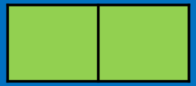
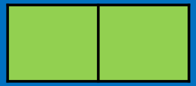
ANN

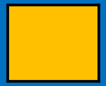
Densely connected neural network



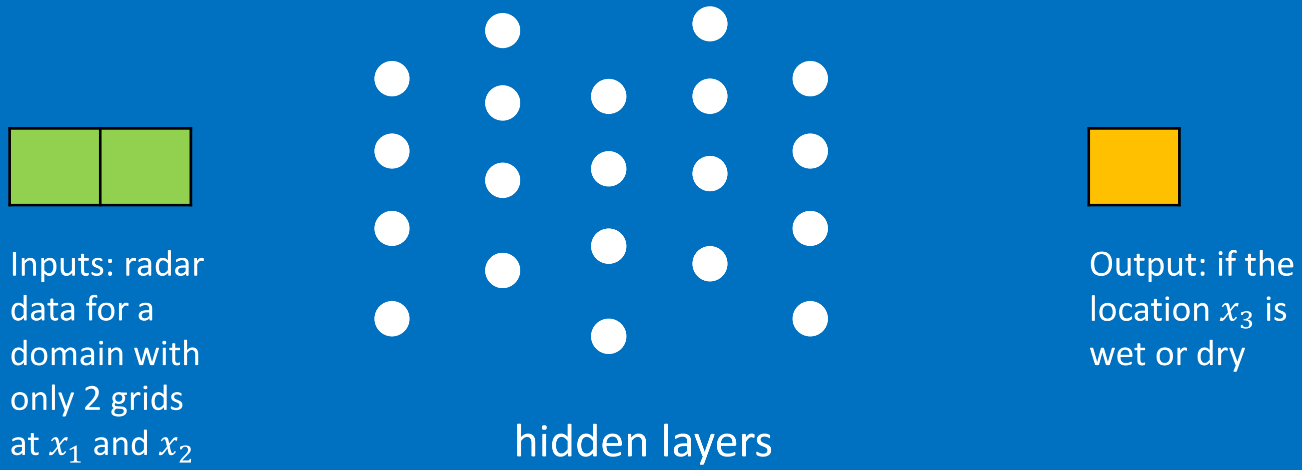
Inputs: radar data for a domain with only 2 grids at x_1 and x_2

