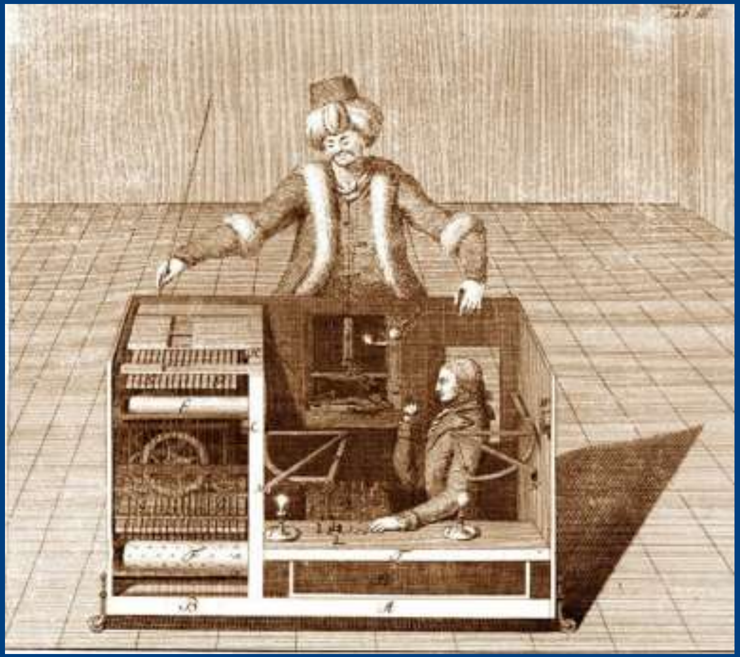
Make your own neural network



Qi Liu（刘琦)

Shaanxi normal university

flyliuqi@gmail.com

The content of the book

- some powerful and efficient math.
- enough Python.
- improve the neural network and have fun.

How they work

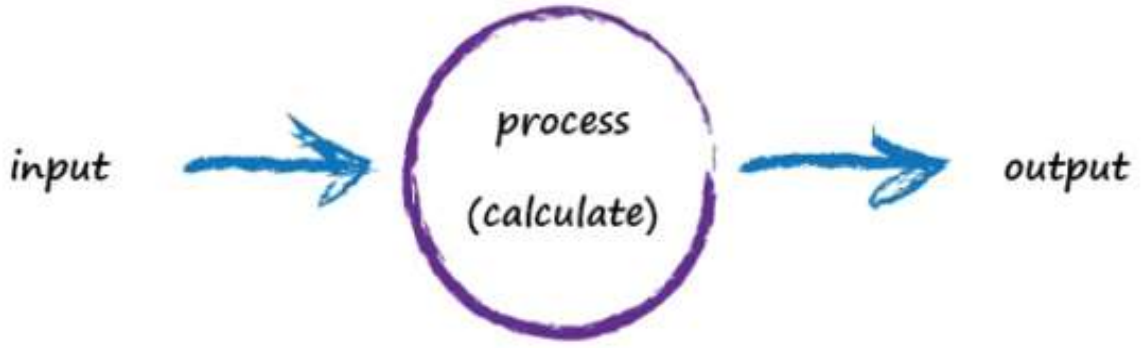
Take inspiration from all the small things around you.

Algorithm → give an impression of a human

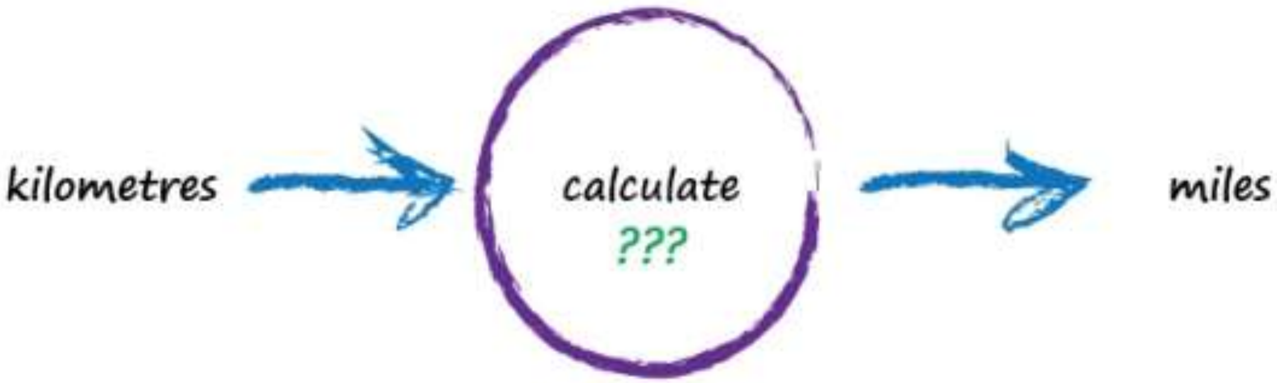
Human:

Computer:

