Neural Networks

Supervised learning

The Data Science Process

- Ask an interesting question
- Get the Data
- Explore the Data
- Model the Data
  - Build a model
  - Fit the model
  - Validate the model
- Communicate/Visualize the Results

Machine Learning

	Supervised learning	*Unsupervised learning*
Discrete	Classification of categorization	Clustering
Continuous	Regression	Dimensionality Reduction

Machine Learning

$X_1, X_2, \ldots, X_p$
predictors
features
covariates

$Y_1, Y_2, \ldots, Y_m$
outcome
response variable
Continuous variable

TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5

n observations

p predictors

Machine Learning

$X_1, X_2, \ldots, X_p$
predictors
features
covariates

Regression

$Y_1, Y_2, \ldots, Y_m$
outcome
response variable
Continuous variable

TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5

n observations

p predictors

Machine Learning

$X_1, X_2, \ldots, X_p$
predictors
features
covariates

Classification

$Y_1, Y_2, \ldots, Y_m$
outcome
response variable
Binary variable

TV	Radio	Newspaper	Sales	Sales > T (15)
230.1	37.8	69.2	22.1	1
44.5	39.3	45.1	10.4	0
17.2	45.9	69.3	9.3	0
151.5	41.3	58.5	18.5	1

n observations

p predictors

Statistical Model

We assume that the response variable, Y, relates to the predictors, X, through some unknown function expressed generally as:

$$Y = f(X) + b$$

Here, f is the unknown function expressing an underlying rule for relating Y to X, b is the amount (unrelated to X) that Y differs from the rule $f(X)$.

A ***statistical model*** is any algorithm that estimates f. We denote the estimated function as $\hat{f}$.

Statistical Model

We assume that the response variable, Y, relates to the predictors, X, through some unknown function expressed generally as:

$$Y = f(X) + b$$

The question is then, how well do we want to know f / $\hat{f}$?

- We want to understand the mechanism underlying the data: in this case we really need to find the “correct” expression for f.

- We only want to use $\hat{f}$ to predict the response for yet unknown predictor values: we may not need a correct expression for f, as long as the prediction is correct!

Statistical Model

We assume that the response variable, Y, relates to the predictors, X, through some unknown function expressed generally as:

$$Y = f(X) + b$$

Building a model, non necessarily realistic, for f:

- Try polynomial function
- Try any mathematical function

Statistical Model

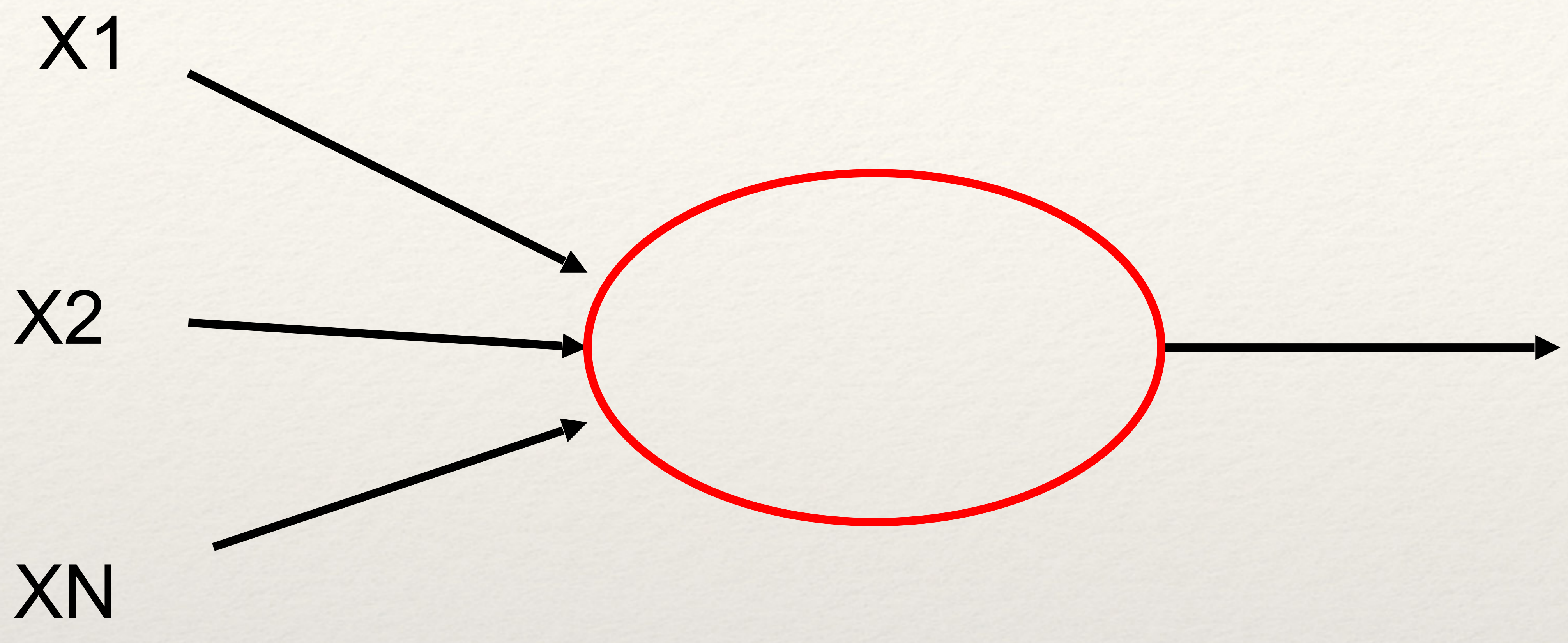
We assume that the response variable, Y, relates to the predictors, X, through some unknown function expressed generally as:

$$Y = f(X) + b$$

Building a model, non necessarily realistic, for f:

- Try polynomial function
- Try any mathematical function
- Use a simple tool for building f: the perceptron!

The perceptron



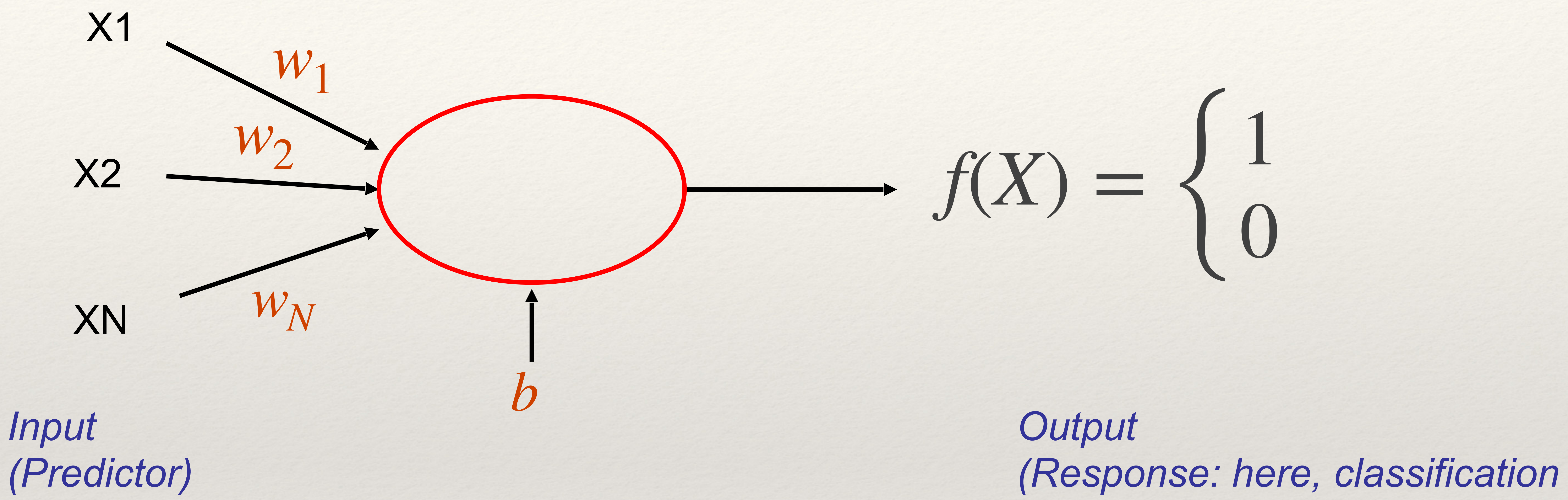
$$f(X) = \begin{cases} 1 \\ 0 \end{cases}$$

Input
(Predictor)

Output
(Response: here, classification

The perceptron classifies the input vector X into two categories.

