ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
VIETNAM - KOREA UNIVERSITY OF INFORMATION AND COMMUNICATION TECHNOLOGY
한-베정보통신기술대학교

Nhân bản – Phụng sự – Khai phóng

Chapter 8

Neural Networks & Deep Learning

Machine Learning

CONTENTS

CONTENTS

- **Perceptron**
- Neural networks
- Gradient descent
- Backpropagation

1950s Age of the Perceptron

1957 The Perceptron (Rosenblatt)

1969 Perceptrons (Minsky, Papert)

1980s Age of the Neural Network

1986 Back propagation (Hinton)

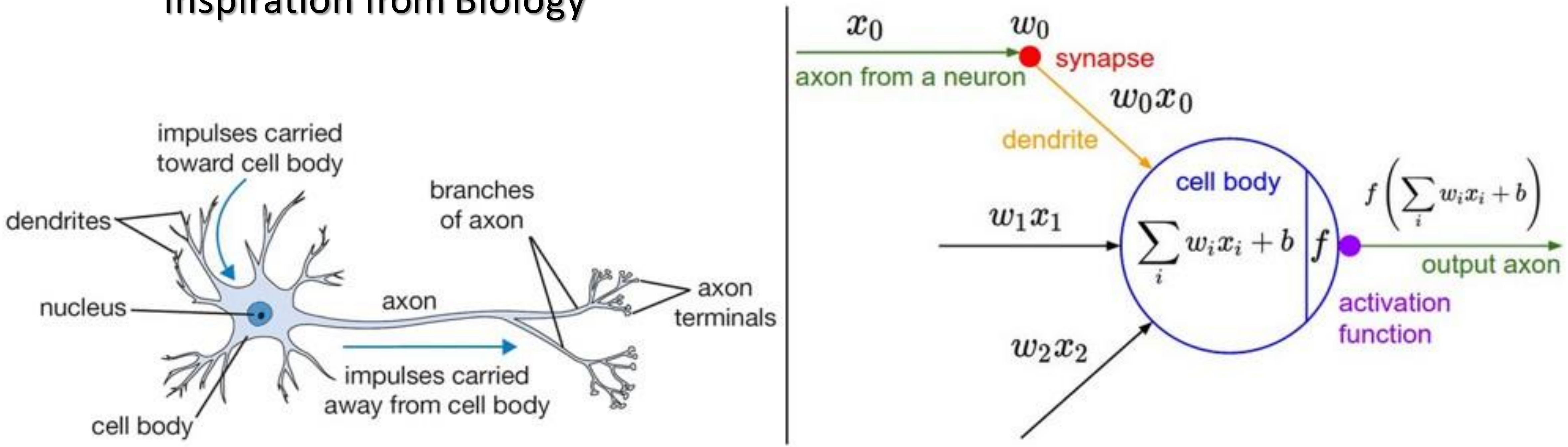
1990s Age of the Graphical Model

2000s Age of the Support Vector Machine

2010s Age of the Deep Network

Deep Learning = Known algorithms + Computing power + Big data

Inspiration from Biology



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

- Neural nets/perceptrons are **loosely** inspired by biology.
- But they certainly are **not** a model of how the brain works, or even how neurons work.

…Perceptron

$$\textbf{function } \textsc{Perceptron Algorithm}$$

$$\boldsymbol{w}^{(0)} \leftarrow \mathbf{0}$$

$$\textbf{for } t = 1, \ldots, T \textbf{ do}$$

$$\textsc{Receive}(\boldsymbol{x}^{(t)})$$

$\boldsymbol{x} \in \{0, 1\}^N$ N-d binary vector

$$\hat{y}_A^{(t)} = \operatorname{sign}\left(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^{(t)} \rangle \right)$$

sign of zero is +1

perceptron is just one line of code!

$$\textsc{Receive}(y^t)$$

$y \in \{1, -1\}$

$$w_n^{(t)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$$

initialized w = 0

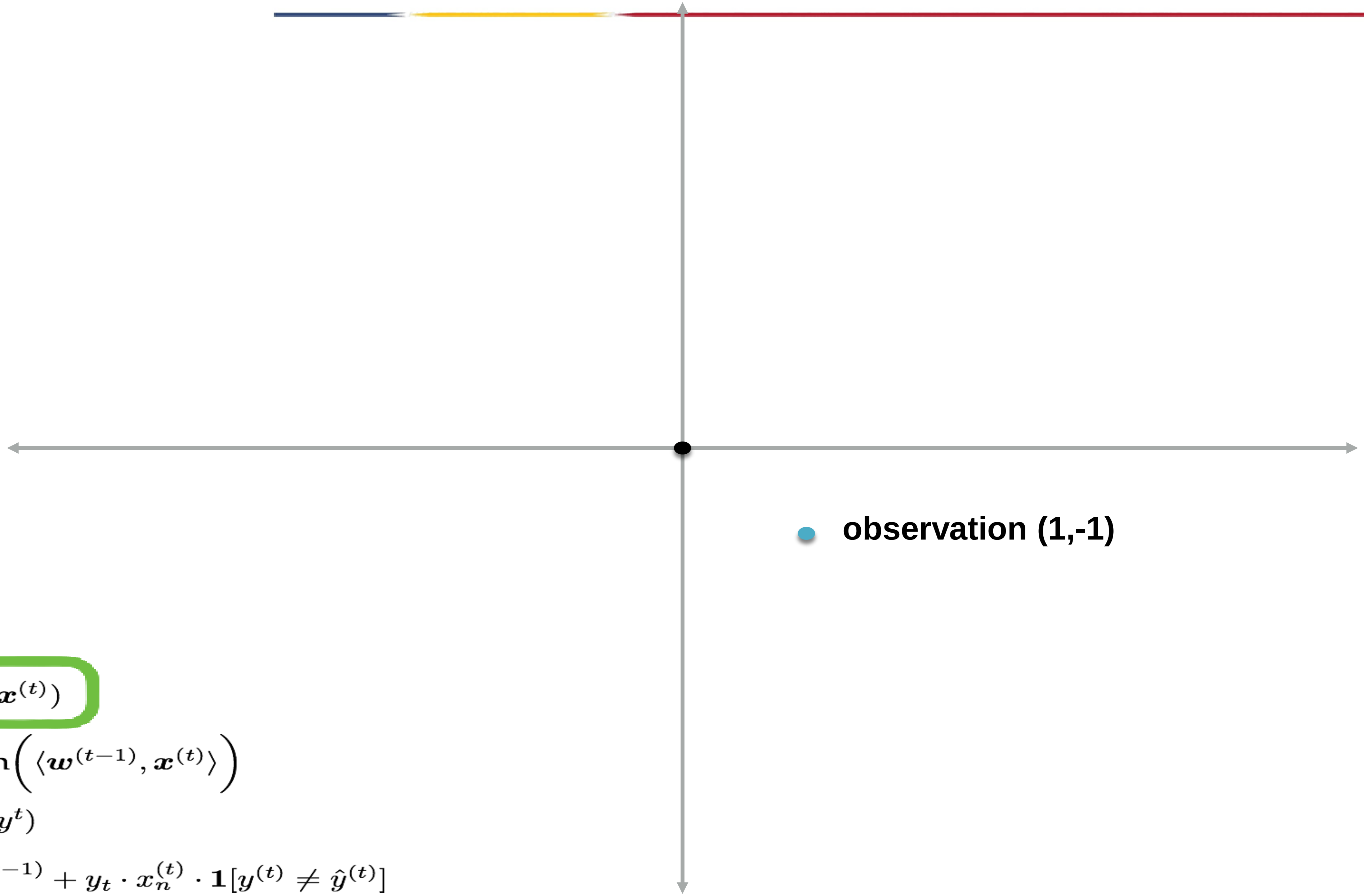
$\textsc{Receive}(\boldsymbol{x}^{(t)})$

$$\hat{y}_A^{(t)} = \mathbf{sign}\Big(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^{(t)} \rangle\Big)$$

$\textsc{Receive}(y^t)$

$$w_n^{(t)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$$

…Perceptron

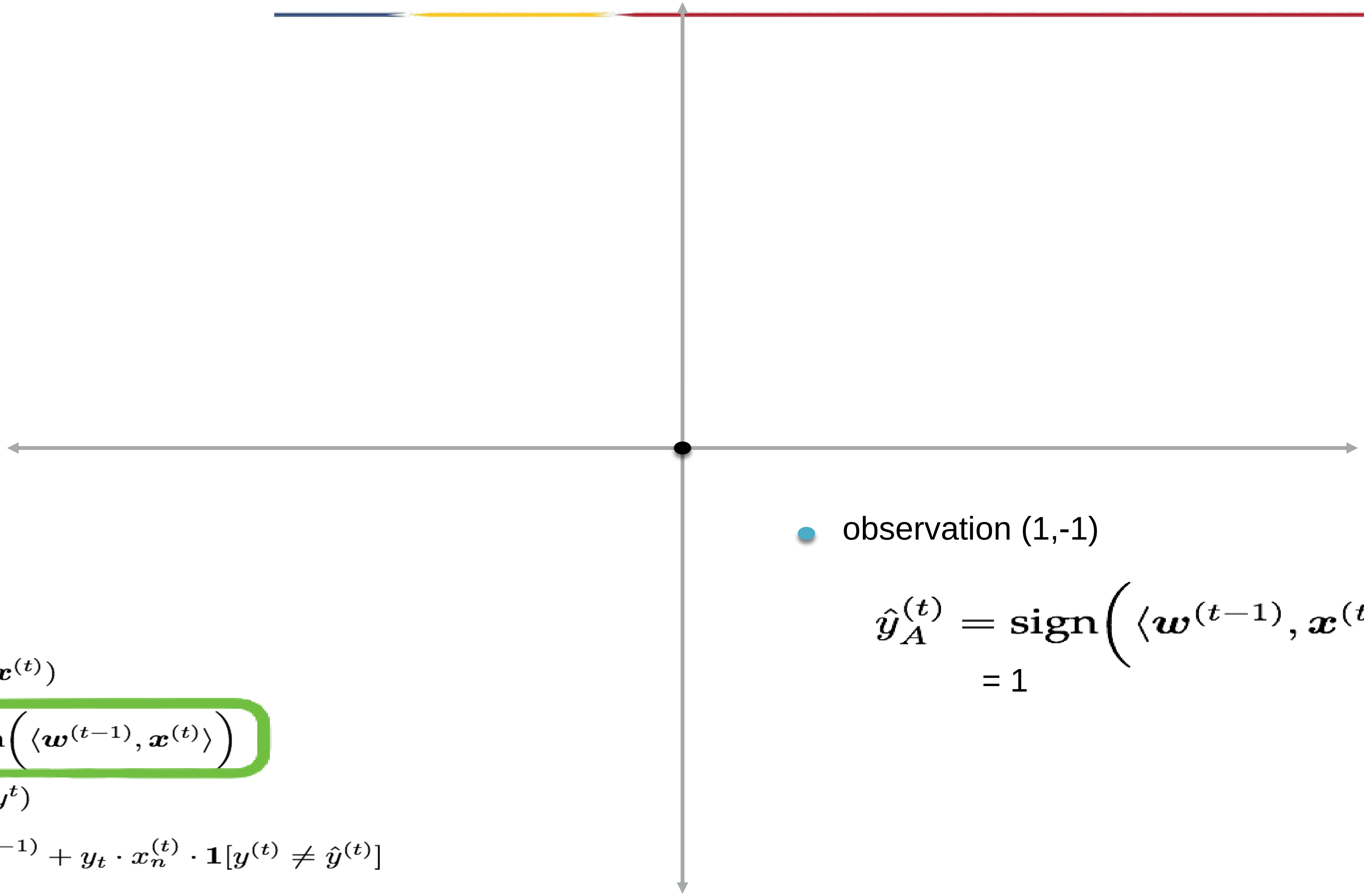


$$\boxed{\textsc{Receive}(\boldsymbol{x}^{(t)})}$$

$$\hat{y}_A^{(t)} = \text{sign}\Big(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^{(t)} \rangle\Big)$$

$$\textsc{Receive}(y^t)$$

$$w_n^{(t)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$$



$$\hat{y}_A^{(t)} = \mathbf{sign}\left(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^{(t)} \rangle \right)$$

= 1

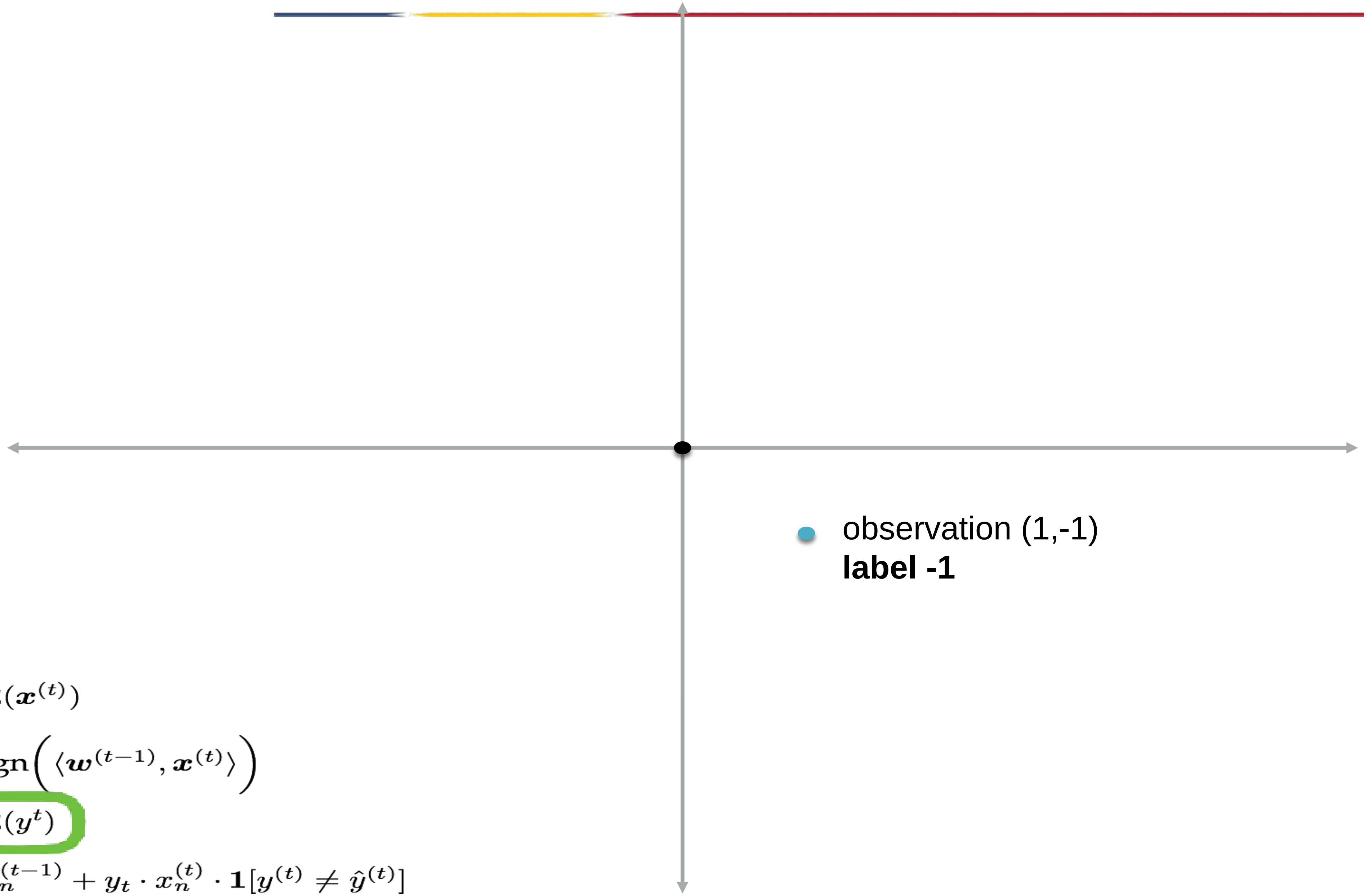
$\textsc{Receive}(\boldsymbol{x}^{(t)})$

$$\boxed{\hat{y}_A^{(t)} = \mathbf{sign}\left(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^{(t)} \rangle \right)}$$

$\textsc{Receive}(y^{t})$

$$w_n^{(t)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$$

…Perceptron



$$\textsc{Receive}(\boldsymbol{x}^{(t)})$$

$$\hat{y}_A^{(t)} = \mathrm{sign}\Big(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^{(t)} \rangle\Big)$$