A Simple Predicting Machine

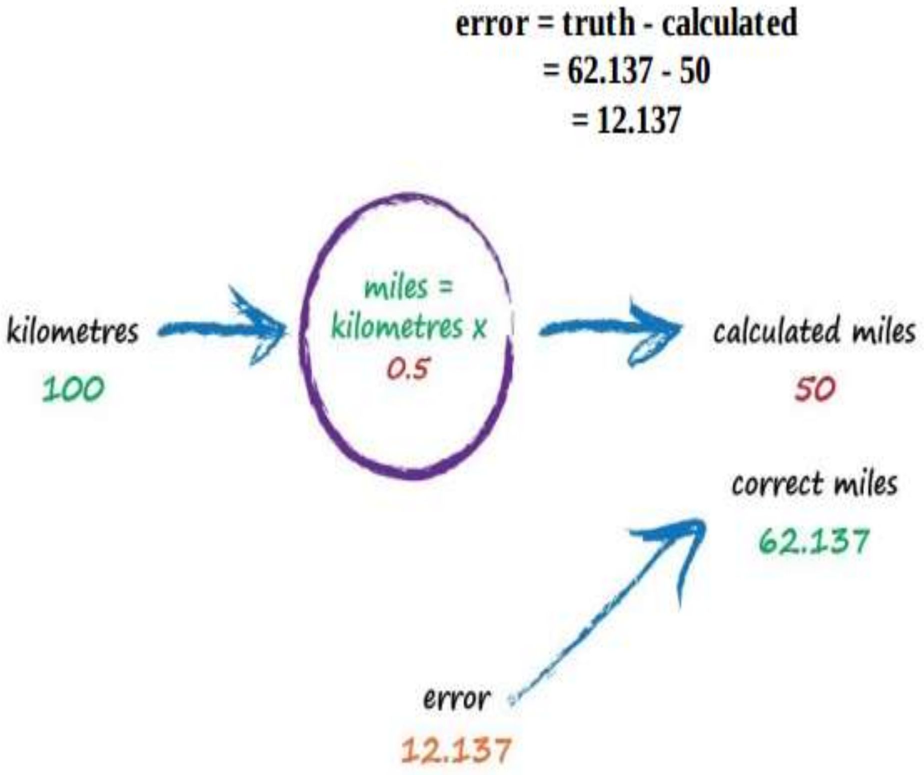


Ramp up complexity just a tiny notch

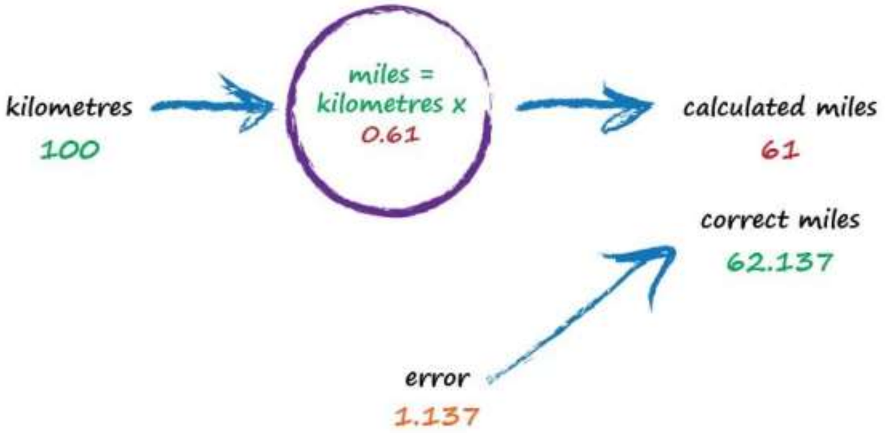


Linear model: input doubles, output doubles.

Try and test it

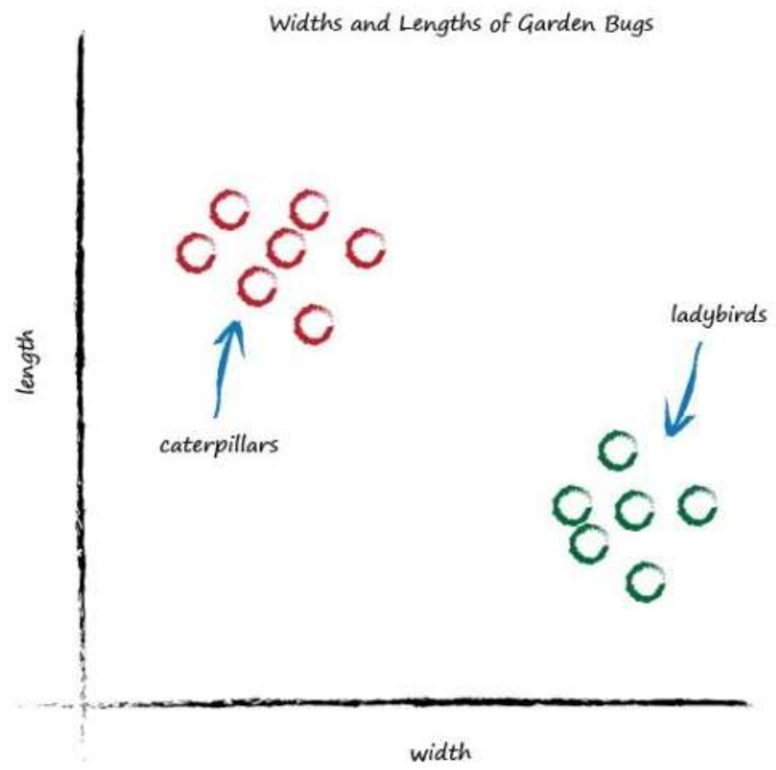


Error: truth-calculated



A good way of refining these models is to adjust the parameters based on how wrong the model is compared to known true examples.