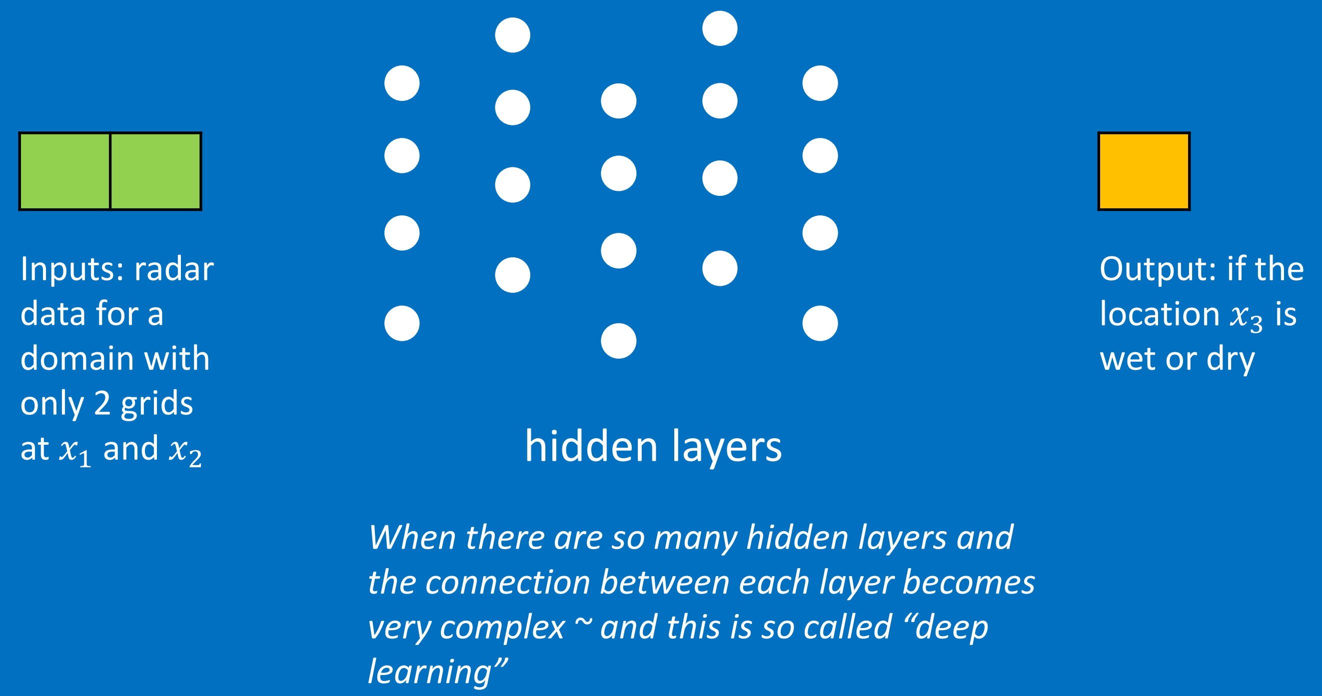


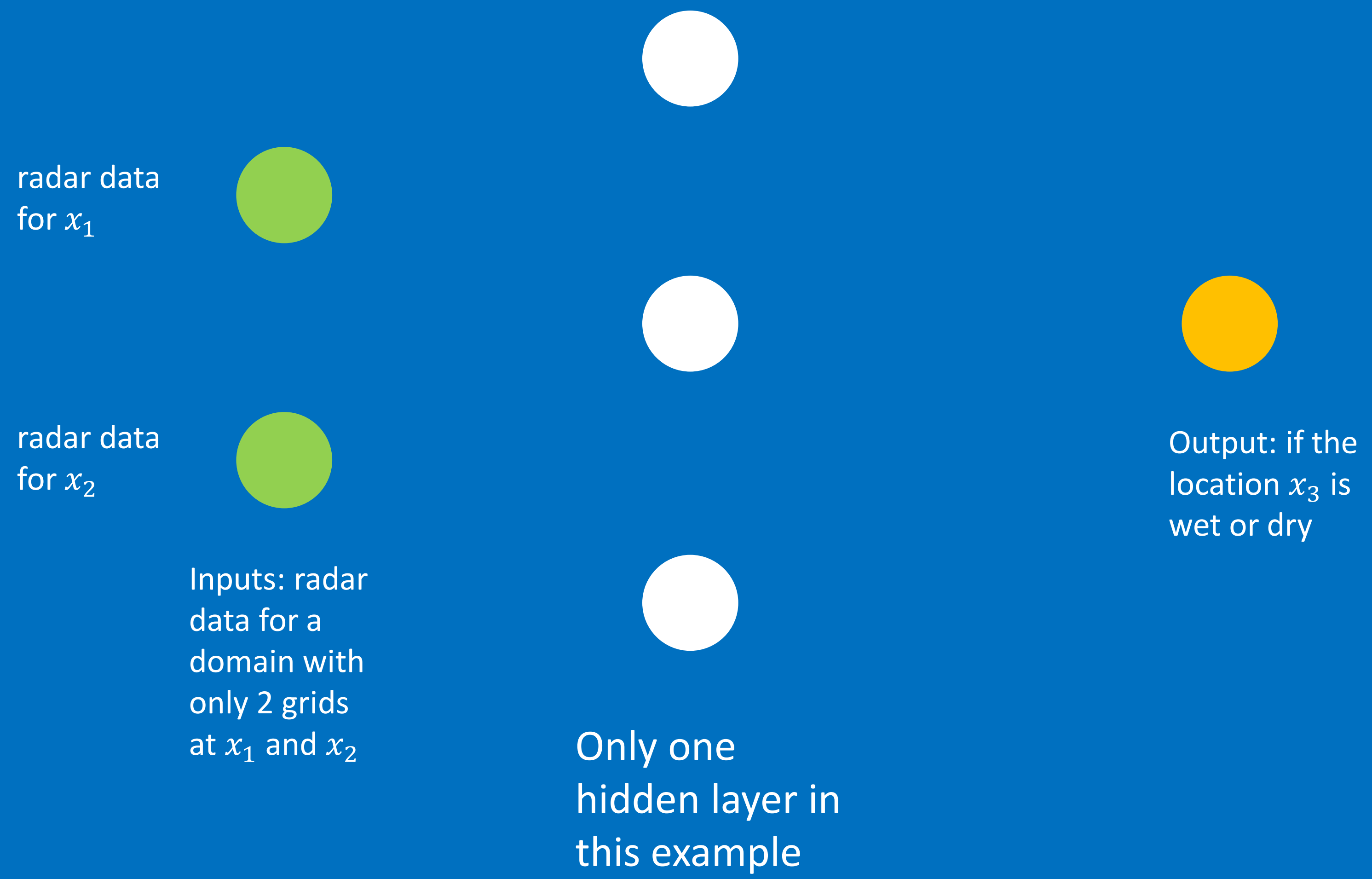
Inputs: radar data for a domain with only 2 grids at x_1 and x_2