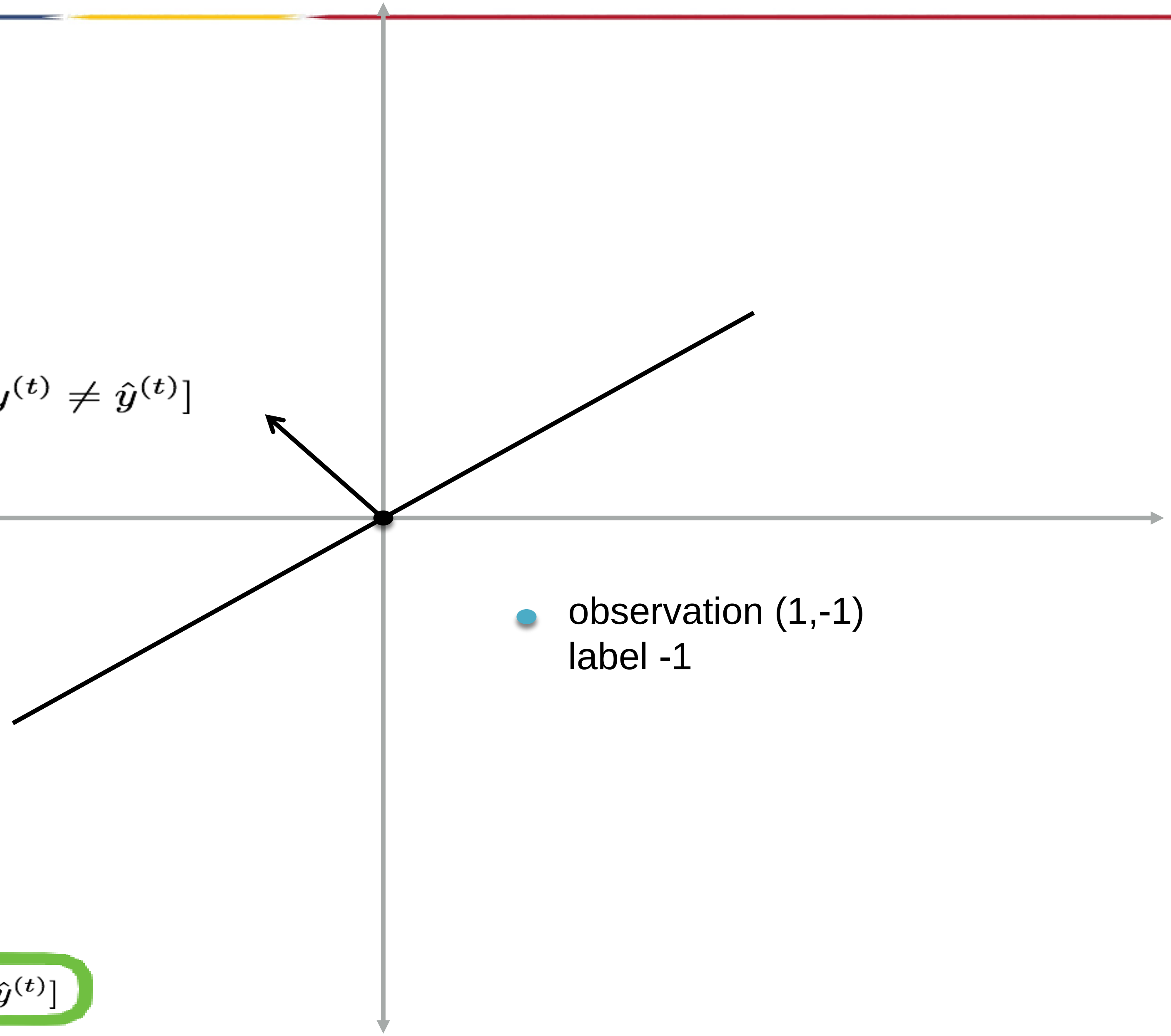
$$\textsc{Receive}(y^t)$$

$$w_n^{(t)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$$

…Perceptron

update w

$$w_n^{(t)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$$



$\textsc{Receive}(\boldsymbol{x}^{(t)})$

$$\hat{y}_A^{(t)} = \text{sign}\Big(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^{(t)} \rangle\Big)$$

$\textsc{Receive}(y^t)$

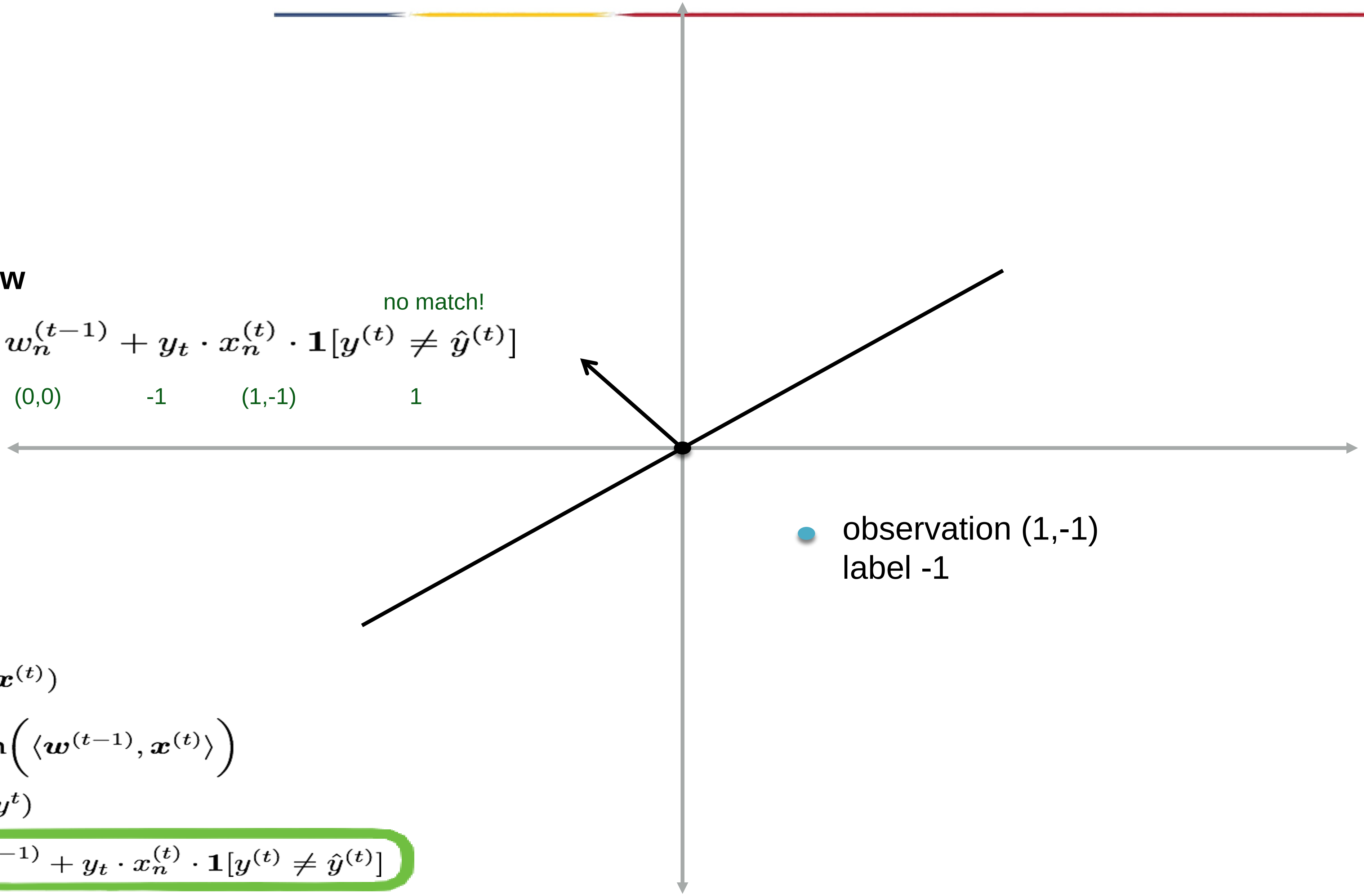
$$\boxed{w_n^{(t)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]}$$

update w

no match!

$$w_n^{(t)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$$

(-1,1) (0,0) -1 (1,-1) 1



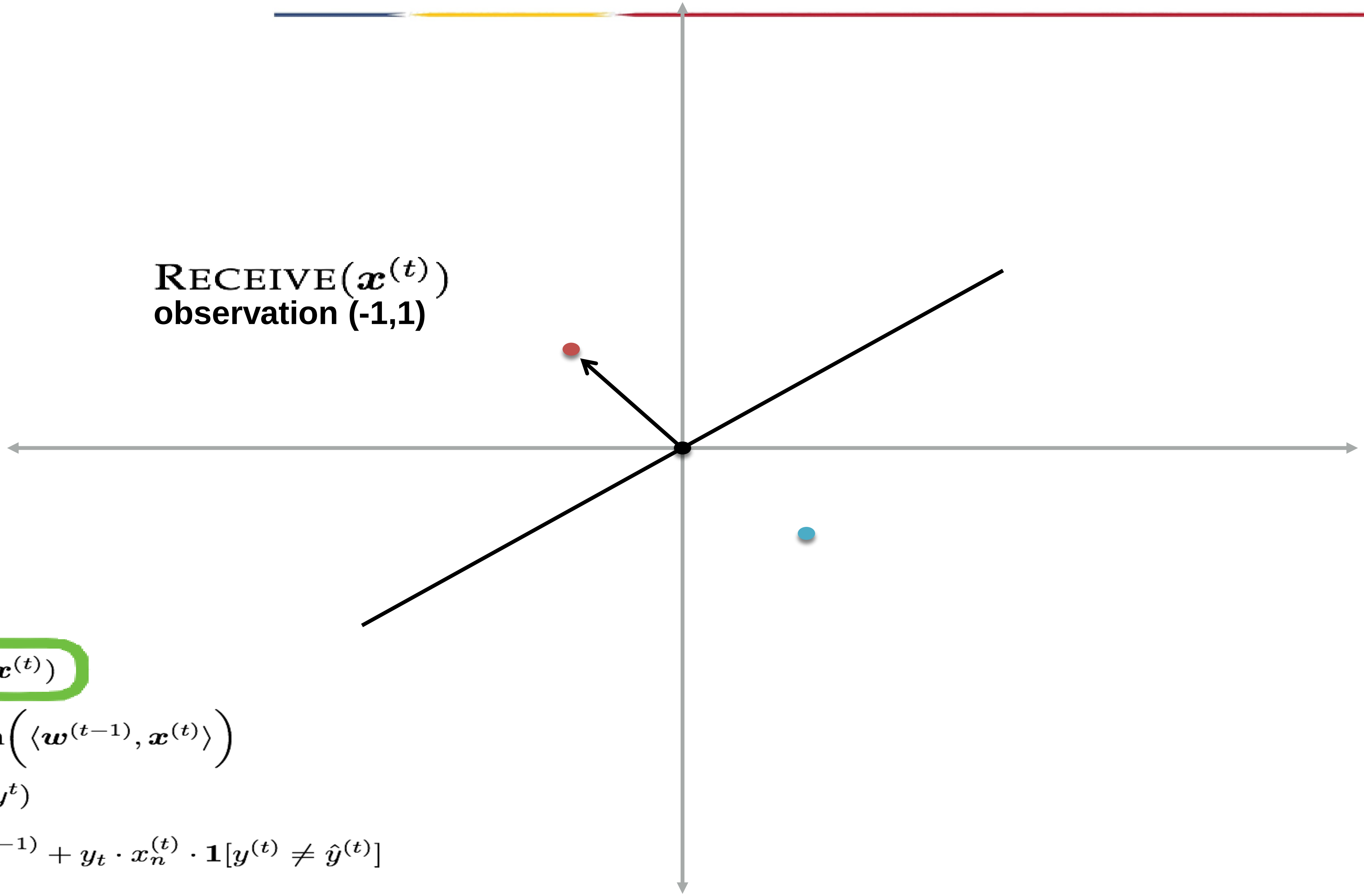
$\textsc{Receive}(\boldsymbol{x}^{(t)})$

$$\hat{y}_A^{(t)} = \mathrm{sign}\Big(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^{(t)} \rangle\Big)$$

$\textsc{Receive}(y^t)$

$$w_n^{(t)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$$

…Perceptron



$\textsc{Receive}(\boldsymbol{x}^{(t)})$

$$\hat{y}_A^{(t)} = \text{sign}\Big(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^{(t)} \rangle\Big)$$

$\textsc{Receive}(y^t)$

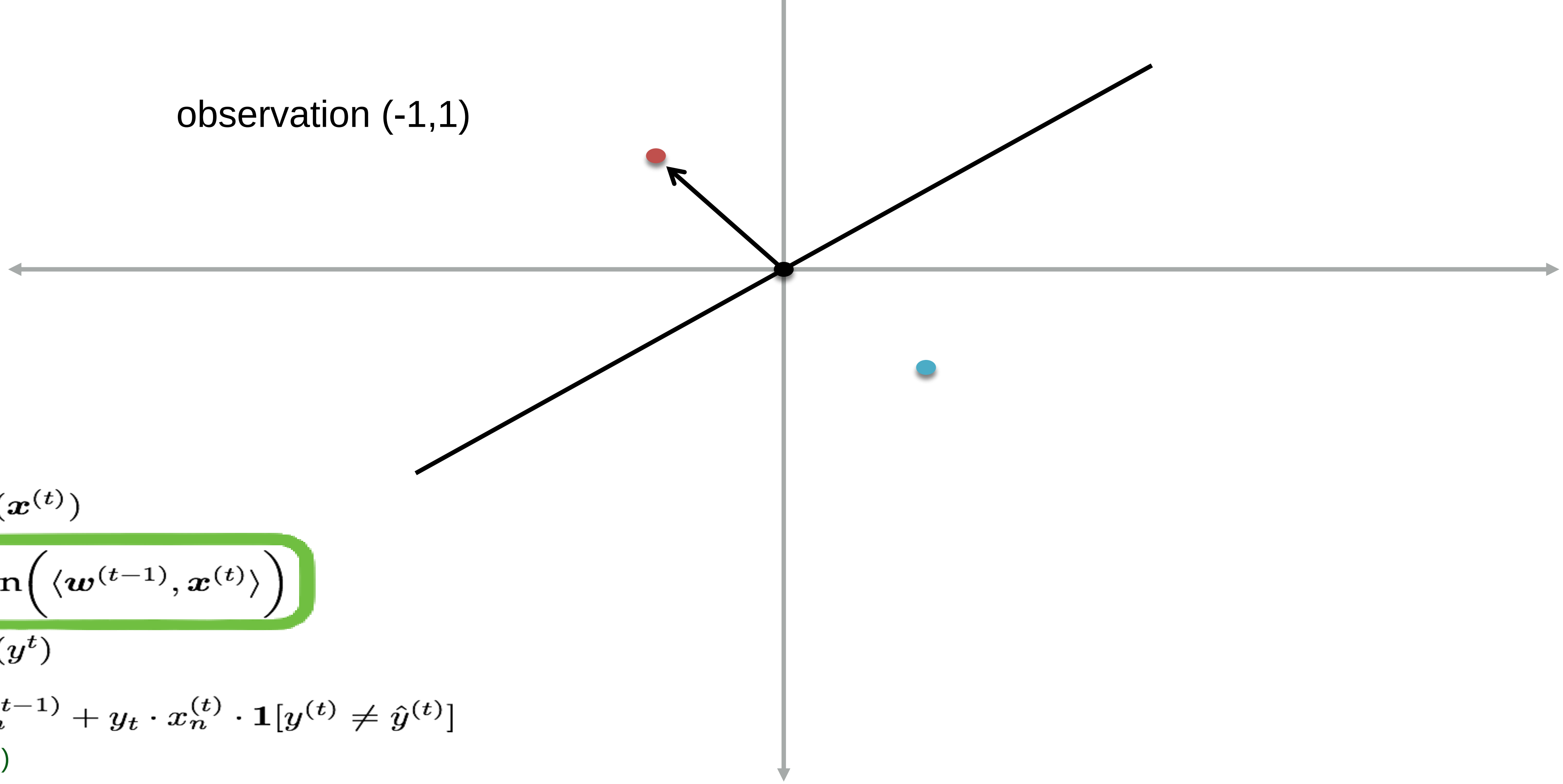
$$w_n^{(t)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$$

(-1,1)

…Perceptron

$$\hat{y}_A^{(t)} = \mathbf{sign}\left(\langle \underset{(-1,1)}{\boldsymbol{w}^{(t-1)}}, \underset{(-1,1)}{\boldsymbol{x}^{(t)}}\rangle\right)$$

= 1



$\textsc{Receive}(\boldsymbol{x}^{(t)})$

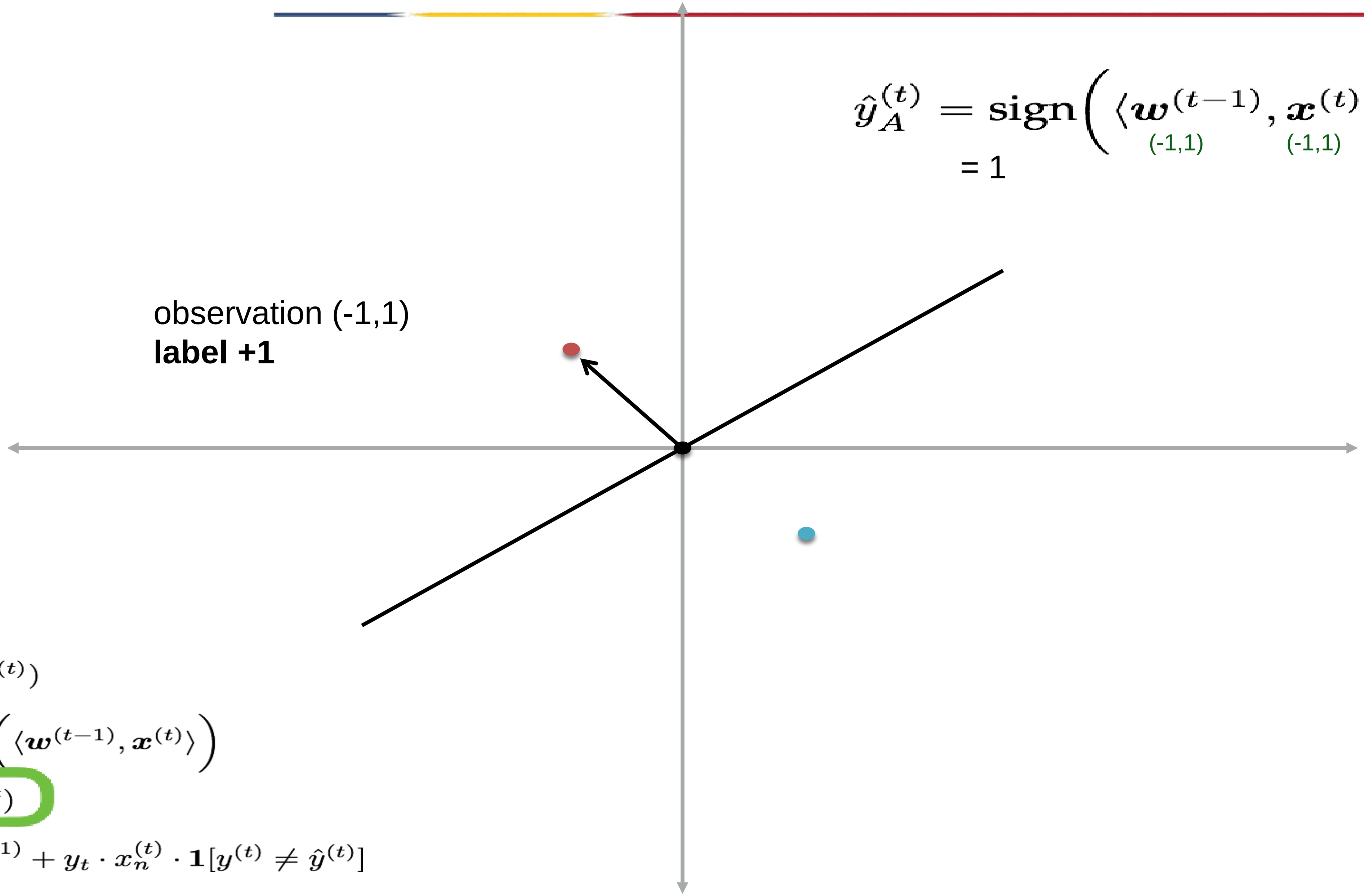
$$\boxed{\hat{y}_A^{(t)} = \mathbf{sign}\left(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^{(t)}\rangle\right)}$$