Classifying is Not Very Different from Predicting



Use the straight line to **separate** different classes.

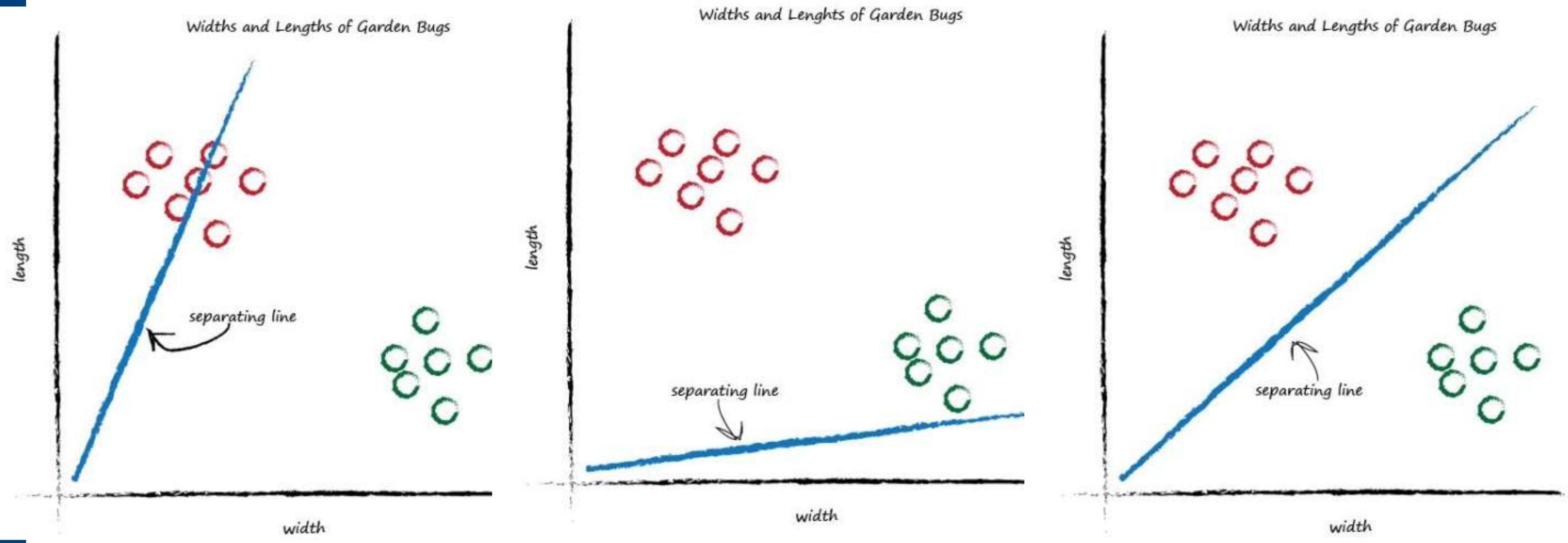
Solve it

$$E = t - y = Ax + (\Delta A)x - Ax$$

$$E = (\Delta A)x$$

$$\Delta A = E / x$$

陕西师范大学



Ingore some of the previous value which was arrived at through potentially many previous training iterations.

Moderate the updates

陕西师範大學

instead of jumping enthusiastically to each new A, we take a fraction of the change of A. This way we move in the direction that the training example suggests, but do so slightly cautiously

↓

Learning rate

$$\Delta A = L(E/x)$$

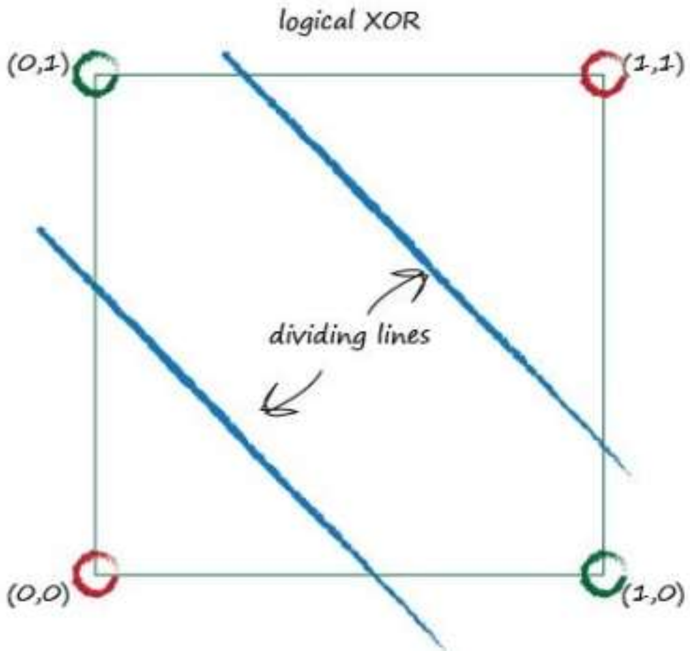
no single training example totally dominates the learning

