the number epochs.



The best predictive and fitted model would be where the validation error has its global minimum.

If the validation error increases(positive slope) while the training error steadily decreases(negative slope) then a situation of overfitting might have occurred.

Neural Network Training Basic Workflow



Data Collection

Network Creation

Configure Network

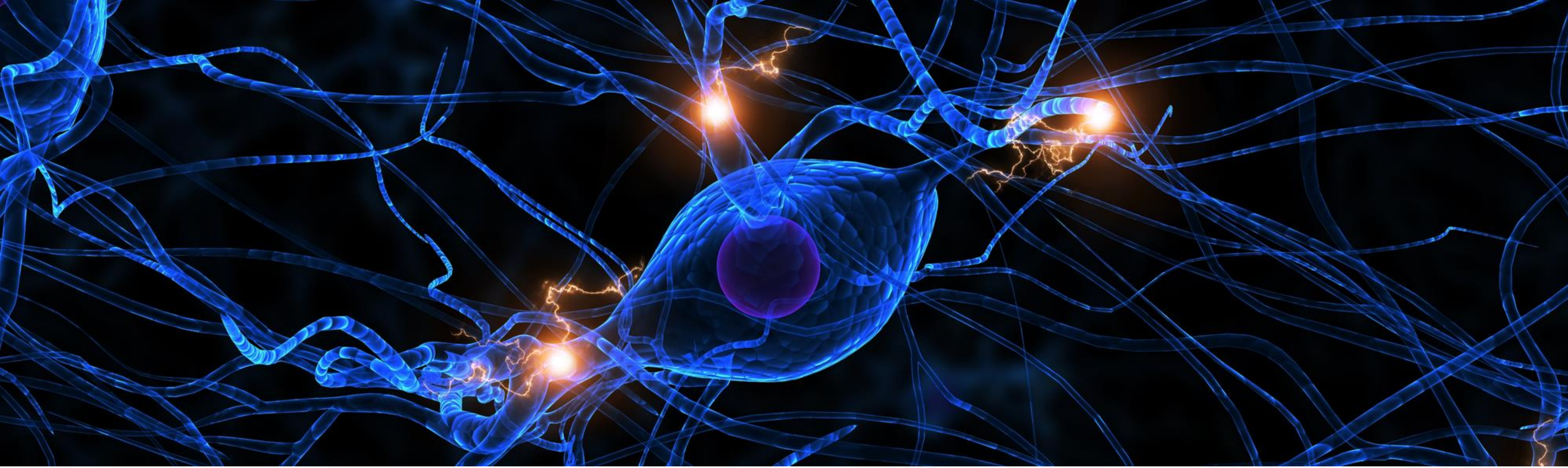
Initialise weights and biases

Network Training

Network Validation

Neural Network Training Basic Workflow

- **Data Collection:** Collecting the features along with the corresponding labels.
- **Network Creation:** Deciding the number of hidden units and create a basic Neural Network architecture.
- **Configuring Network:** Selecting the error function and optimization algorithm, convergence condition, stopping criteria, train-test data splitting criteria.
- **Weights and biases Initialization:** This is done for faster convergence; depends on experience.
- **Network Training :** Multiple passes for mapping inputs(features) to outputs(labels).
- **Network Validation :**It is done by predicting the validation data samples followed by calculating various performance metrics e.g. confusion matrix.



2. CASE STUDY:
wine classification

Let's to our first case.

Case study:

wine classification

Pattern recognition neural network for wines classification by winery based on its chemical characteristics.

