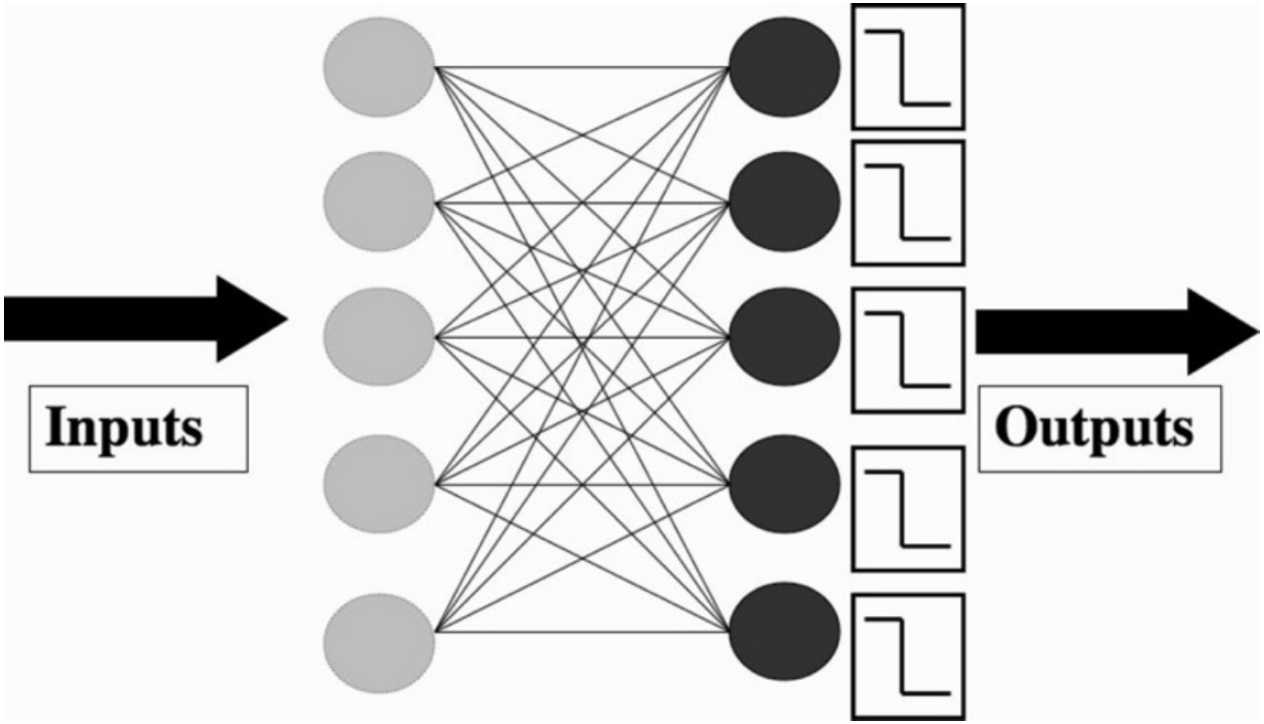
Machine Learning
Perceptron

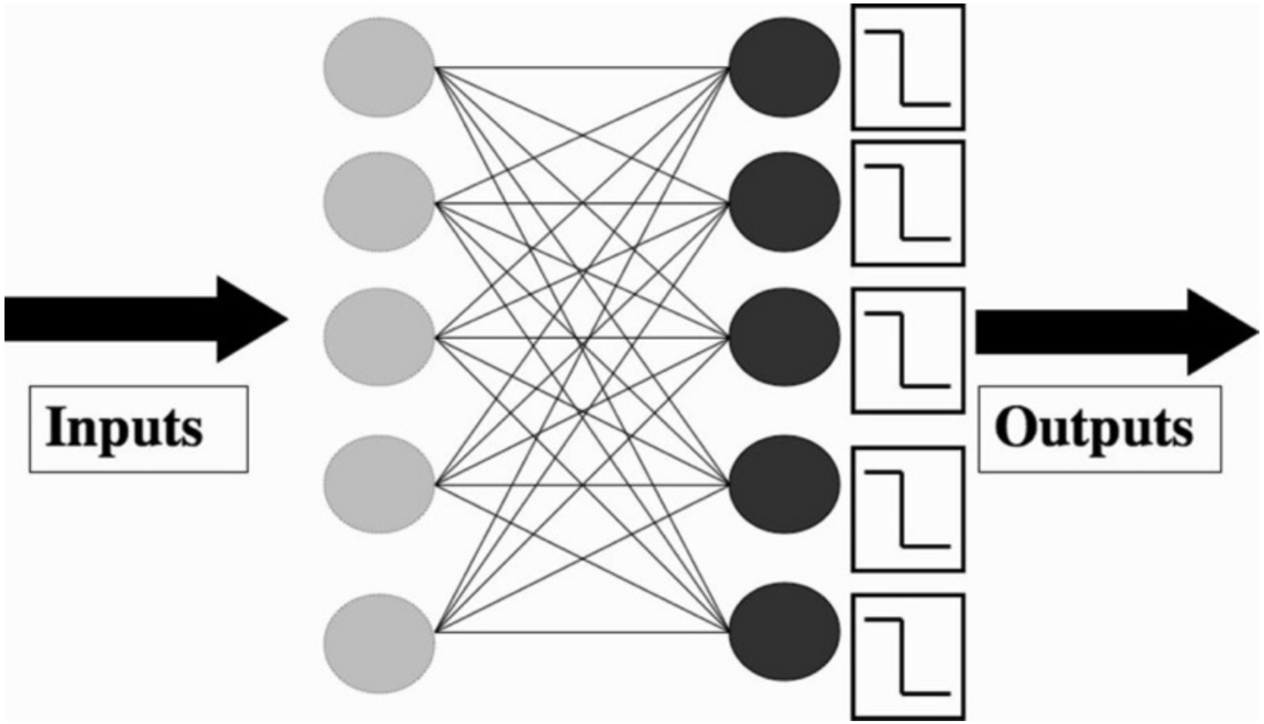
Jian Liu



Machine Learning
Perceptron

Jian Liu

Part 2: Error Functions



Training the perceptron or training neural network models

Learning happens through **optimisation**.

We define an error function, and then an optimisation algorithm finds the parameters that obtain the minimum error.