dampen the impact of those errors or noise and smooth them out

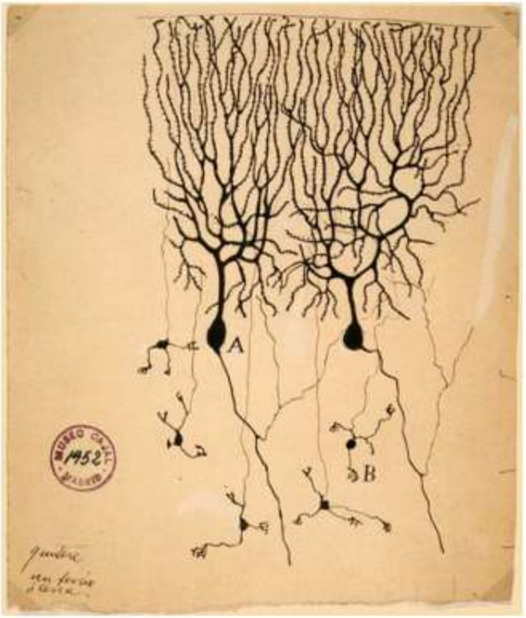
One classifier may be not enough

Example

Input A	Input B	Logical XOR
0	0	0
0	1	1
1	0	1
1	1	0



many linear lines can start to separate off even unusually shaped regions for classification.



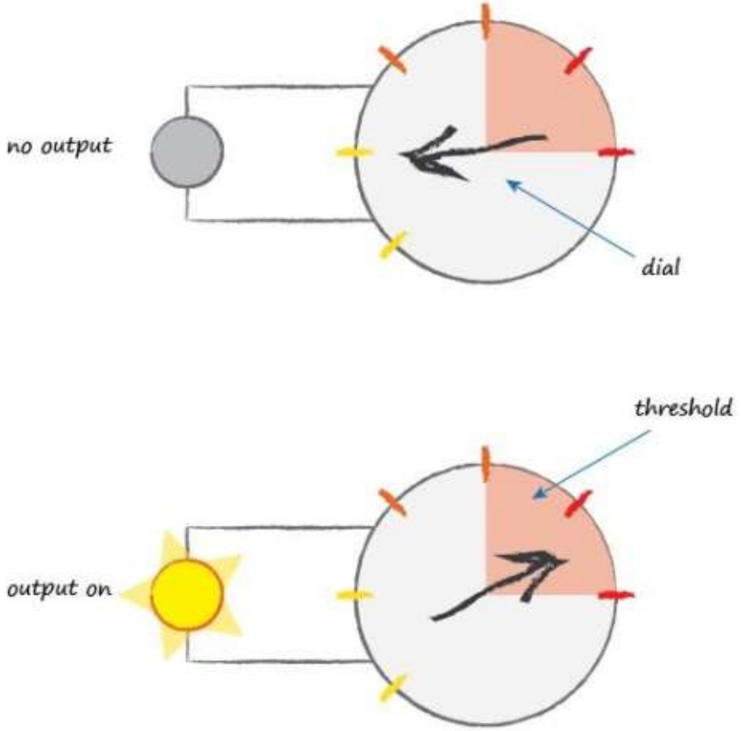
process signals in parall

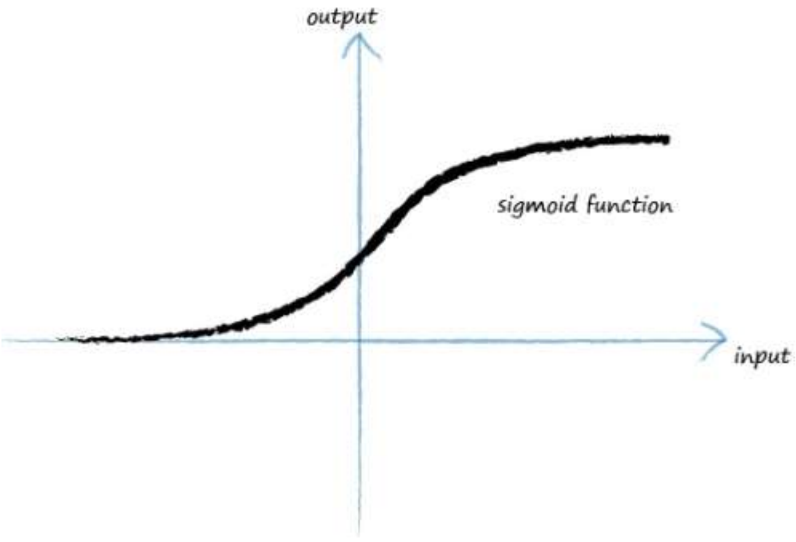
fuzziness

incredibly resilient to damage and imperfect signals

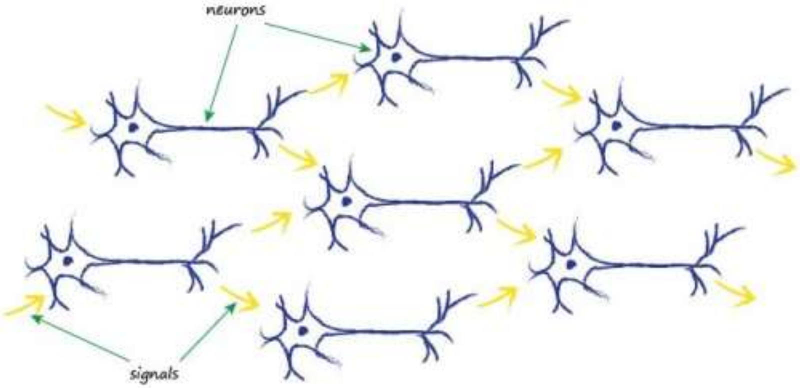
suppress the input until it has grown so large that it triggers an output