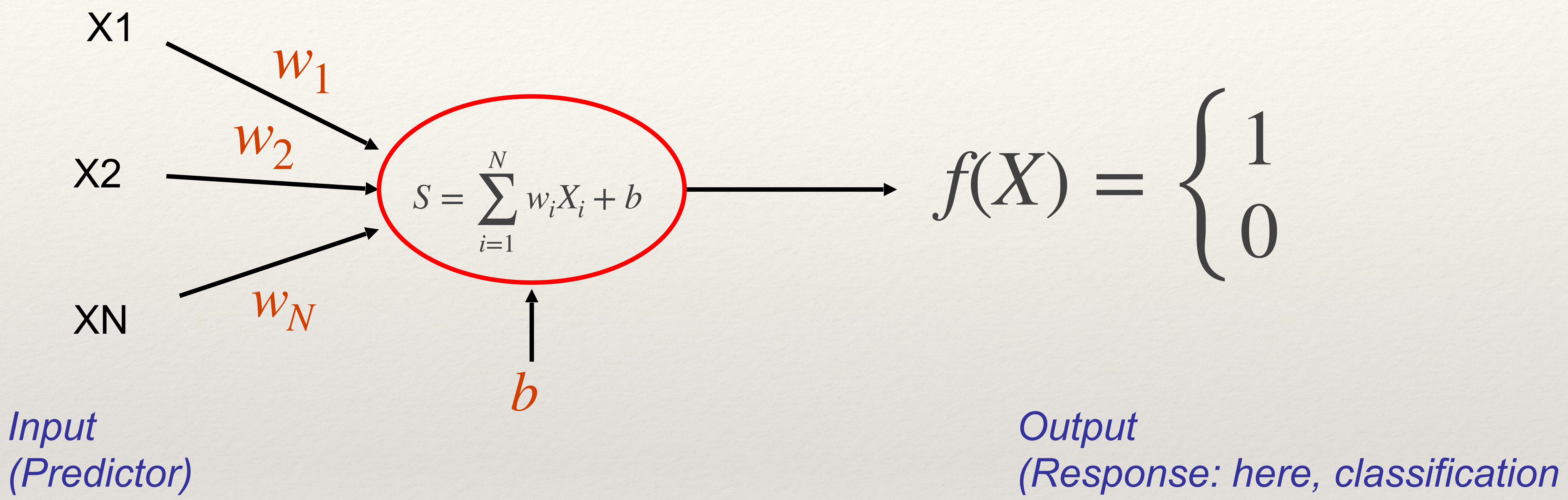
The perceptron



The perceptron classifies the input vector X into two categories.

Add weights ***w*** to all predictors, and a bias ***b***

The perceptron

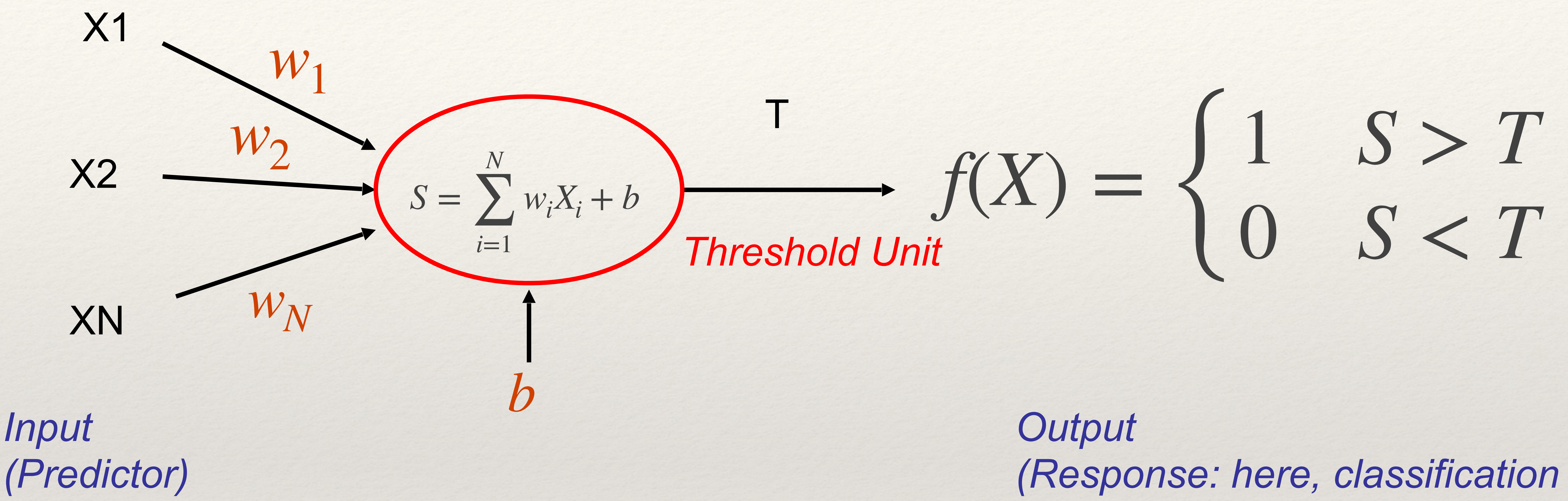


The perceptron classifies the input vector X into two categories.

Add weights ***w*** to all predictors, and a bias ***b***

Define simple "function" S as weighted sum of the predictors plus a bias

The perceptron



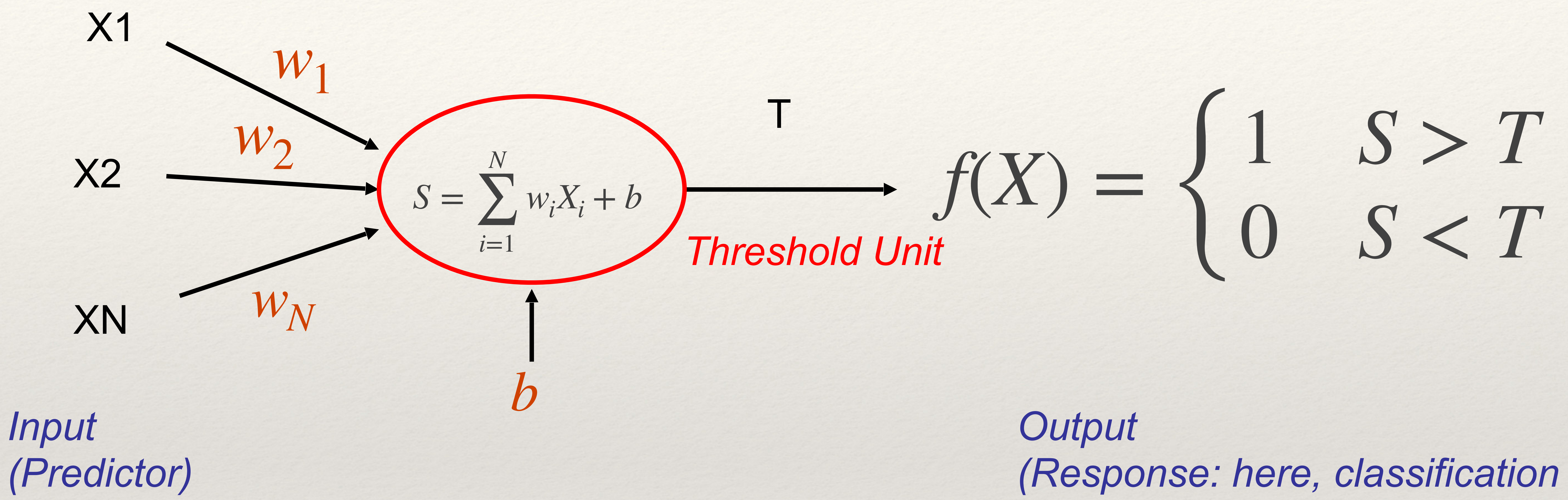
The perceptron classifies the input vector X into two categories.

Add weights $\boldsymbol{w}$ to all predictors, and a bias $\boldsymbol{b}$

Define simple "function" S as weighted sum of the predictors plus a bias

Add threshold to the function S

The perceptron

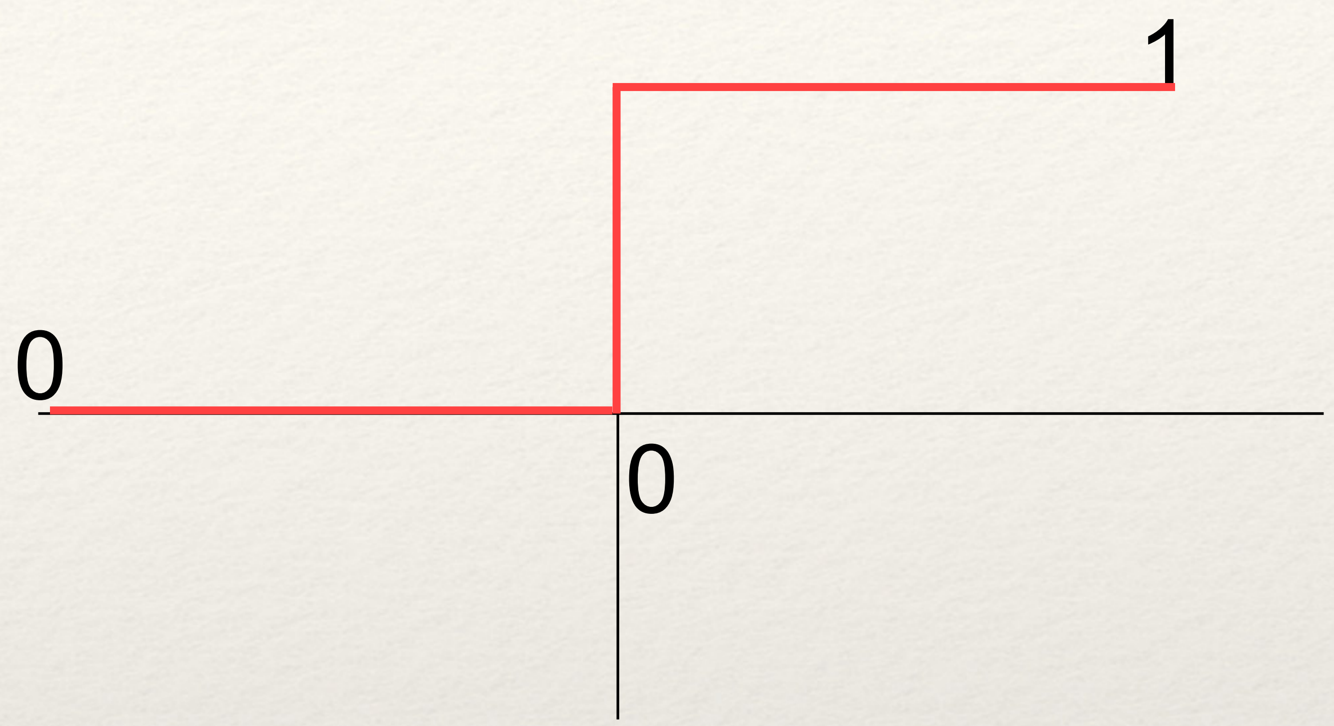


If the weights, bias, and threshold T are not known in advance, the perceptron must be trained. Ideally, the perceptron must be trained to return the correct answer on all training examples, and perform well on examples it has never seen.