Case study:

wine classification

In this example we attempt to build a neural network that can classify wines from 3 wineries by 13 attributes:

1. Alcohol
2. Malic acid
3. Ash
4. Alcalinity of ash
5. Magnesium
6. Total phenols
7. Flavanoids
8. Nonflavanoid phenols
9. Proanthocyanins
10. Color intensity
11. Hue
12. OD280/OD315 of diluted wines
13. Proline

2. CASE STUDY: wine classification

Preparing the Data

1. Load the dataset

```
[x,t] = wine_dataset;
```

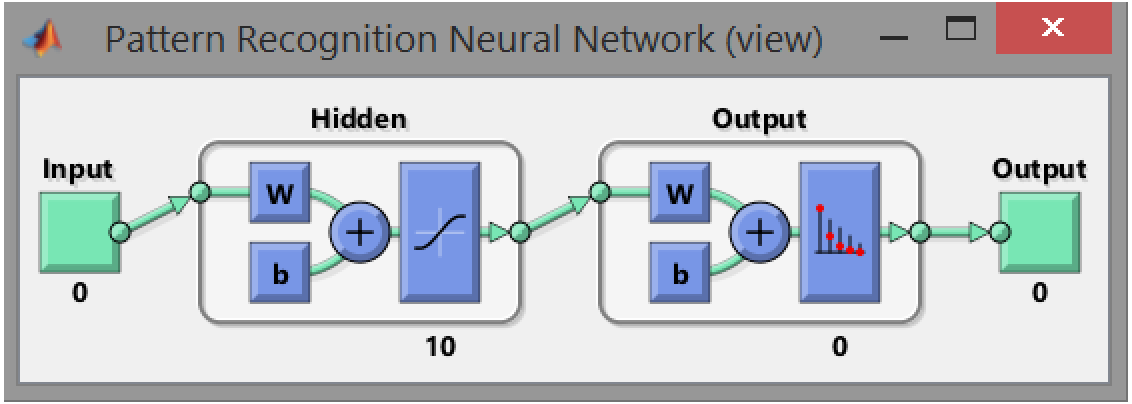
- The data is organized into 2 matrices → the input matrix *x* and the target matrix *t*.
- Both matrices have 178 columns → represent 178 wine sample attributes (inputs) and associated winery class vectors (targets).
- Input matrix x has 13 rows → represent the 13 attributes.
- Target matrix t has 3 rows → represent the 3 wineries.

2. CASE STUDY: wine classification

Creating Neural Network for Pattern Recognition

2. Create a neural network that will learn to classify the wines

```
net = patternnet(10);
view(net);
```


Pattern Recognition Neural Network (view)

2. CASE STUDY: wine classification

Configuring the Neural Network

3. Configuration of neural network

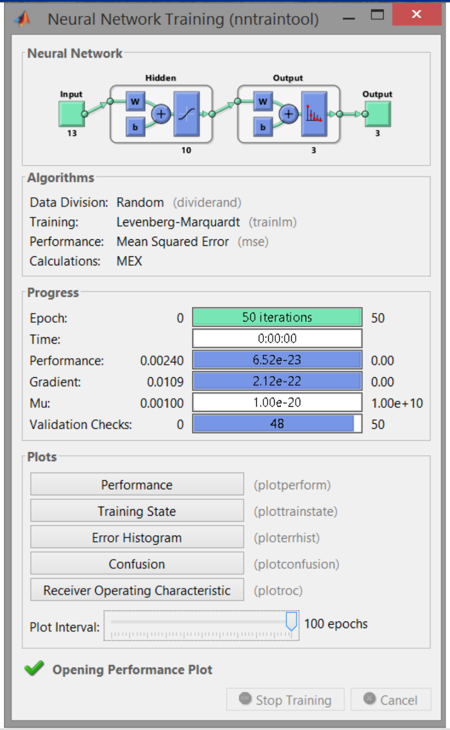
```
net.trainFcn='trainlm';   ‘Levenberg-Marquardt’
net.performFcn='mse';
net.trainParam.min_grad=0;
net.trainParam.epochs=50
net.trainParam.max_fail =50;
net.divideParam.trainRatio=0.7;
net.divideParam.valRatio=0.15;
net.divideParam.testRatio=0.15;
```

2. CASE STUDY: wine classification

Training the Neural Network

4. Train of Neural Network

```
[net,tr] = train(net,x,t);
```



Neural Network

Algorithms

Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	50 iterations	50
Time:		0:00:00	
Performance:	0.00240	6.52e-23	0.00
Gradient:	0.0109	2.12e-22	0.00
Mu:	0.00100	1.00e-20	1.00e+10
Validation Checks:	0	48	50