For example, the error function is the total number of mistakes:

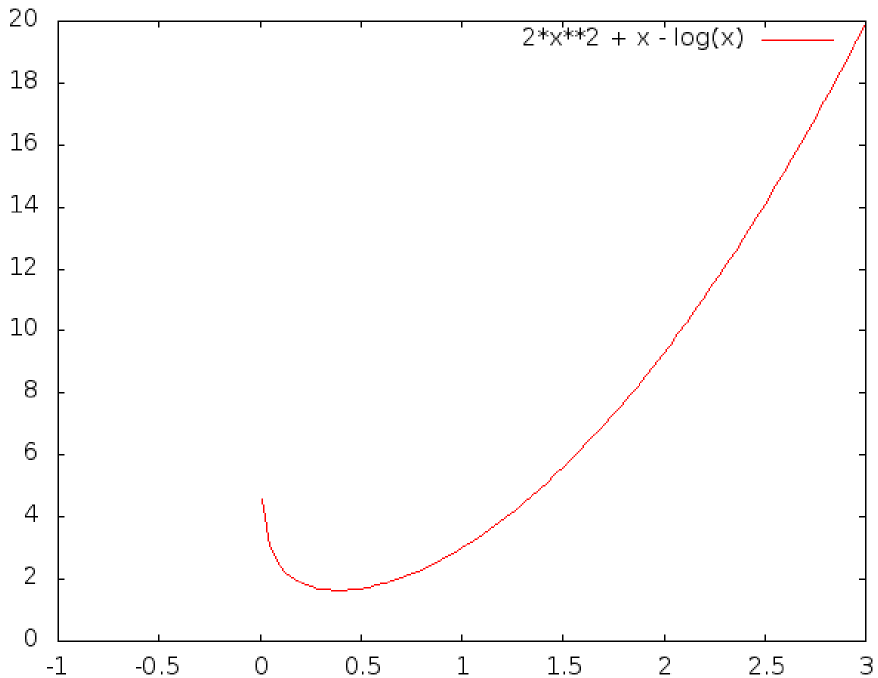
$$E(\boldsymbol{X}) = \sum_{\vec{x}_n \in \boldsymbol{X}} |y_n - t_n|$$

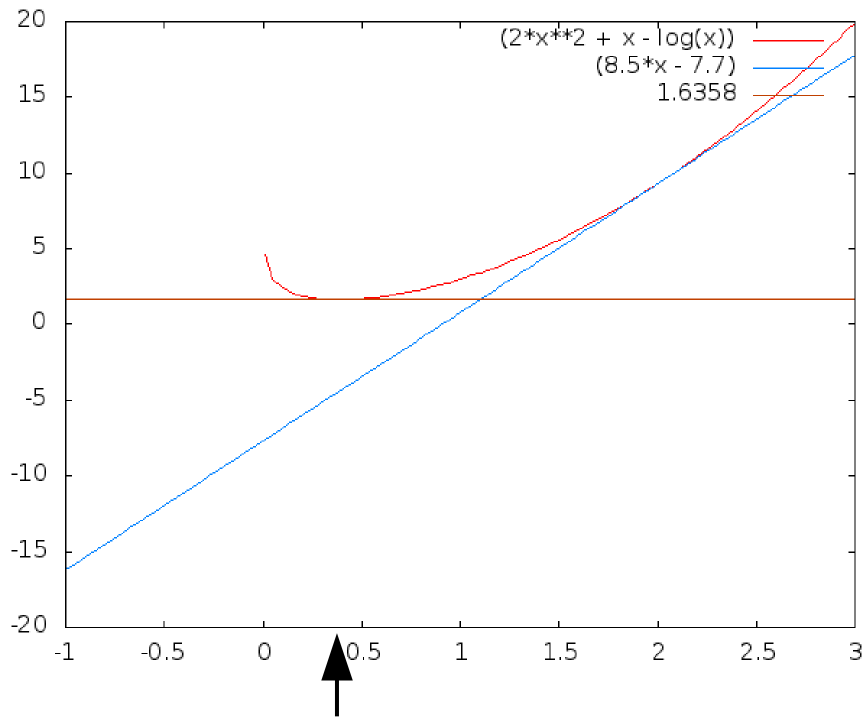
Where y_n is the output of the perceptron on point n, and $t_n \in \{0,1\}$ is the desired class, and $\boldsymbol{X}$ is the dataset.

Elements of Local Optimisation

Goal

Find the minimum point of a given function:





The minimum is at 0.39

Local Methods

Gradient descent

First order: gradient descent

$$x_{t+1} = x_t - \eta \nabla f(x_t)$$

step parameter

Second order: Newton's method

$$f(x_n + \Delta x) \approx f(x_n) + f'(x_n)\Delta x + \frac{1}{2} f''(x_n)\Delta x^2$$

Taylor's expansion

$$\frac{\partial}{\partial \Delta x} f(x_n + \Delta x) = f'(x_n) + f''(x_n)\Delta x = 0$$

Optimal step

$$\Delta x = \frac{-f'(x_n)}{f''(x_n)}$$

Many dimensions: $x_{t+1} = x_t - H^{-1}|_{x_n} \nabla f$

Question

The current point is <1,0>

compute the next point following gradient descent on the function $f(x,y) = x^3 + 2y^2 - y$ with step size 0.1.

Question

We want to compute: $x_{t+1} = \langle 1,0 \rangle - 0.1\, \nabla f(x_t)$

$\nabla f = \langle 3x^2, 4y - 1 \rangle$ Evaluated in <1,0> is <3,-1>

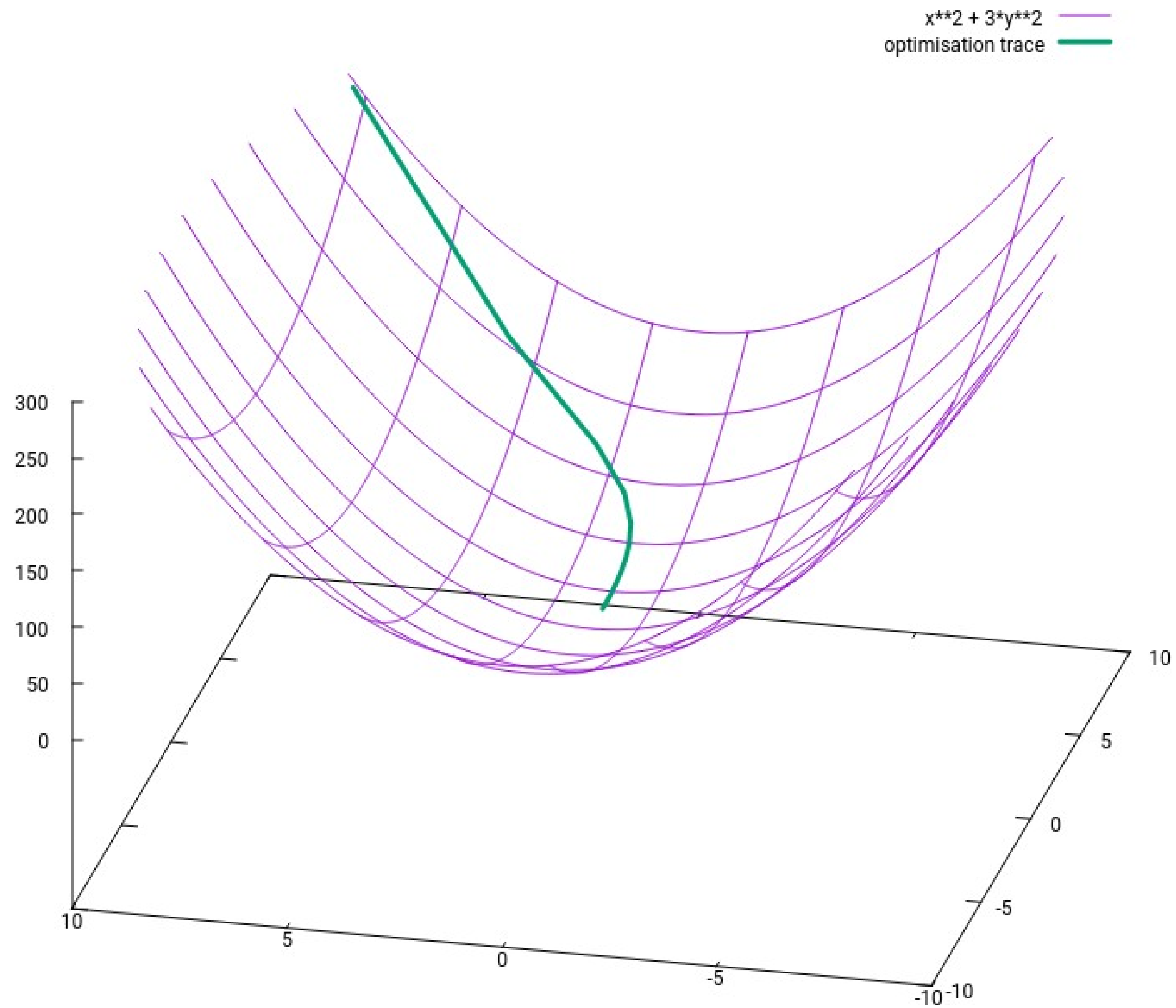
$$x_{t+1} = \langle 1,0 \rangle - 0.1 \cdot \langle 3,-1 \rangle = \langle 0.7, 0.1 \rangle$$