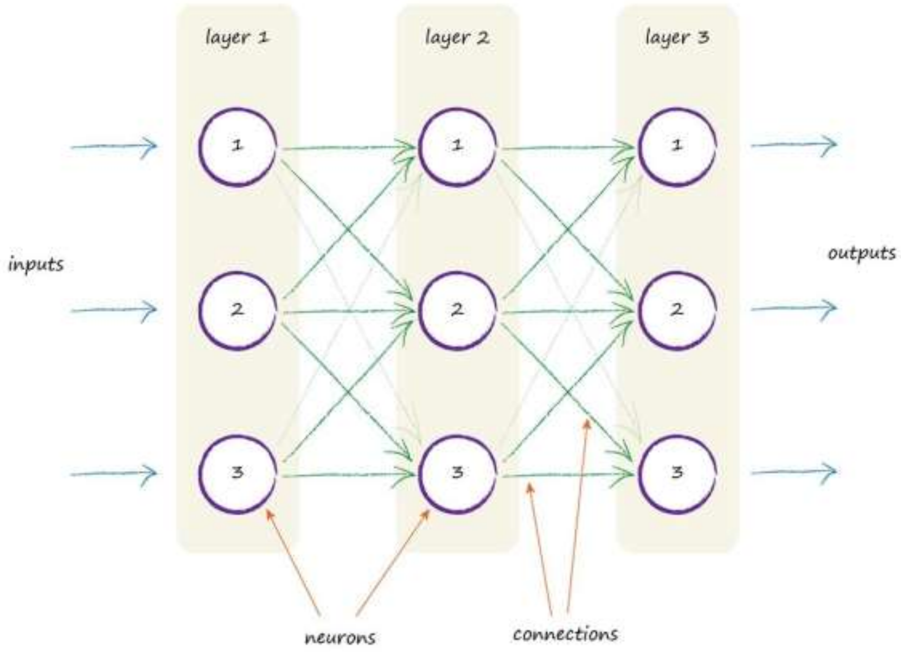
threshold ⇨ activation function





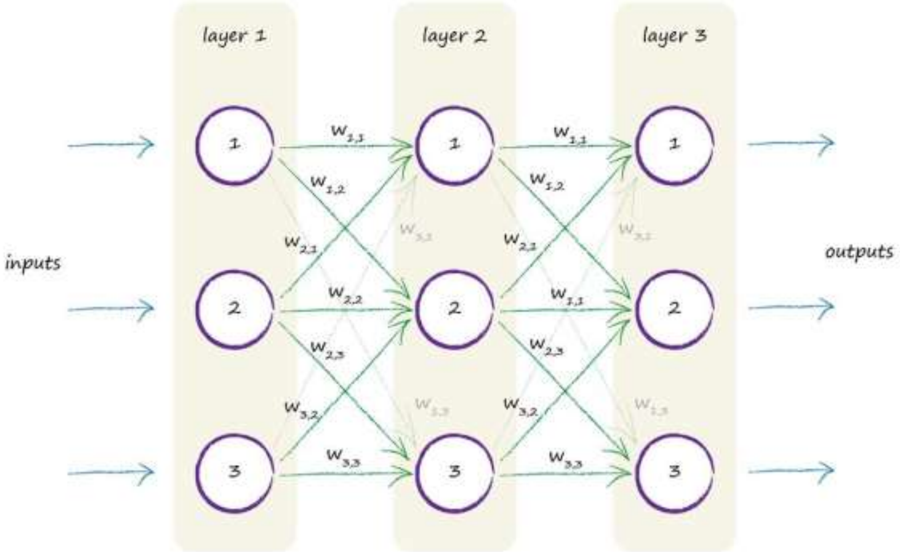
$$y = \frac{1}{1 + e^{-x}}$$





connected neurons **rtificial neural networks**

weight → To adjust the slope



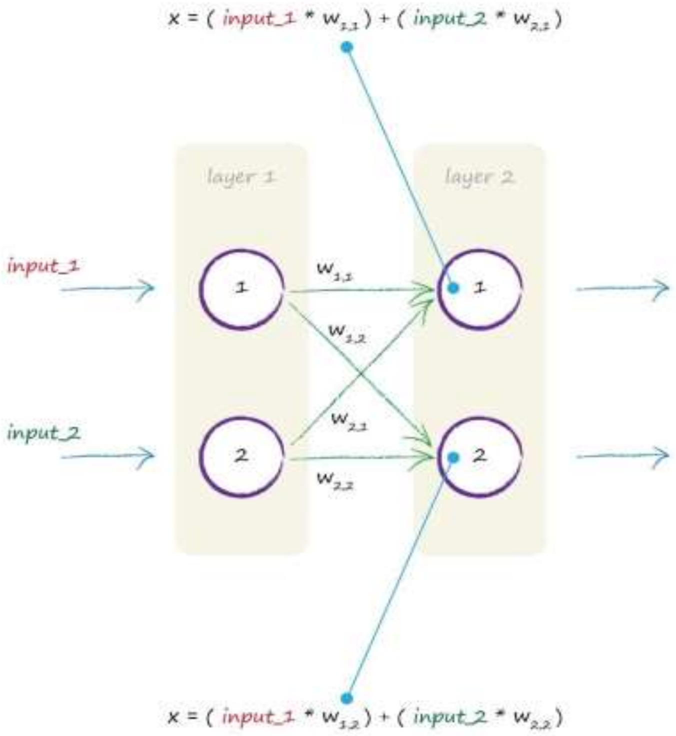
A low weight will de-emphasise a signal, and a high weight will amplify it.