Plots

Performance	(plotperform)
Training State	(plottrainstate)
Error Histogram	(ploterrhist)
Confusion	(plotconfusion)
Receiver Operating Characteristic	(plotroc)

Plot Interval: 100 epochs

✓ **Opening Performance Plot**

2. CASE STUDY: wine classification

Testing the Neural Network

5. Test the performance of Neural Network

```
testX = x(:,tr.testInd);
testT = t(:,tr.testInd);
testY = net(testX);
testIndices = vec2ind(testY)
```

2. CASE STUDY: wine classification

Checking the performance

5. Plot confusion

```
plotconfusion(testT,testY)
```

Or

```
[c,cm] = confusion(testT,testY);
fprintf(‘% Correct Classification : %f%%\n', 100*(1-c));
fprintf(‘% Incorrect Classification : %f%%\n', 100*c);
```

2. CASE STUDY: wine classification

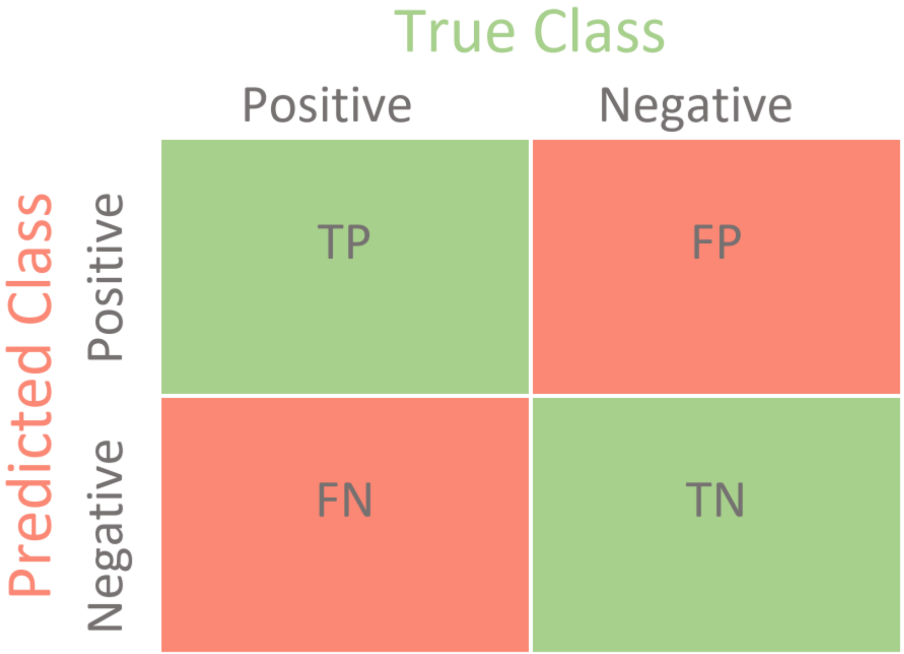
Confusion Matrix

TP-True Positive
True class(+)=Predicted class(+)

TN-True Negative
True class(-)=Predicted class(-)

FP-False Positive
True class(-)=Predicted class(+)

FN-False Negative
True class(+)=Predicted class(-)