$f(1,0) = 1$

Our solution has improved!

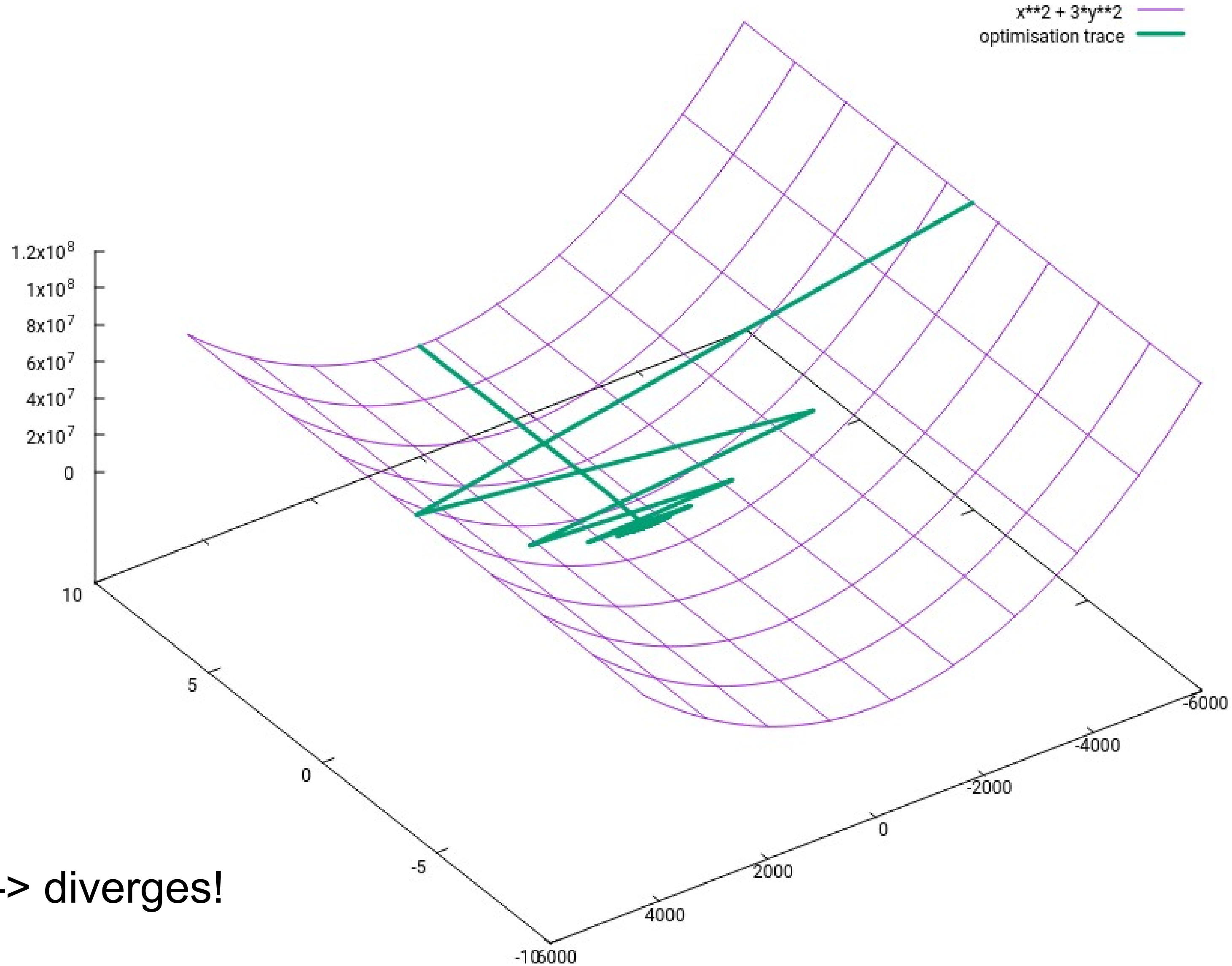
$f(0.7, 0.1) = 0.263$

In 3D



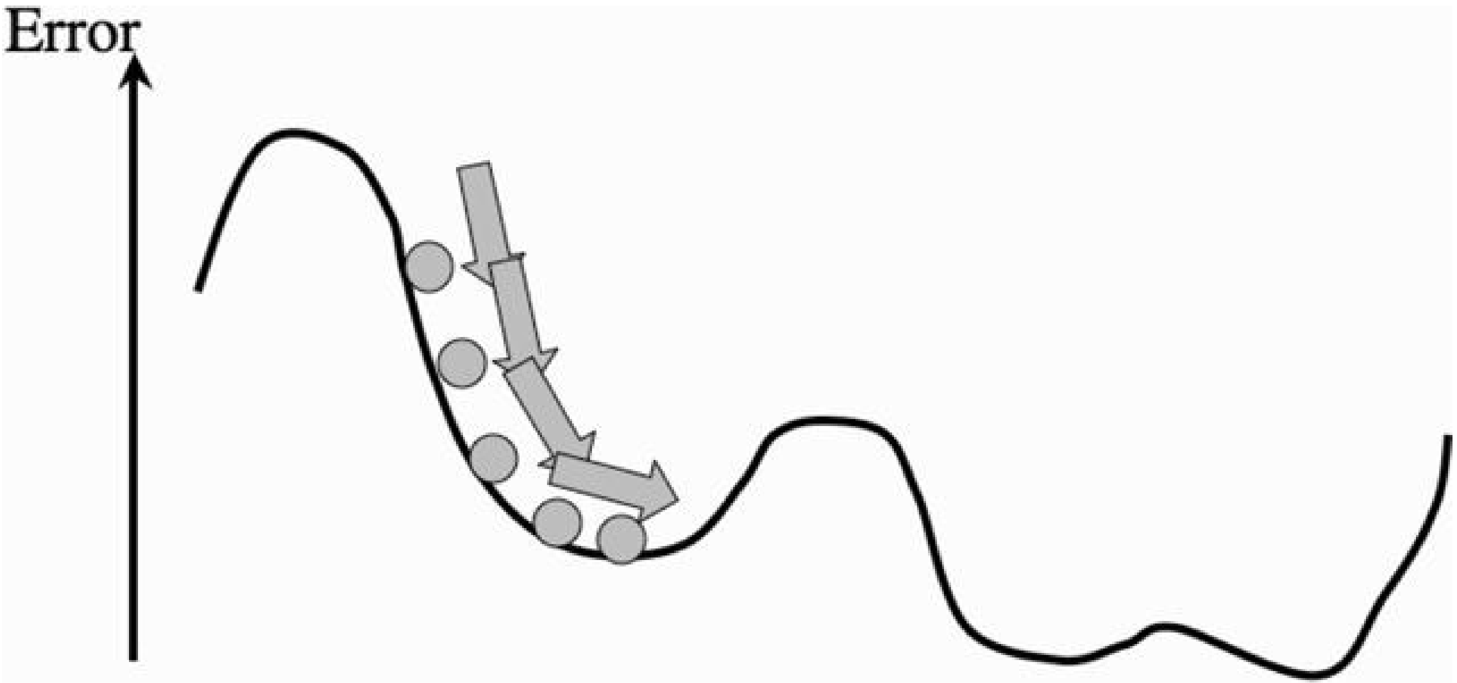
Step size: 0.1

In 3D



Step size: 0.14 –> diverges!

Local Minima

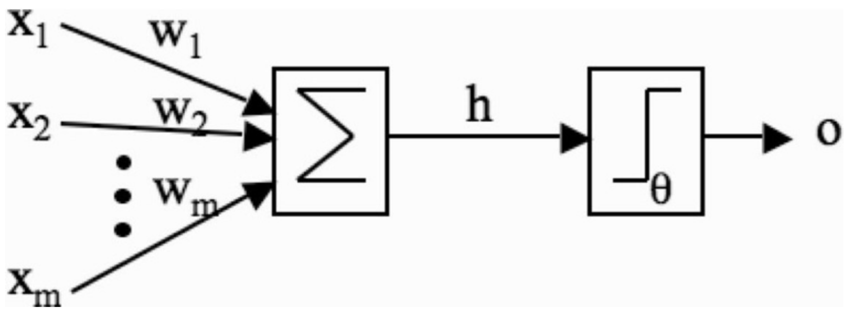


Training the perceptron

We want to apply gradient descent:

$$x_{t+1} = x_t - \eta \nabla f(x_t)$$

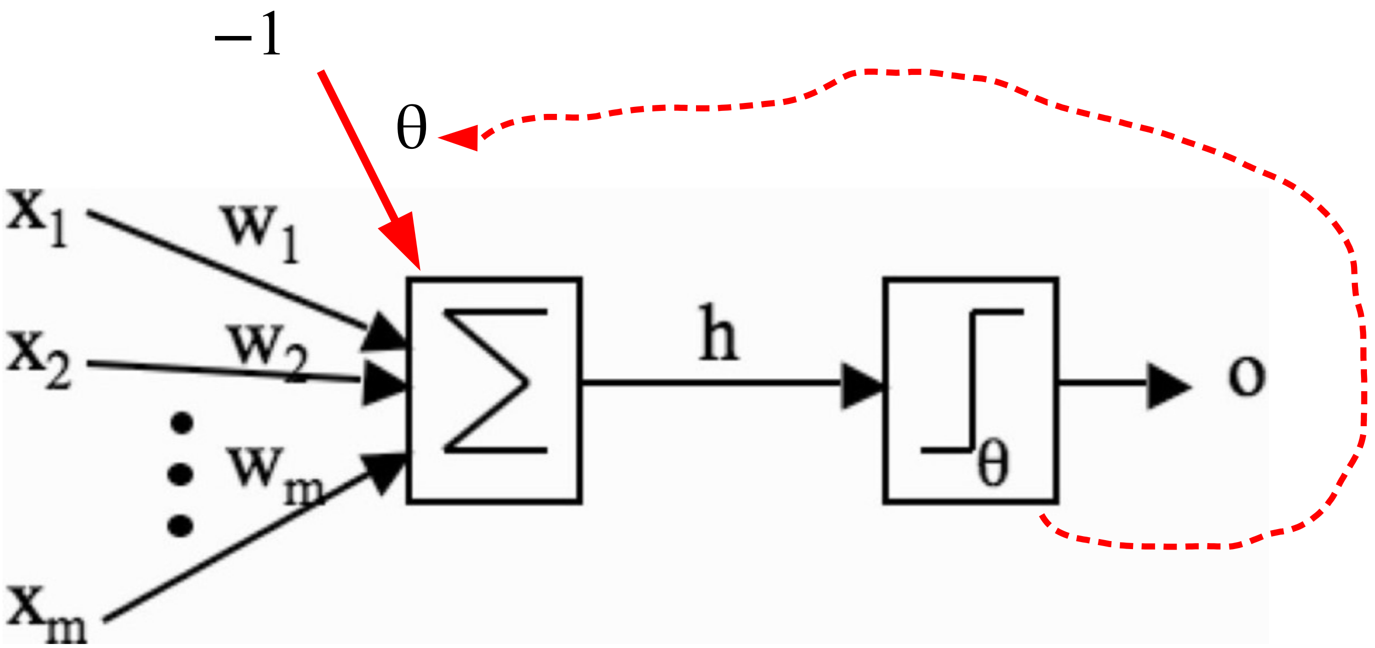
To the parameters of a perceptron:



So as to minimise an error (or loss) function, such as:

$$E(\boldsymbol{X}) = \sum_{\vec{x}_n \in \boldsymbol{X}} |y_n - t_n|$$

Bias input



$$h_w(\boldsymbol{x}) = \sum_i w_i x_i = \boldsymbol{w} \cdot \boldsymbol{x}$$

$$o(h_w) = \begin{cases} 1 & \text{if } h_w > \theta \\ 0 & \text{if } h_w \leq \theta \end{cases}$$

$$\longrightarrow$$

$$h_w(\boldsymbol{x}) = \sum_i w_i x_i - \theta$$

$$\boldsymbol{x}_{new} = \langle \boldsymbol{x}, -1 \rangle \quad \boldsymbol{w}_{new} = \langle \boldsymbol{w}, \theta \rangle$$

$$h_w(\boldsymbol{x}_{new}) = \boldsymbol{w}_{new} \cdot \boldsymbol{x}_{new}$$

$$w_1 x_1 + w_2 x_2 + \cdots + w_n x_n - 1 \cdot \theta > 0$$

$$o(h_w) = \begin{cases} 1 & \text{if } h_w > 0 \\ 0 & \text{if } h_w \leq 0 \end{cases}$$