Pay attention: **The first layer: not do anything other than represent the input signals. not apply an activation function.**

Why full connectivity??

- easier to encode as computer instructions
- some weights become zero or close to zero, the link is effectively broken.

Matrix Multiplication is useful



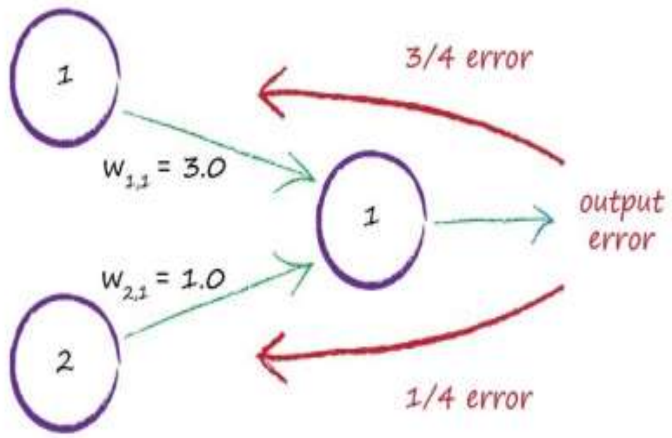
$$\begin{pmatrix} w_{1,1} & w_{2,1} \\ w_{1,2} & w_{2,2} \end{pmatrix} \begin{pmatrix} \text{input\_1} \\ \text{input\_2} \end{pmatrix} = \begin{pmatrix} (\text{input\_1} * w_{1,1}) + (\text{input\_2} * w_{2,1}) \\ (\text{input\_1} * w_{1,2}) + (\text{input\_2} * w_{2,2}) \end{pmatrix}$$

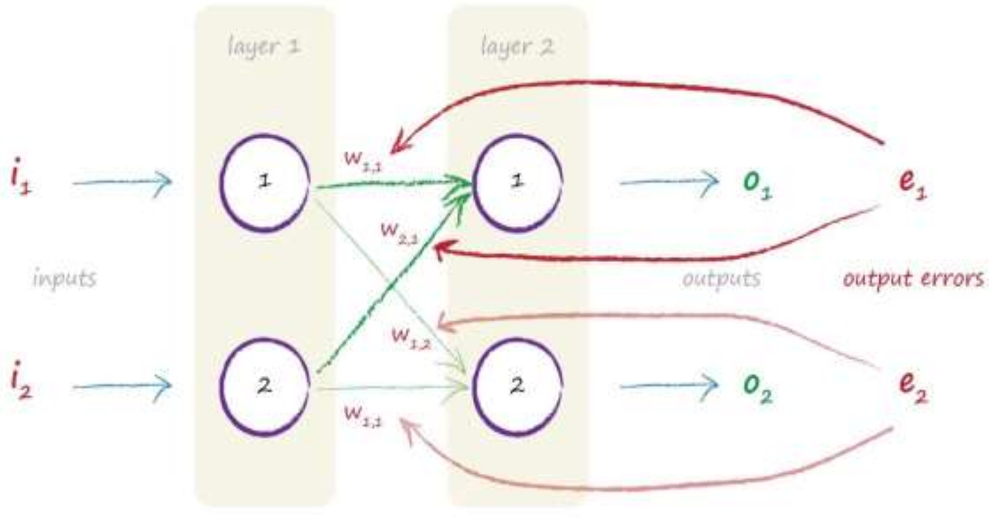
$$\mathbf{X} = \mathbf{W} \cdot \mathbf{I}$$