$\textsc{Receive}(y^t)$

$$w_n^{(t)} = \underset{(-1,1)}{w_n^{(t-1)}} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$$

$$\hat{y}_A^{(t)} = \operatorname{sign}\Big(\langle \underbrace{\boldsymbol{w}^{(t-1)}}_{(-1,1)}, \underbrace{\boldsymbol{x}^{(t)}}_{(-1,1)} \rangle \Big)$$

$= 1$



$\textsc{Receive}(\boldsymbol{x}^{(t)})$

$$\hat{y}_A^{(t)} = \operatorname{sign}\Big(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^{(t)} \rangle \Big)$$

$\textsc{Receive}(y^t)$

$$\underbrace{w_n^{(t)}}_{(-1,1)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$$

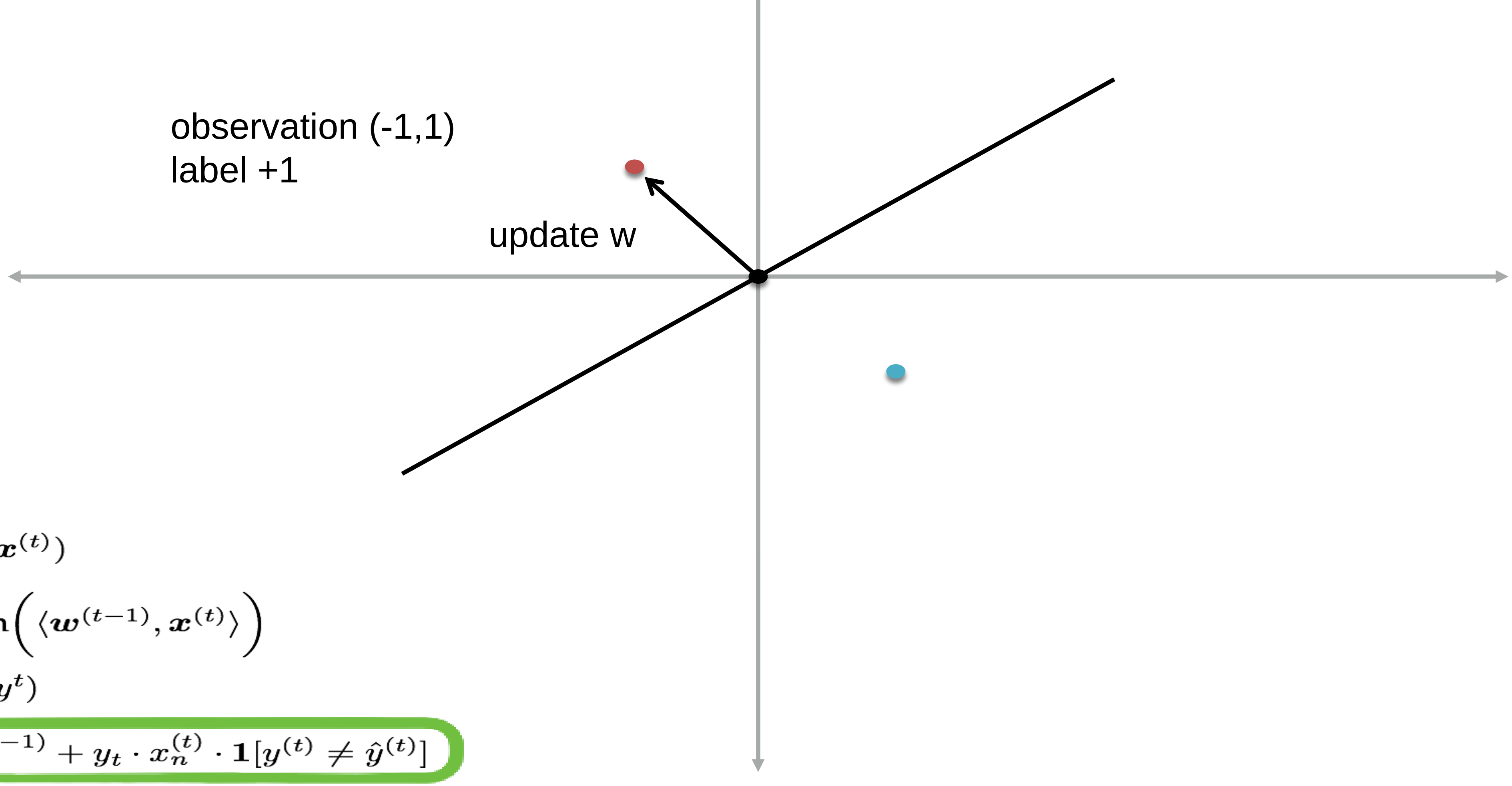
…Perceptron

update w

match!

$$w_n^{(t)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$$

(-1,1) (-1,1) +1 (-1,1) 0

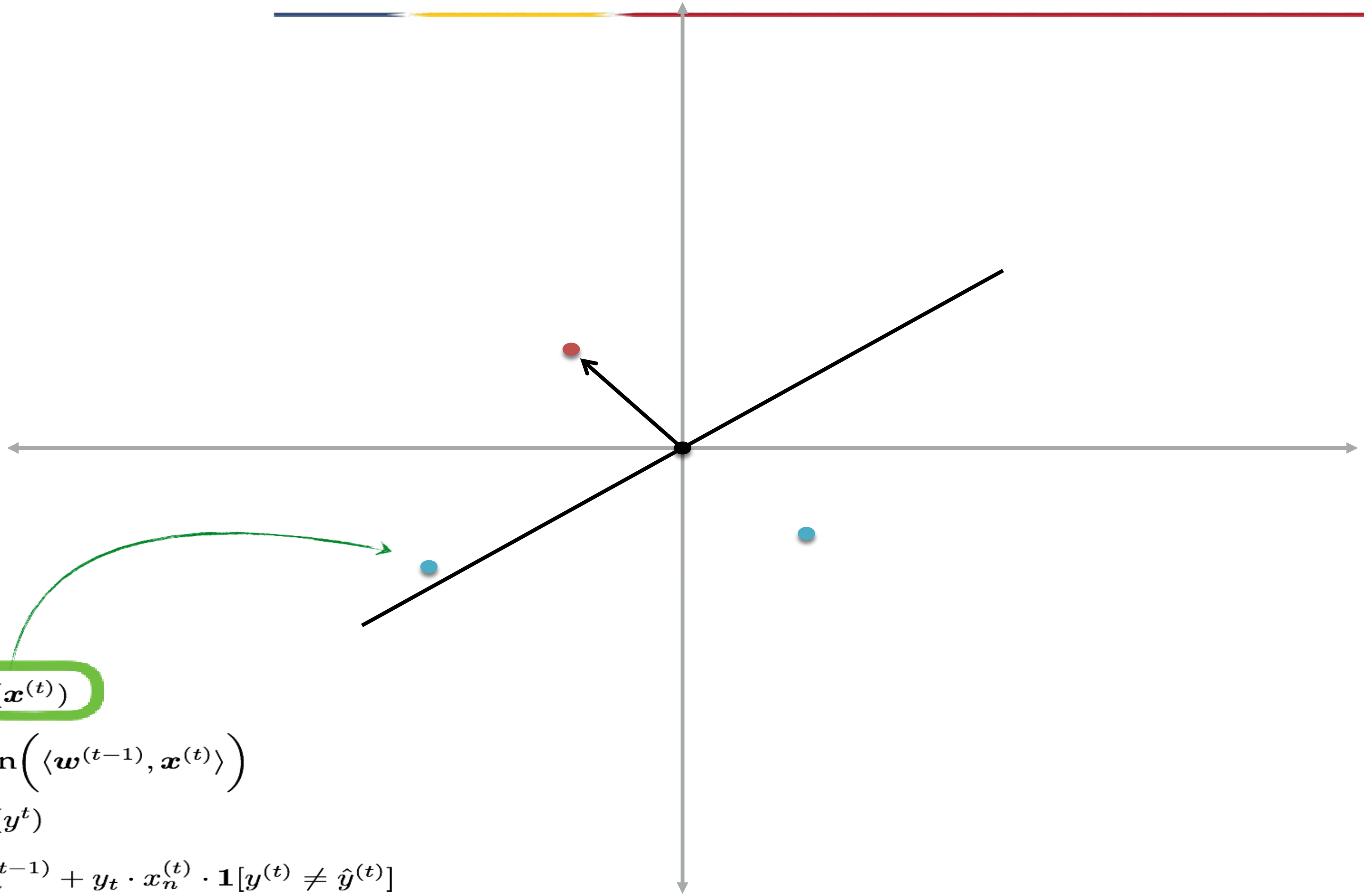


$\textsc{Receive}(\boldsymbol{x}^{(t)})$

$$\hat{y}_A^{(t)} = \mathrm{sign}\Big(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^{(t)} \rangle\Big)$$

$\textsc{Receive}(y^{t})$

$$w_n^{(t)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$$

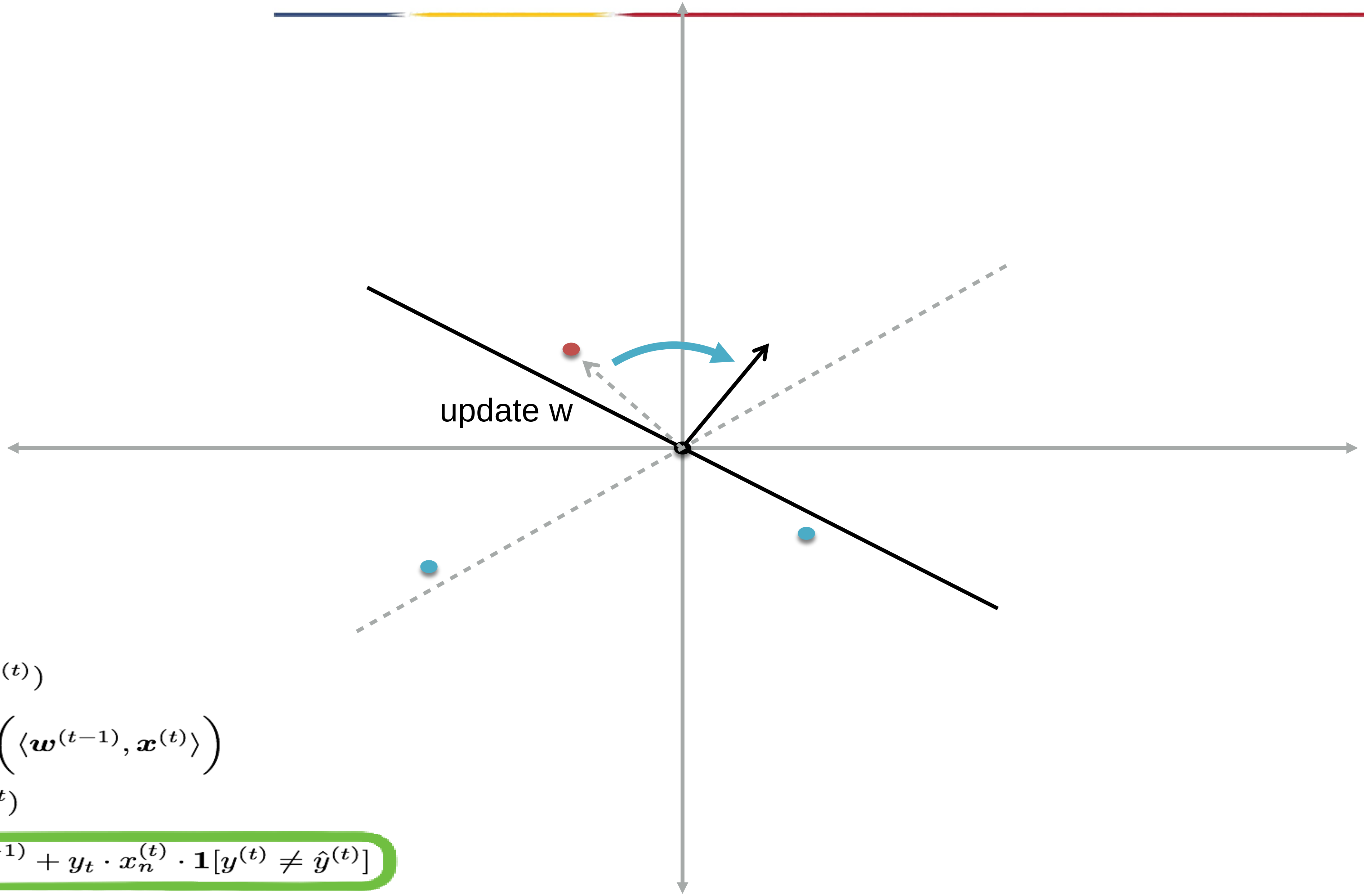


$\textsc{Receive}(\boldsymbol{x}^{(t)})$

$\hat{y}_A^{(t)} = \mathrm{sign}\Big(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^{(t)} \rangle\Big)$

$\textsc{Receive}(y^t)$

$w_n^{(t)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$



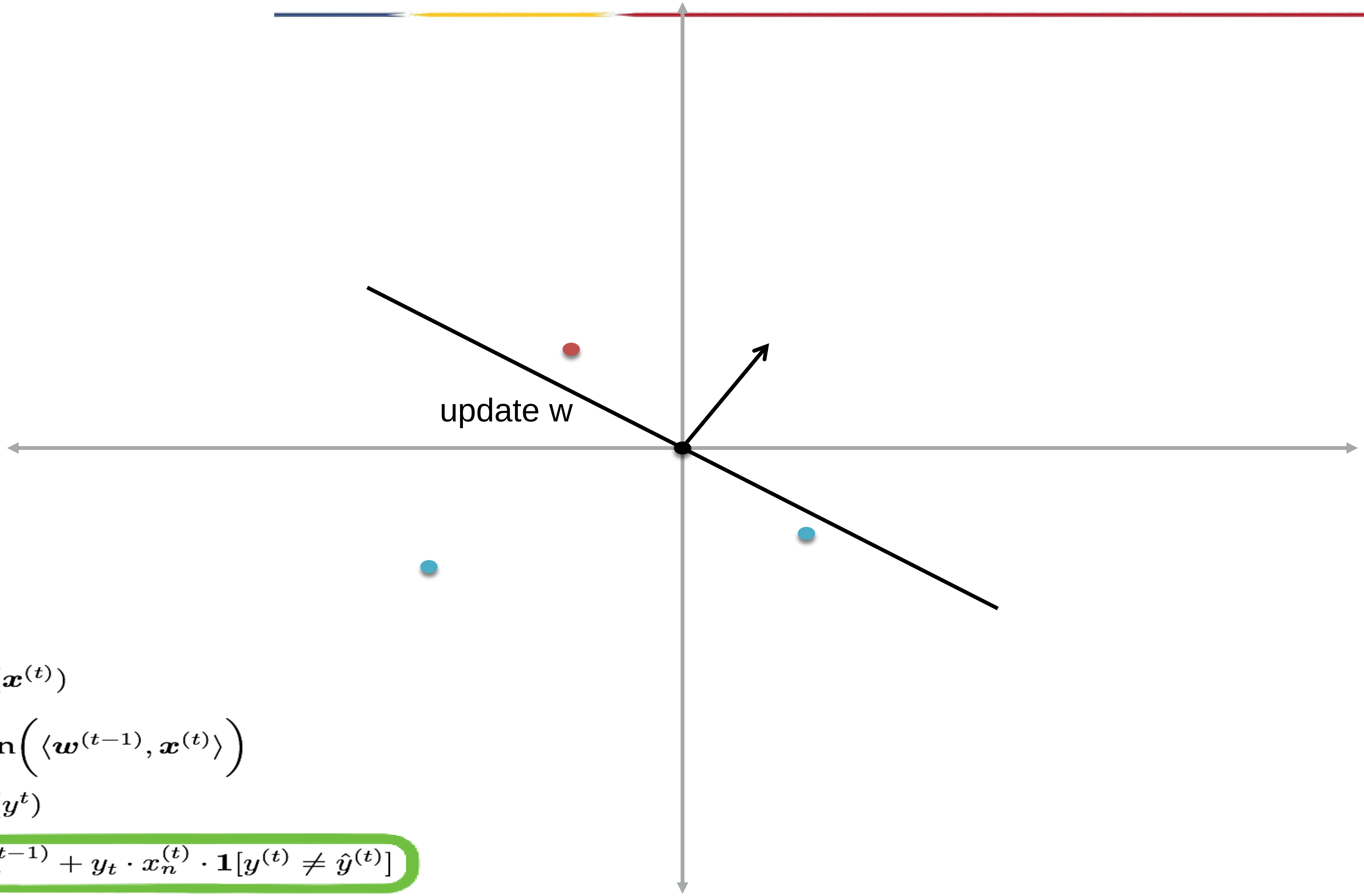
$\textsc{Receive}(\boldsymbol{x}^{(t)})$

$$\hat{y}_A^{(t)} = \mathrm{sign}\Big(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^{(t)} \rangle\Big)$$