The training set must contain both type of data (i.e. with “1” and “0” output).

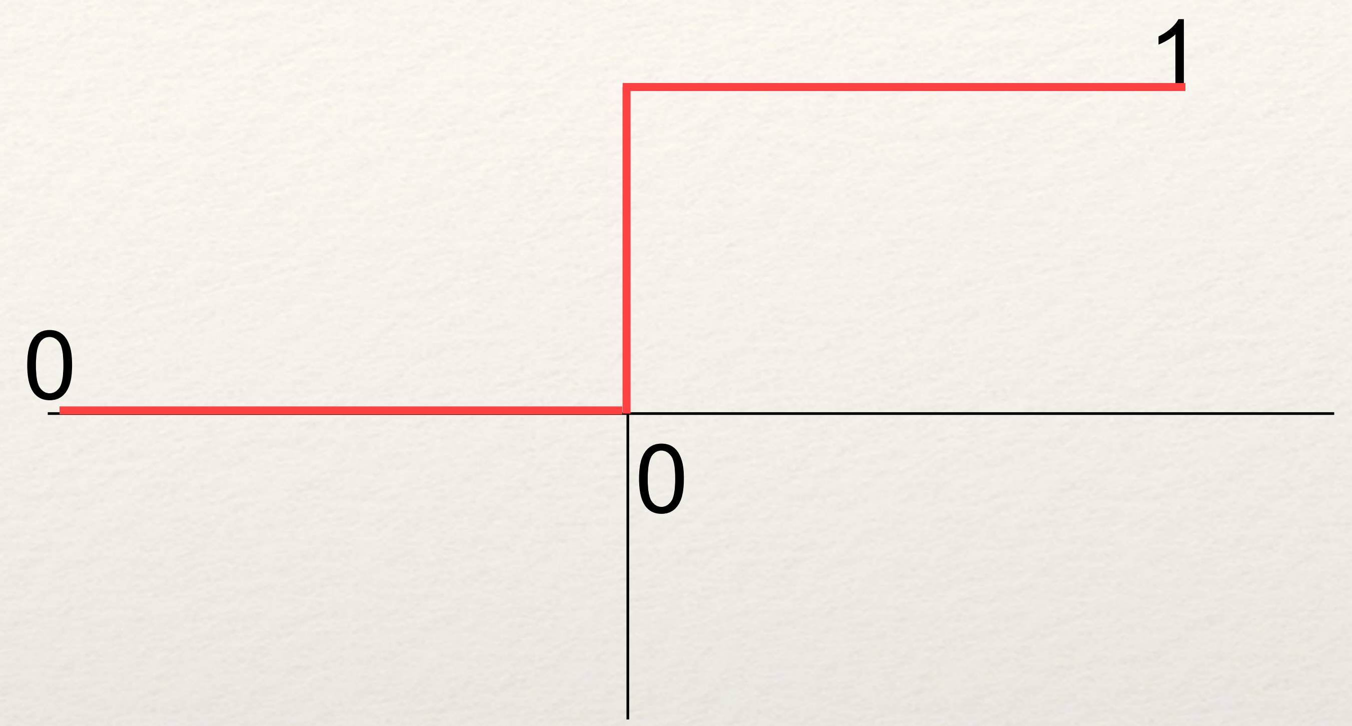
The perceptron

The output F is a function of S: it is often set discrete (i.e. 1 or 0), in which case the function is the step function.



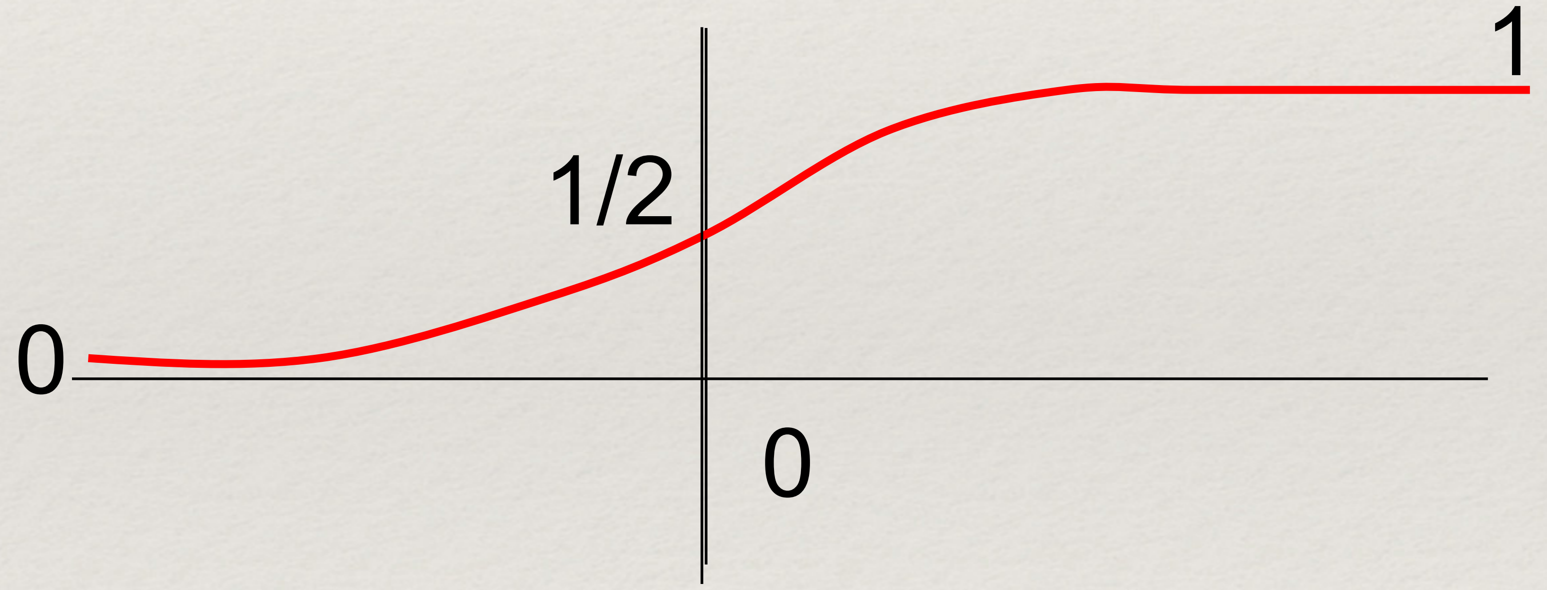
The perceptron

The output F is a function of S: it is often set discrete (i.e. 1 or 0), in which case the function is the step function.



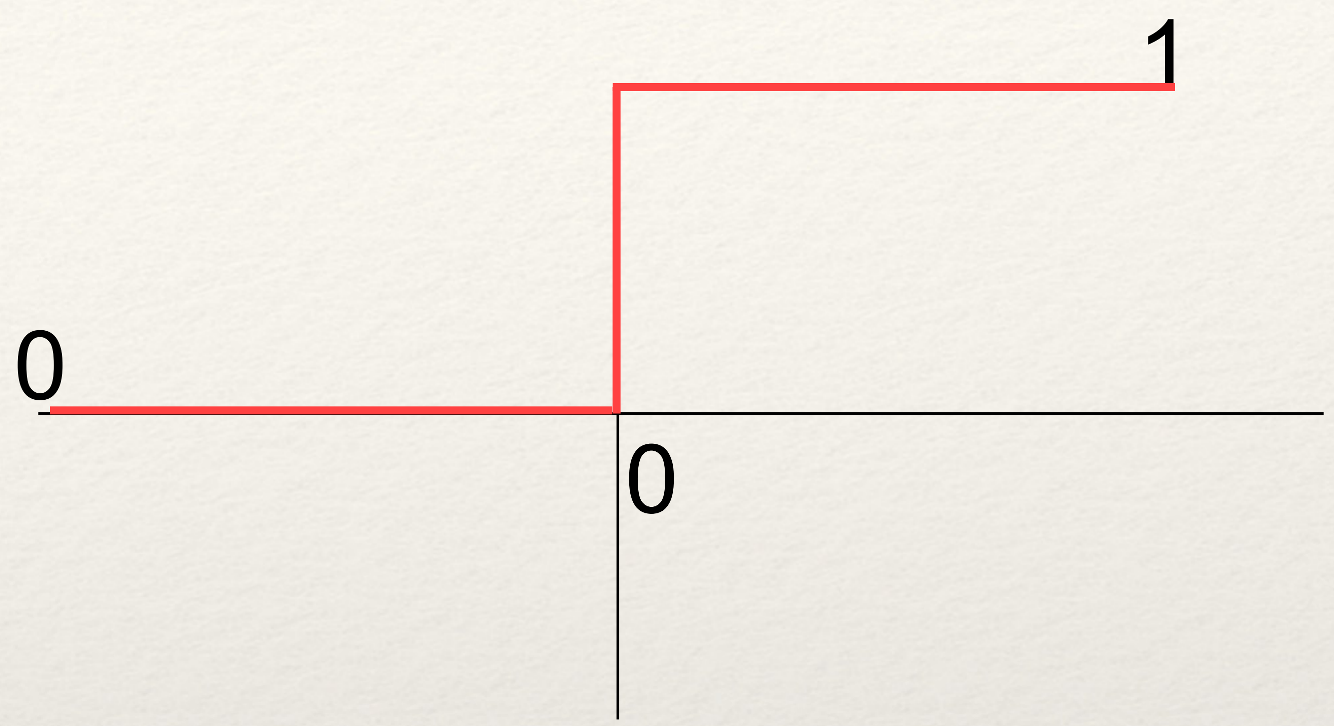
For continuous output, often use a sigmoid:

$$F(X) = F(S) = \frac{1}{1 + e^{-S}}$$



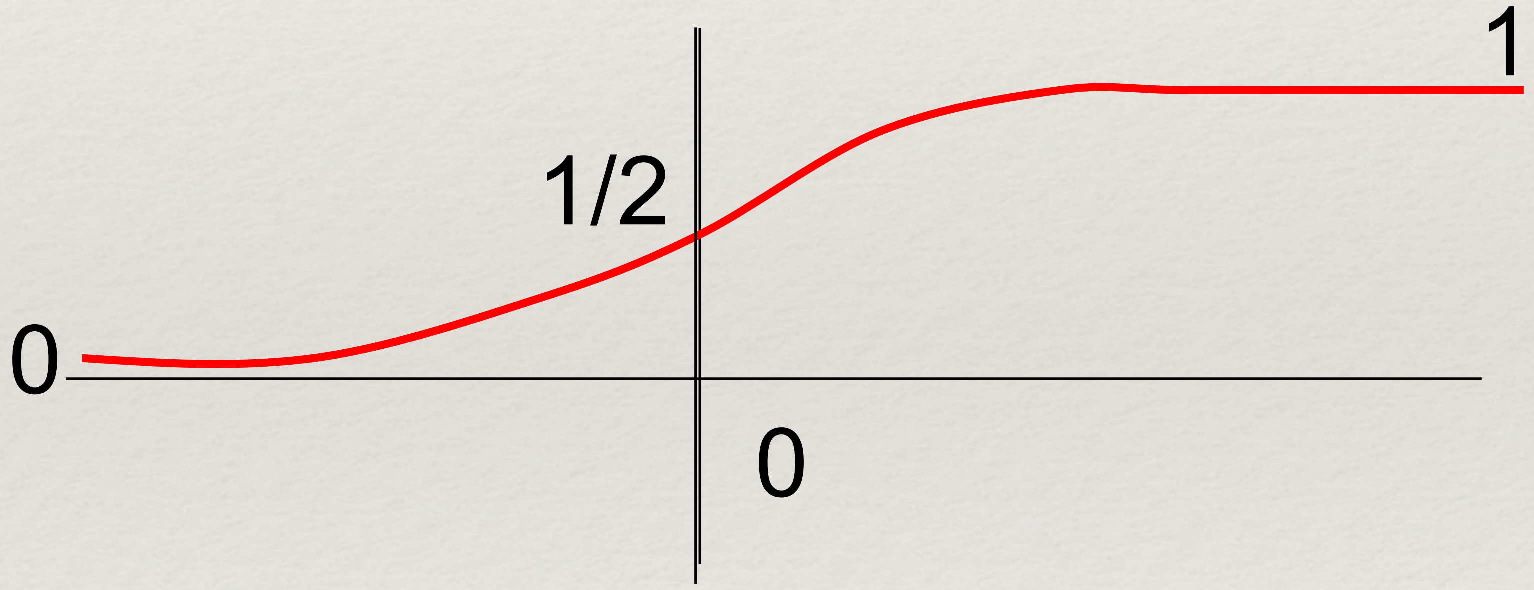
The perceptron

The output F is a function of S: it is often set discrete (i.e. 1 or 0), in which case the function is the step function.



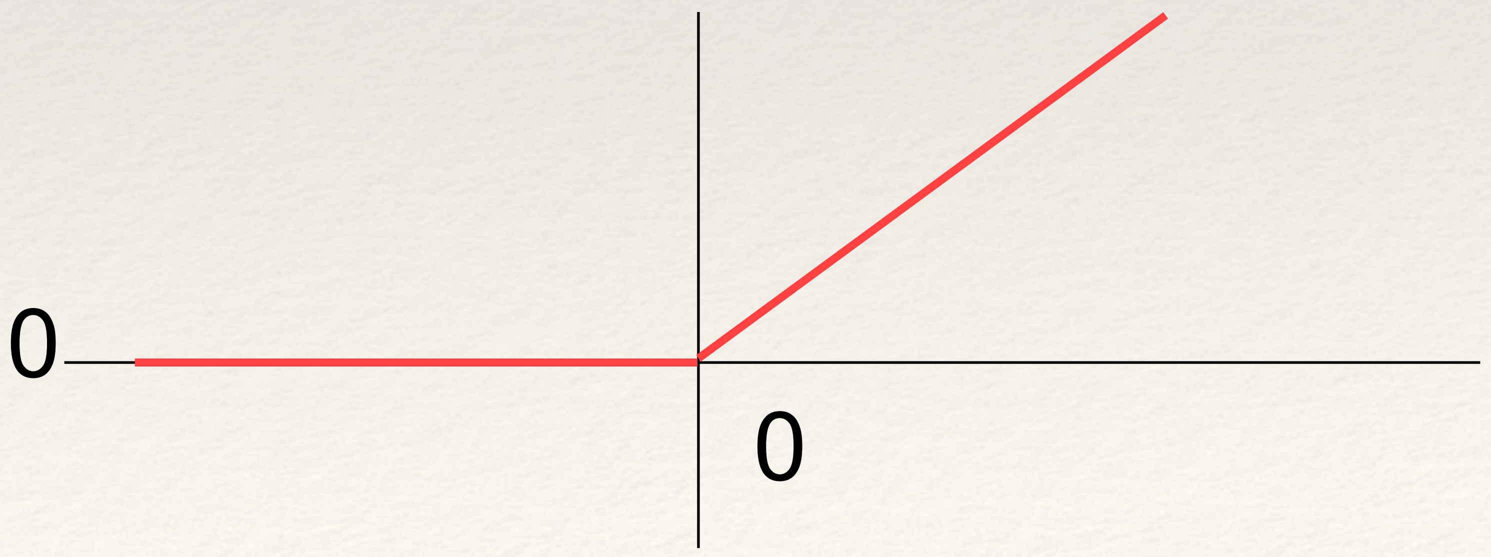
For continuous output, often use a sigmoid:

$$F(X) = F(S) = \frac{1}{1 + e^{-S}}$$



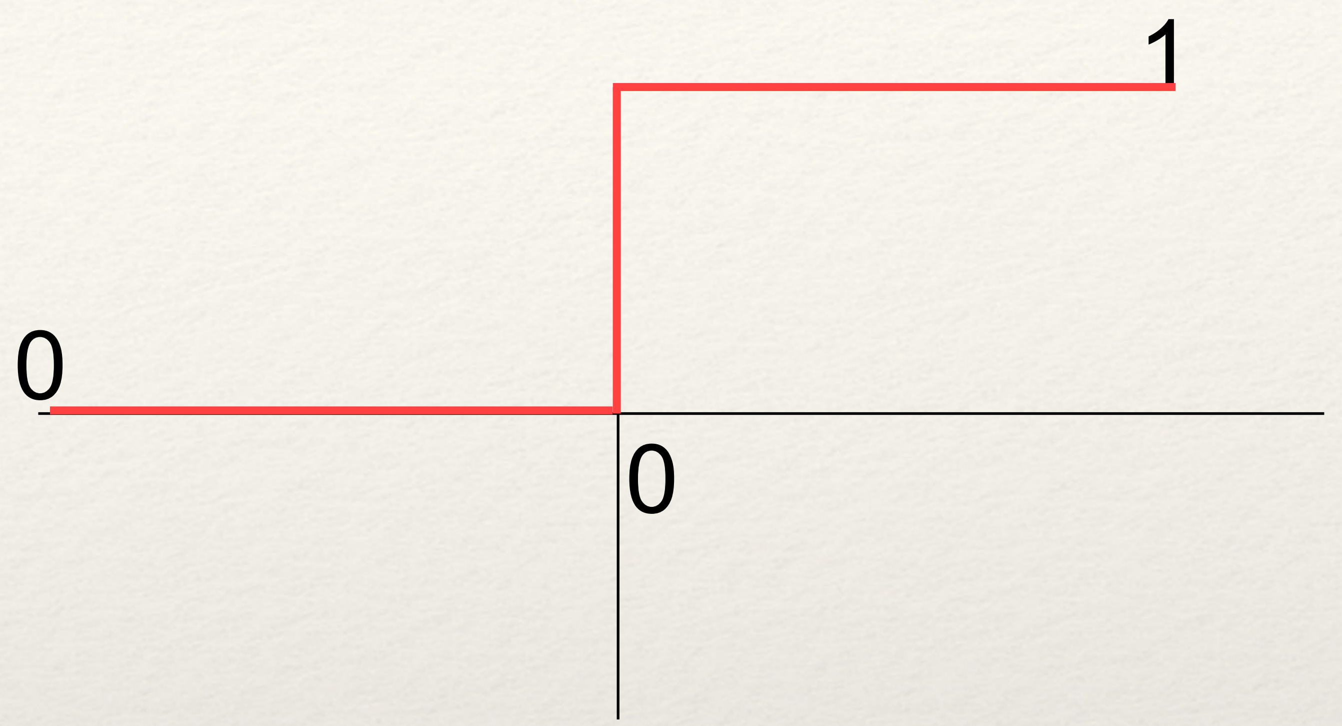
A popular alternative is the Rectifier Unit (ReLu):