$$O = \text{sigmoid}(X)$$

陕西师范大学

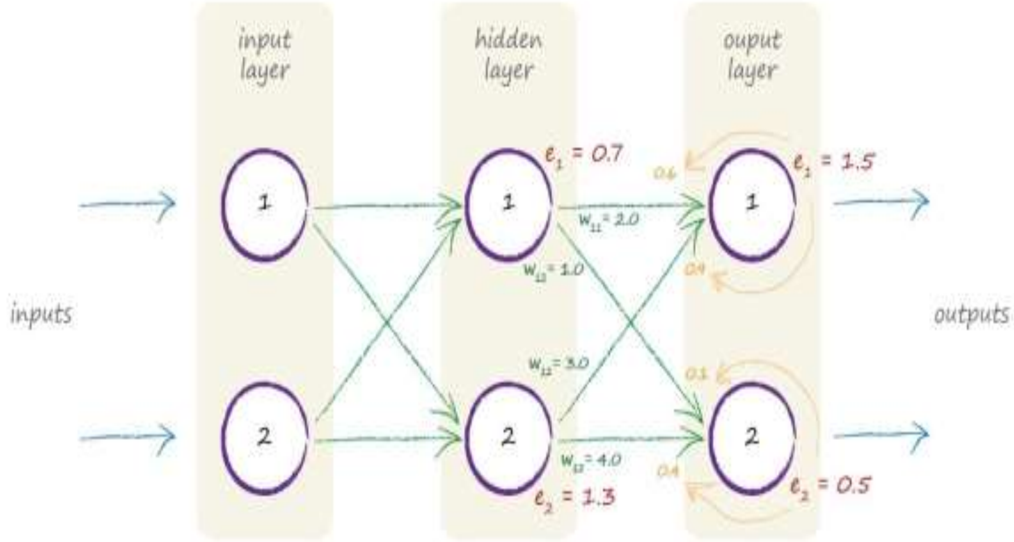
use that error to refine the neural network



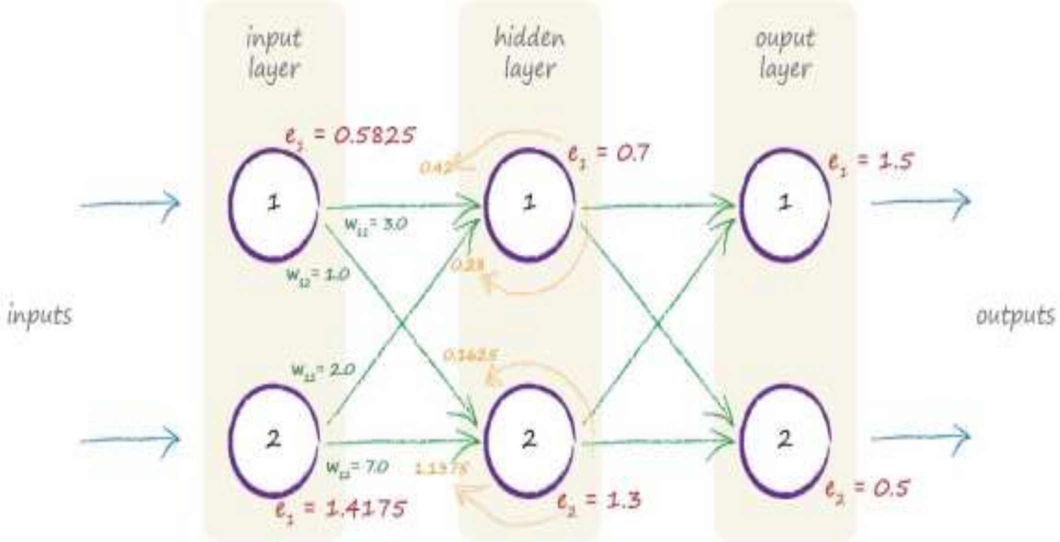


backpropagation

Backpropagating Errors To More Layers



1



2

Backpropagating Errors with Matrix Multiplication

陕西師範大學

$$error_{hidden} = \begin{pmatrix} (e_1 * w_{11}) + (e_2 * w_{12}) \\ (e_1 * w_{21}) + (e_2 * w_{22}) \end{pmatrix}$$

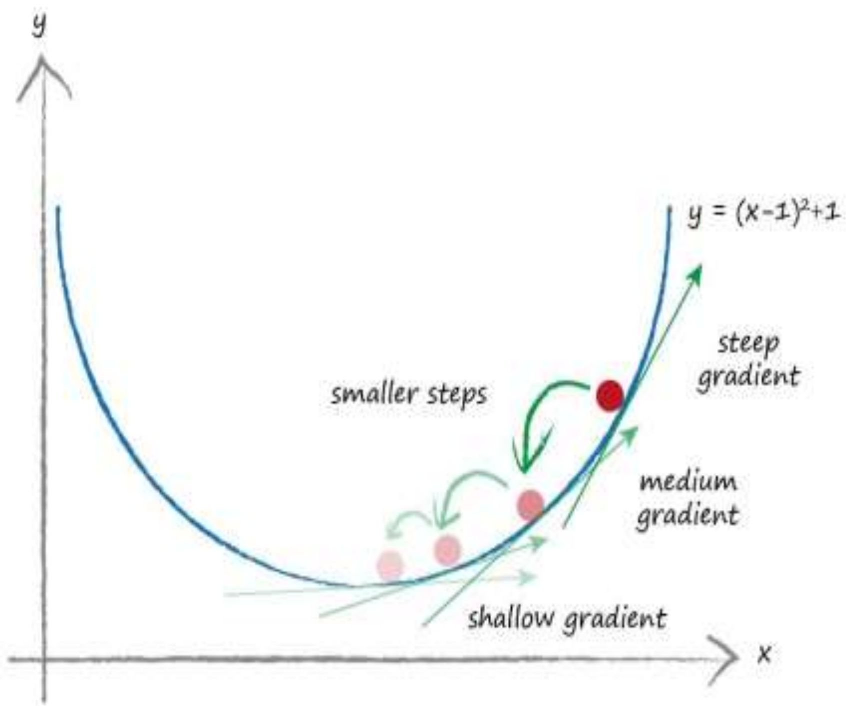
$$error_{hidden} = w^T_{hidden\_output} \cdot error_{output}$$

$$error_{hidden} = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \cdot \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}$$