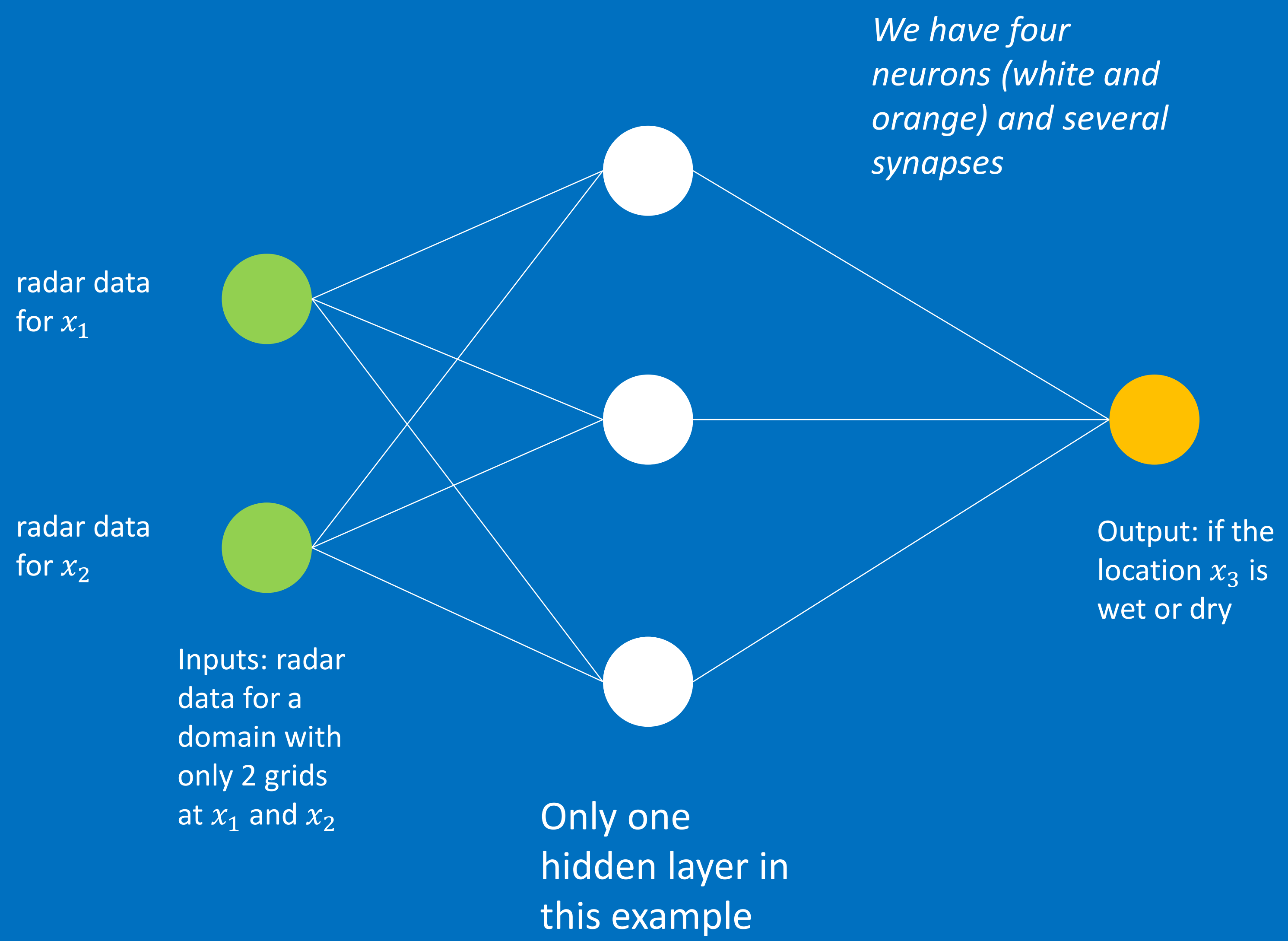


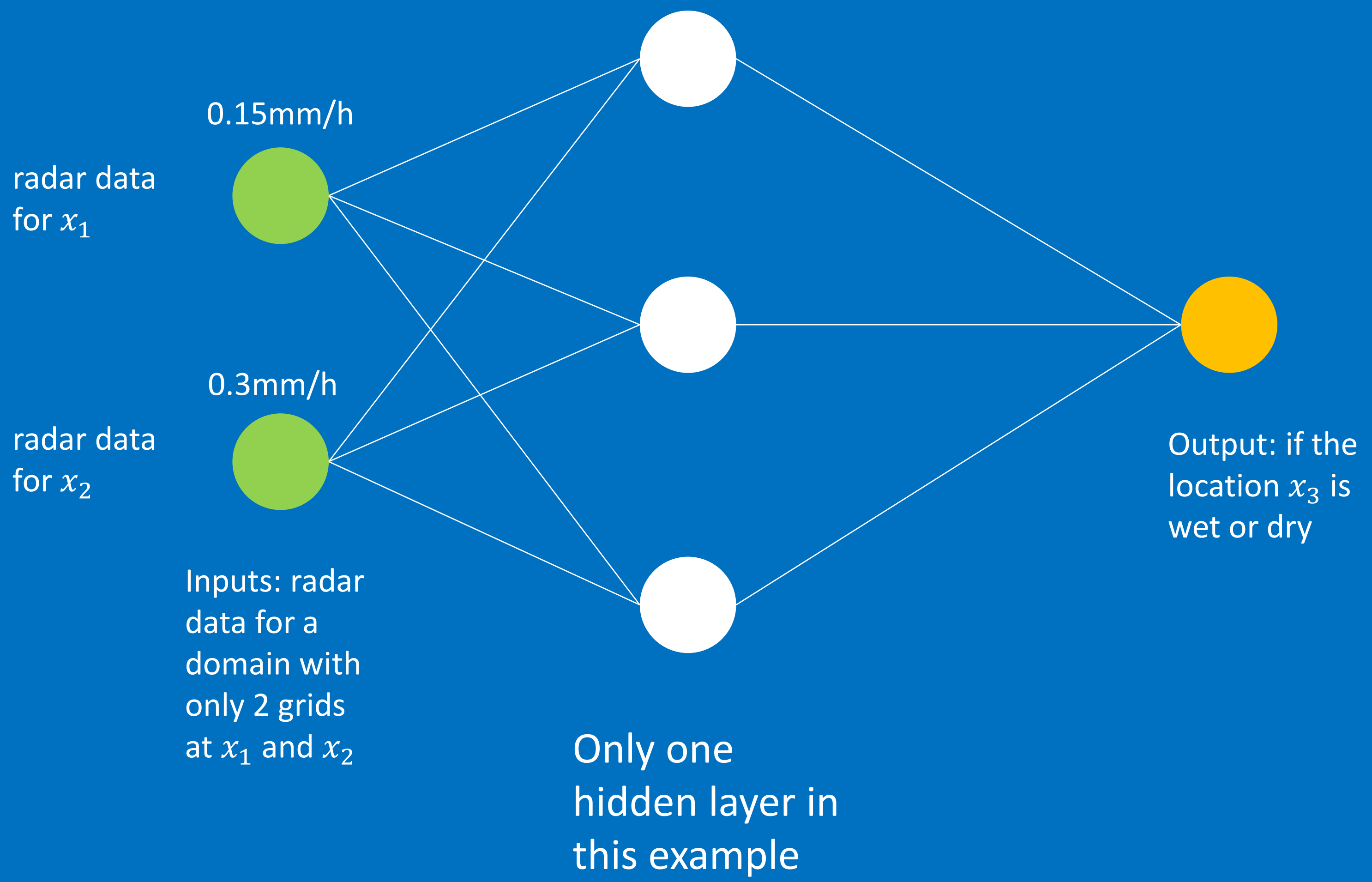
Output: if the location x_3 is wet or dry


Inputs: radar data for a domain with only 2 grids at x_1 and x_2
hidden layers
Output: if the location x_3 is wet or dry


Inputs: radar data for a domain with only 2 grids at x_1 and x_2
hidden layers
When there are so many hidden layers and the connection between each layer becomes very complex ~ and this is so called "deep learning"
Output: if the location x_3 is wet or dry


radar data for x_1
radar data for x_2
Inputs: radar data for a domain with only 2 grids at x_1 and x_2
Only one hidden layer in this example
Output: if the location x_3 is wet or dry

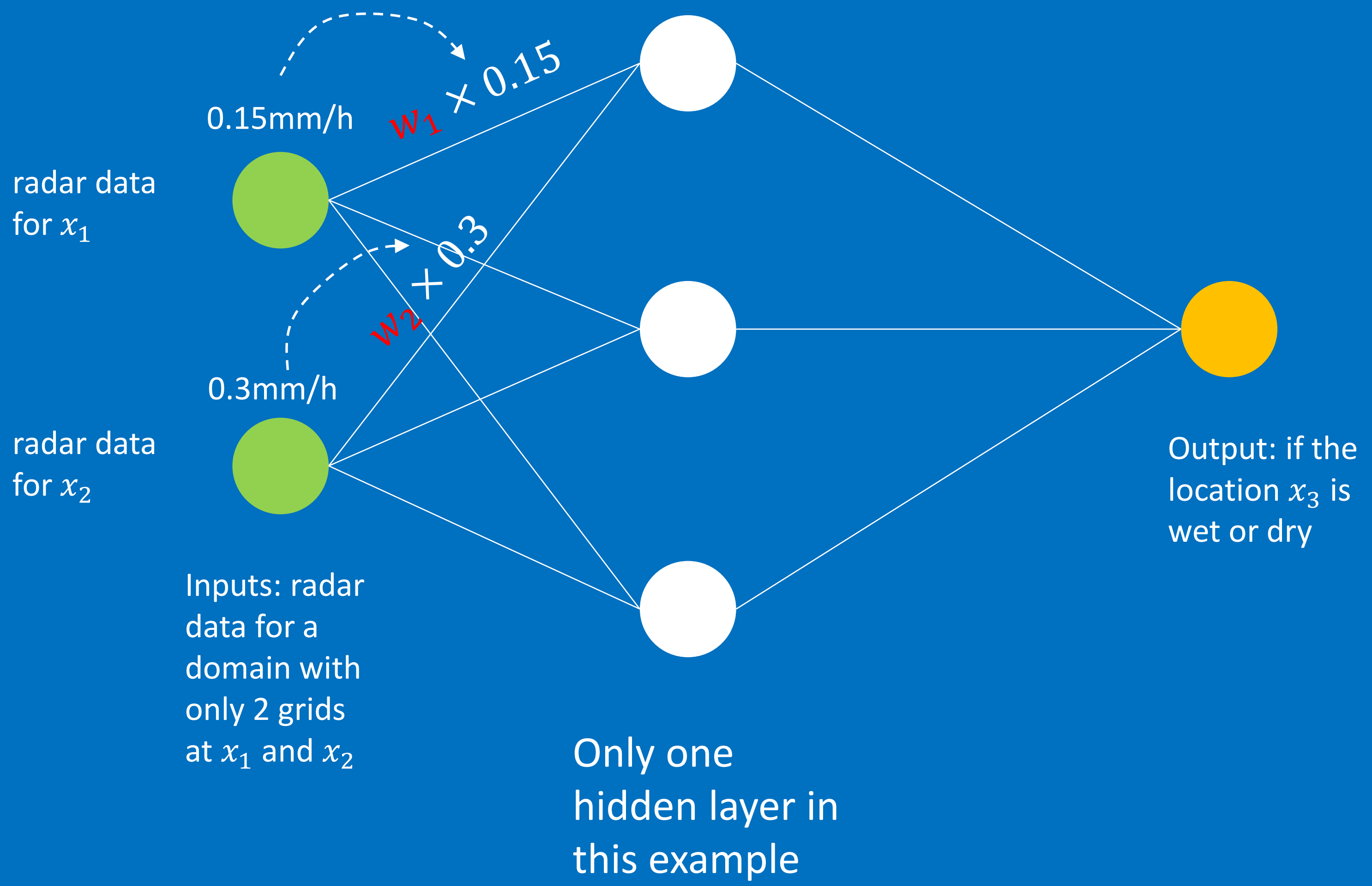

We have four neurons (white and orange) and several synapses
radar data for x_1
radar data for x_2
Inputs: radar data for a domain with only 2 grids at x_1 and x_2
Output: if the location x_3 is wet or dry
Only one hidden layer in this example