$$\mathrm{ReLu}(S) = \max(0, S)$$



The perceptron

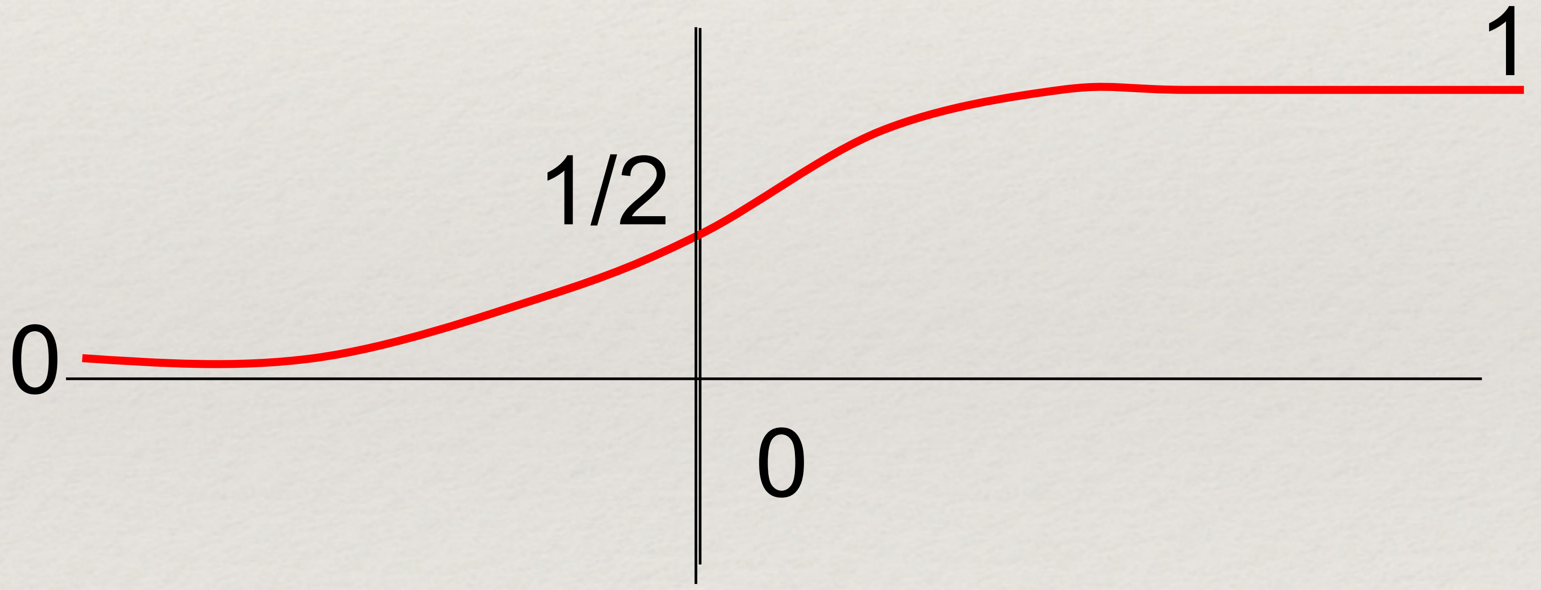
The output F is a function of S: it is often set discrete (i.e. 1 or 0), in which case the function is the step function.



For the step function, we can set the bias to 0

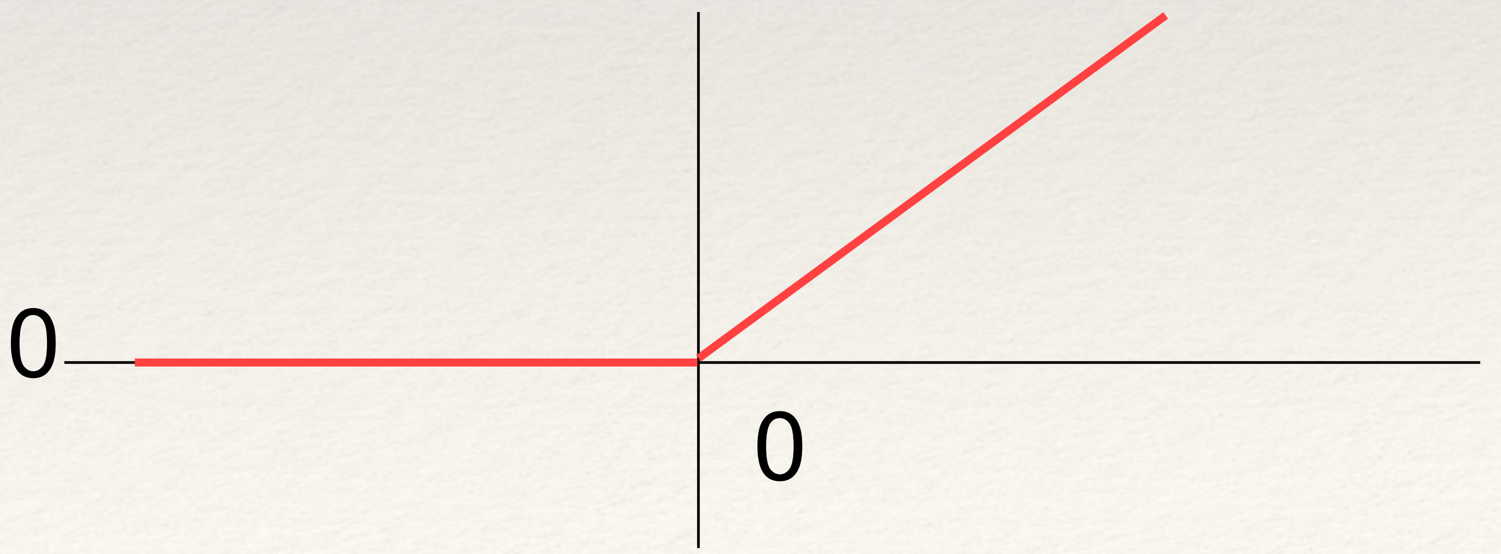
For continuous output, often use a sigmoid:

$$F(X) = F(S) = \frac{1}{1 + e^{-S}}$$

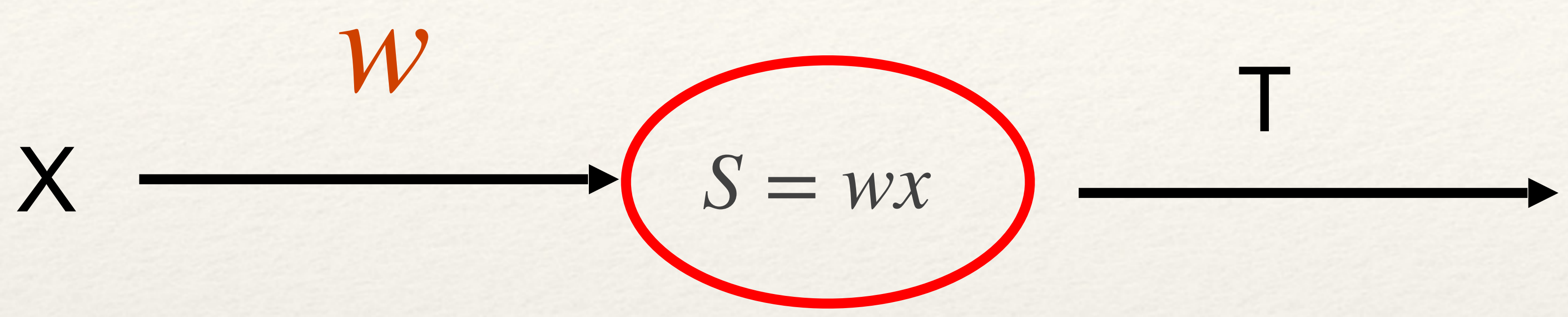


A popular alternative is the Rectifier Unit (ReLu):

$$\text{ReLu}(S) = \max(0, S)$$



Example: the NOT gate



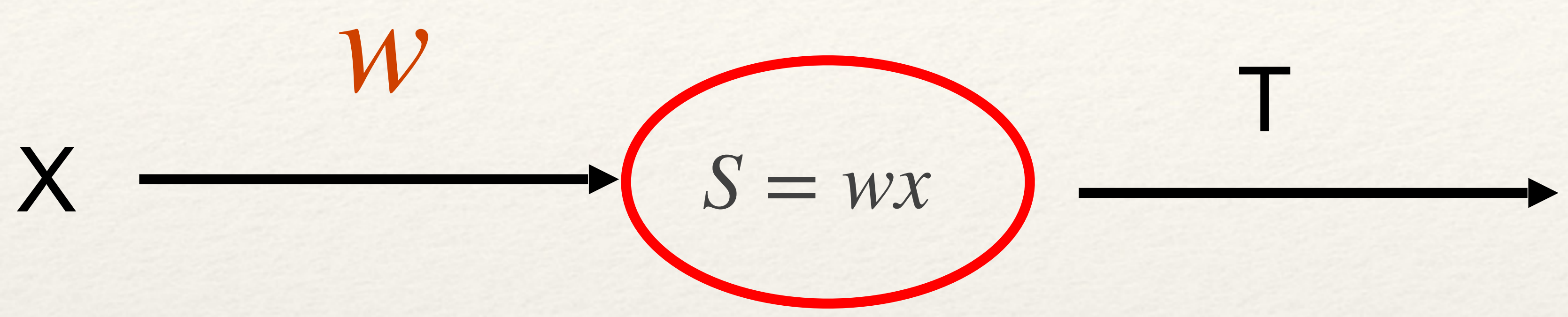
$$f(X) = \begin{cases} 1 & S > T \\ 0 & S < T \end{cases}$$