Question

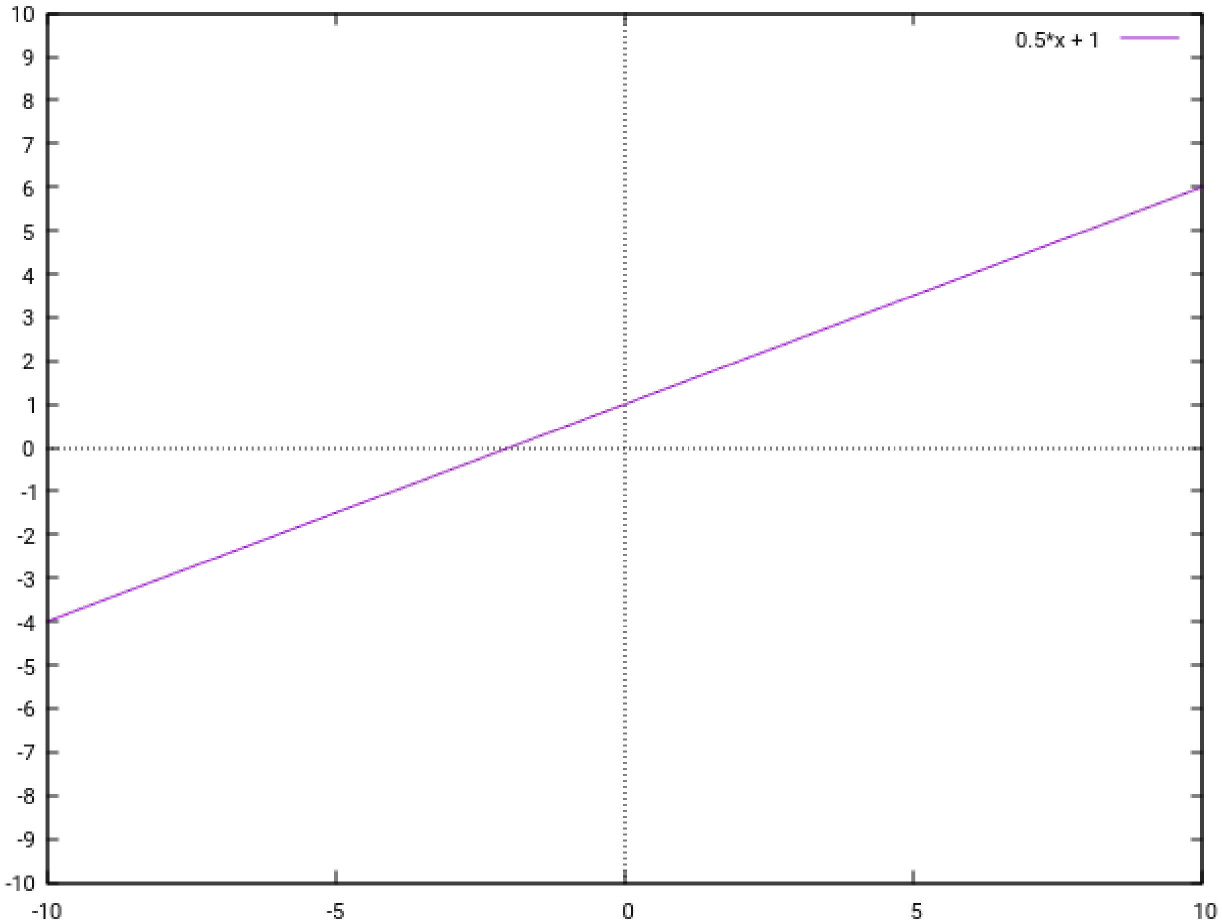
The decision boundary of the perceptron is the function below:

$$f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} = 0$$

Plot the following function: $\frac{1}{2}x - y + 1 = 0$

Solution

Plot the following function: $\frac{1}{2}x - y + 1 = 0$



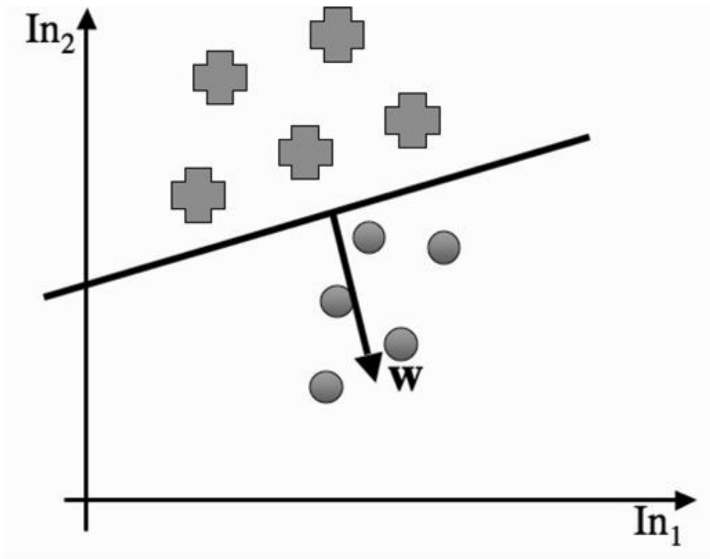
Linear separability

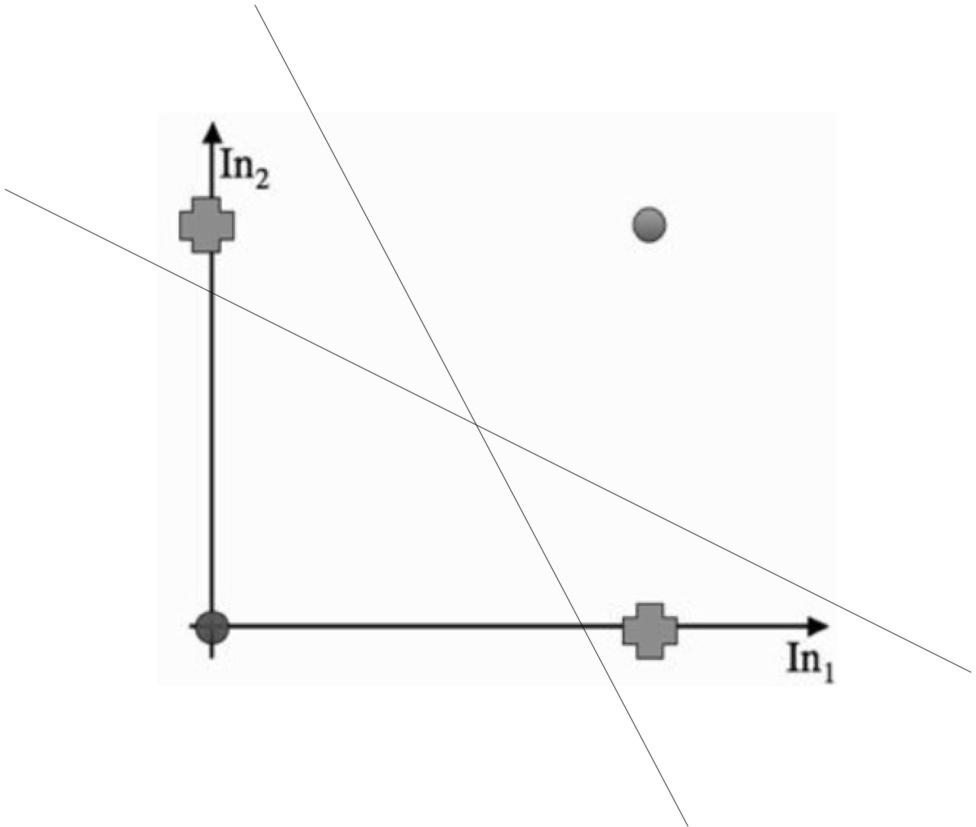
We have established that the decision boundary is a hyperplane.