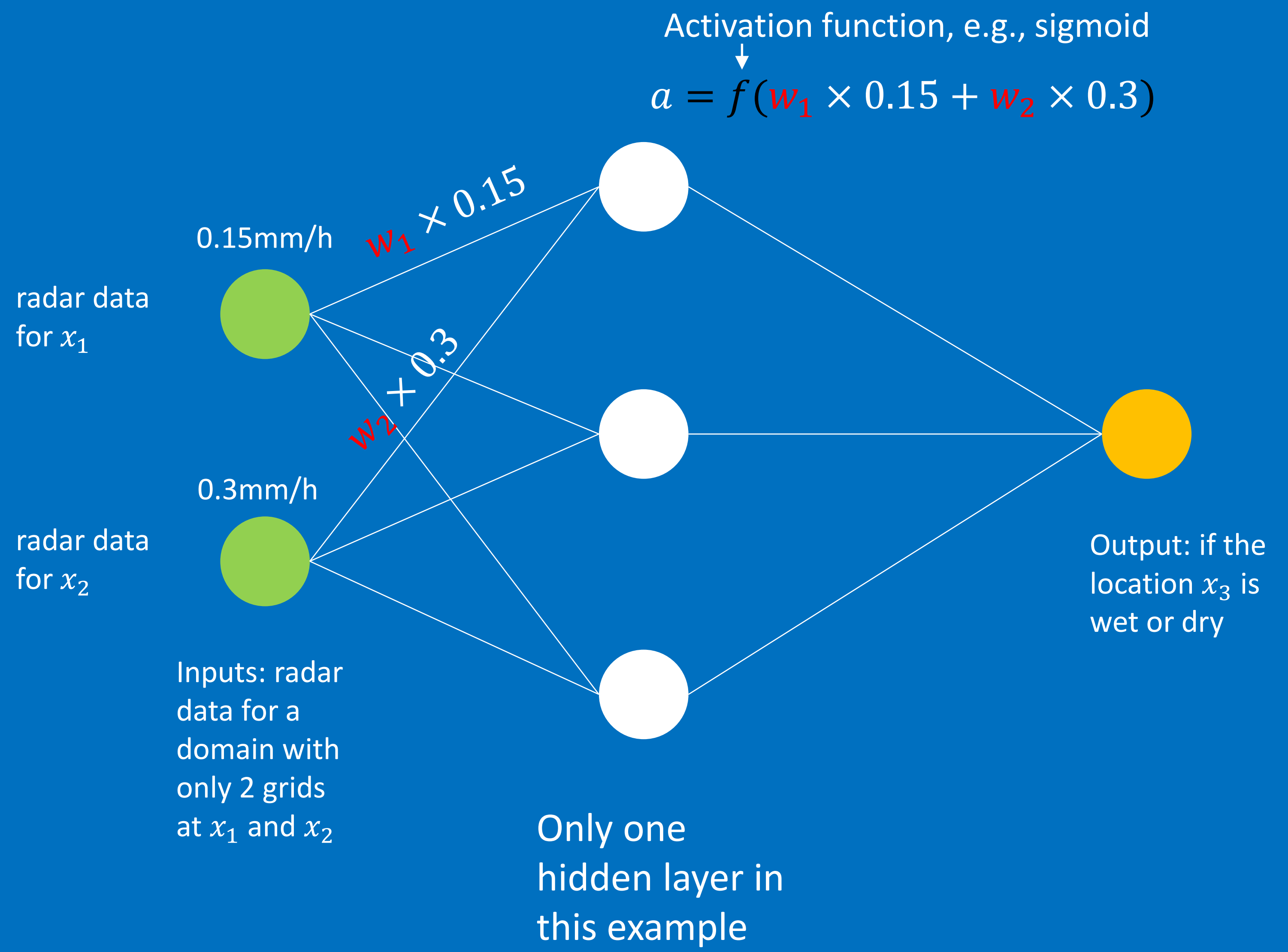


0.15mm/h
radar data for x_1
0.3mm/h
radar data for x_2
Inputs: radar data for a domain with only 2 grids at x_1 and x_2
Only one hidden layer in this example
Output: if the location x_3 is wet or dry

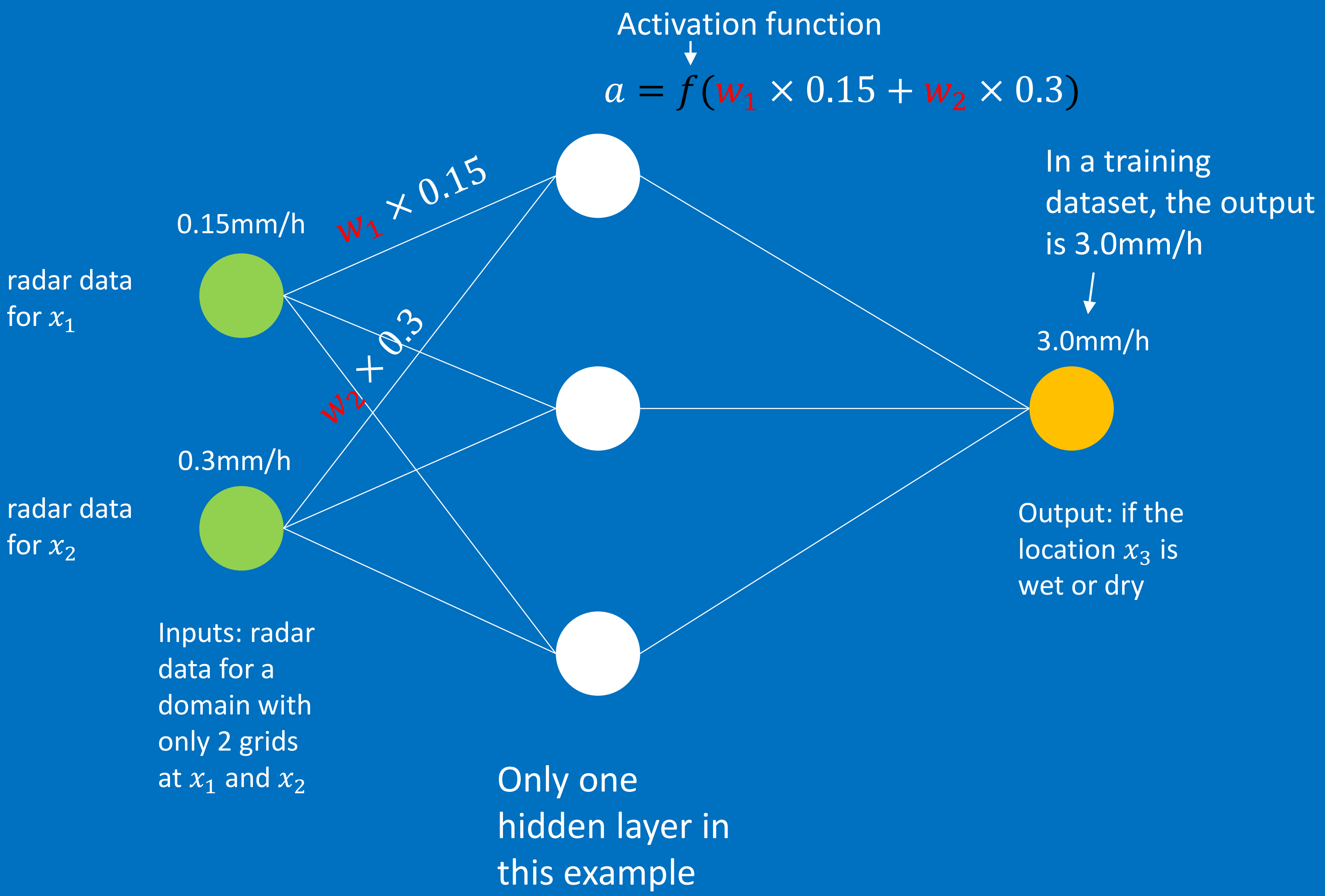
Each synapse contains:

- a weight (e.g., w_1 and w_2)
- and the contribution from previous layer (e.g., 0.15 and 0.3)





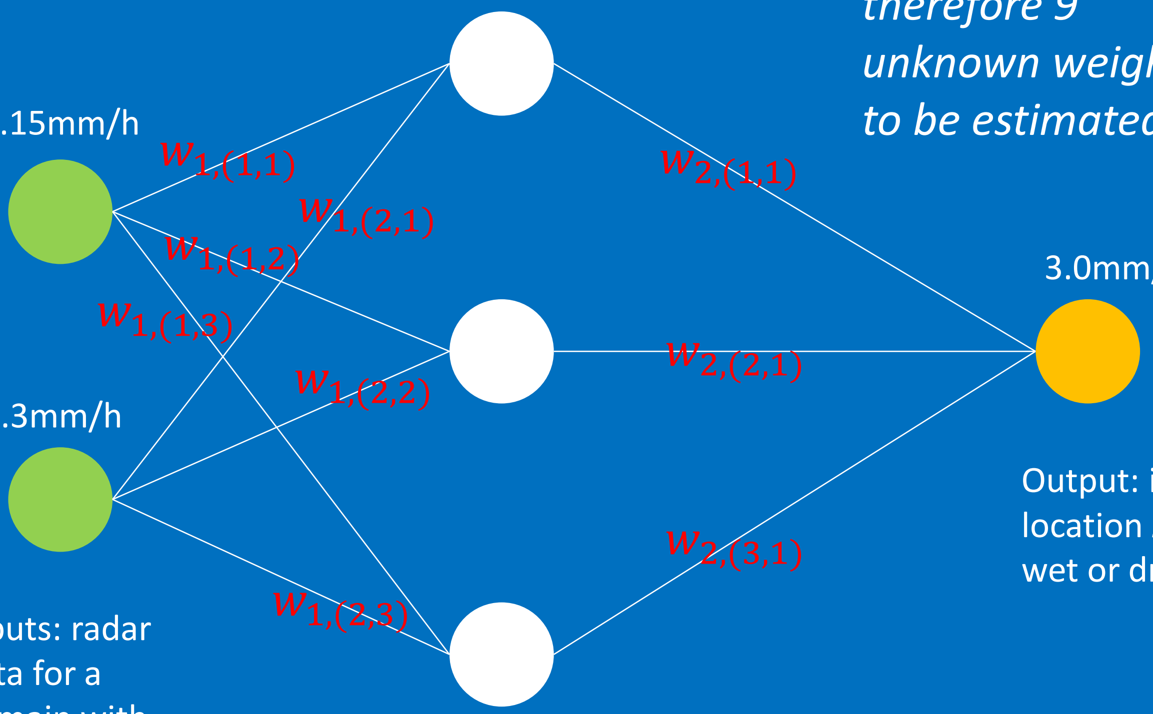
Activation function, e.g., sigmoid
$a = f(w_1 \times 0.15 + w_2 \times 0.3)$
$w_1 \times 0.15$
$w_2 \times 0.3$
0.15mm/h
radar data for x_1
0.3mm/h
radar data for x_2
Inputs: radar data for a domain with only 2 grids at x_1 and x_2
Only one hidden layer in this example
Output: if the location x_3 is wet or dry


Activation function
$a = f(w_1 \times 0.15 + w_2 \times 0.3)$
0.15mm/h
$w_1 \times 0.15$
radar data for x_1
$w_2 \times 0.3$
0.3mm/h
radar data for x_2
Inputs: radar data for a domain with only 2 grids at x_1 and x_2
Only one hidden layer in this example
In a training dataset, the output is 3.0mm/h
3.0mm/h
Output: if the location x_3 is wet or dry

During the training, there are 9 *synapses and therefore 9 unknown weights to be estimated*