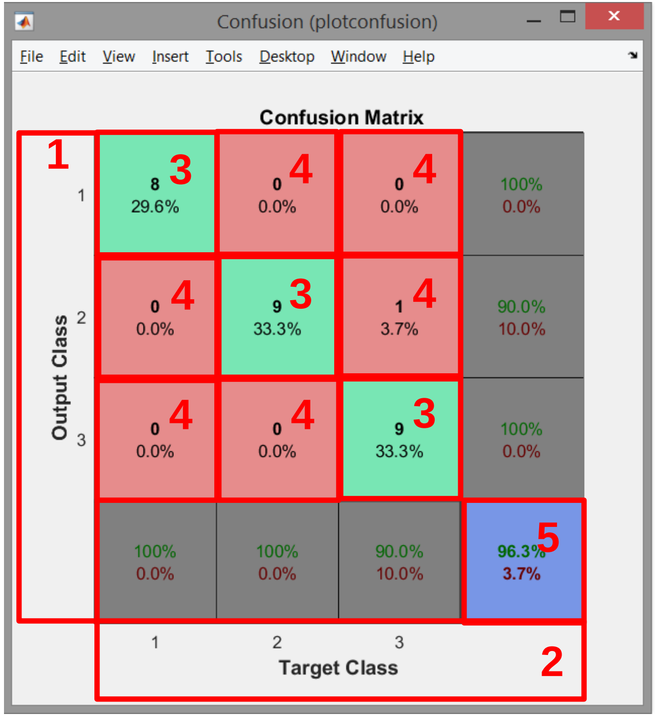
		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

2. CASE STUDY: wine classification

Checking the performance

5. Plot confusion

1. The rows correspond to the predicted class: Output Class
2. the columns correspond to the true class: Target Class.
3. The diagonal cells correspond to observations that are correctly classified(**True Positives/True Negatives**).
4. The off-diagonal cells correspond to incorrectly classified observations(**False Negatives/False Positives**).
5. The cell in the bottom right of the plot shows the overall accuracy



Confusion Matrix

Output Class

1	8 29.6%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	9 33.3%	1 3.7%	90.0% 10.0%
3	0 0.0%	0 0.0%	9 33.3%	100% 0.0%
	100% 0.0%	100% 0.0%	90.0% 10.0%	96.3% 3.7%
	1	2	3	

Target Class

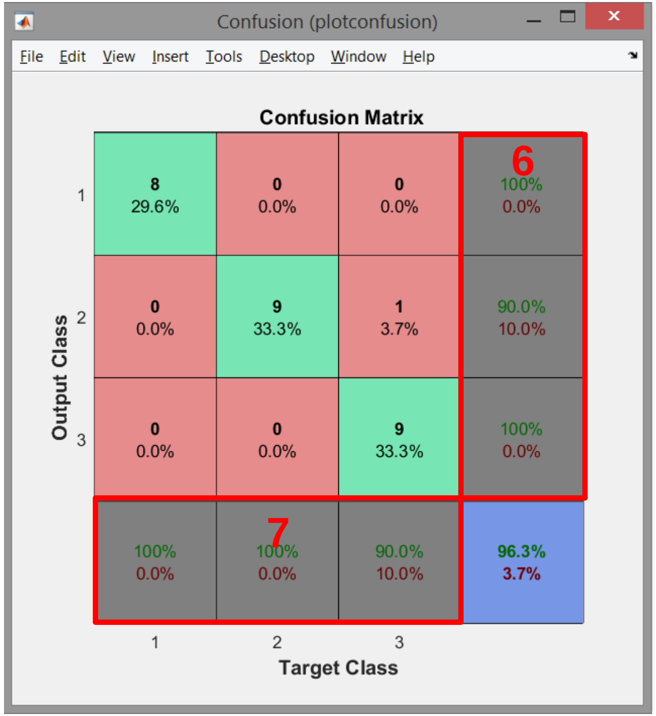
2. CASE STUDY: wine classification

Checking the performance

5. Plot confusion

6. Precision (positive predictive value) and false (positive) discovery rate: The column on the far right of the plot shows the percentages of all the examples predicted to belong to each class that are correctly and incorrectly classified.

7. Recall (or true positive rate) and false negative rate: The row at the bottom of the plot shows the percentages of all the examples belonging to each class that are correctly and incorrectly classified.



Confusion Matrix

	1	2	3	
1	8 29.6%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	9 33.3%	1 3.7%	90.0% 10.0%
3	0 0.0%	0 0.0%	9 33.3%	100% 0.0%
	100% 0.0%	100% 0.0%	90.0% 10.0%	96.3% 3.7%

2. CASE STUDY: wine classification

Checking the performance

5. Understanding confusion matrix for multi-classes

Analyzing the confusion matrix for multi-class classification is actually one-vs-all classification.
One-vs-all means one class is treated as positive class, while other classes are treated as negative classes, at a time.

