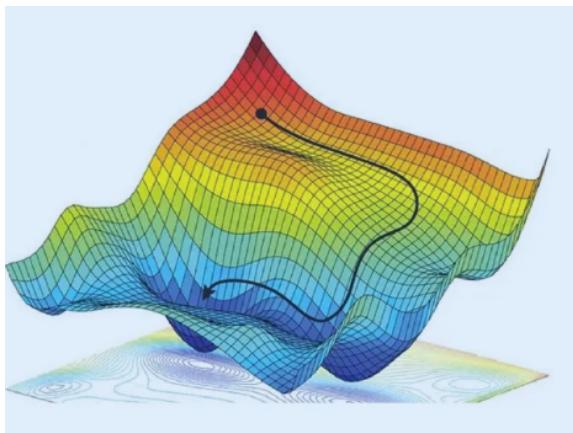
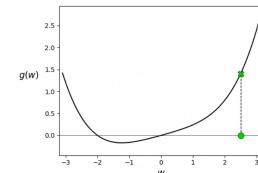


# W2.2 continued: Gradient Descent

[Link back to classification and logistic reasoning notes](#)



Demo Example for Gradient Descent



UNIVERSITY OF  
BIRMINGHAM  
[https://github.com/jermwatt/machine\\_learning\\_refined/tree/main/pages/notes/3\\_First\\_order\\_methods/videos/animation\\_6.mp4](https://github.com/jermwatt/machine_learning_refined/tree/main/pages/notes/3_First_order_methods/videos/animation_6.mp4)

## Gradient Descent

- A general strategy to minimize cost functions in ML algorithms.
- Goal: minimize the cost function  $g(w_0, w_1)$

```

Start at a random point say  $w_0 = 0, w_1 = 0$ 
Repeat until no change occurs
    Update  $w_0, w_1$  by taking
        a small step in the direction of the steepest descent of cost
Return  $w_0, w_1$ .
```

UNIVERSITY OF  
BIRMINGHAM

## Gradient Descent – More General...

- Goal: minimize the cost function  $g(\mathbf{w})$ , where  $\mathbf{w} = (w_0, w_1, \dots)$

```

Input:  $\alpha > 0$ 
Initialise  $\mathbf{w}$ . //at 0 or some random value
Repeat until convergence
     $\mathbf{w} := \mathbf{w} - \alpha \nabla g(\mathbf{w})$ 
Return  $\mathbf{w}$ .
```

UNIVERSITY OF  
BIRMINGHAM

Learning rate  
or step size,  
e.g. 0.01

Gradient or steepest  
direction

Looking at the first image instead of going from the starting point directly to the end arrow instantly you can go little by little down the more curved, complex path.

### Back to Univariate Linear Regression

Back to two dimensional function  $g(w_0, w_1)$ :

- The vector of partial derivatives is called the **gradient vector**.

$$\nabla g(\mathbf{w}) = \begin{pmatrix} \frac{\partial g}{\partial w_0} \\ \frac{\partial g}{\partial w_1} \end{pmatrix}, \text{ where } \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}$$

- Recall: Partial derivative with respect to one variable is the ordinary derivative of the function by treating the others as constants.
- The negative of the gradient evaluated at a location  $(\hat{w}_0, \hat{w}_1)$  gives us the direction of the steepest descent from that location.
- We take a small step in that direction – using the learning rate  $\alpha$ .



### Applying GD to solve univariate linear regression

- Recall: we aim to minimizing the cost function

$$g(w_0, w_1) = \frac{1}{N} \sum_{n=1}^N ((w_1 x^{(n)} + w_0) - y^{(n)})^2$$

- Using the chain rule, we have \*:

$$\frac{\partial g}{\partial w_0} = \frac{2}{N} \sum_{n=1}^N ((w_1 x^{(n)} + w_0) - y^{(n)})$$

$$\frac{\partial g}{\partial w_1} = \frac{2}{N} \sum_{n=1}^N ((w_1 x^{(n)} + w_0) - y^{(n)}) x^{(n)}$$

UNIVERSITY OF BIRMINGHAM \*For a very detailed explanations of all steps watch: <https://www.youtube.com/watch?v=sDv4I4zSB8>