radar data for x_1

radar data for x_2

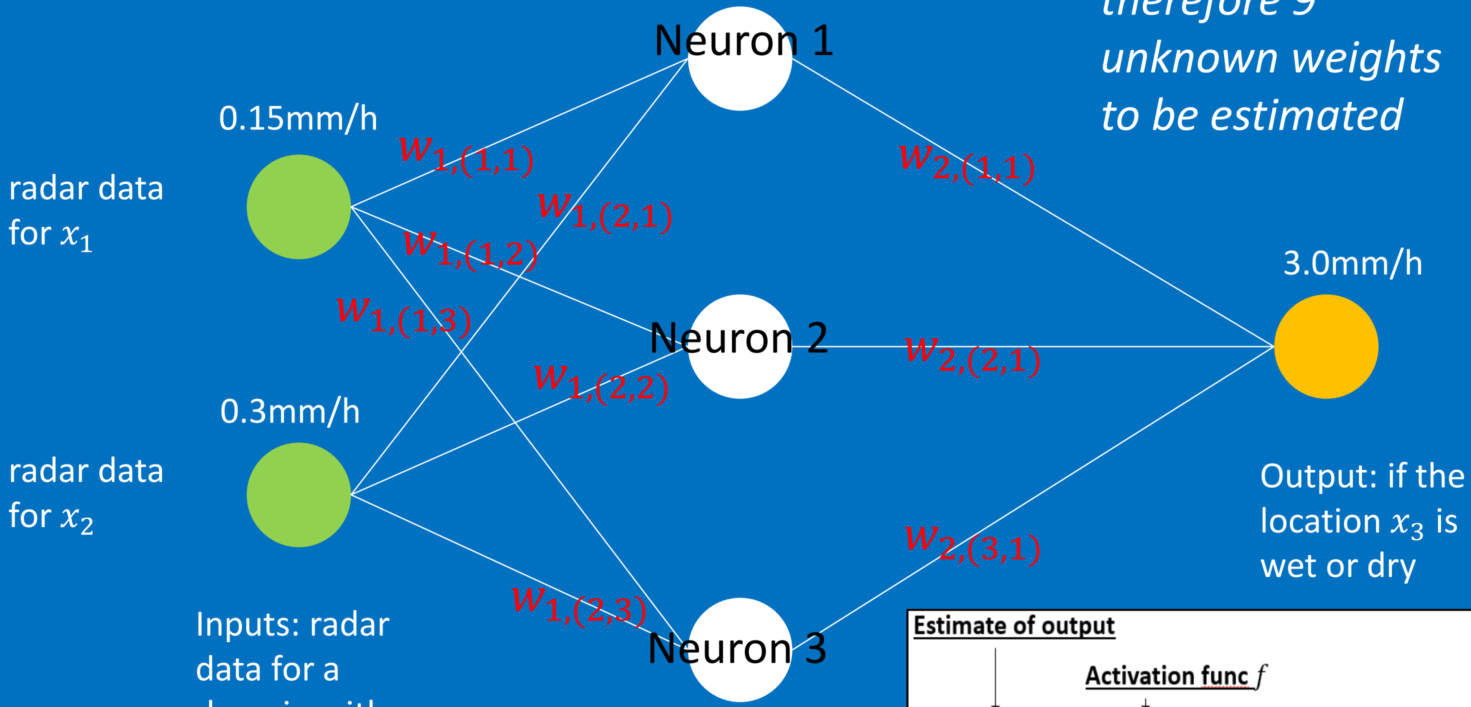


Output: if the location x_3 is wet or dry

Inputs: radar data for a domain with only 2 grids at x_1 and x_2

Only one hidden layer in this example

During the training, there are 9 *synapses and therefore 9 unknown weights to be estimated*



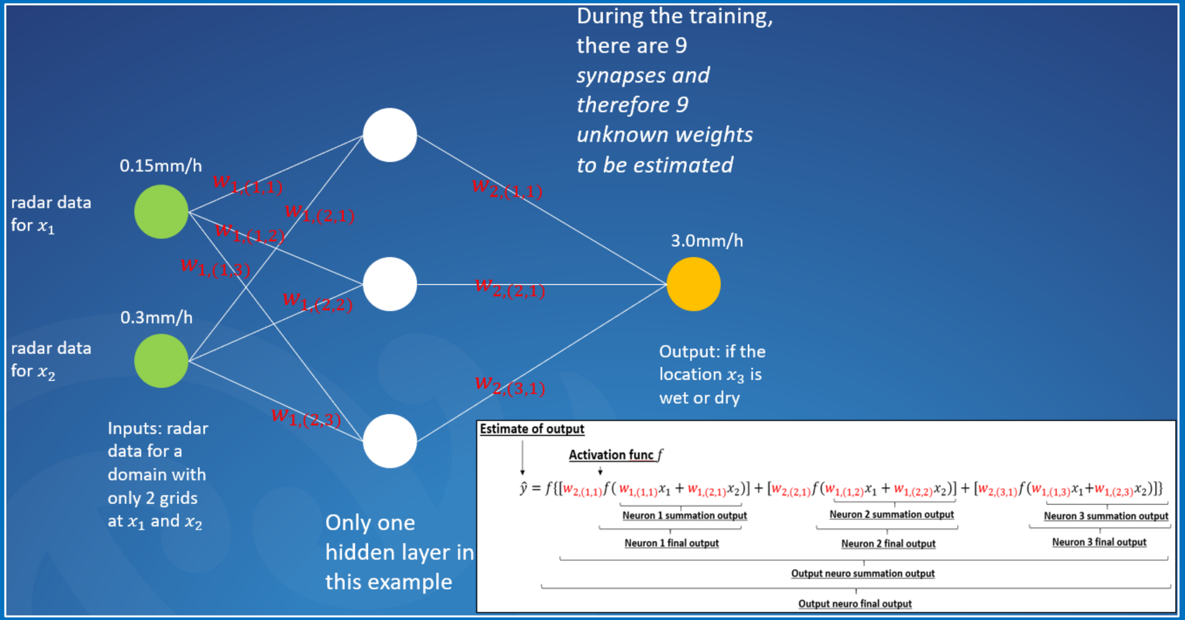
Inputs: radar data for a domain with only 2 grids at x_1 and x_2

Only one hidden layer in this example

Estimate of output

Activation func f

$$\hat{y} = \underbrace{f\{\underbrace{[w_{2,(1,1)}\underbrace{f(\underbrace{w_{1,(1,1)}x_1 + w_{1,(2,1)}x_2}_{\text{Neuron 1 summation output}})}_{\text{Neuron 1 final output}}] + [w_{2,(2,1)}\underbrace{f(\underbrace{w_{1,(1,2)}x_1 + w_{1,(2,2)}x_2}_{\text{Neuron 2 summation output}})}_{\text{Neuron 2 final output}}] + [w_{2,(3,1)}\underbrace{f(\underbrace{w_{1,(1,3)}x_1 + w_{1,(2,3)}x_2}_{\text{Neuron 3 summation output}})}_{\text{Neuron 3 final output}}]}_{\text{Output neuro summation output}}\}}_{\text{Output neuro final output}}$$



first layer weight

$$J = \sum_{i=1}^{3} \{Y - f[f(XW_n)]W_m\}_i^{\,2}$$

second layer weight

X: input layer ($m \times n$)

W : weights we want to get ($k \times p$)

Y : output layer ($m \times 1$)

f : activation function

- *m: the number of training dataset (in this case: 3)*
- *n: the number of input neuros (in this case: 2)*
- *k: the number of neuros for previous layer*
- *p: the number of neuros for next layer*

Assuming that we have 3 training datasets, the matrix form of the cost function can be constructed as:

first training dataset

first ground truth

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ x_{3,1} & x_{3,2} \end{bmatrix} = \begin{bmatrix} 0.15 & 0.3 \\ 5 & 2.4 \\ 0.5 & 2.0 \end{bmatrix} \qquad Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 3.0 \\ 0.8 \\ 1.5 \end{bmatrix}$$

$$W_1 = \begin{bmatrix} w_{1,(1,1)} & w_{1,(1,2)} & w_{1,(1,3)} \\ w_{1,(2,1)} & w_{1,(2,2)} & w_{1,(2,3)} \end{bmatrix} \qquad W_2 = \begin{bmatrix} w_{2,(1,1)} \\ w_{2,(2,1)} \\ w_{2,(3,1)} \end{bmatrix}$$

During the training, there are 9

radar data for x_1

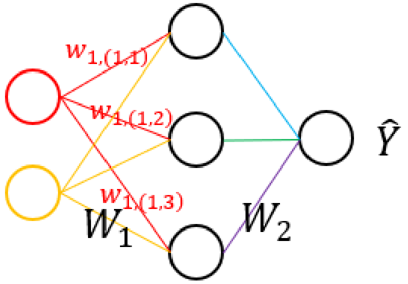
radar data for x_2

e want to get

$$\begin{bmatrix} 3.0 \\ 0.8 \\ 1.5 \end{bmatrix}$$

The structure of the neural network:

$$\hat{Y} = f\{[f(XW_1)]W_2\}$$



The components of the neural network:

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ x_{3,1} & x_{3,2} \end{bmatrix} \quad W_1 = \begin{bmatrix} w_{1,(1,1)} & w_{1,(1,2)} & w_{1,(1,3)} \\ w_{1,(2,1)} & w_{1,(2,2)} & w_{1,(2,3)} \end{bmatrix} \quad W_2 = \begin{bmatrix} w_{2,(1,1)} \\ w_{2,(2,1)} \\ w_{2,(3,1)} \end{bmatrix} \quad \hat{Y} = \begin{bmatrix} \hat{y_1} \\ \hat{y_2} \\ \hat{y_2} \end{bmatrix}$$