		Target Class		
		1	2	3
Predicted Class	1	8	0	0
	2	0	9	1
	3	0	0	0

2. CASE STUDY: wine classification

Checking the performance

5. Understanding confusion matrix for multi-classes

Analyzing the confusion matrix for multi-class classification is actually one-vs-all classification.
One-vs-all means one class is treated as positive class, while other classes are treated as negative classes, at a time.

Target Class

Predicted Class

	1	2	3
1	8	0	0
2	0	9	1
3	0	0	0

2. CASE STUDY: wine classification

Checking the performance

5. Plot confusion

Target Class

Predicted Class

	1	2	3
1	8	0	0
2	0	9	1
3	0	0	0

Target Class

Predicted Class

	1	0	0
1	TP	FP	FP
0	FN	TN	TN
0	FN	TN	TN

Similar procedure can be followed for class 2 and 3, followed by calculating accuracy, sensitivity and specificity.

2. CASE STUDY: wine classification

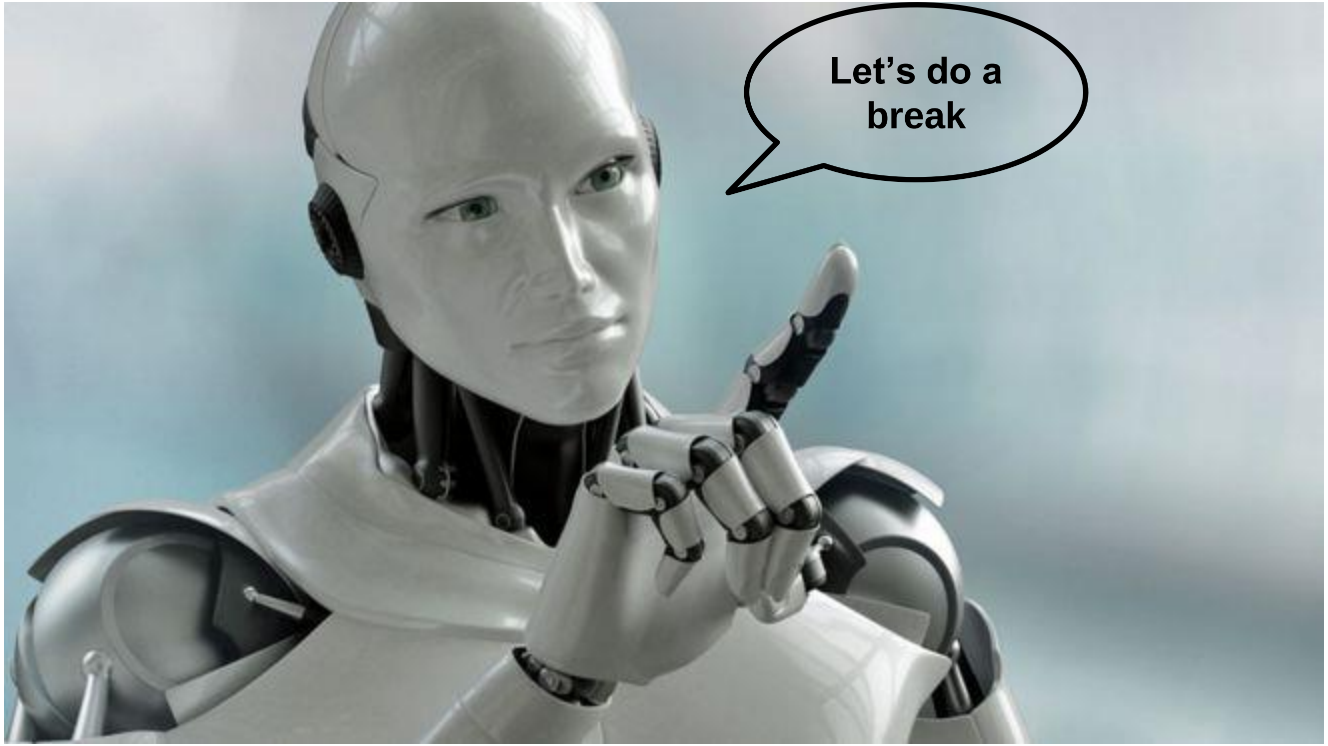
Checking the performance