$\textsc{Receive}(y^t)$

$$\boxed{w_n^{(t)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]}$$

…Perceptron



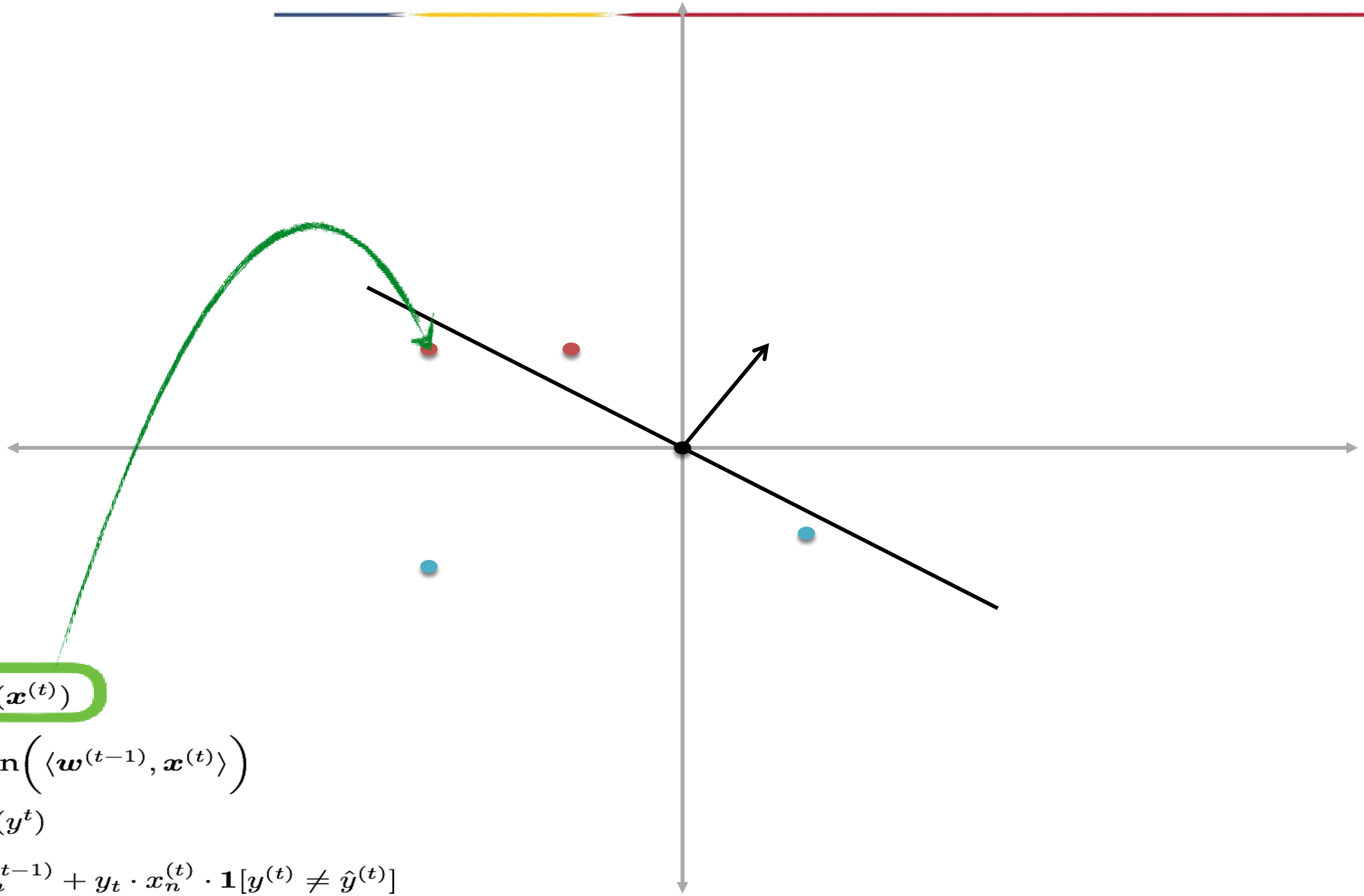
$\textsc{Receive}(\boldsymbol{x}^{(t)})$

$$\hat{y}_A^{(t)} = \text{sign}\left(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^{(t)} \rangle\right)$$

$\textsc{Receive}(y^t)$

$$\boxed{w_n^{(t)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]}$$

...Perceptron



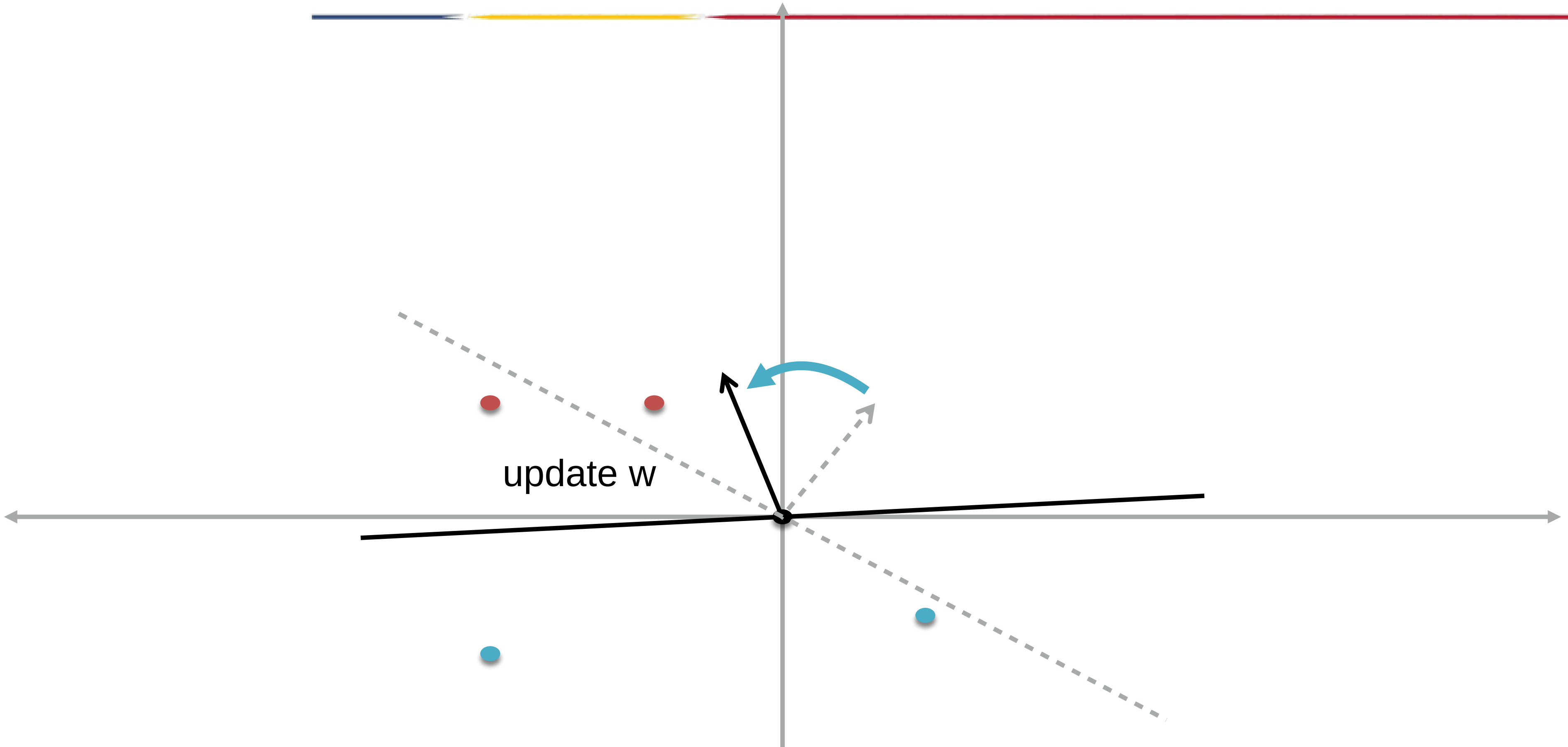
$\textsc{Receive}(\boldsymbol{x}^{(t)})$

$\hat{y}_A^{(t)} = \text{sign}\Big(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^{(t)} \rangle\Big)$

$\textsc{Receive}(y^t)$

$w_n^{(t)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$

…Perceptron

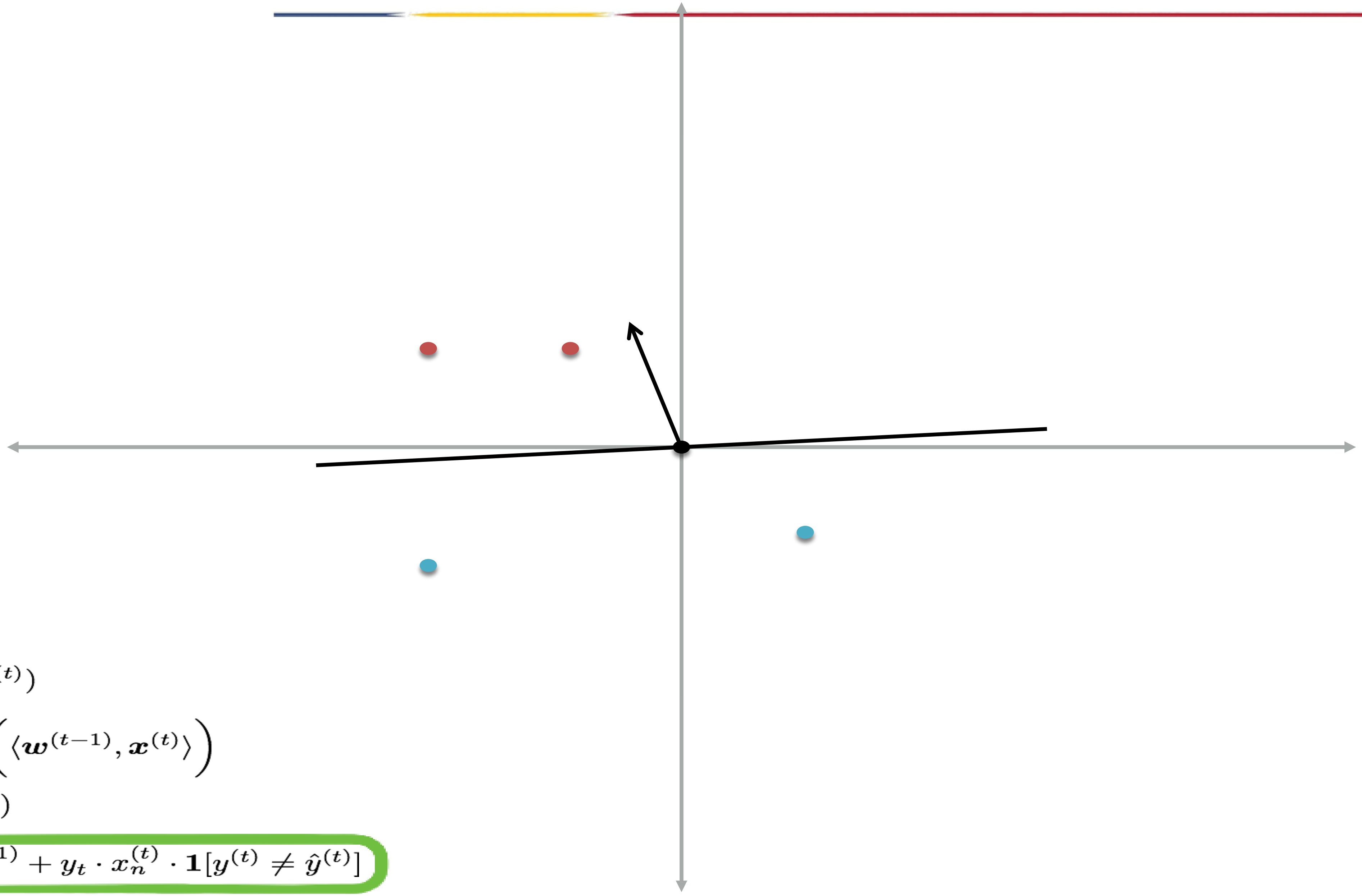


$$\textsc{Receive}(\boldsymbol{x}^{(t)})$$

$$\hat{y}_A^{(t)} = \mathrm{sign}\Big(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^{(t)} \rangle\Big)$$

$$\textsc{Receive}(y^t)$$

$$\boxed{w_n^{(t)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]}$$



$\textsc{Receive}(\boldsymbol{x}^{(t)})$

$\hat{y}_A^{(t)} = \text{sign}\Big(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}^{(t)} \rangle\Big)$

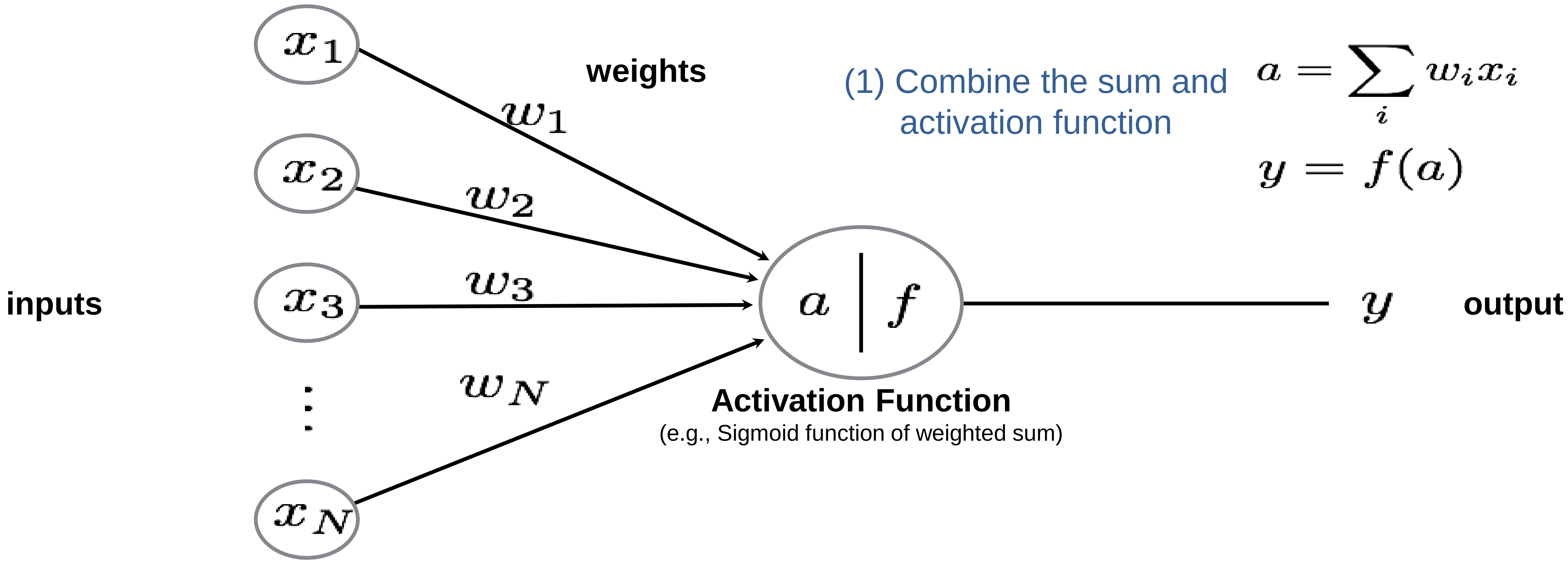
$\textsc{Receive}(y^t)$

$w_n^{(t)} = w_n^{(t-1)} + y_t \cdot x_n^{(t)} \cdot \mathbf{1}[y^{(t)} \neq \hat{y}^{(t)}]$

repeat …

…Perceptron

Another way to draw it…



(1) Combine the sum and activation function

$$a = \sum_i w_i x_i$$

$$y = f(a)$$

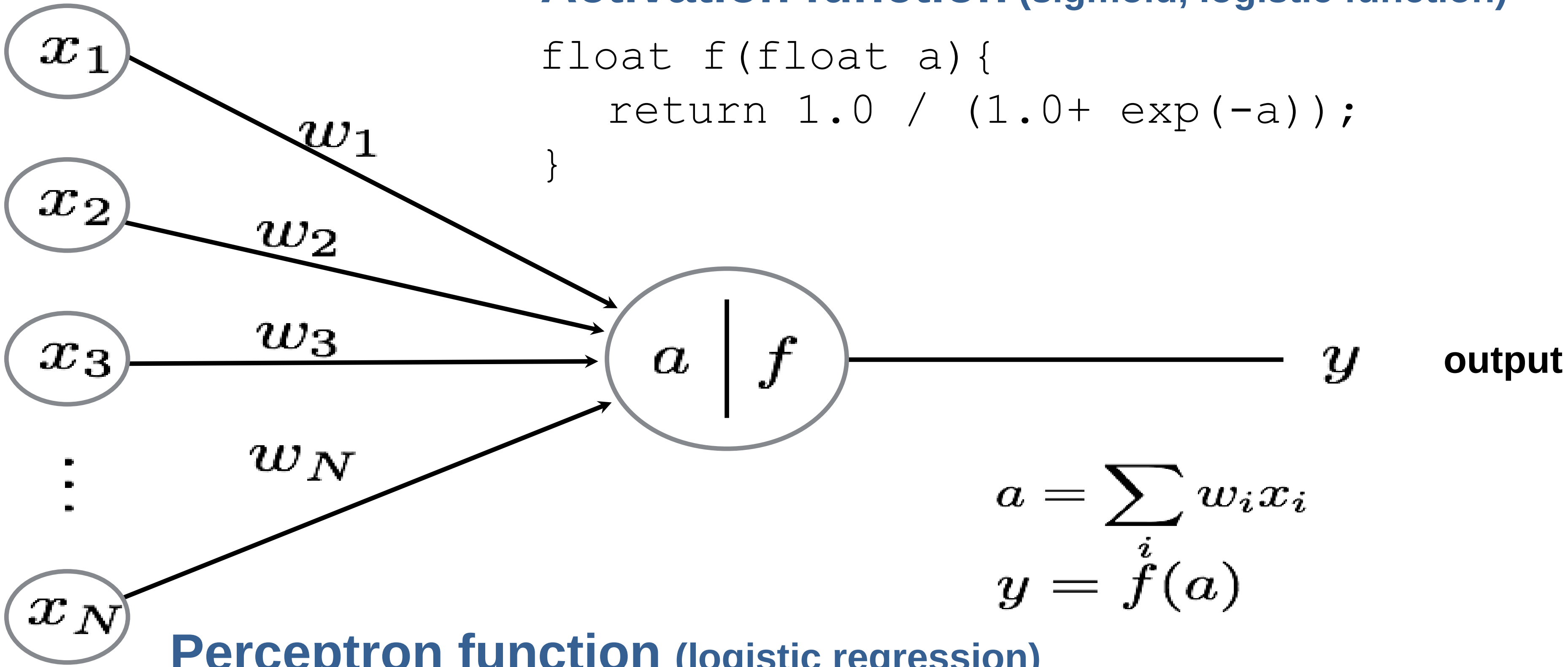
(2) suppress the bias term (less clutter)

$$x_N = 1$$
$$w_N = b$$

Programming the 'forward pass'