X1	Output	S
1	0	w
0	1	0

$w = -1$

$T = -0.5$

Example: the NOT gate



$$f(X) = \begin{cases} 1 & S > T \\ 0 & S < T \end{cases}$$

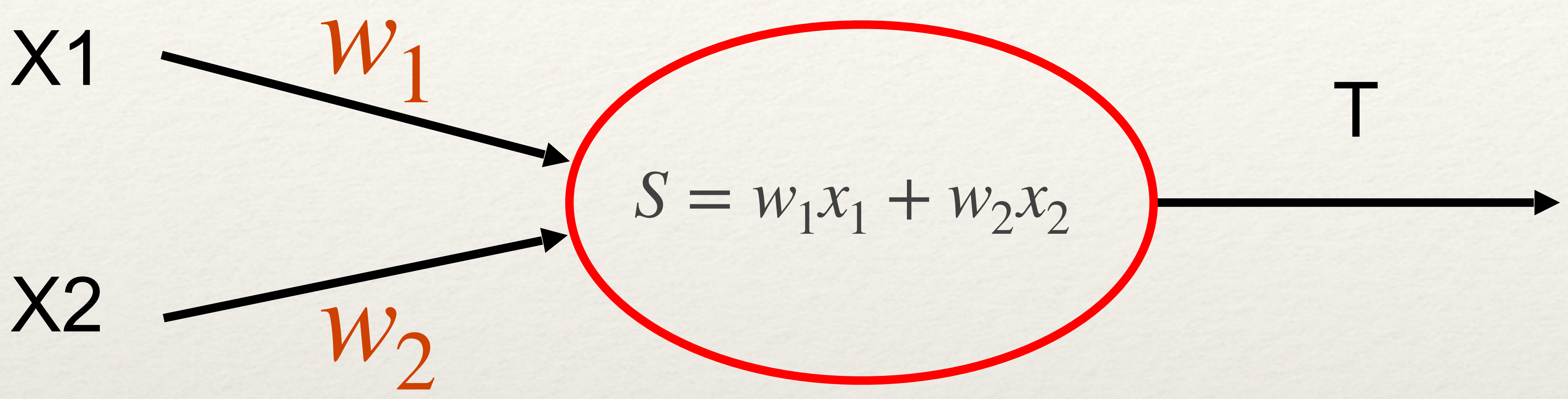
X1	Output	S
1	0	w
0	1	0

Possible solution:

$w = -1$

$T = -0.5$

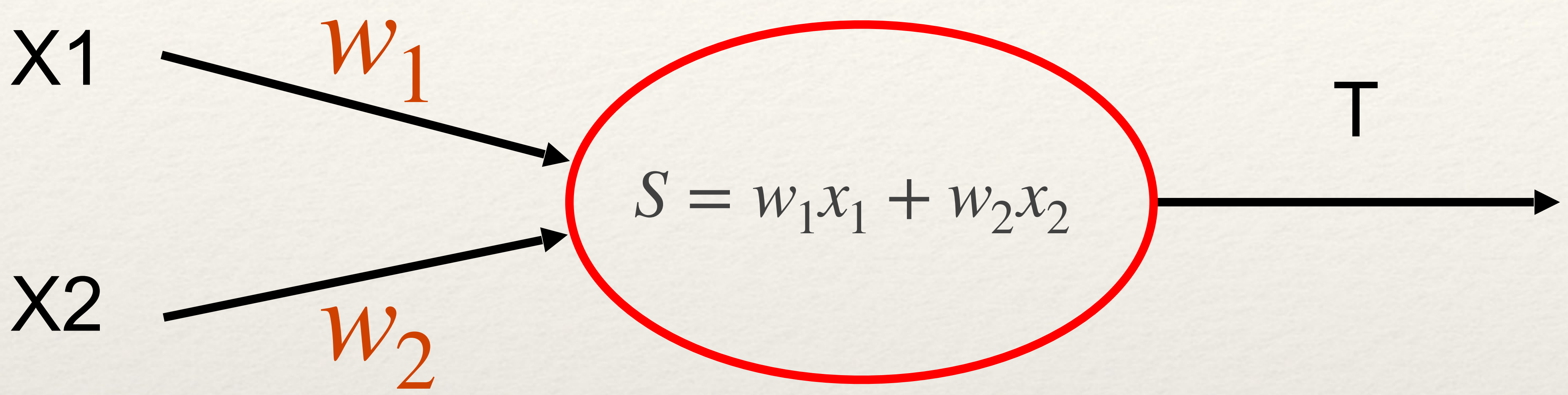
Example: the AND gate



$$f(X) = \begin{cases} 1 & S > T \\ 0 & S < T \end{cases}$$

X1	X2	Output	S
1	1	1	$w_1 + w_2$
1	0	0	w_1
0	1	0	w_2
0	0	0	0

Example: the AND gate



$$f(X) = \begin{cases} 1 & S > T \\ 0 & S < T \end{cases}$$

X1	X2	Output	S
1	1	1	$w_1 + w_2$
1	0	0	w_1
0	1	0	w_2
0	0	0	0

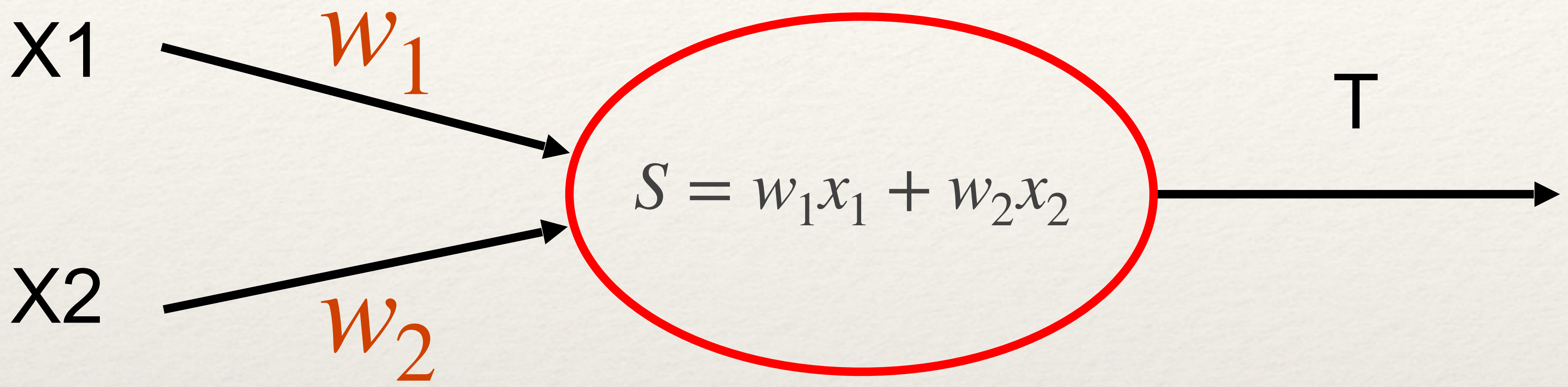
Possible solution:

$w_1 = 1$

$w_2 = 1$

$T = 1.5$

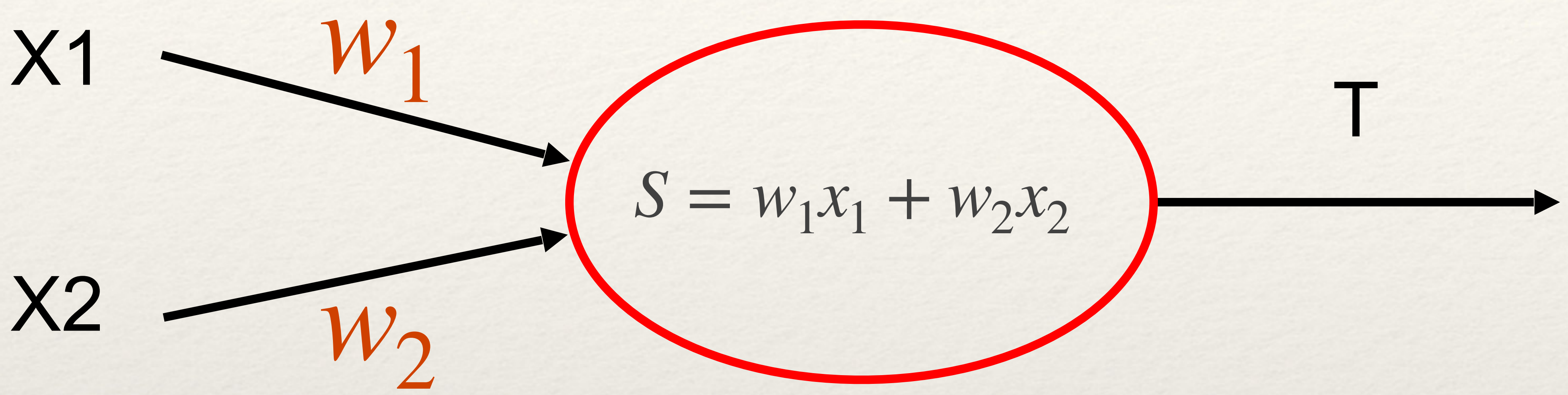
Example: the OR gate



$$f(X) = \begin{cases} 1 & S > T \\ 0 & S < T \end{cases}$$

X1	X2	Output	S
1	1	1	$w_1 + w_2$
1	0	1	w_1
0	1	1	w_2
0	0	0	0

Example: the OR gate



$$f(X) = \begin{cases} 1 & S > T \\ 0 & S < T \end{cases}$$

X1	X2	Output	S
1	1	1	$w_1 + w_2$
1	0	1	w_1
0	1	1	w_2
0	0	0	0

Possible solution:

$w_1 = 1$

$w_2 = 1$

$T = 0.5$

The perceptron

Training a perceptron:

Find the weights W that minimizes the error function:

$$E = \sum_{i=1}^{P} \left(F(X_i^T W) - t(X_i) \right)^2$$

P: number of training data
X_i: training vectors
$F(X_i^T W)$: output of the perceptron
$t(X_i)$: target value for X_i

Use steepest descent:

- compute gradient:

$$\nabla E = \left(\frac{\delta E}{\delta w_1}, \frac{\delta E}{\delta w_2}, \ldots, \frac{\delta E}{\delta w_N} \right)$$

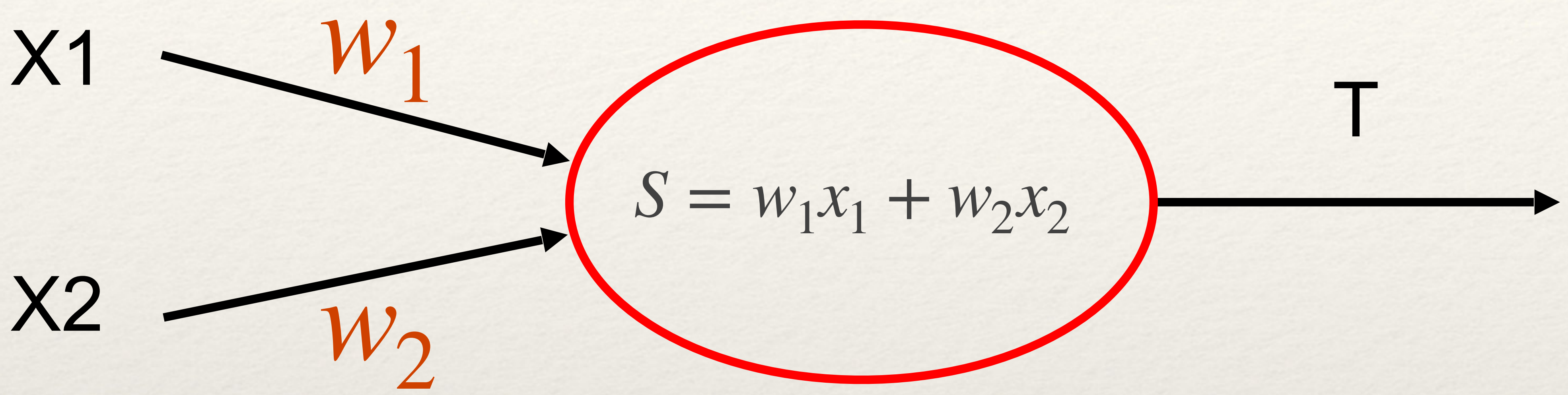
- update weight vector:

$$W_{new} = W_{old} - \epsilon \nabla E$$

- iterate

(ε: learning rate)

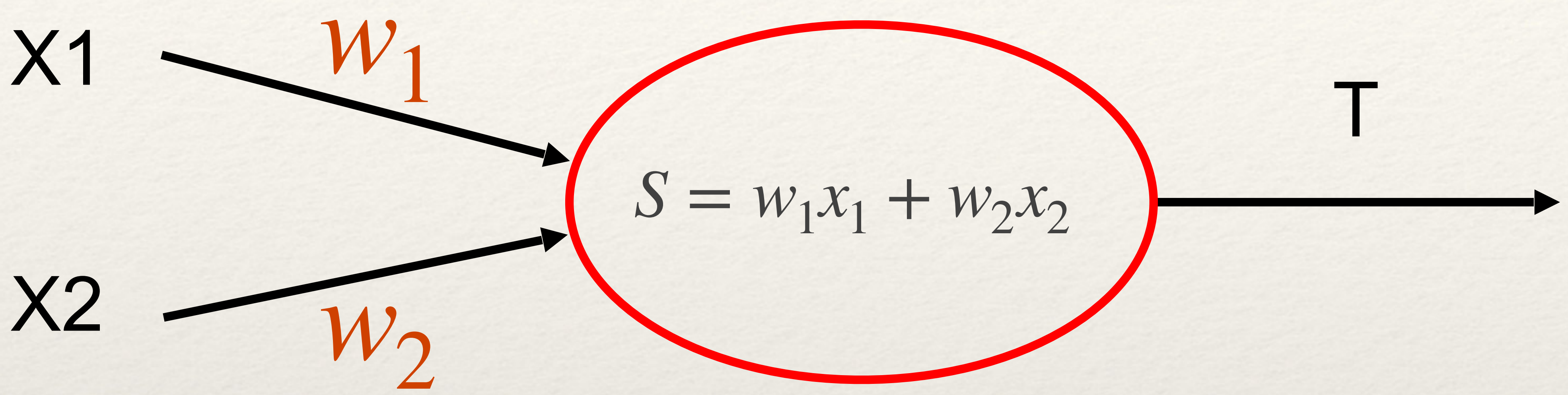
Example: the XOR gate



$$f(X) = \begin{cases} 1 & S > T \\ 0 & S < T \end{cases}$$

X1	X2	Output	S
1	1	0	$w_1 + w_2$
1	0	1	w_1
0	1	1	w_2
0	0	0	0

Example: the XOR gate

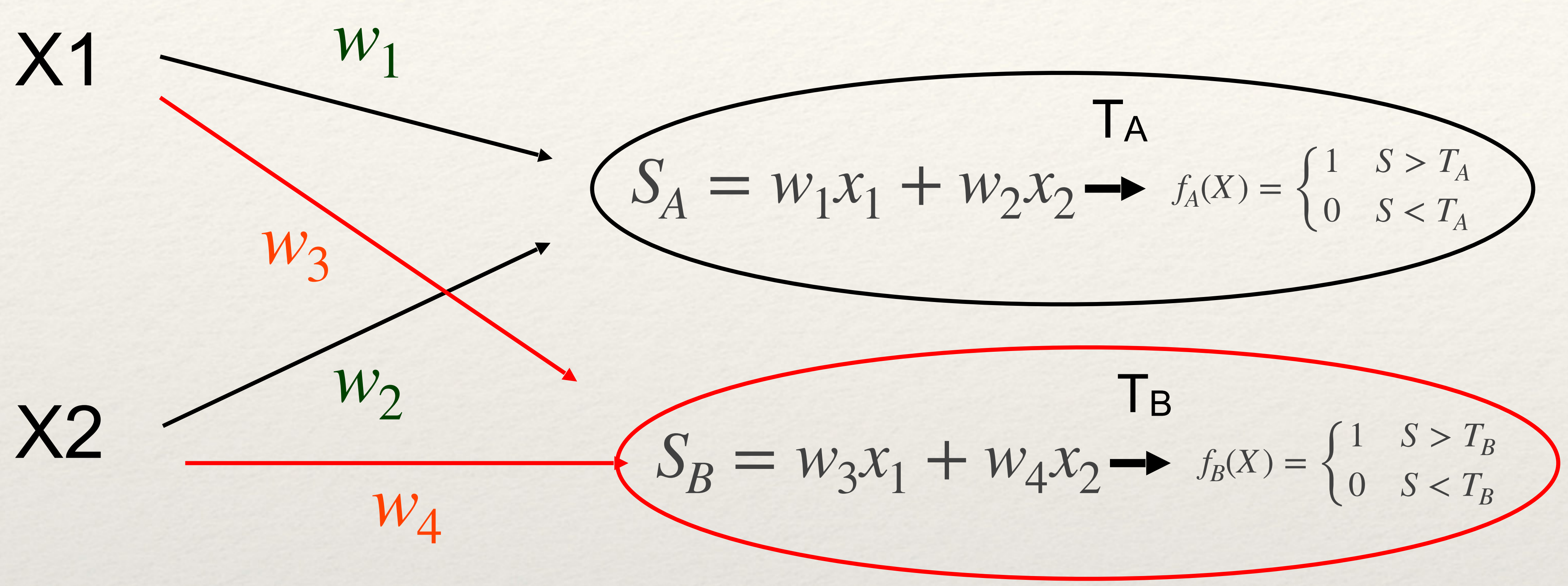


$$f(X) = \begin{cases} 1 & S > T \\ 0 & S < T \end{cases}$$

X1	X2	Output	S
1	1	0	$w_1 + w_2$
1	0	1	w_1
0	1	1	w_2
0	0	0	0

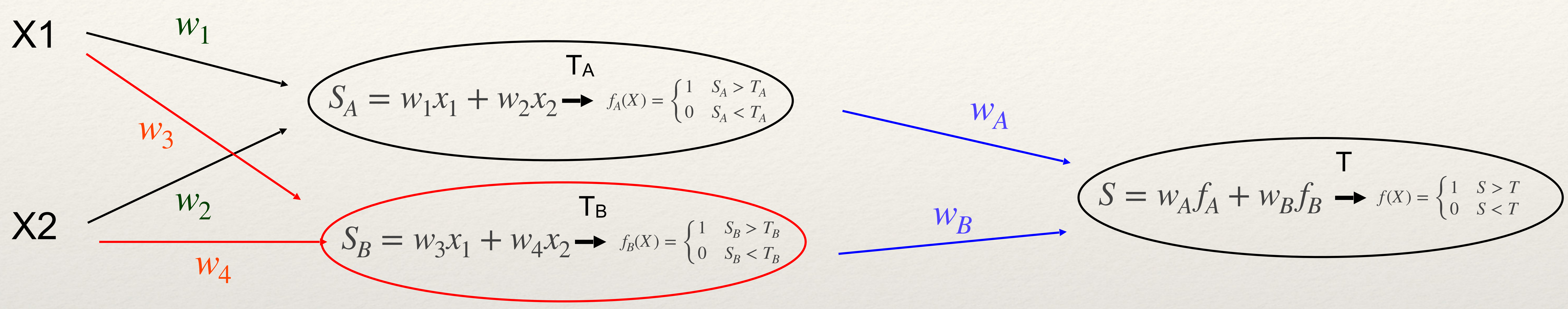
Cannot find a solution with a "simple" perceptron