$$h_w(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 = 0$$

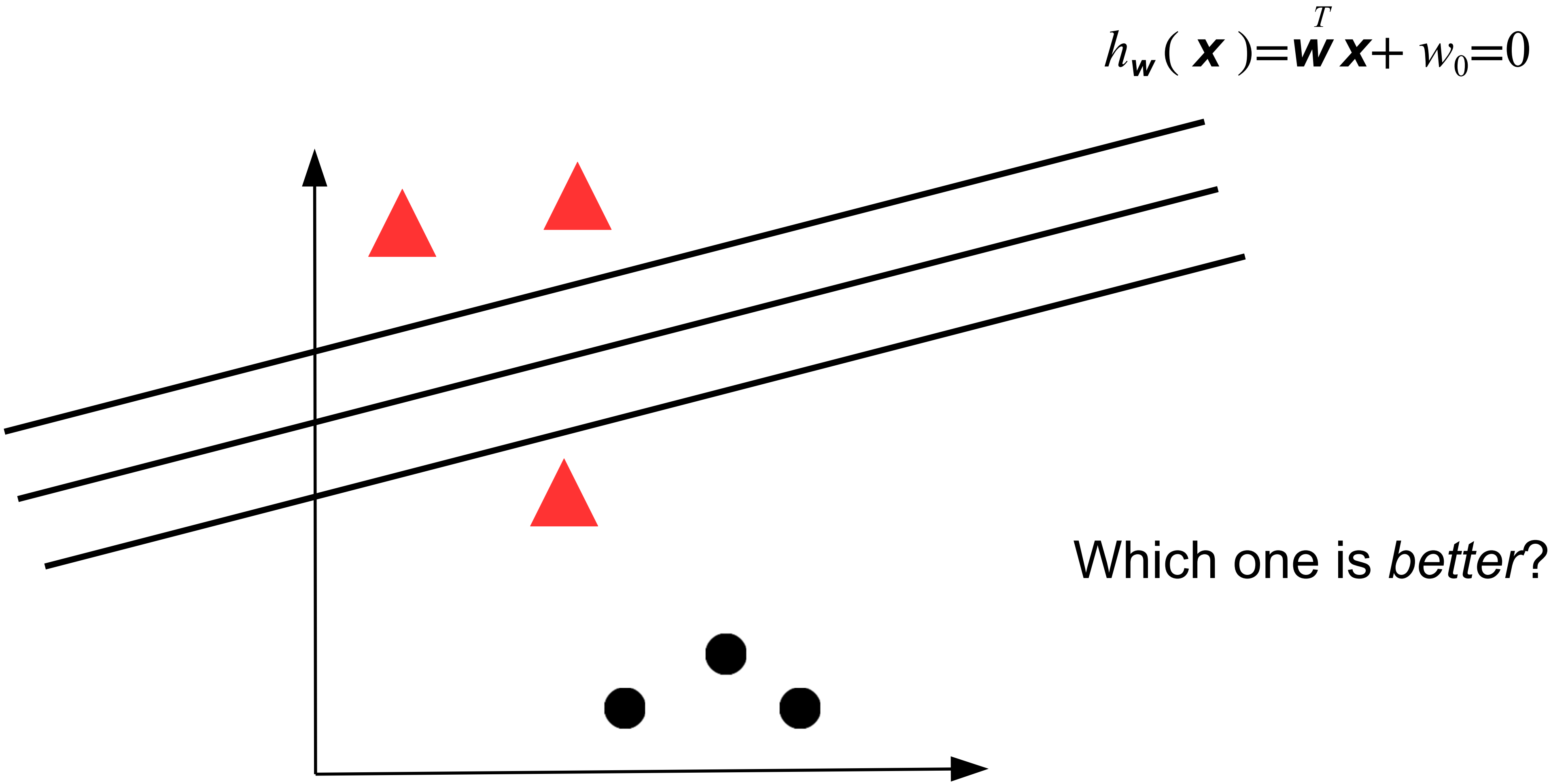
XOR

Not linearly separable!





Number of mistakes as Error

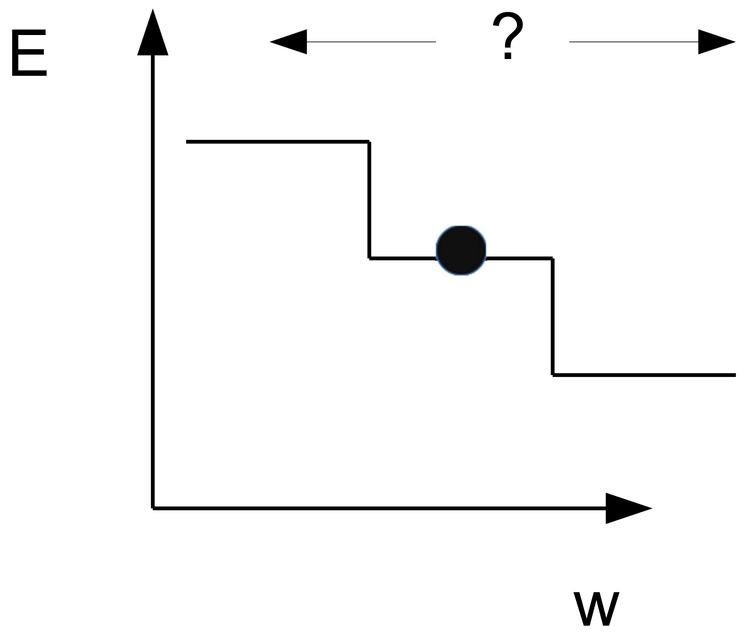


Which one is *better*?

Number of mistakes

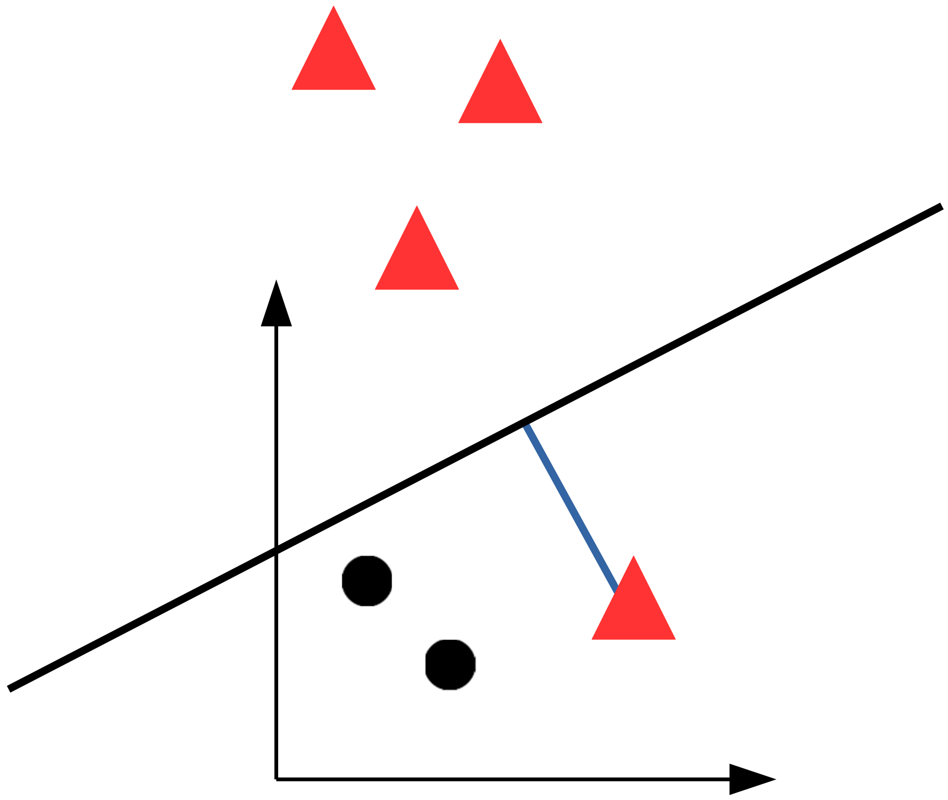
$$E(\boldsymbol{X}) = \sum_{\vec{x}_n \in \boldsymbol{X}} |y_n - t_n|$$

Number of mistakes on the dataset. Piecewise constant → no gradient.