Activation function (sigmoid, logistic function)

```
float f(float a){
    return 1.0 / (1.0+ exp(-a));
}
```



$$a = \sum_i w_i x_i$$

$$y = f(a)$$

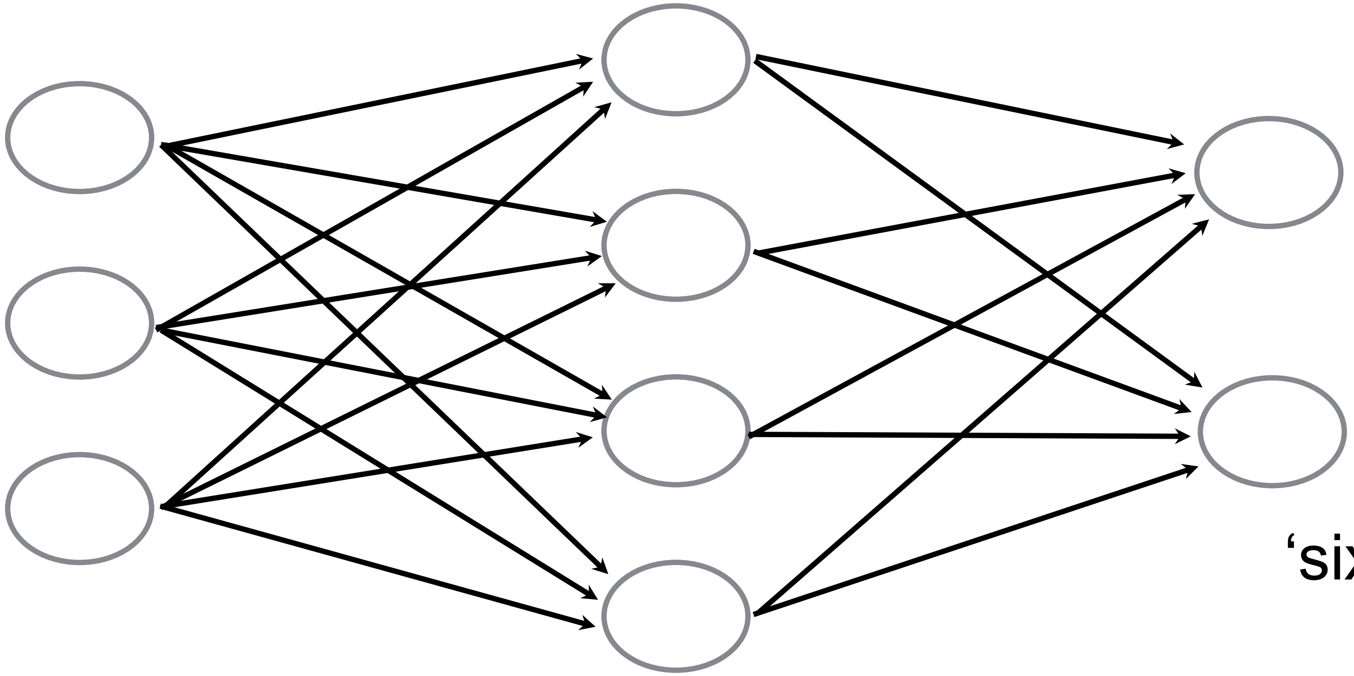
Perceptron function (logistic regression)

```
float perceptron(vector<float> x, vector<float> w){
    float a  = dot(x,w);
    return f(a);
}
```

CONTENTS

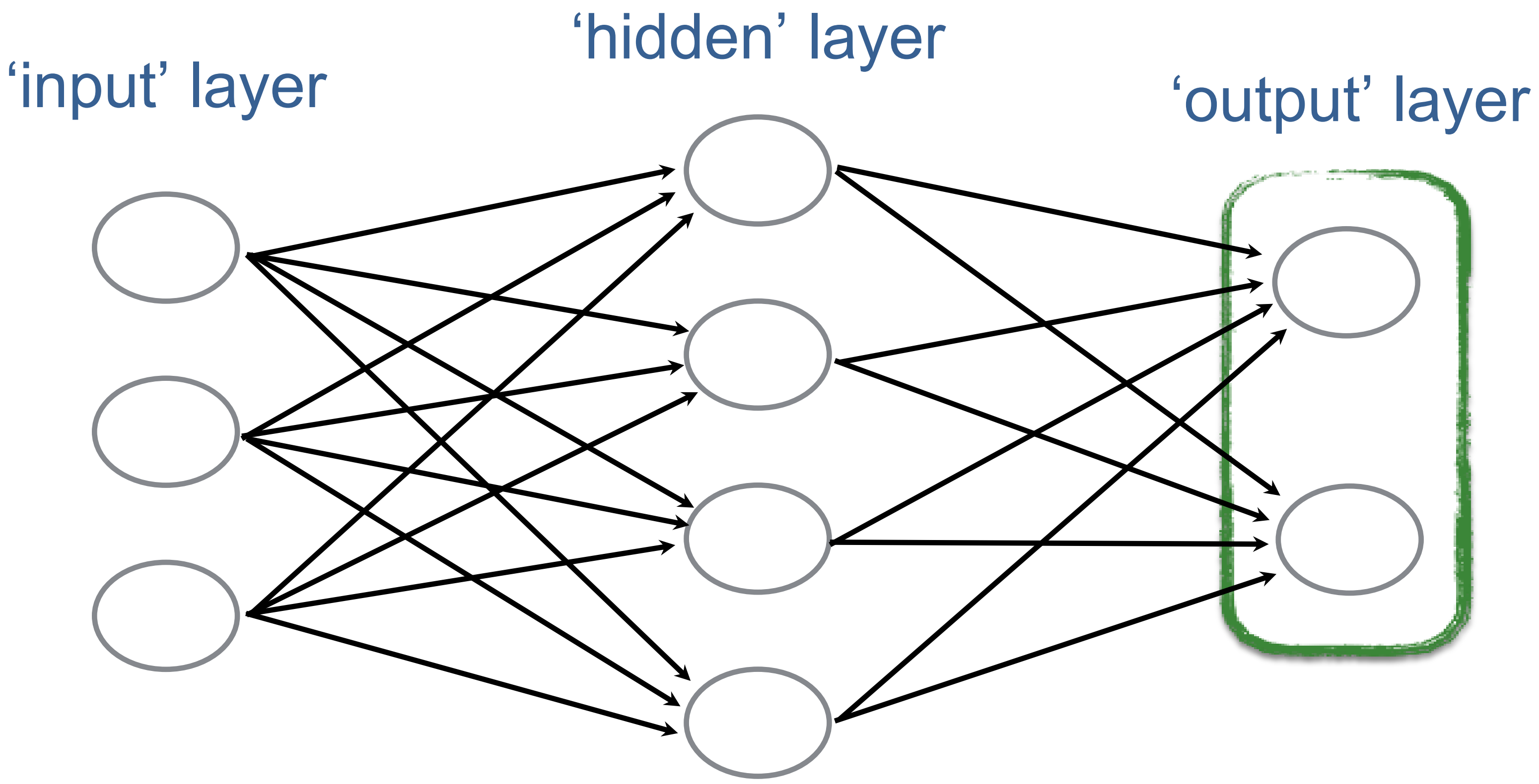
Connect a bunch of perceptrons together …

a collection of connected perceptrons



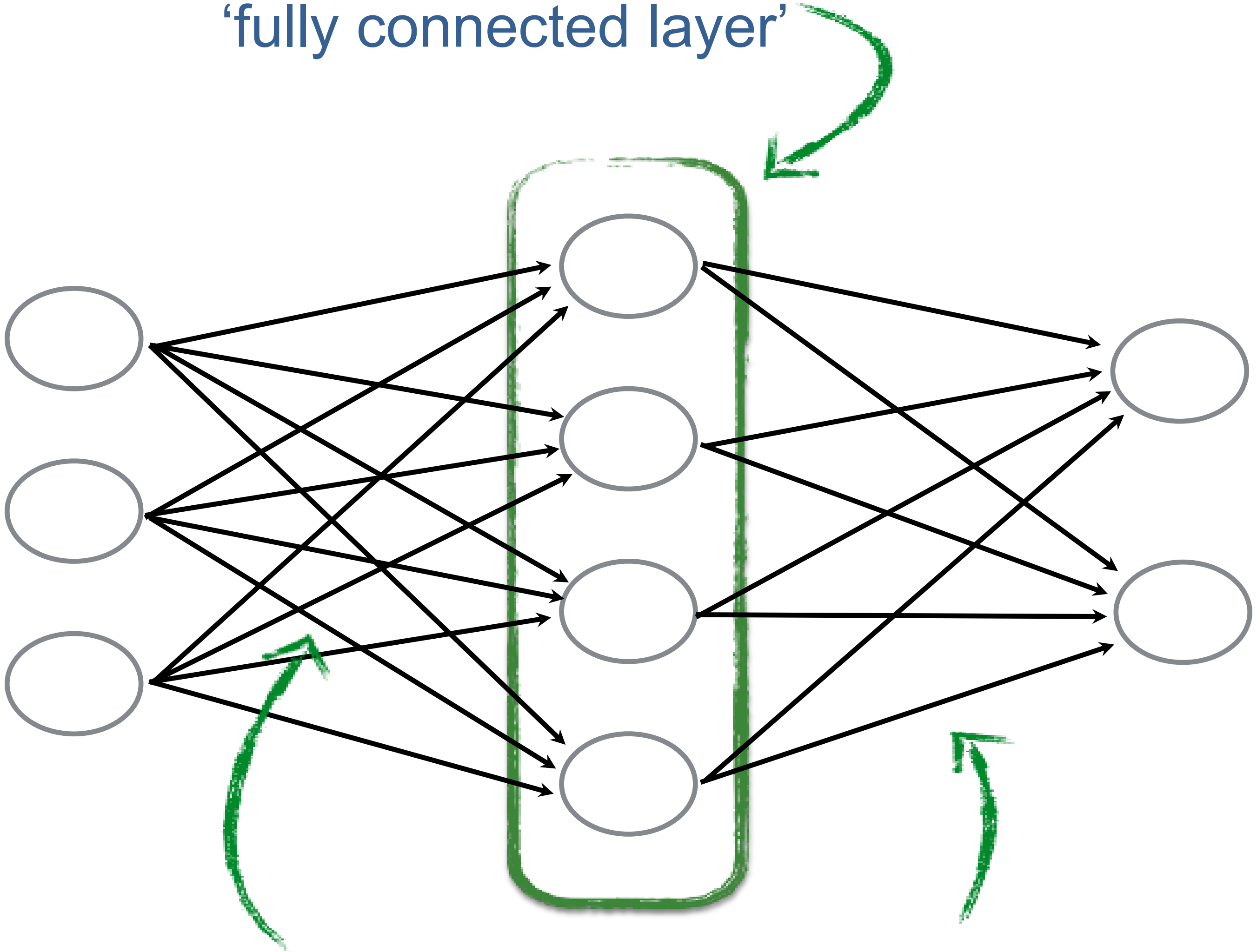
'six perceptrons'

Some terminology…



…also called a **Multi-layer Perceptron** (MLP)

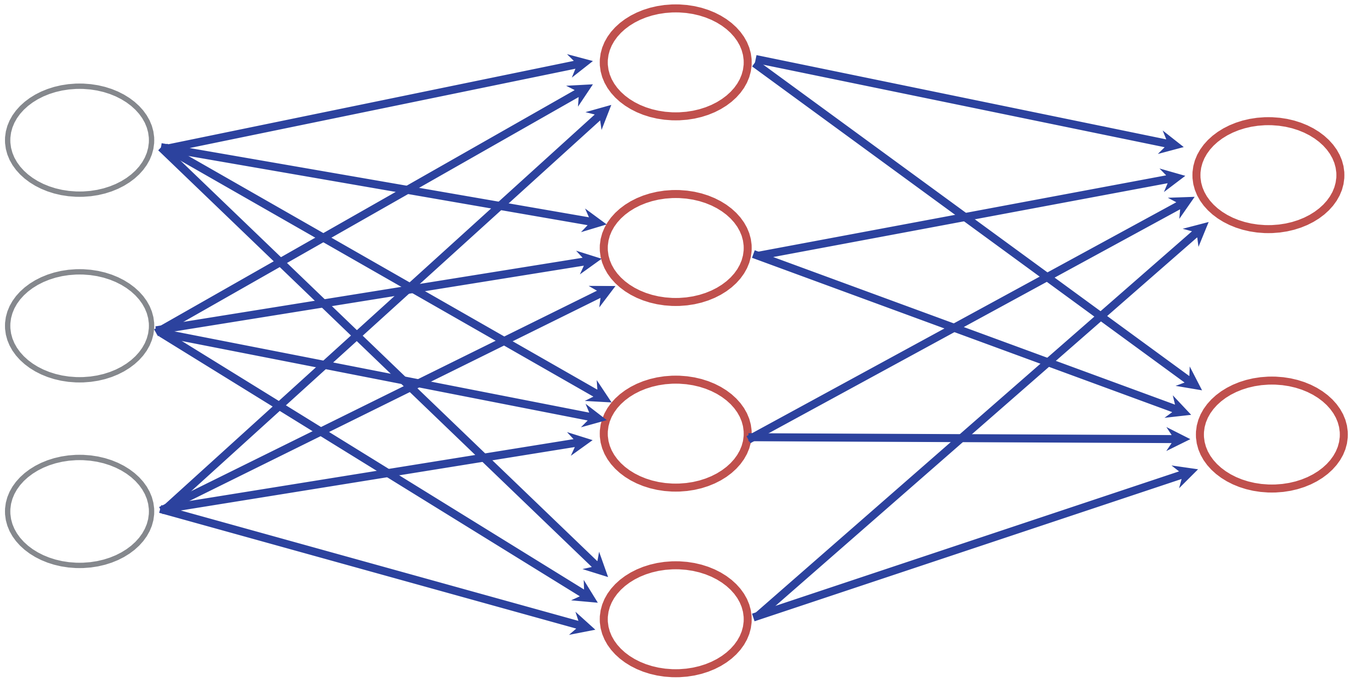
this layer is a
'fully connected layer'



all pairwise neurons between layers are connected

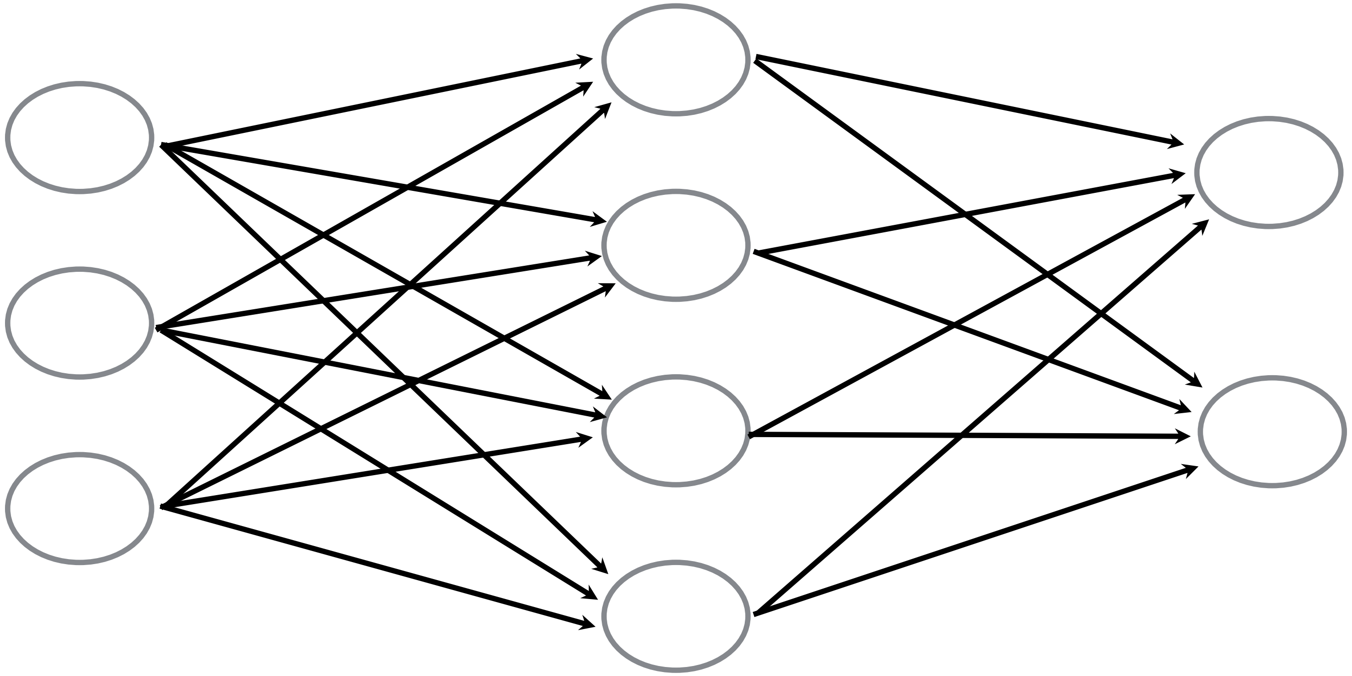
How many neurons (perceptrons)? **4 + 2 = 6**

How many weights (edges)? **(3 x 4) + (4 x 2) = 20**



How many learnable parameters total? **20 + 6 = 26**

performance usually tops out at 2-3 layers,
deeper networks don't really improve performance...