A more complex network the XOR gate



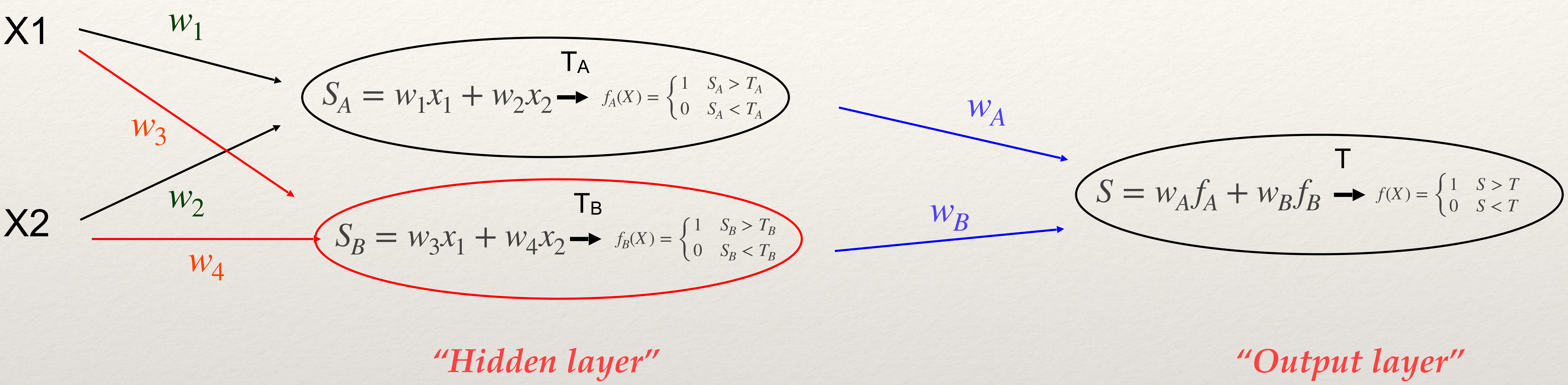
X1	X2	Output
1	1	0
1	0	1
0	1	1
0	0	0

A more complex network the XOR gate



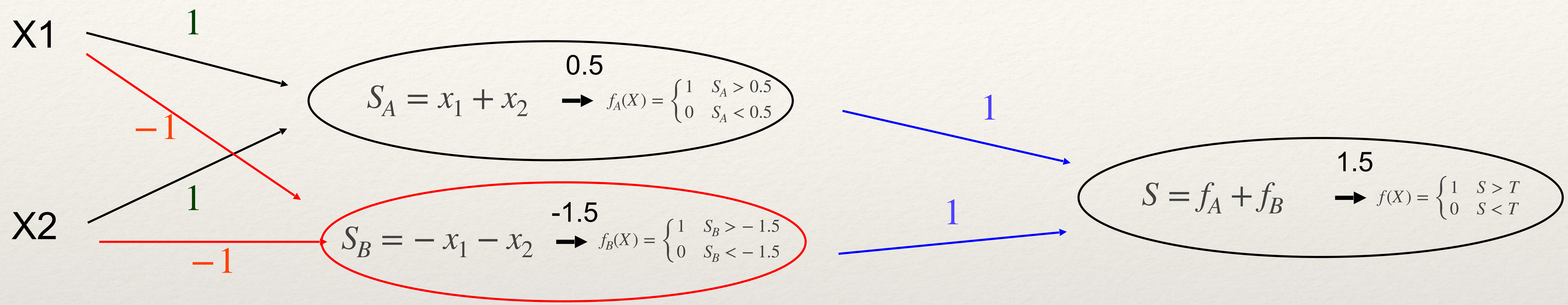
X1	X2	Output
1	1	0
1	0	1
0	1	1
0	0	0

A more complex network the XOR gate



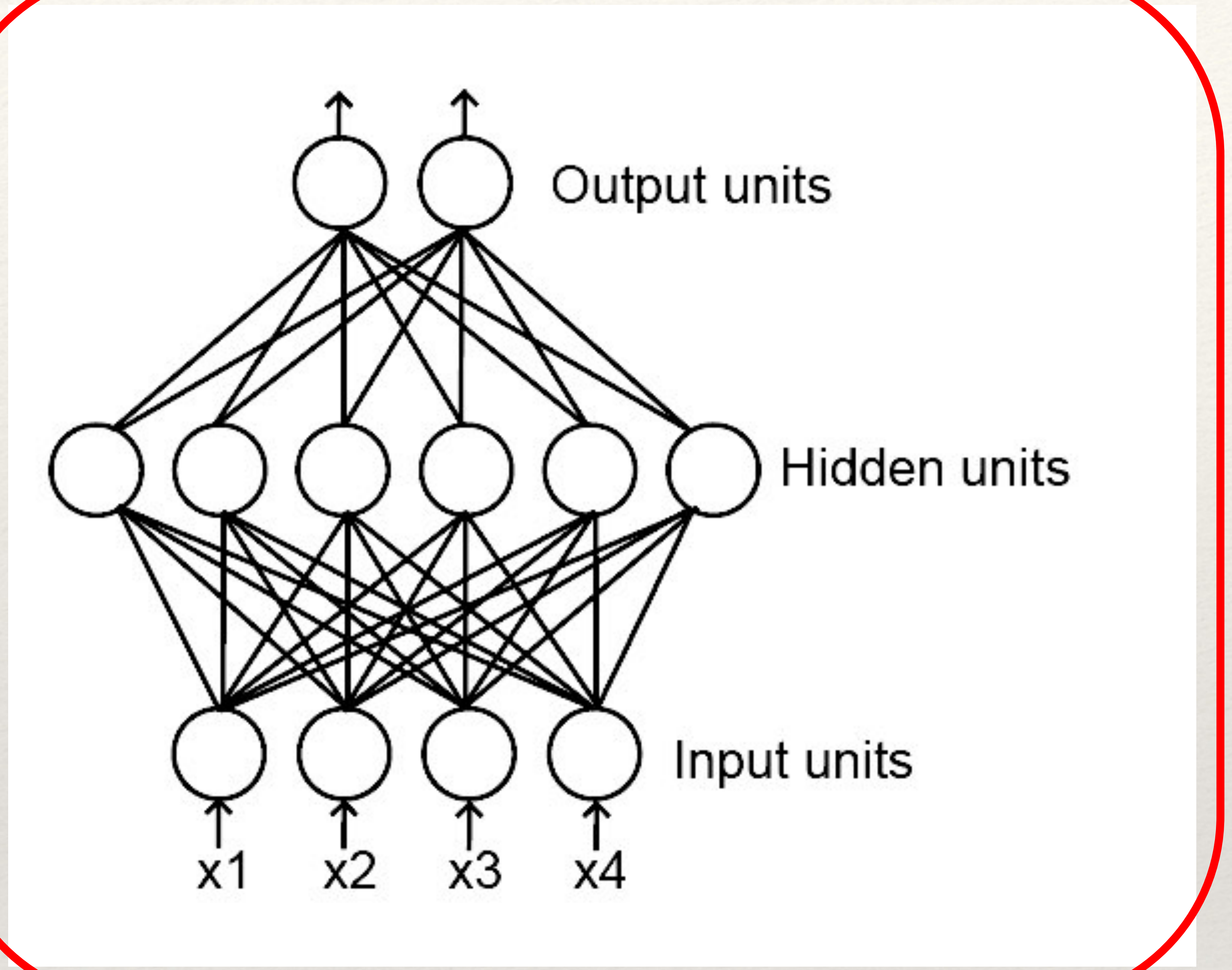
X1	X2	Output	S_A	S_B	S
1	1	0	$w_1 + w_2$	$w_1 + w_2$	$w_Af_A + w_Bf_B$
1	0	1	w_1	w_1	$w_Af_A + w_Bf_B$
0	1	1	w_2	w_2	$w_Af_A + w_Bf_B$
0	0	0	0	0	$w_Af_A + w_Bf_B$

A more complex network the XOR gate



X1	X2	Output
1	1	0
1	0	1
0	1	1
0	0	0

Neural Network



A complete neural network is a set of perceptrons interconnected such that the outputs of some units becomes the inputs of other units. Many topologies are possible!

Neural networks are trained just like perceptron, by minimizing an error function:

$$E = \sum_{i=1}^{Ndata} \left(NN(X_i) - t(X_i) \right)^2$$

Machine Learning / AI