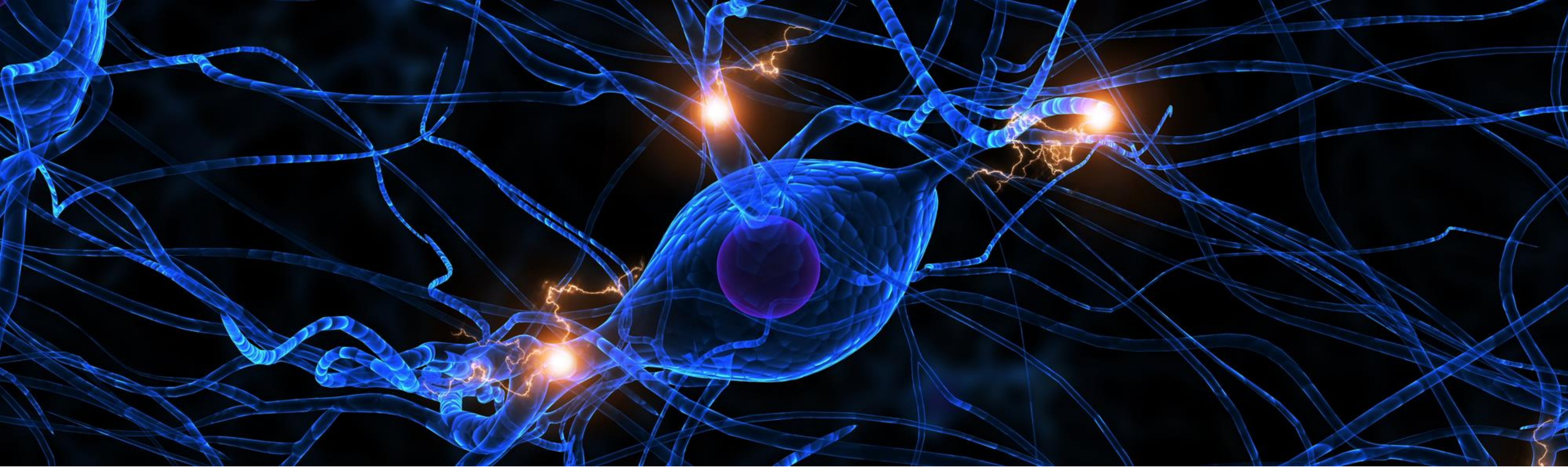
5. Performance Metrics

$$Accuracy = \frac{TP+TN}{P+N}$$

$$Sensitivity = \frac{TP}{TP+FN}$$

$$Specificity = \frac{TN}{TN+FP}$$


Let’s do a break



3. CASE STUDY:
Body fat

Let's to the third case.

Case study:

body fat

Fitting neural network for estimate the percentage of body fat of someone from various measures.

1. Age (years)
2. Weight (lbs)
3. Height (inches)
4. Neck circumference (cm)
5. Chest circumference (cm)
6. Abdomen 2 circumference (cm)
7. Hip circumference (cm)
8. Thigh circumference (cm)
9. Knee circumference (cm)
10. Ankle circumference (cm)
11. Biceps (extended) circumference (cm)
12. Forearm circumference (cm)
13. Wrist circumference (cm)

3. CASE STUDY: body fat

Pattern recognition neural network & Fitting neural network

Artificial neural network for fitting is quite similar for classifying

Instead of using:

`net = patternnet(10);`

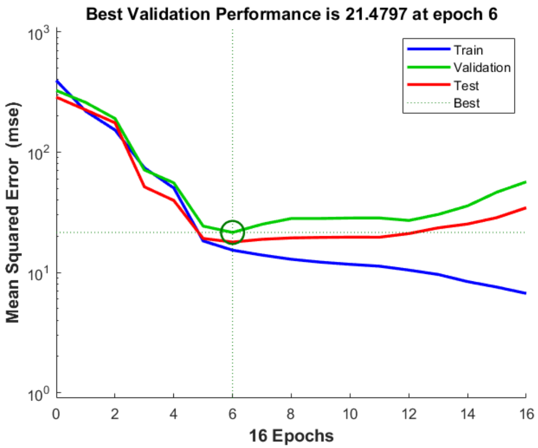
We use:

`net = fitnet(10);`

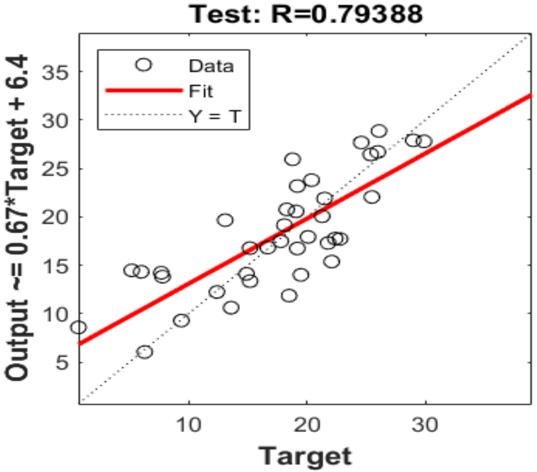
3. CASE STUDY: body fat

Fitting neural network

Analysing Results



Performance during training process



Regression Fit

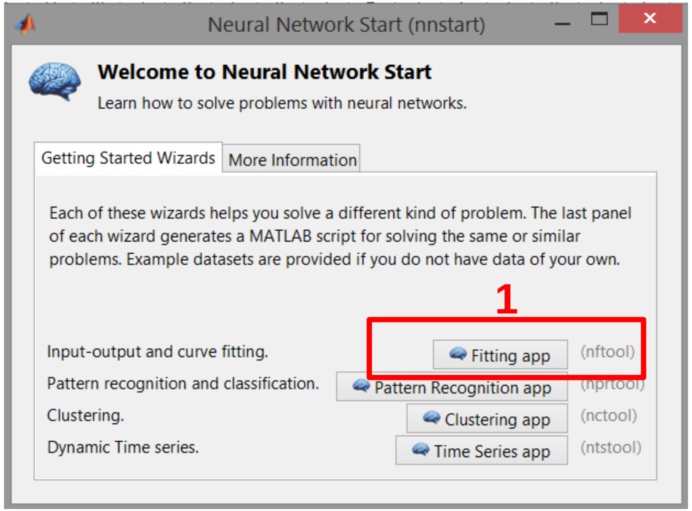
3. CASE STUDY: body fat

Neural Network Start GUI

[bodyfatInputs,bodyfatTargets]= bodyfat_dataset;

Open the Neural Network Start GUI with this command: nnstart

The rest of the process is similar to the previous case study



Neural Network Start (nnstart)

Welcome to Neural Network Start

Learn how to solve problems with neural networks.

Each of these wizards helps you solve a different kind of problem. The last panel of each wizard generates a MATLAB script for solving the same or similar problems. Example datasets are provided if you do not have data of your own.

Input-output and curve fitting.	Fitting app	(nftool)
Pattern recognition and classification.	Pattern Recognition app	(nprtool)
Clustering.	Clustering app	(nctool)
Dynamic Time series.	Time Series app	(ntstool)

NOTES AND TIPS

Now, the most important thing is to practice to gain sensitivity for creating neural models

For practicing
Datasets: Deep Learning Toolbox Sample Data Sets

Use the Neural Network Start GUI. Use the script button to reproduce the neural network and, then, adapt it to solve similar problems.

Thanks!