...with the exception of **Convolutional Neural Networks** for images

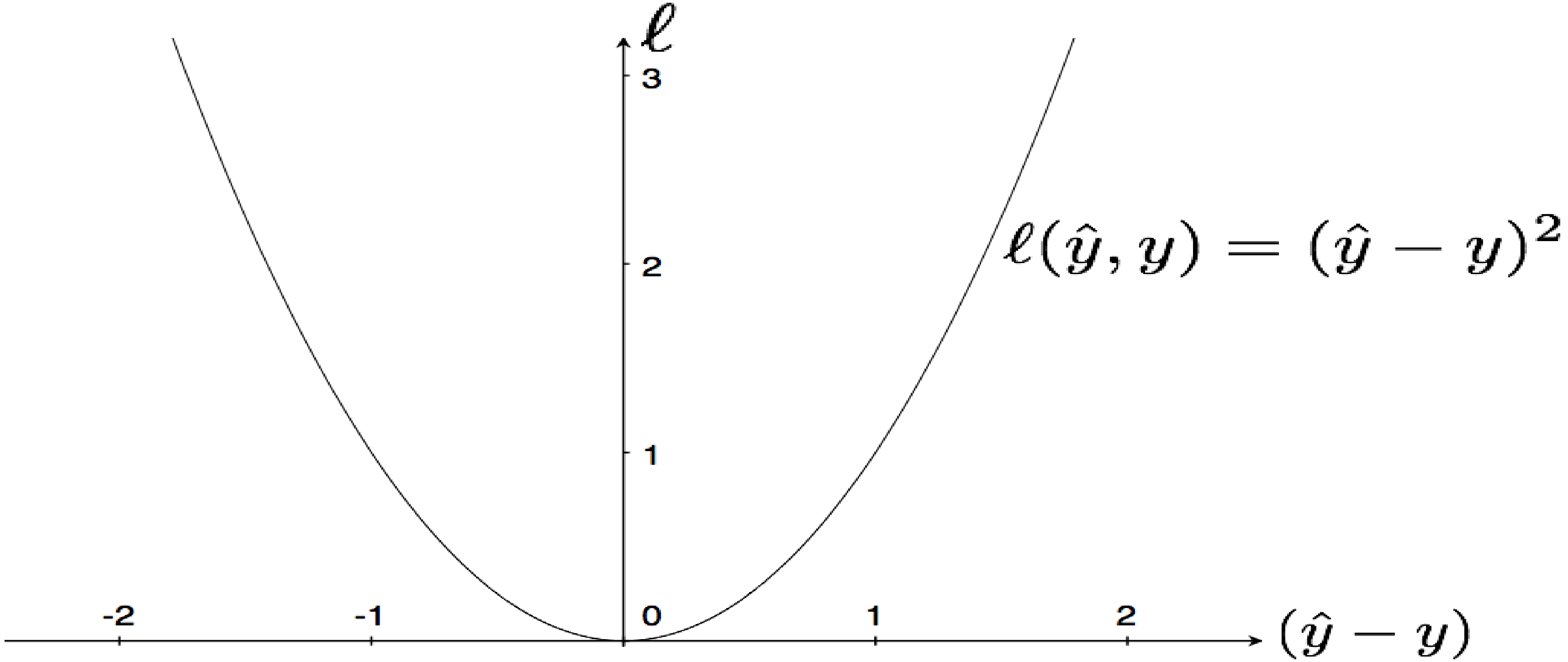
CONTENTS

- Perceptron.
- Neural networks.
- **Gradient descent**
- Backpropagation.
- Stochastic gradient descent.

Loss Function: defines what is means to be **close** to the true solution

⇨**chose the loss function!** (some are better than others depending on what you want to do)

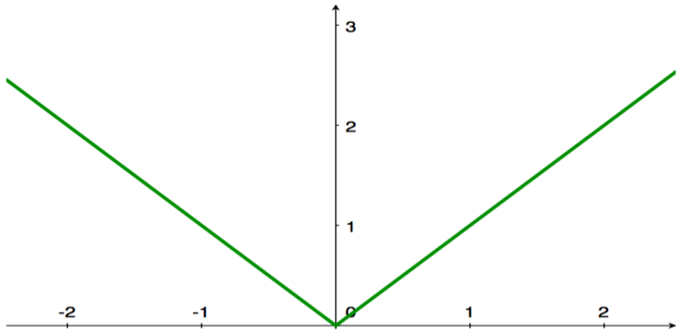
Squared Error
(a popular loss function)



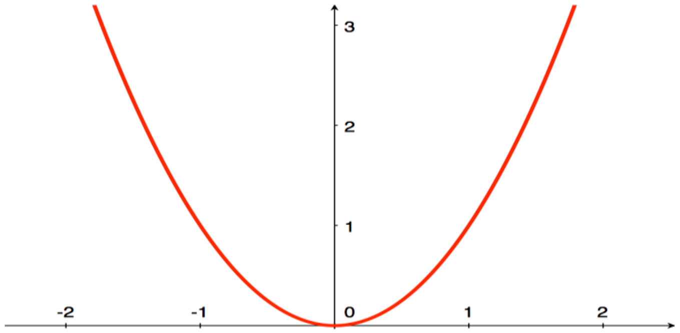
Loss Function: defines what is means to be **close** to the true solution

⇨**chose the loss function!** (some are better than others depending on what you want to do)

$$\ell(\hat{y}, y) = |\hat{y} - y|$$



$$\ell(\hat{y}, y) = (\hat{y} - y)^2$$



Gradient descent

world's smallest perceptron!

$$x \xrightarrow{w} f \longrightarrow y = wx$$

(a.k.a. line equation, linear regression)

Given several examples

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

and a perceptron

$$\hat{y} = wx$$

Modify weight w such that $\hat{y}$ gets '**closer**' to y

perceptron parameter **perceptron output** **true label**

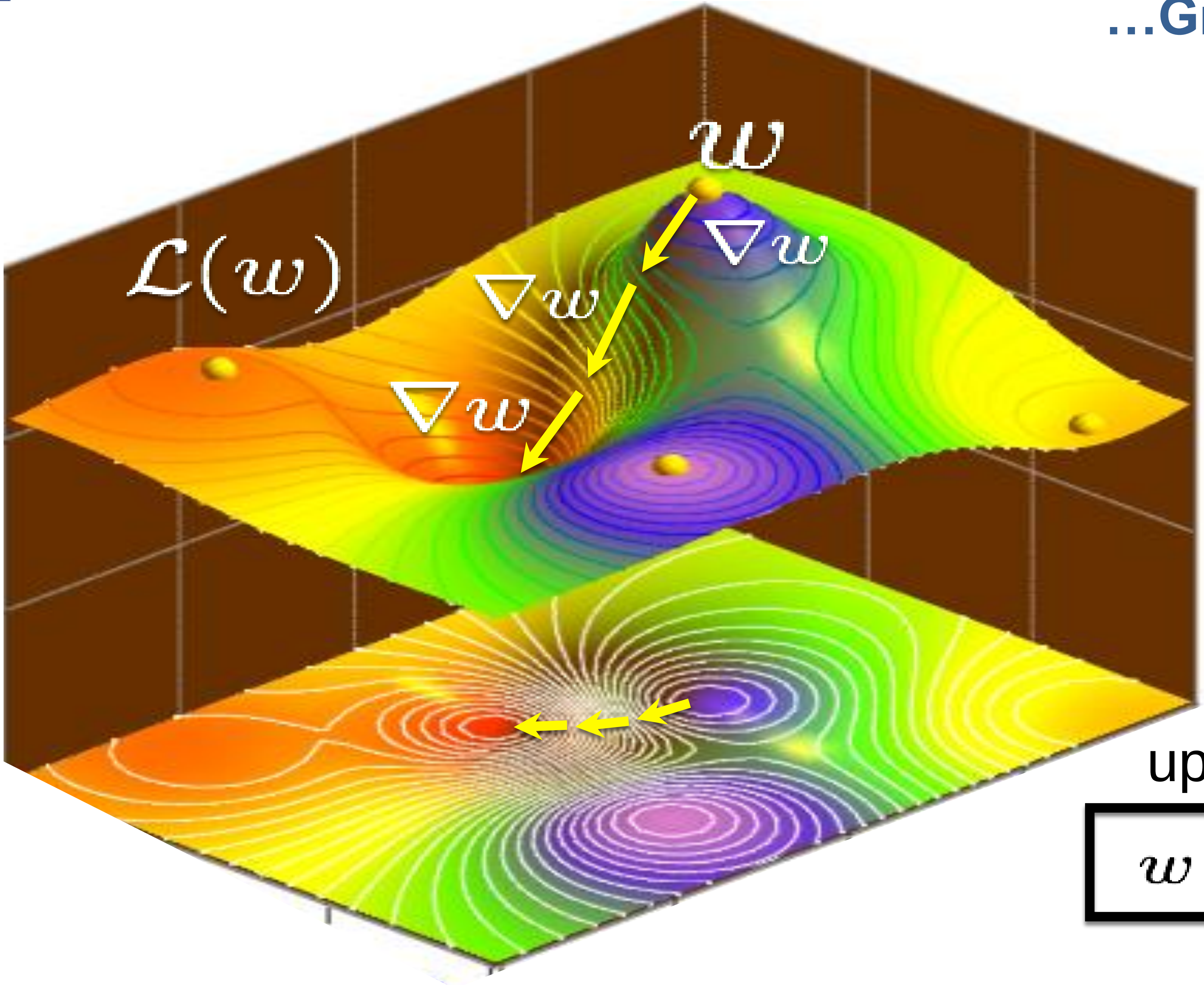
Code to train perceptron:

$$
\begin{aligned}
&\text{for } n = 1 \ldots N \\
&\quad w = w + (y_n - \hat{y})x_i;
\end{aligned}
$$

just one line of code!

Now where does this come from?

…Gradient descent



update rule:

$$\boxed{w = w - \nabla w}$$

CONTENTS

Backpropagation

$$x \xrightarrow{\;w\;} f \longrightarrow y = wx$$

⇨ function of **ONE** parameter!

Training the world's smallest perceptron

$$\begin{aligned}&\text{for } \; n = 1 \ldots N \\ &\qquad w = w + \underline{(y_n - \hat{y})x_i};\end{aligned}$$

This is just gradient descent, that means…

this should be the gradient of the loss function

Now where does this come from?

$$\frac{d\mathcal{L}}{dw}$$ …is the rate at which **this** will change…

$$\mathcal{L} = \frac{1}{2}(y - \hat{y})^2$$

the loss function

… per unit change of **this**

$$y = wx$$

the weight parameter

Let's compute the derivative…

Compute the derivative

$$\frac{d\mathcal{L}}{dw} = \frac{d}{dw}\left\{\frac{1}{2}(y-\hat{y})^2\right\}$$

$$= -(y-\hat{y})\frac{dwx}{dw}$$

$$= -(y-\hat{y})x = \nabla w$$

just shorthand

That means the weight update for **gradient descent** is:

$$w = w - \nabla w$$

move in direction of negative gradient

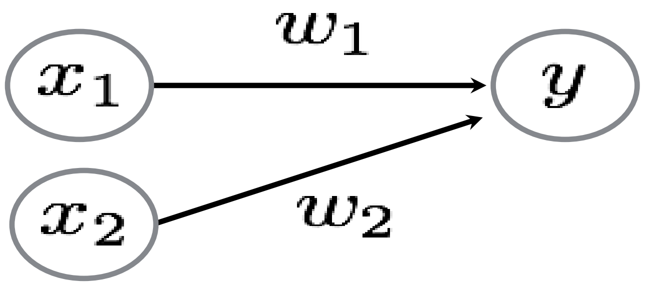
$$= w + (y-\hat{y})x$$

Gradient Descent (world's smallest perceptron)

For each sample $\{x_i, y_i\}$

1. Predict
   a. Forward pass $\hat{y} = wx_i$
   b. Compute Loss $\mathcal{L}_i = \frac{1}{2}(y_i - \hat{y})^2$
2. Update
   a. Back Propagation $\frac{d\mathcal{L}_i}{dw} = -(y_i - \hat{y})x_i = \nabla w$
   b. Gradient update $w = w - \nabla w$

world's (second) smallest **perceptron**!



function of **two** parameters!

Derivative computation

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_1} &= \frac{\partial}{\partial w_1}\left\{\frac{1}{2}(y-\hat{y})^2\right\} \\ &= -(y-\hat{y})\frac{\partial \hat{y}}{\partial w_1} \\ &= -(y-\hat{y})\frac{\partial \sum_i w_i x_i}{\partial w_1} \\ &= -(y-\hat{y})\frac{\partial w_1 x_1}{\partial w_1} \\ &= -(y-\hat{y})x_1 = \nabla w_1\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_2} &= \frac{\partial}{\partial w_2}\left\{\frac{1}{2}(y-\hat{y})^2\right\} \\ &= -(y-\hat{y})\frac{\partial \hat{y}}{\partial w_2} \\ &= -(y-\hat{y})\frac{\partial \sum_i w_i x_i}{\partial w_1} \\ &= -(y-\hat{y})\frac{\partial w_2 x_2}{\partial w_2} \\ &= -(y-\hat{y})x_2 = \nabla w_2\end{aligned}$$

Gradient Update

$$\begin{aligned}w_1 &= w_1 - \eta\nabla w_1 \\ &= w_1 + \eta(y-\hat{y})x_1\end{aligned}$$

$$\begin{aligned}w_2 &= w_2 - \eta\nabla w_2 \\ &= w_2 + \eta(y-\hat{y})x_2\end{aligned}$$

Gradient Descent

For each sample $\{x_i, y_i\}$

1. Predict
   a. Forward pass $\hat{y} = f_{\text{MLP}}(x_i; \theta)$
   b. Compute Loss $\mathcal{L}_i = \frac{1}{2}(y_i - \hat{y})$

(side computation to track loss. not needed for backprop)

2. Update
   a. Back Propagation

two lines now

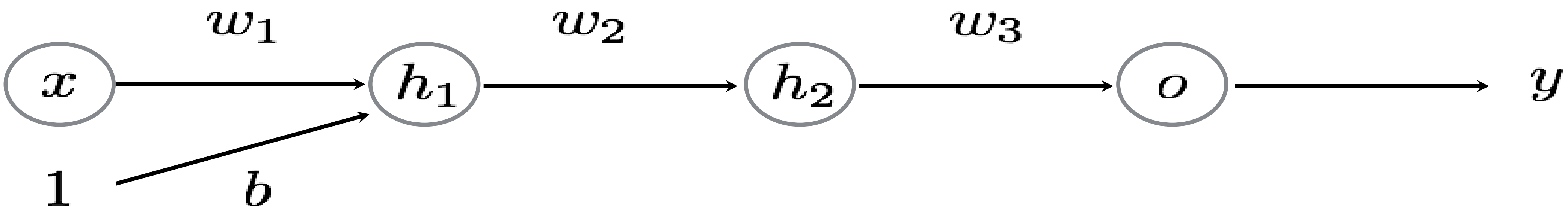
$$\nabla w_{1i} = -(y_i - \hat{y})x_{1i}$$
$$\nabla w_{2i} = -(y_i - \hat{y})x_{2i}$$

   b. Gradient update

$$w_{1i} = w_{1i} + \eta(y - \hat{y})x_{1i}$$
$$w_{2i} = w_{2i} + \eta(y - \hat{y})x_{2i}$$

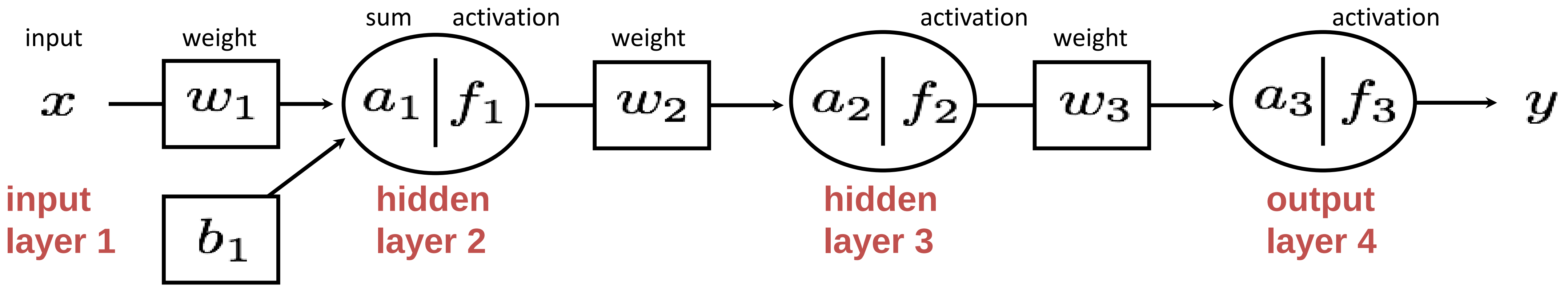
(adjustable step size)

multi-layer perceptron

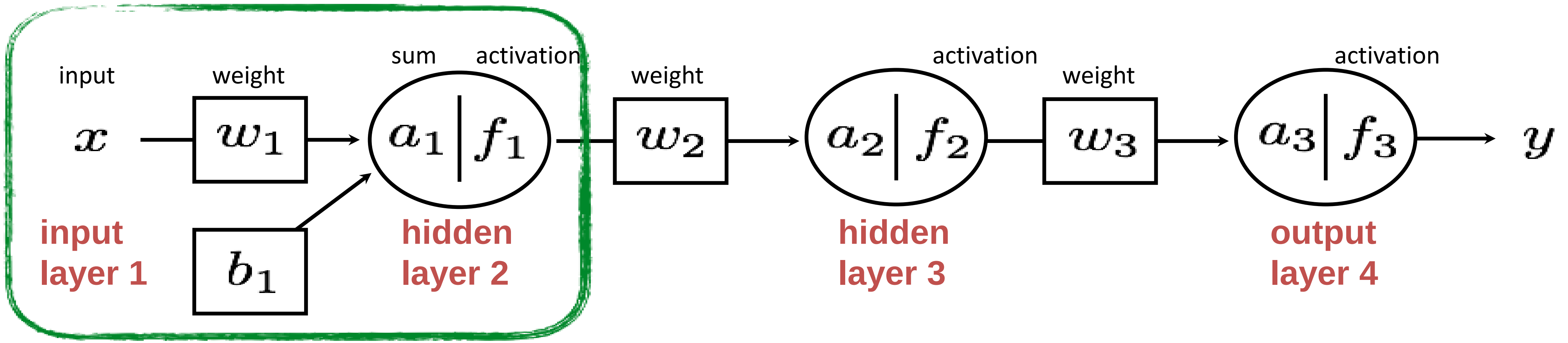


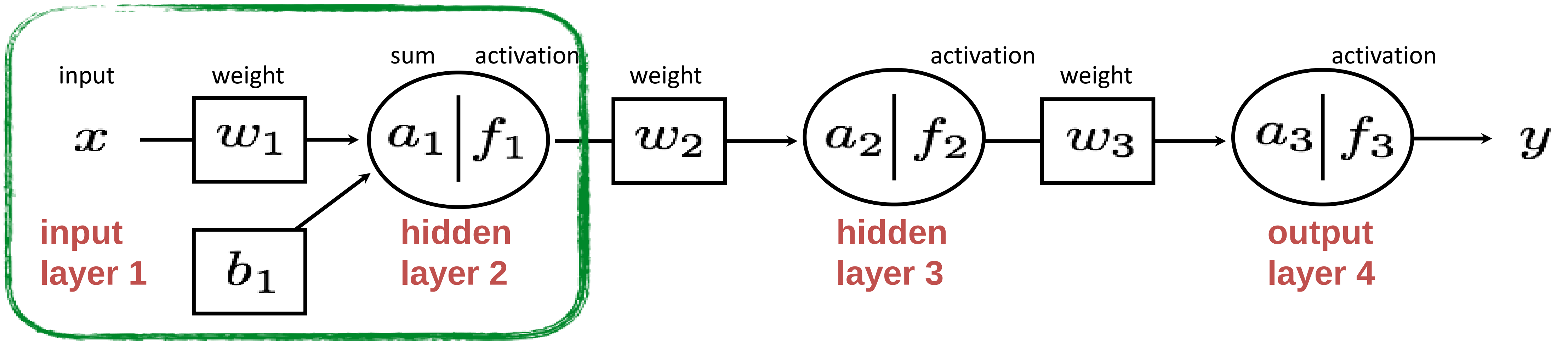
function of **FOUR** parameters and **FOUR** layers!