There is no local information on the direction of improvement

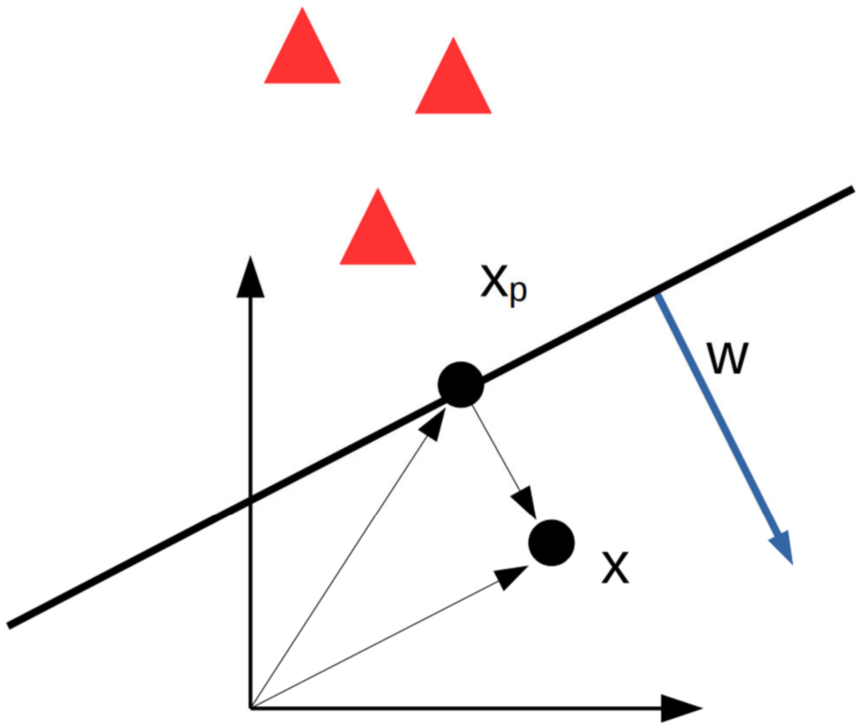
Towards a better error function



For each misclassified point, we would like to know not only that they are on the wrong side, but also **by how much**.

Towards a better error function

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 = 0$$



Distance to the hyperplane

$$\boldsymbol{x} = \boldsymbol{x}_p + d \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$$

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}\left(\boldsymbol{x}_p + d \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right) + w_0$$

$$= \cancel{\boldsymbol{w}\,\boldsymbol{x}_p + w_0} + d \frac{\boldsymbol{w}^T \boldsymbol{w}}{\|\boldsymbol{w}\|} = d\|\boldsymbol{w}\|$$

Recall that:

$$\boldsymbol{w}^T \boldsymbol{w} = w_1^2 + w_2^2 + \cdots + w_n^2 = \|\boldsymbol{w}\|^2$$

Towards a better error function

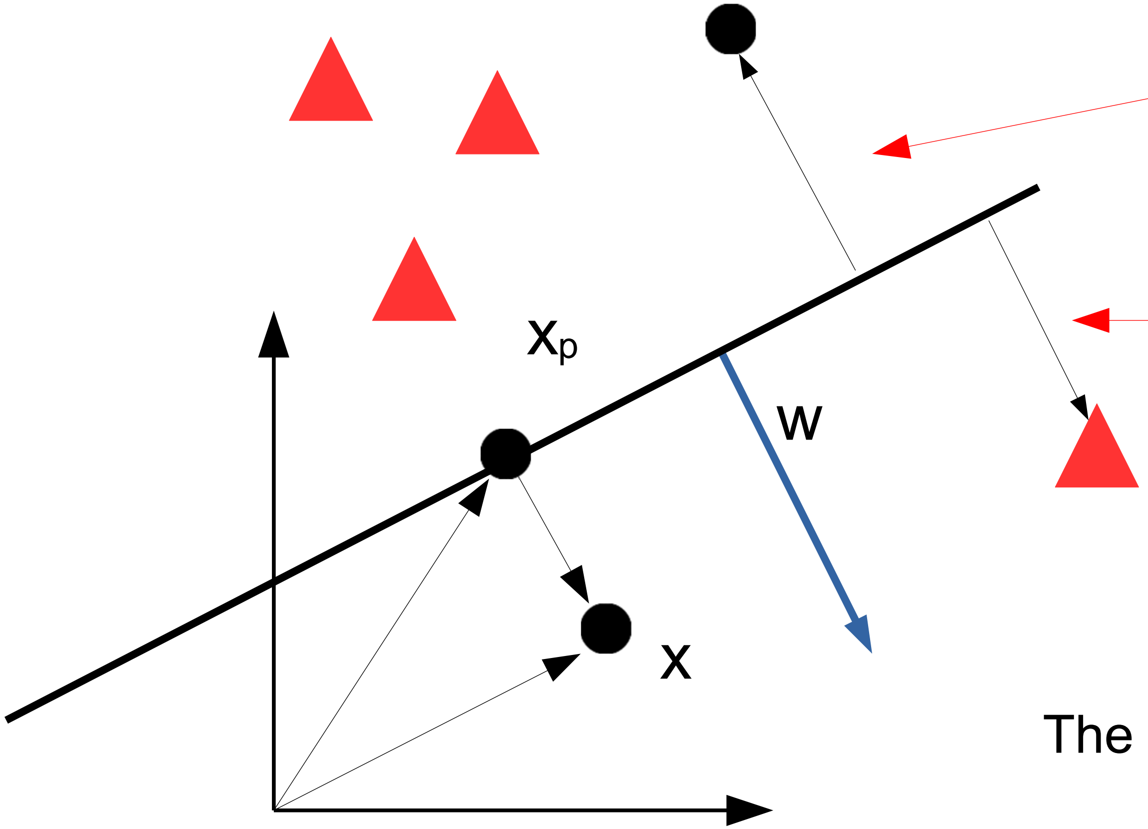
$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 = d\,\|\boldsymbol{w}\|$$

$$E(\boldsymbol{X}) = \sum_{\boldsymbol{x}_n \in \boldsymbol{X}} (\boldsymbol{w}^T \boldsymbol{x}_n + w_0)$$

Is this a good error?

Towards a better error function

$$\boldsymbol{x} = \boldsymbol{x}_p + d \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$$



This d is negative

This d is positive

The two errors cancel each other out!