## Algorithm for univariate linear regression using GD

```

Input:  $\alpha > 0$ , training set  $\{(x^{(n)}, y^{(n)}): n = 1, 2 \dots N\}$ 
Initialise  $w_0 = 0, w_1 = 0$ 
Repeat
    for  $n = 1, 2 \dots N$  //more efficient to update after each data point
         $w_0 := w_0 - \alpha((w_1 x^{(n)} + w_0) - y^{(n)})$ 
         $w_1 := w_1 - \alpha((w_1 x^{(n)} + w_0) - y^{(n)}) x^{(n)}$ 
Until change in cost remains below a very small threshold
Return  $w_0, w_1$ .

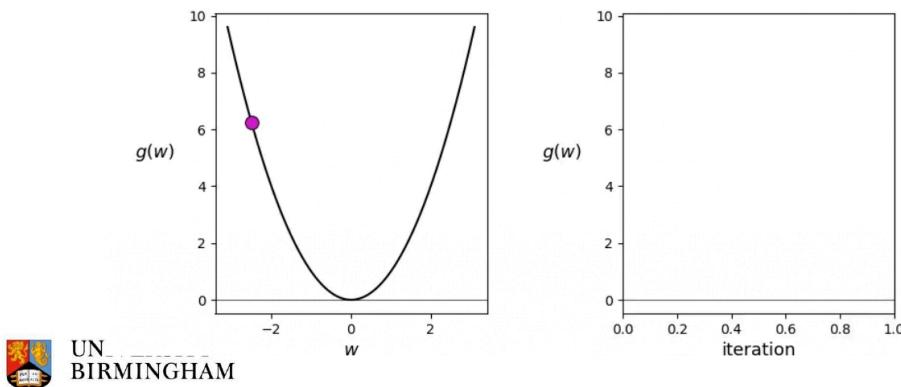
```



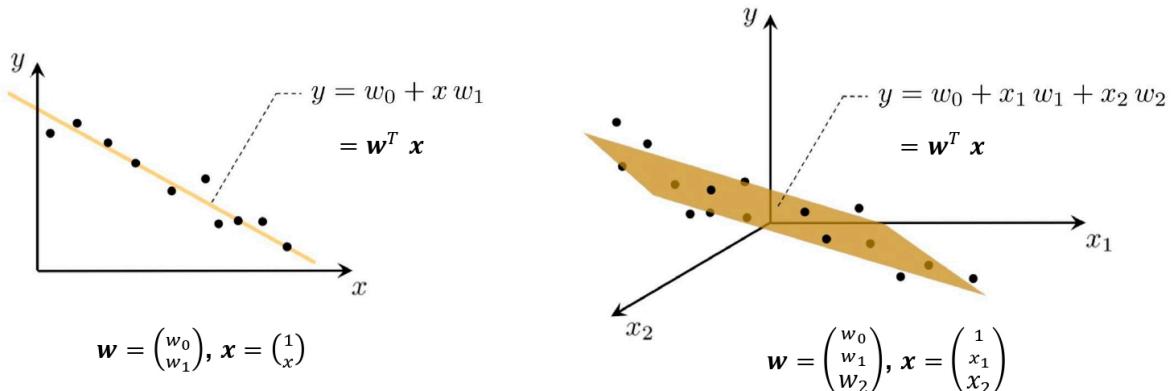
UNIVERSITY OF  
BIRMINGHAM

## Effect of the learning rate

Whether or not we descend in the function when taking this step depends completely on how far along it we travel.



## Multivariate linear regression



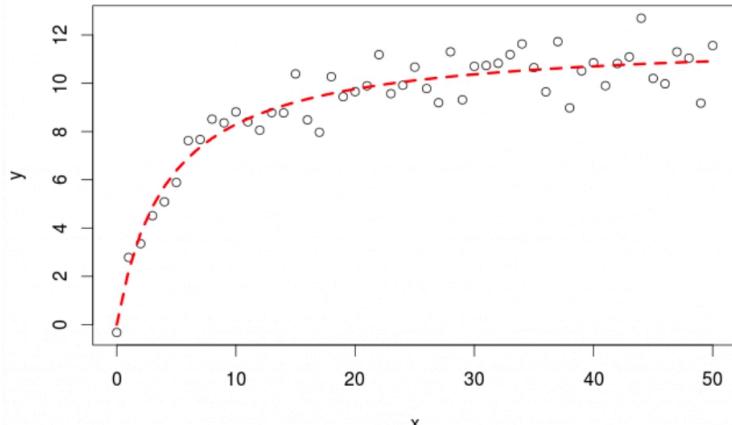
Left shows univariate linear regression and right shows multivariate linear regression in order to highlight the difference

The  $\mathbf{w}^T$  is needed to allow multiplication of the vectors.

Without it you'd have a mismatch of dimensions

## Univariate nonlinear regression

$$y = w_0 + w_1 x$$



$$\mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_m \end{pmatrix}, \mathbf{x} = \begin{pmatrix} 1 \\ x \\ x^2 \\ \dots \\ x^m \end{pmatrix}$$

This is an m-th order polynomial regression model.

## Advantages of vector notation

- Vector notation is more concise.
- With the vectors  $\mathbf{w}$  and  $\mathbf{x}$  populated appropriately (and differently in each case, as on the previous 2 slides), these models are still linear in the parameter vector.
- The cost function is the L2 as before.
- The gradient remains:

$$\nabla g(\mathbf{w}) = 2(\mathbf{w}^T \mathbf{x}^{(n)} - y^{(n)}) \mathbf{x}^{(n)}$$

- Ready to be plugged into the general gradient descent algorithm.



UNIVERSITY OF  
BIRMINGHAM