Backpropagation



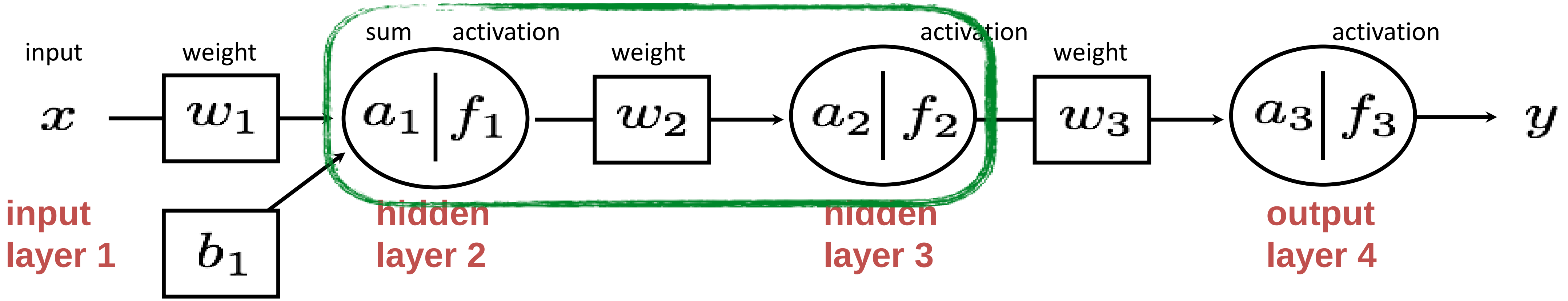
Backpropagation





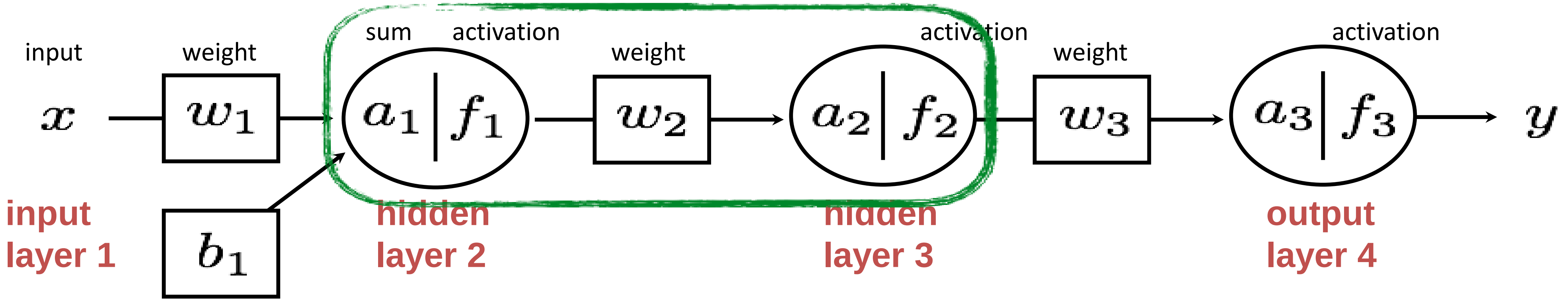
$$a_1 = w_1 \cdot x + b_1$$

Backpropagation



$$a_1 = w_1 \cdot x + b_1$$

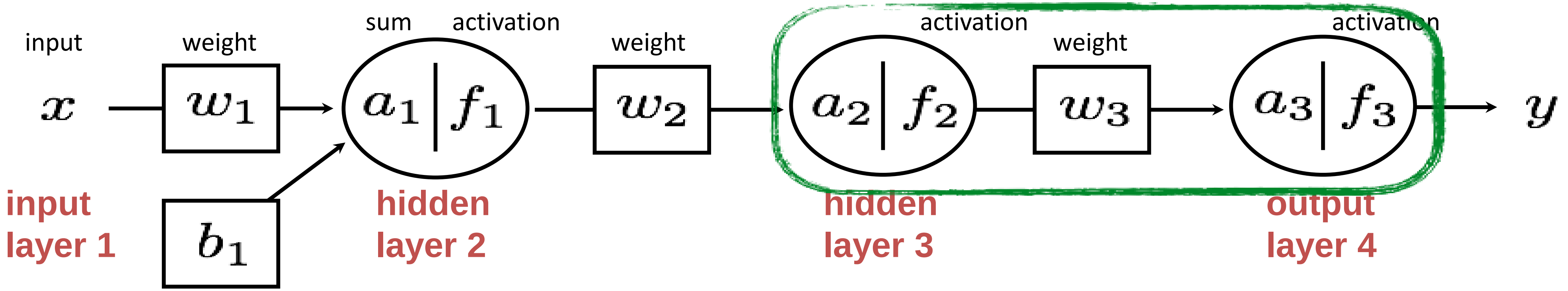
Backpropagation



$$a_1 = w_1 \cdot x + b_1$$

$$a_2 = w_2 \cdot f_1(w_1 \cdot x + b_1)$$

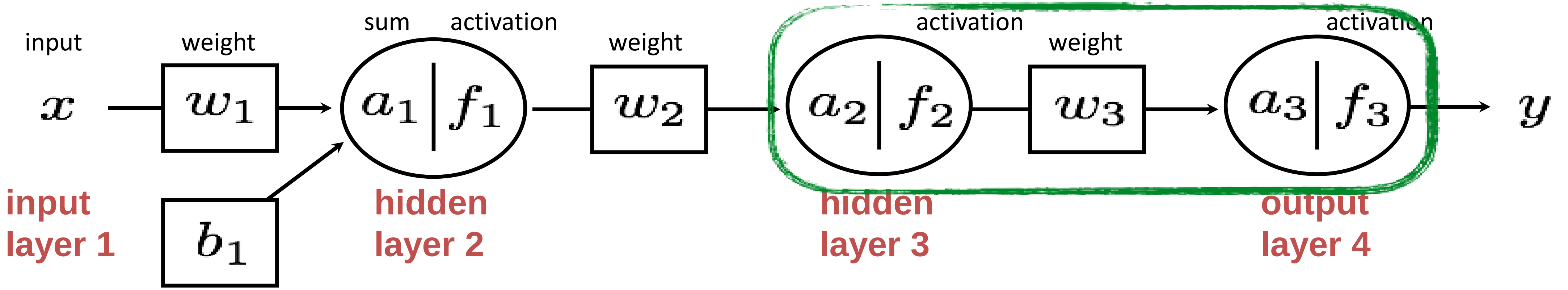
Backpropagation



$$a_1 = w_1 \cdot x + b_1$$

$$a_2 = w_2 \cdot f_1(w_1 \cdot x + b_1)$$

Backpropagation

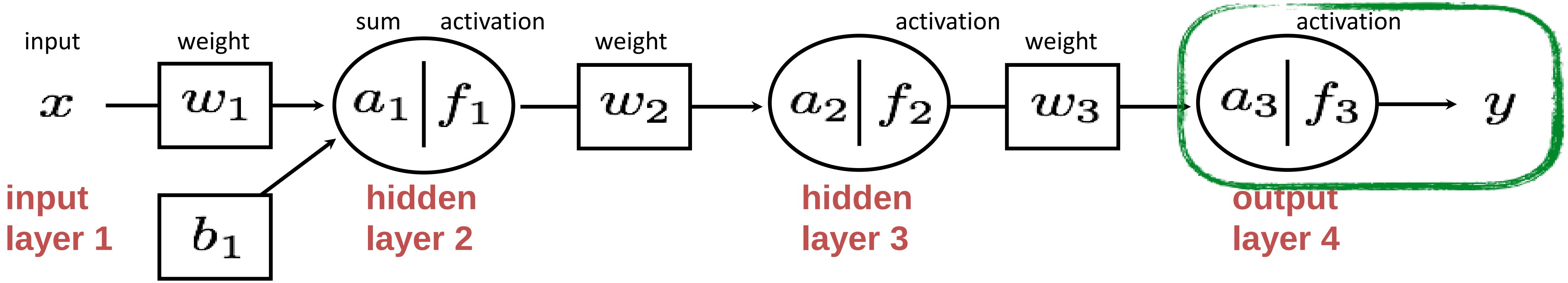


$$a_1 = w_1 \cdot x + b_1$$

$$a_2 = w_2 \cdot f_1(w_1 \cdot x + b_1)$$

$$a_3 = w_3 \cdot f_2(w_2 \cdot f_1(w_1 \cdot x + b_1))$$

Backpropagation

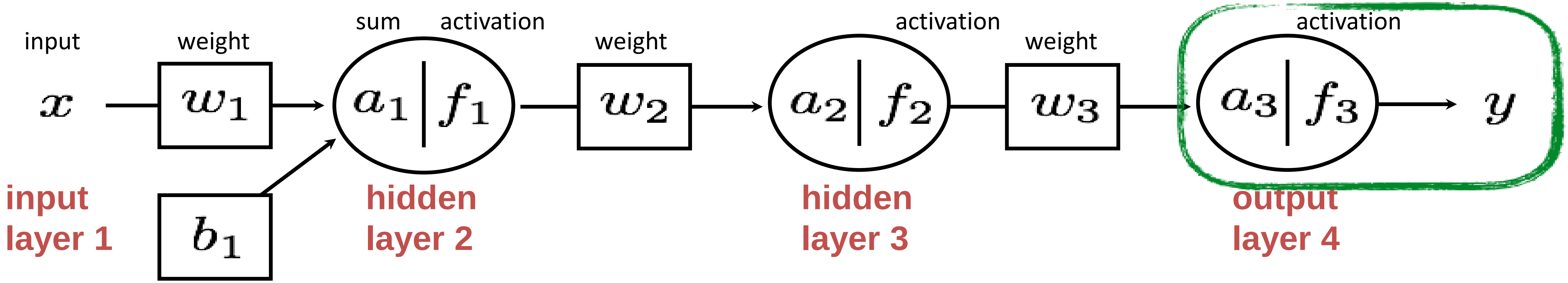


$$a_1 = w_1 \cdot x + b_1$$

$$a_2 = w_2 \cdot f_1(w_1 \cdot x + b_1)$$

$$a_3 = w_3 \cdot f_2(w_2 \cdot f_1(w_1 \cdot x + b_1))$$

Backpropagation



$$a_1 = w_1 \cdot x + b_1$$

$$a_2 = w_2 \cdot f_1(w_1 \cdot x + b_1)$$

$$a_3 = w_3 \cdot f_2(w_2 \cdot f_1(w_1 \cdot x + b_1))$$

$$y = f_3(w_3 \cdot f_2(w_2 \cdot f_1(w_1 \cdot x + b_1)))$$

Entire network can be written out as one long equation

$$y = f_3(w_3 \cdot f_2(w_2 \cdot f_1(w_1 \cdot x + b_1)))$$

We need to train the network:

What is known? What is unknown?

Entire network can be written out as a long equation

$$y = f_3(w_3 \cdot f_2(w_2 \cdot f_1(w_1 \cdot x + b_1)))$$

known

We need to train the network:

What is known? What is unknown?

Entire network can be written out as a long equation

$$y = f_3(w_3 \cdot f_2(w_2 \cdot f_1(w_1 \cdot x + b_1)))$$

activation function sometimes has parameters

unknown

We need to train the network:

What is known? What is unknown?

Learning an MLP

Given a set of samples and a MLP

$$\{x_i, y_i\}$$

$$y = f_{\mathrm{MLP}}(x; \theta)$$

Estimate the parameters of the MLP

$$\theta = \{f, w, b\}$$

Gradient Descent

For each **random** sample $\{x_i, y_i\}$

1. Predict

    a. Forward pass $\hat{y} = f_{\text{MLP}}(x_i; \theta)$

    b. Compute Loss

2. Update

    a. Back Propagation $\frac{\partial \mathcal{L}}{\partial \theta}$ vector of parameter partial derivatives

    b. Gradient update $\theta \leftarrow \theta - \eta \nabla \theta$

vector of parameter update equations

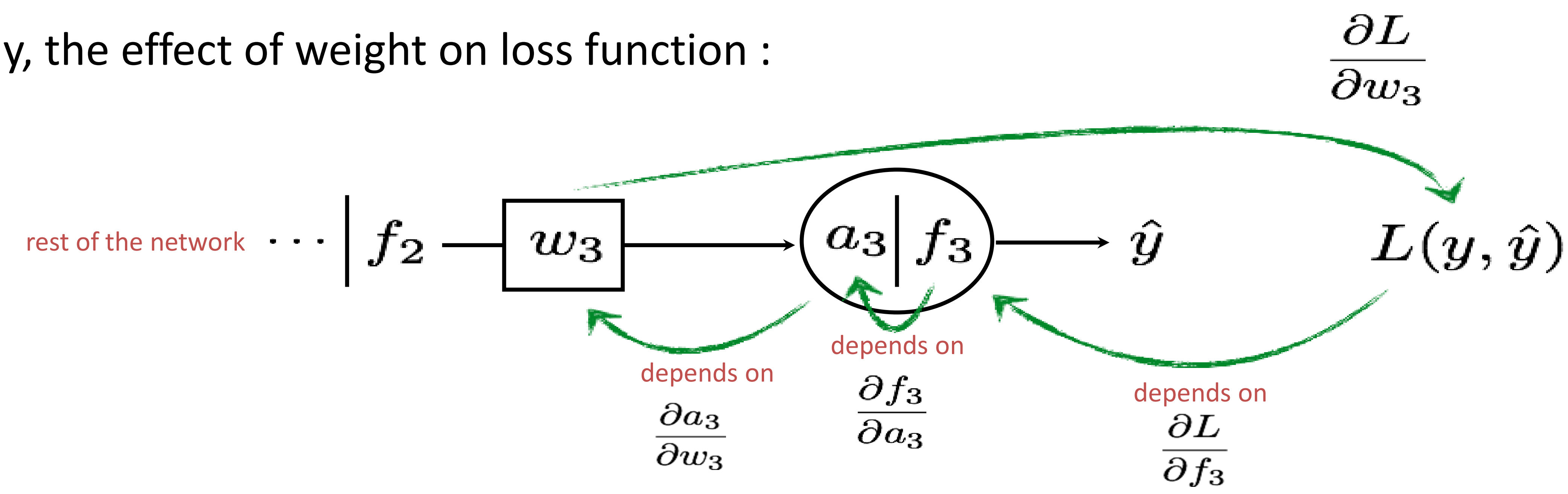
So we need to compute the partial derivatives

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \left[\frac{\partial \mathcal{L}}{\partial w_3} \frac{\partial \mathcal{L}}{\partial w_2} \frac{\partial \mathcal{L}}{\partial w_1} \frac{\partial \mathcal{L}}{\partial b}\right]$$

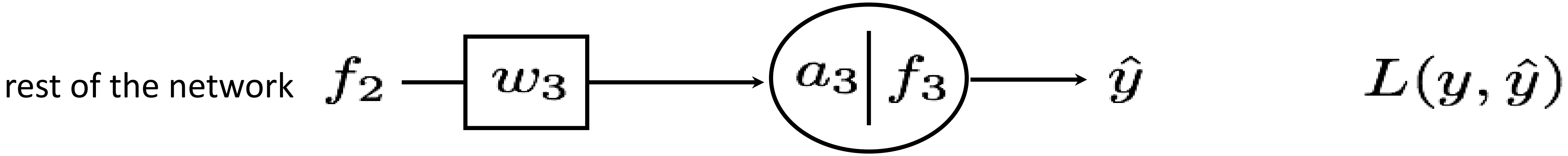
According to the chain rule…

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial w_3}$$

Intuitively, the effect of weight on loss function :



Backpropagation



$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial w_3}$$

Chain Rule!