The perceptron criterion

$h_w(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 = 0$ apply the bias input

if $\boldsymbol{w}^T \boldsymbol{x} > 0$ then $y = 1$ In case of mistake: $t = 0$ $(y - t) = 1$

if $\boldsymbol{w}^T \boldsymbol{x} \leq 0$ then $y = 0$ In case of mistake: $t = 1$ $(y - t) = -1$

Therefore, if mistake: $\boldsymbol{w}^T \boldsymbol{x}(y - t) > 0$

$$E(\boldsymbol{X}) = \sum_{\boldsymbol{x}_n \in \boldsymbol{X}} |y_n - t_n|$$

$$E_p(\boldsymbol{X}) = \sum_{\boldsymbol{x}_n \in \boldsymbol{X}} \boldsymbol{w}^T \boldsymbol{x}_n (y_n - t_n)$$

Number of mistakes on the dataset. Piecewise constant → gradient useless.

Proportional to distance of misclassified points from surface. → gradient ok.

Question

Given the perceptron error (below), what is the gradient with respect to **w**?

$$E_p(\boldsymbol{X}) = \mathbf{w}^T \mathbf{x}(y-t) = (w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n)(y-t)$$

Solution

$$E_p(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}(y-t) = w_0 x_0 (y-t) + w_1 x_1 (y-t) + \cdots + w_m x_m (y-t)$$

$$\nabla E_p(x) = \begin{bmatrix} \frac{\partial}{\partial w_0} E_p(x) \\ \frac{\partial}{\partial w_1} E_p(x) \\ \frac{\partial}{\partial w_2} E_p(x) \\ \dots \\ \frac{\partial}{\partial w_n} E_p(x) \end{bmatrix} = \begin{bmatrix} x_0(y-t) \\ x_1(y-t) \\ x_2(y-t) \\ \dots \\ x_n(y-t) \end{bmatrix}$$

Gradient descent

$$\nabla E_p(\boldsymbol{X}) = \sum_{\boldsymbol{x}_n \in \boldsymbol{X}} \boldsymbol{x}_n (y_n - t_n)$$

Recall that gradient descent does the following update:

$$w_{k+1} = w_k - \eta \nabla f(w_k)$$

Which leads us to the update rule for the perceptron:

$$w_{k+1} = w_k - \eta \sum_{\boldsymbol{x}_n \in \boldsymbol{X}} \boldsymbol{x}_n (y_n - t_n)$$

Stochastic gradient descent

$$E_p(\boldsymbol{X}) = \frac{1}{N} \sum_{\boldsymbol{x}_n \in \boldsymbol{X}} \boldsymbol{w}^T \boldsymbol{x}_n (y_n - t_n) = \boldsymbol{E}[\boldsymbol{w}^T \boldsymbol{x}_n (y_n - t_n)]$$

Gradient:

$$\boldsymbol{w}_{\boldsymbol{k+1}} = \boldsymbol{w}_{\boldsymbol{k}} - \eta \frac{1}{N} \sum_{\boldsymbol{x}_n \in \boldsymbol{X}} \boldsymbol{x}_n (y_n - t_n)$$

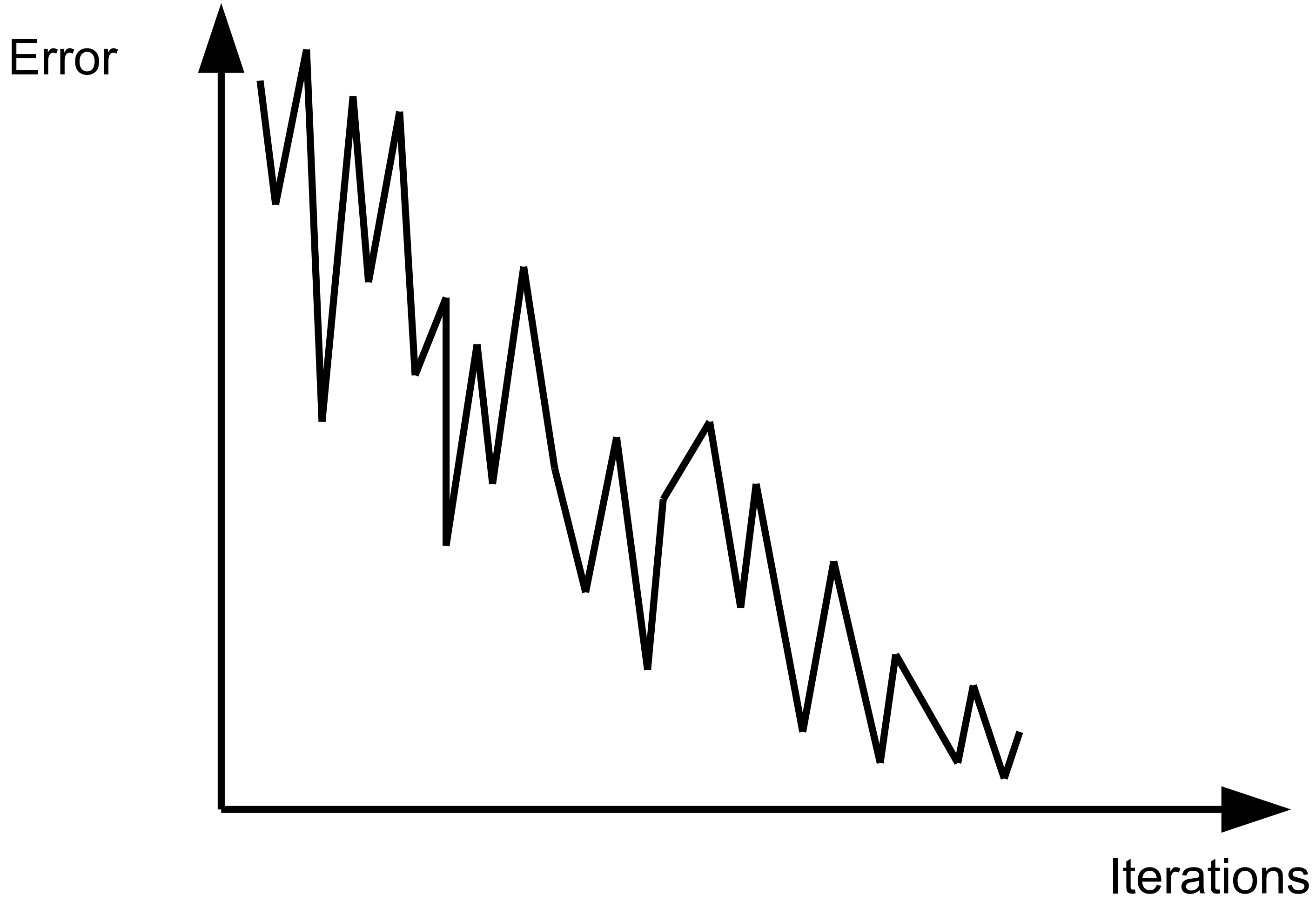
Stochastic gradient descent:

$$\boldsymbol{w}_{\boldsymbol{k+1}} = \boldsymbol{w}_{\boldsymbol{k}} - \eta \boldsymbol{x} (y - t)$$

GD: you have to run through ALL the samples in your training set to do a single update for a parameter in a particular iteration

SGD: you use ONLY ONE or SUBSET of training sample from your training set to do the update for a parameter in a particular iteration. If you use SUBSET, it is called Minibatch Stochastic gradient Descent.

Stochastic gradient descent



Summary

- Define an appropriate error function for the perceptron.
- Derive the corresponding update algorithm.
- Describe the difference between gradient descent and stochastic gradient descent.