It’s too hard, right?

$$o_k = \frac{1}{1 + e^{-\sum_{j=1}^{3}\left(w_{j,k} \cdot \frac{1}{1+e^{-\sum_{i=1}^{3}(w_{i,j} \cdot x_i)}}\right)}}$$

~~brute force~~ **gradient descent**


y
y = (x-1)²+1
steep
gradient
smaller steps
medium
gradient
shallow gradient
x

Cost function

Network Output	Target Output	Error (target - actual)	Error |target - actual|	Error $(target - actual)^2$
0.4	0.5	0.1	0.1	0.01
0.8	0.7	-0.1	0.1	0.01
1.0	1.0	0	0	0
Sum		0	0.2	0.02

(target - actual): positive and negative errors cancel each other out.

|target - actual|: no continuous near the minimum

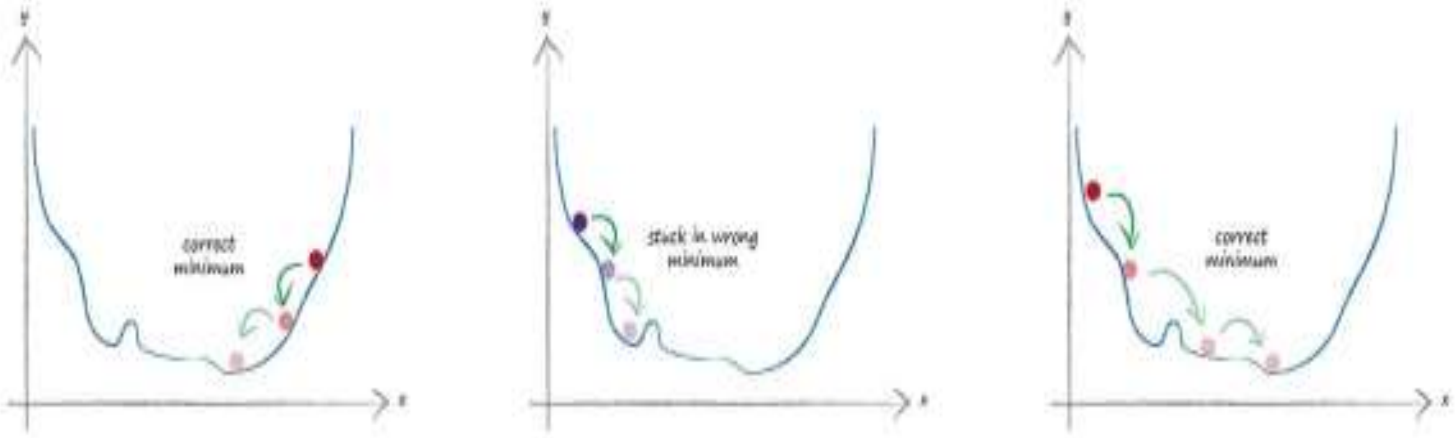
Not get smaller closer to the minimum

(target - actual)2: easy enough

smooth and continuous

gets smaller nearer the minimum

How to avoid ending up in the wrong valley?



starting from different points

choosing **different starting link weights**

陕西師範大學

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} \Sigma_n (t_n - o_n)^2$$

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} (t_k - o_k)^2$$

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial o_k} \cdot \frac{\partial o_k}{\partial w_{jk}}$$

$$\frac{\partial E}{\partial w_{jk}} = -2(t_k - o_k) \cdot \frac{\partial o_k}{\partial w_{jk}}$$

$$\frac{\partial E}{\partial w_{jk}} = -2(t_k - o_k) \cdot \frac{\partial}{\partial w_{jk}} \text{sigmoid}(\Sigma_j w_{jk} \cdot o_j)$$

$$\frac{\partial}{\partial x} \text{sigmoid}(x) = \text{sigmoid}(x)\left(1 - \text{sigmoid}(x)\right)$$

陕西師範大學

$$\frac{\partial E}{\partial w_{jk}} = -2(t_k - o_k) \cdot sigmoid(\Sigma_j w_{jk} \cdot o_j)(1 - sigmoid(\Sigma_j w_{jk} \cdot o_j)) \cdot \frac{\partial}{\partial w_{jk}}(\Sigma_j w_{jk} \cdot o_j)$$

$$= -2(t_k - o_k) \cdot sigmoid(\Sigma_j w_{jk} \cdot o_j)(1 - sigmoid(\Sigma_j w_{jk} \cdot o_j)) \cdot o_j$$

$$\boxed{\frac{\partial E}{\partial w_{jk}} = -(t_k - o_k) \cdot sigmoid(\Sigma_j w_{jk} \cdot o_j)(1 - sigmoid(\Sigma_j w_{jk} \cdot o_j)) \cdot o_j}$$

陕西师範大學

$$new\ w_{jk} = old\ w_{jk} - \alpha \cdot \frac{\partial E}{\partial w_{jk}}$$

$$\Delta w_{jk} = \alpha * E_k * sigmoid(O_k) * (1 - sigmoid(O_k)) \cdot O_j^T$$

thanks