Backpropagation

rest of the network $f_2 \longrightarrow \boxed{w_3} \longrightarrow \left(a_3 \mid f_3 \right) \longrightarrow \hat{y}$ $\qquad L(y, \hat{y})$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial f_3} \frac{\partial f_3}{\partial a_3} \frac{\partial a_3}{\partial w_3}$$

$$= -\eta(y - \hat{y}) \frac{\partial f_3}{\partial a_3} \frac{\partial a_3}{\partial w_3}$$

Just the partial derivative of L2 loss

Backpropagation

rest of the network $f_2 \longrightarrow \boxed{w_3} \longrightarrow \left(a_3 \,|\, f_3 \right) \longrightarrow \hat{y}$ $\qquad L(y, \hat{y})$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial w_3}$$

$$= -\eta(y - \hat{y})\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial w_3}$$

Let's use a Sigmoid function

$$\frac{ds(x)}{dx} = s(x)(1 - s(x))$$

Backpropagation

rest of the network $f_2 \longrightarrow \boxed{w_3} \longrightarrow (a_3 \mid f_3) \longrightarrow \hat{y}$ $\qquad L(y, \hat{y})$

$$
\begin{aligned}
\frac{\partial L}{\partial w_3} &= \frac{\partial L}{\partial f_3} \frac{\partial f_3}{\partial a_3} \frac{\partial a_3}{\partial w_3} \\
&= -\eta(y - \hat{y}) \frac{\partial f_3}{\partial a_3} \frac{\partial a_3}{\partial w_3} \\
&= -\eta(y - \hat{y}) f_3(1 - f_3) \frac{\partial a_3}{\partial w_3}
\end{aligned}
$$

Let's use a Sigmoid function

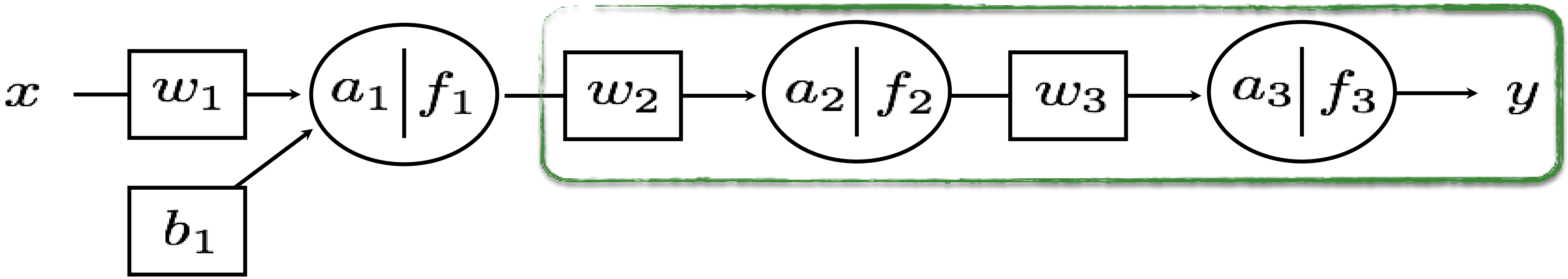
$$\frac{ds(x)}{dx} = s(x)(1 - s(x))$$

Backpropagation

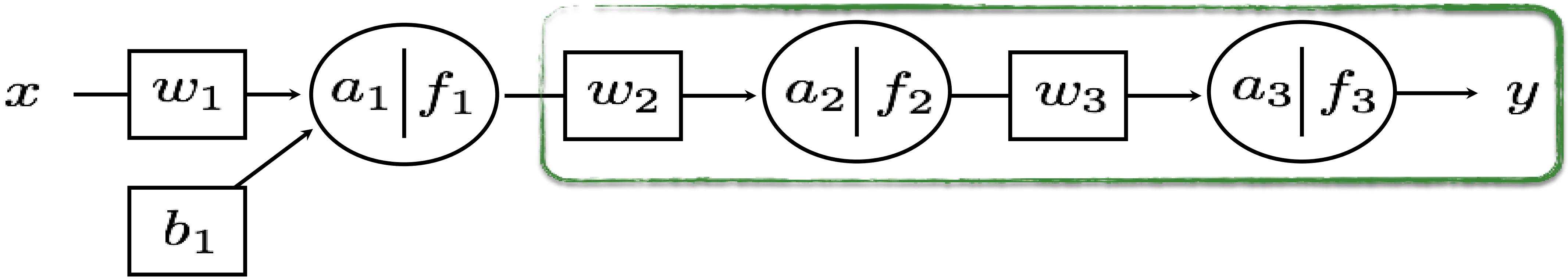
rest of the network $f_2 \longrightarrow \boxed{w_3} \longrightarrow \left(a_3 \mid f_3 \right) \longrightarrow \hat{y}$ $\qquad L(y, \hat{y})$

$$
\begin{aligned}
\frac{\partial L}{\partial w_3} &= \frac{\partial L}{\partial f_3} \frac{\partial f_3}{\partial a_3} \frac{\partial a_3}{\partial w_3} \\
&= -\eta(y - \hat{y}) \frac{\partial f_3}{\partial a_3} \frac{\partial a_3}{\partial w_3} \\
&= -\eta(y - \hat{y}) f_3(1 - f_3) \frac{\partial a_3}{\partial w_3} \\
&= -\eta(y - \hat{y}) f_3(1 - f_3) f_2
\end{aligned}
$$

Backpropagation



$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial f_3} \frac{\partial f_3}{\partial a_3} \frac{\partial a_3}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial w_2}$$



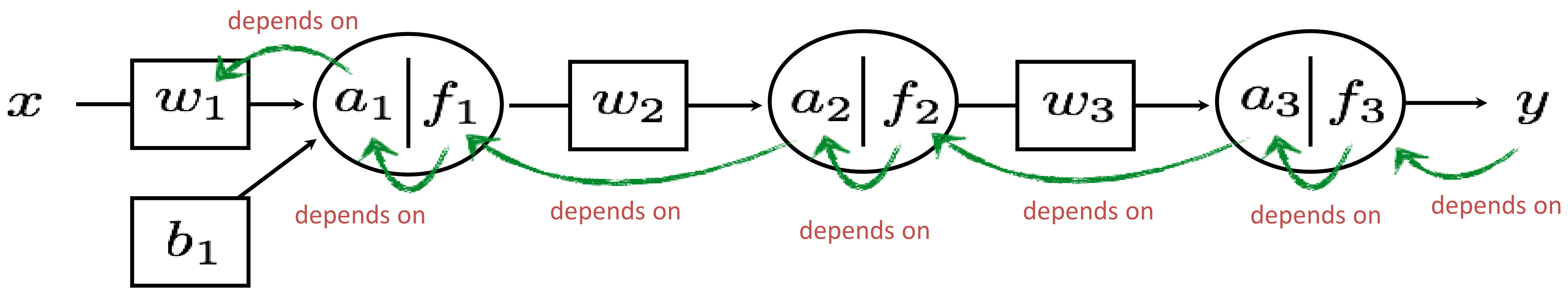
$$\frac{\partial L}{\partial w_2} = \boxed{\frac{\partial L}{\partial f_3} \frac{\partial f_3}{\partial a_3}} \frac{\partial a_3}{\partial f_2} \frac{\partial f_2}{\partial a_2} \frac{\partial a_2}{\partial w_2}$$

already computed.
re-use (propagate)!

The Chain Rule

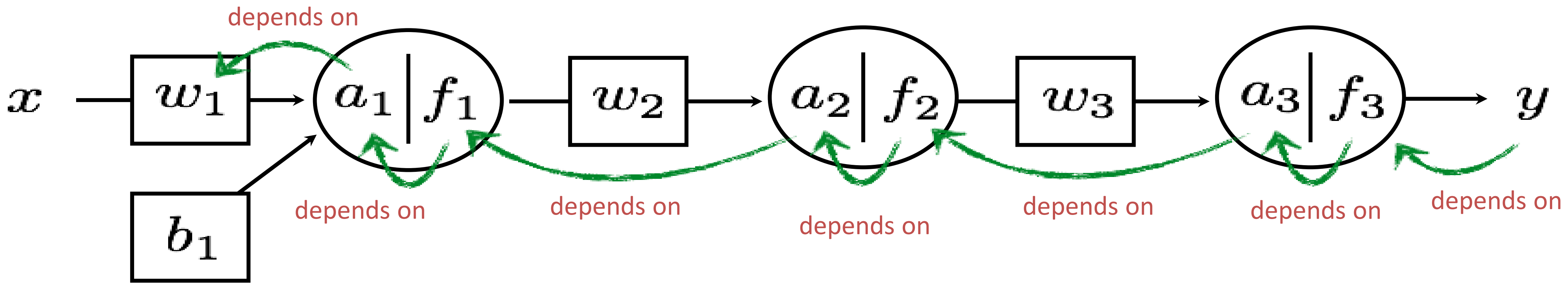
a.k.a. backpropagation

The chain rule says…



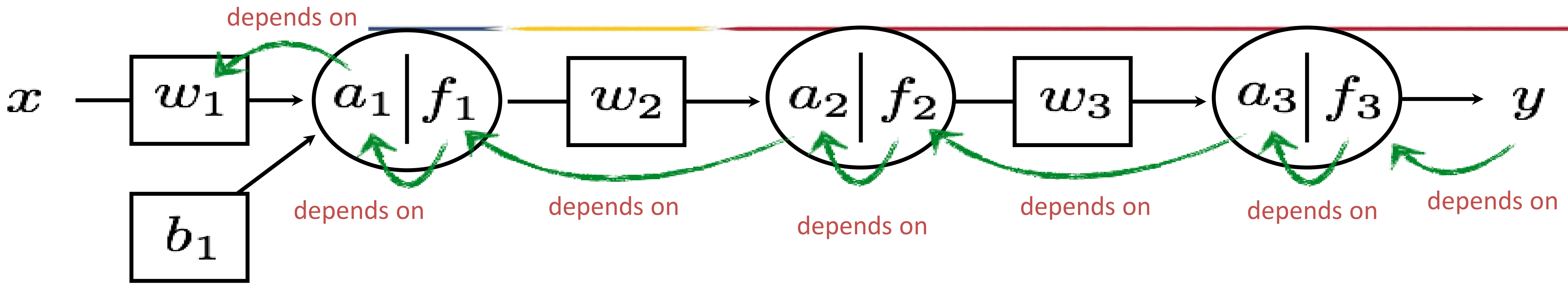
$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}\frac{\partial a_1}{\partial w_1}$$

The chain rule says...



$$\frac{\partial L}{\partial w_1} = \boxed{\frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}\frac{\partial a_1}{\partial w_1}$$

already computed.
re-use (propagate)!

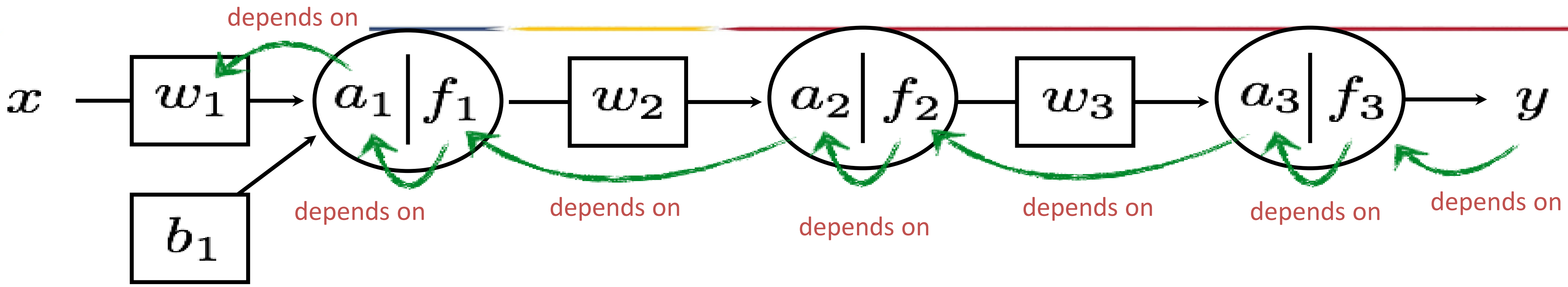


$$\frac{\partial \mathcal{L}}{\partial w_3} = \boxed{\frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}}\frac{\partial a_3}{\partial w_3}$$

$$\frac{\partial \mathcal{L}}{\partial w_2} = \boxed{\frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial w_2}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}\frac{\partial a_1}{\partial w_1}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}\frac{\partial a_1}{\partial b}$$

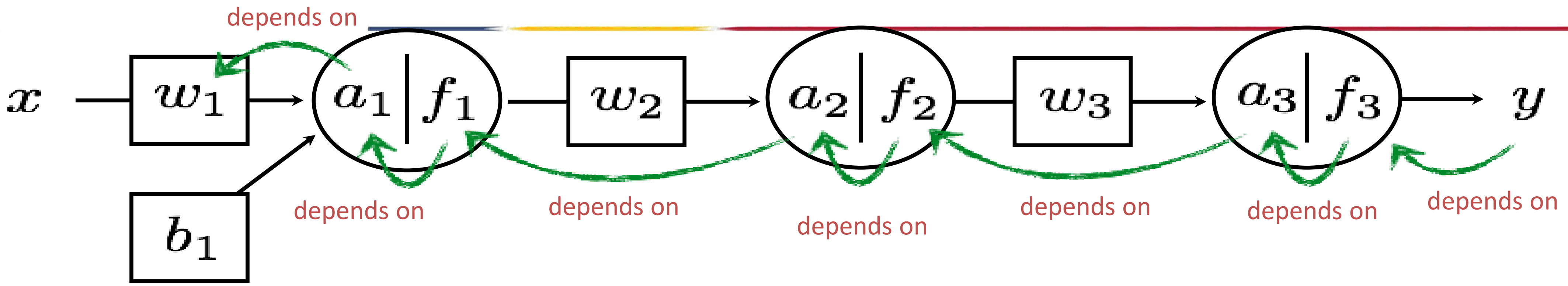


$$\frac{\partial \mathcal{L}}{\partial w_3} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial w_3}$$

$$\frac{\partial \mathcal{L}}{\partial w_2} = \boxed{\frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}}\frac{\partial a_2}{\partial w_2}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \boxed{\frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}\frac{\partial a_1}{\partial w_1}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}\frac{\partial a_1}{\partial b}$$



$$\frac{\partial \mathcal{L}}{\partial w_3} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial w_3}$$

$$\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial w_2}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \boxed{\frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}}\frac{\partial a_1}{\partial w_1}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \boxed{\frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}}\frac{\partial a_1}{\partial b}$$

Gradient Descent

For each example sample $\{x_i, y_i\}$

1. Predict
   a. Forward pass $\hat{y} = f_{\text{MLP}}(x_i; \theta)$
   b. Compute Loss $\mathcal{L}_i$
2. Update
   a. Back Propagation

$$\frac{\partial \mathcal{L}}{\partial w_3} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial w_3}$$
$$\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial w_2}$$
$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}\frac{\partial a_1}{\partial w_1}$$
$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}\frac{\partial a_1}{\partial b}$$

   b. Gradient update

$$w_3 = w_3 - \eta \nabla w_3$$
$$w_2 = w_2 - \eta \nabla w_2$$
$$w_1 = w_1 - \eta \nabla w_1$$
$$b = b - \eta \nabla b$$

Gradient Descent

For each example sample $\{x_i, y_i\}$

1. Predict

   a. Forward pass $\hat{y} = f_{\text{MLP}}(x_i; \theta)$

   b. Compute Loss $\mathcal{L}_i$

2. Update

   a. Back Propagation $\frac{\partial \mathcal{L}}{\partial \theta}$

   vector of parameter partial derivatives

   b. Gradient update $\theta \leftarrow \theta + \eta \frac{\partial \mathcal{L}}{\partial \theta}$

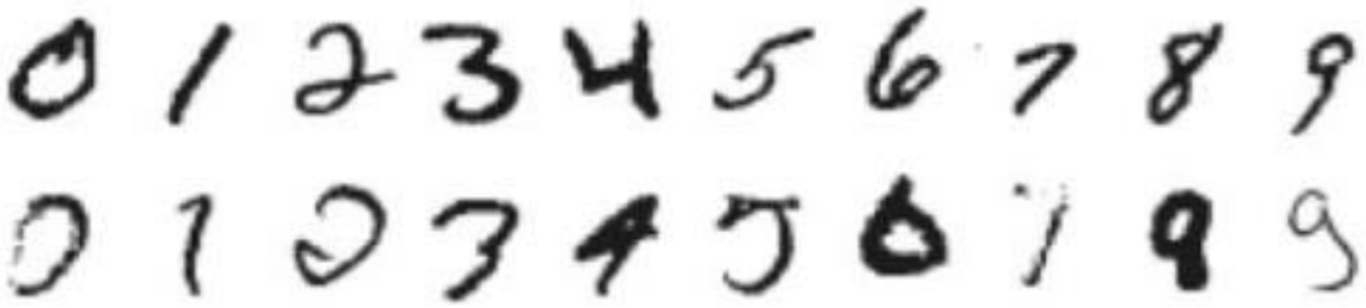
   vector of parameter update equations

SUMMARY

- Perceptron
- Neural networks
- Gradient descent
- Backpropagation

Experiments with the MNIST database

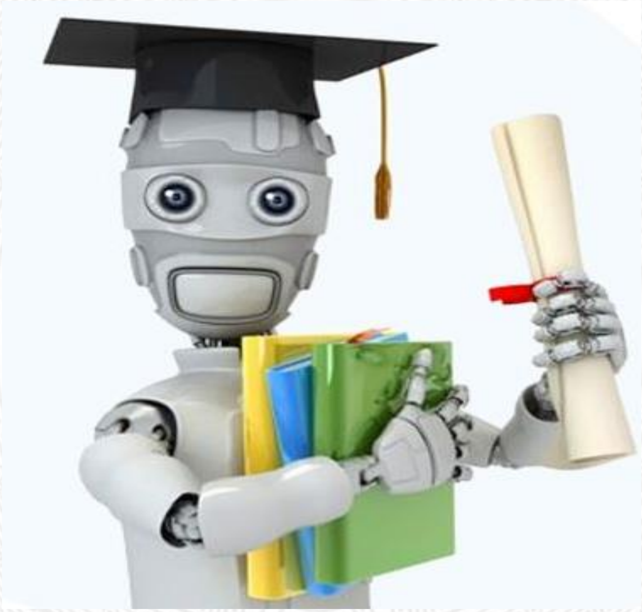
- The MNIST database of handwritten digits
- Training set of 60,000 examples, test set of 10,000 examples
- Vectors in R^{784} (28x28 images)
- Labels are the digits they represent
- Various methods have been tested with this training set and test set



- Linear models: 7% - 12% error
- KNN: 0.5%- 5% error
- Neural networks: 0.35% - 4.7% error
- Convolutional NN: 0.23% - 1.7% error

[Tinker With a Neural Network Right Here in Your Browser]

- Open source software to play with neural networks in your browser.
- The dots are colored orange or blue for positive and negative examples.
- It’s possible to choose the activation function, architecture, rate etc.
- Very well done! Let’s check it out!



Enjoy the Course…!