Solve the network (step1):

(3x2)x(2x3)=>(3x3)

$$XW_1 = \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ x_{3,1} & x_{3,2} \end{bmatrix} \begin{bmatrix} w_{1,(1,1)} & w_{1,(1,2)} & w_{1,(1,3)} \\ w_{1,(2,1)} & w_{1,(2,2)} & w_{1,(2,3)} \end{bmatrix} = \begin{bmatrix} x_{1,1}w_{1,(1,1)} + x_{1,2}w_{1,(2,1)} & x_{1,1}w_{1,(1,2)} + x_{1,2}w_{n,(2,2)} & x_{1,1}w_{1,(1,3)} + x_{1,2}w_{1,(2,3)} \\ x_{2,1}w_{1,(1,1)} + x_{2,2}w_{1,(2,1)} & x_{2,1}w_{1,(1,2)} + x_{2,2}w_{n,(2,2)} & x_{2,1}w_{1,(1,3)} + x_{2,2}w_{1,(2,3)} \\ x_{3,1}w_{1\,(1,1)} + x_{3,2}w_{1,(2,1)} & x_{3,1}w_{1,(1,2)} + x_{3,2}w_{n,(2,2)} & x_{3,1}w_{1,(1,3)} + x_{3,2}w_{1,(2,3)} \end{bmatrix}$$

Solve the network (step2):

(3x3)=>(3x3)

$$f(XW_1) = \begin{bmatrix} f(x_{1,1}w_{1,(1,1)} + x_{1,2}w_{1,(2,1)}) & f(x_{1,1}w_{1,(1,2)} + x_{1,2}w_{n,(2,2)}) & f(x_{1,1}w_{1,(1,3)} + x_{1,2}w_{1,(2,3)}) \\ f(x_{2,1}w_{1,(1,1)} + x_{2,2}w_{1,(2,1)}) & f(x_{2,1}w_{1,(1,2)} + x_{2,2}w_{n,(2,2)}) & f(x_{2,1}w_{1,(1,3)} + x_{2,2}w_{1,(2,3)}) \\ f(x_{3,1}w_{1\,(1,1)} + x_{3,2}w_{1,(2,1)}) & f(x_{3,1}w_{1,(1,2)} + x_{3,2}w_{n,(2,2)}) & f(x_{3,1}w_{1,(1,3)} + x_{3,2}w_{1,(2,3)}) \end{bmatrix}$$

Solve the network (step3):

(3x3)x(3x1)=>(3x1)

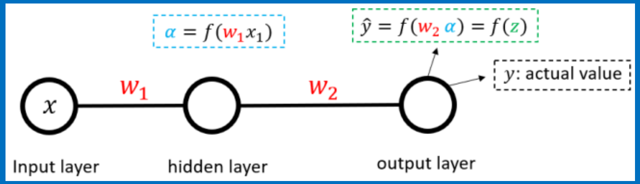
$$[f(XW_1)]W_2 = \begin{bmatrix} f(x_{1,1}w_{1,(1,1)} + x_{1,2}w_{1,(2,1)}) & f(x_{1,1}w_{1,(1,2)} + x_{1,2}w_{n,(2,2)}) & f(x_{1,1}w_{1,(1,3)} + x_{1,2}w_{1,(2,3)}) \\ f(x_{2,1}w_{1,(1,1)} + x_{2,2}w_{1,(2,1)}) & f(x_{2,1}w_{1,(1,2)} + x_{2,2}w_{n,(2,2)}) & f(x_{2,1}w_{1,(1,3)} + x_{2,2}w_{1,(2,3)}) \\ f(x_{3,1}w_{1\,(1,1)} + x_{3,2}w_{1,(2,1)}) & f(x_{3,1}w_{1,(1,2)} + x_{3,2}w_{n,(2,2)}) & f(x_{3,1}w_{1,(1,3)} + x_{3,2}w_{1,(2,3)}) \end{bmatrix} \begin{bmatrix} w_{m,(1,1)} \\ w_{m,(2,1)} \\ w_{m,(3,1)} \end{bmatrix}$$

$$= \begin{bmatrix} w_{2,(1,1)}f(x_{1,1}w_{1,(1,1)} + x_{1,2}w_{n,(2,1)}) + w_{2,(2,1)}f(x_{1,1}w_{1,(1,2)} + x_{1,2}w_{n,(2,2)}) + w_{2,(3,1)}f(x_{1,1}w_{n,(1,3)} + x_{1,2}w_{1,(2,3)}) \\ w_{2,(1,1)}f(x_{2,1}w_{n1\,(1,1)} + x_{2,2}w_{n,(2,1)}) + w_{2,(2,1)}f(x_{2,1}w_{1,(1,2)} + x_{2,2}w_{n,(2,2)}) + w_{2,(3,1)}f(x_{2,1}w_{n,(1,3)} + x_{2,2}w_{1,(2,3)}) \\ w_{2,(1,1)}f(x_{3,1}w_{1,(1,1)} + x_{3,2}w_{n,(2,1)}) + w_{2,(2,1)}f(x_{3,1}w_{1,(1,2)} + x_{3,2}w_{n,(2,2)}) + w_{2,(3,1)}f(x_{3,1}w_{n,(1,3)} + x_{3,2}w_{1,(2,3)}) \end{bmatrix}$$

Solve the network (step4):

(3x1)=>(3x1)

$$\begin{bmatrix} \hat{y_1} \\ \hat{y_2} \\ \hat{y_2} \end{bmatrix} = f\{[f(XW_1)]W_2\} = \begin{bmatrix} f\{w_{2,(1,1)}f(x_{1,1}w_{1,(1,1)} + x_{1,2}w_{n,(2,1)}) + w_{2,(2,1)}f(x_{1,1}w_{1,(1,2)} + x_{1,2}w_{n,(2,2)}) + w_{2,(3,1)}f(x_{1,1}w_{n,(1,3)} + x_{1,2}w_{1,(2,3)})\} \\ f\{w_{2,(1,1)}f(x_{2,1}w_{n1\,(1,1)} + x_{2,2}w_{n,(2,1)}) + w_{2,(2,1)}f(x_{2,1}w_{1,(1,2)} + x_{2,2}w_{n,(2,2)}) + w_{2,(3,1)}f(x_{2,1}w_{n,(1,3)} + x_{2,2}w_{1,(2,3)})\} \\ f\{w_{2,(1,1)}f(x_{3,1}w_{1,(1,1)} + x_{3,2}w_{n,(2,1)}) + w_{2,(2,1)}f(x_{3,1}w_{1,(1,2)} + x_{3,2}w_{n,(2,2)}) + w_{2,(3,1)}f(x_{3,1}w_{n,(1,3)} + x_{3,2}w_{1,(2,3)})\} \end{bmatrix}$$



In this case, there are two variables (or weights) w_1 and w_1 to be solved, therefore we can write the total gradient in the format as:

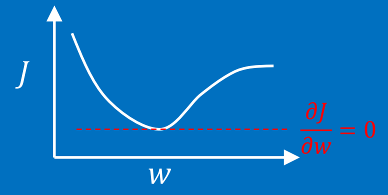
$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial w_1} + \frac{\partial J}{\partial w_2}$$

Where "J" is the error of the estimated state, e.g.,

$$J = \sum_{i=1}^{3} \{Y - f[f(XW_n)]W_m\}_i^2$$

Truth Estimated state

The purpose is to get the minimum "J", and the dependent variable is weight "w", and the minimum "J" happens when the gradient $\frac{\partial J}{\partial w} = 0$ (or close to zero)



The gradient is derived from backward (that's why we call it backpropagation) so first let us look at W_2,

(1) the gradient of W_2 is $\frac{\partial J}{\partial w_2}$, which can be represented as:

$$\frac{\partial J}{\partial w} = \frac{\partial (y-\hat{y})^2}{\partial w}$$

(2) let's multiple ½ to the gradient $\frac{\partial (y_i - \hat{y_i})^2}{\partial w}$ to make it simpler, so the power rule gives us:

$$\frac{\partial J}{\partial w} = \frac{\partial \frac{1}{2}(y-\hat{y})^2}{\partial w} = \frac{\partial (y-\hat{y})}{\partial w}$$

(3) Then we apply the chain rule "$\frac{df(x)}{dx} = \frac{df(x)}{df}\frac{df}{dx}$" to the derivation:

$$\frac{\partial J}{\partial w} = \frac{\partial (y-\hat{y})}{\partial w} = -(y-\hat{y})\frac{\partial (\hat{y})}{\partial w}$$

(4) Since $\hat{y}$ is the function of the activation function, $\hat{y} = f(z)$ then:

$$\frac{\partial J}{\partial w} = \frac{\partial (y-\hat{y})}{\partial w} = -(y-\hat{y})\frac{\partial (\hat{y})}{\partial w} = -(y-\hat{y})f'(z)\frac{\partial (z)}{\partial w}$$

Where $z = w_2\, \alpha$, where α is the value from the upstream layer from the last layer (as Figure 2.14).

(5) Since $z = w_2\, \alpha$ and α is the function of w_1, so $\frac{\partial (z)}{\partial w} = w_2 \frac{\partial (\alpha)}{\partial w_1}$

$$\frac{\partial J}{\partial w} = -(y-\hat{y})f'(z)\frac{\partial (z)}{\partial w} = -(y-\hat{y})f'(w_2\, \alpha)w_2\frac{\partial \alpha}{\partial w_1}$$

(6) we also have $\alpha = f(w_1 x)$, let us assume $\beta = w_1 x$. therefore

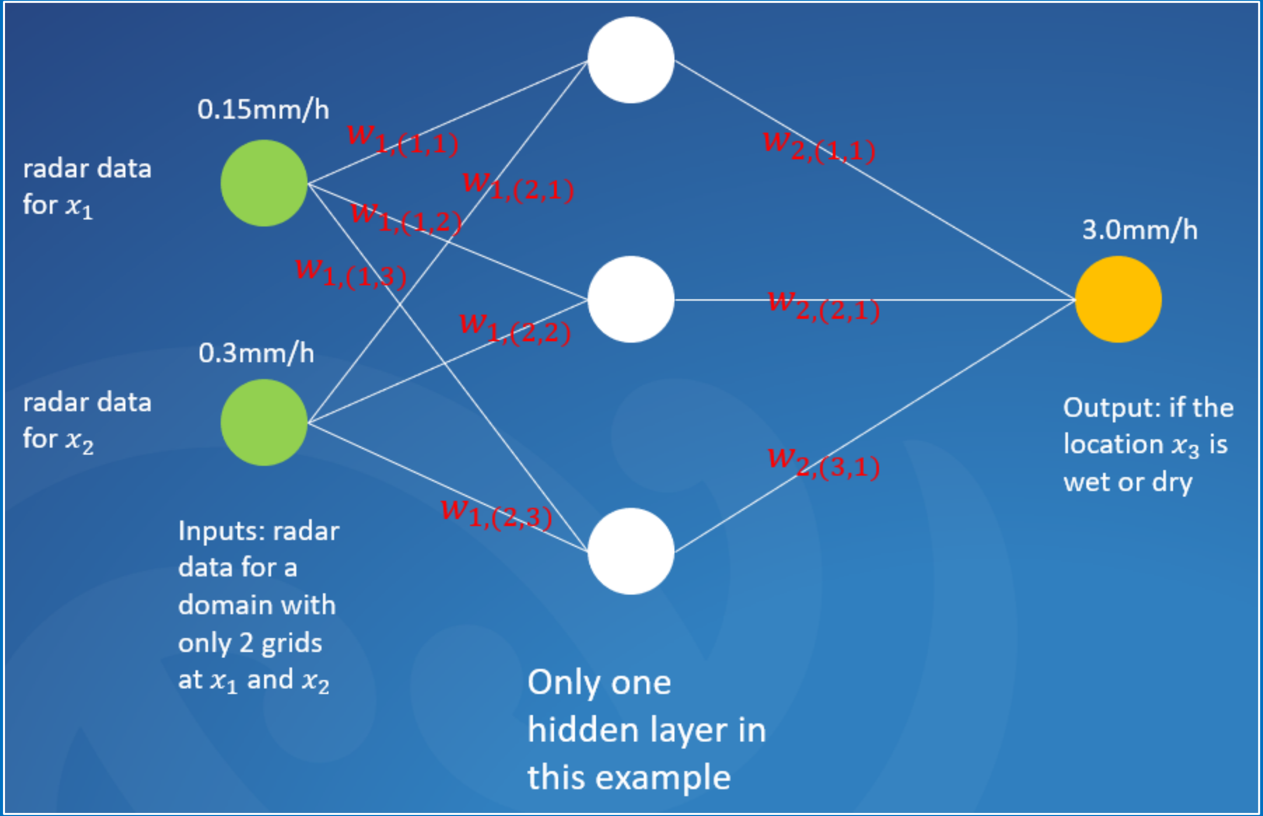
$$\frac{\partial J}{\partial w_n} = -(y-\hat{y})f'(w_2\, \alpha)w_2\frac{\partial \alpha}{\partial w_1} = -(y-\hat{y})f'(w_2\, \alpha)w_2\, f'(w_1\, x)w_1\, x$$

Where $\alpha = f(w_1 x)$:

$$\frac{\partial J}{\partial w} = -(y-\hat{y})w_1\, w_2 f'[w_2\, f(w_1 x)]f'[w_1\, x]x$$

In the most optimal situation, which we can find the gradient equals to zero, with the known x and y in the training process, we can easily get the weights for w_1 and w_2 ~ this process is called the optimal interpolation method.

After the minimization of the above cost function, we can get the optimal combination of $\boldsymbol{W}$, and this combination then is used to do the prediction following the procedure described in Figure 2.12


0.15mm/h
radar data for x_1
0.3mm/h
radar data for x_2
Inputs: radar data for a domain with only 2 grids at x_1 and x_2
$w_{1,(1,1)}$
$w_{1,(2,1)}$
$w_{1,(1,2)}$
$w_{1,(1,3)}$
$w_{1,(2,2)}$
$w_{1,(2,3)}$
$w_{2,(1,1)}$
$w_{2,(2,1)}$
$w_{2,(3,1)}$
3.0mm/h
Output: if the location x_3 is wet or dry
Only one hidden layer in this example