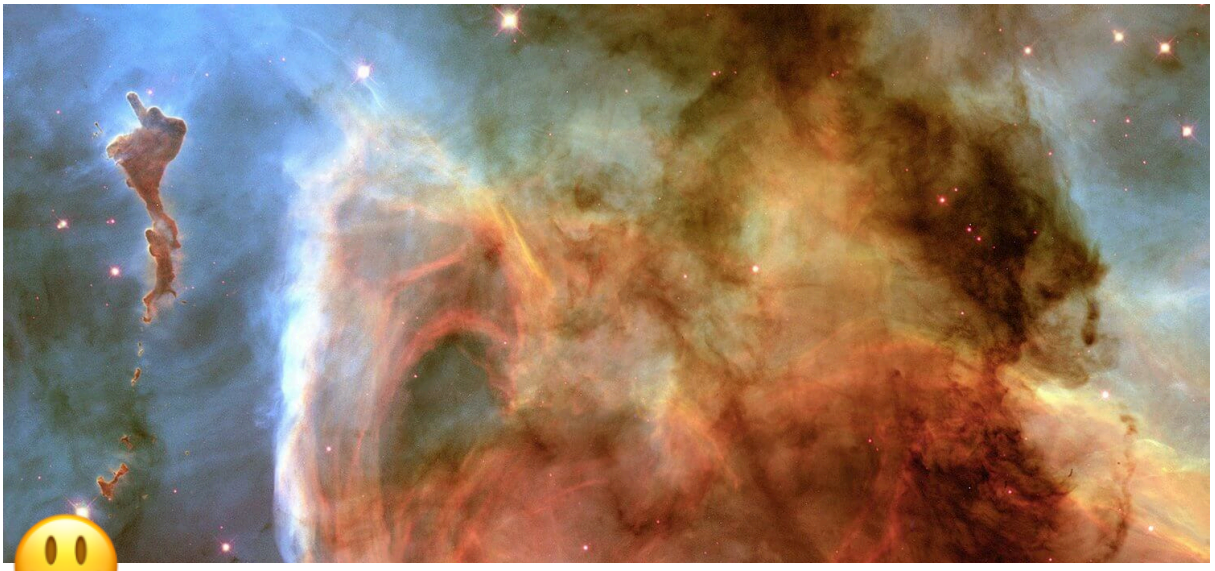
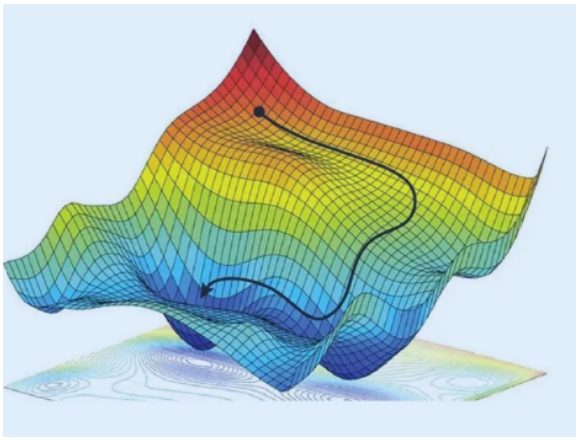
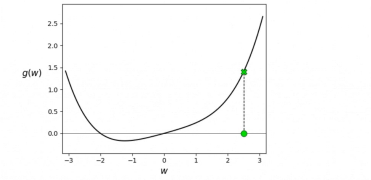


W2.2 continued: Gradient Descent

Link back to classification and logistic reasoning notes



Demo Example for Gradient Descent



UNIVERSITY OF BIRMINGHAM

https://github.com/jermwatt/machine_learning_refined/tree/gh-pages/notes/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 $g(\boldsymbol{w}), where\ \boldsymbol{w} = (w_0, w_1, \ldots)$

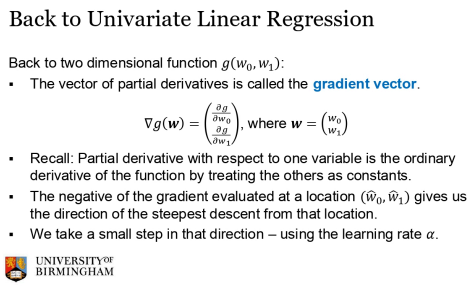
Input: $\alpha > 0$
Initialise $\boldsymbol{w}$. //at 0 or some random value
Repeat until convergence
$\boldsymbol{w} := \boldsymbol{w} - \alpha \nabla g(\boldsymbol{w})$
Return $\boldsymbol{w}$.

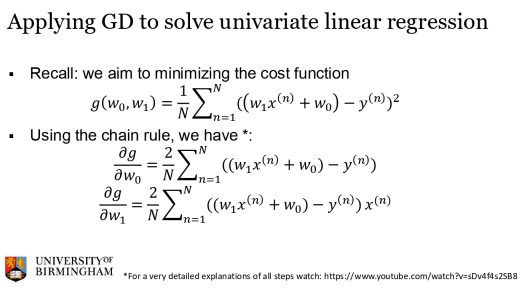
Learning rate or step size, e.g. 0.01

Gradient or steepest direction

UNIVERSITY OF BIRMINGHAM

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.





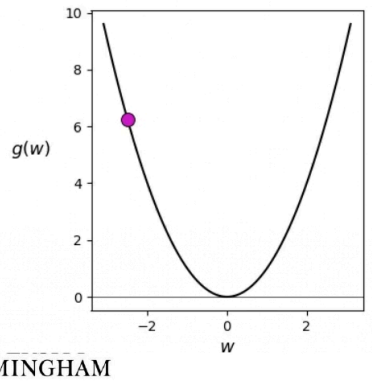
Algorithm for univariate linear regression using GD

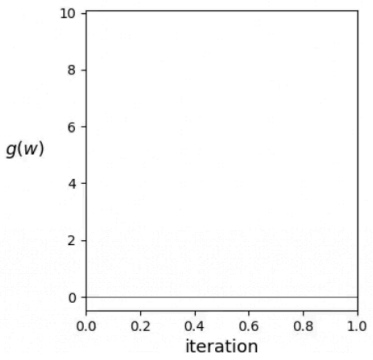
Input: $\alpha > 0$, training set $\{(x^{(n)}, y^{(n)}): n = 1,2 \ldots N\}$
Initialise $w_0 = 0, w_1 = 0$
Repeat
    for n = 1, 2… N //more efficient to update after each data point
        $w_0 := w_0 - \alpha((w_1 x^{(n)} + w_0) - y^{(n)})$
        $w_1 := w_1 - \alpha((w_1 x^{(n)} + w_0) - y^{(n)})x^{(n)}$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

UNIVERSITY OF BIRMINGHAM

Effect of the learning rate

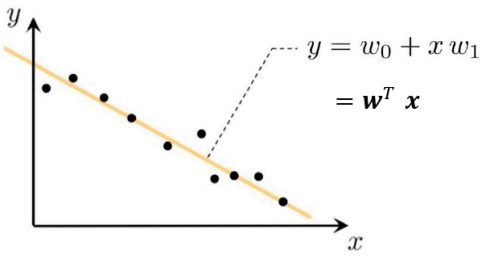
Whether or not we descend in the function when taking this step depends completely on how far along it we travel.



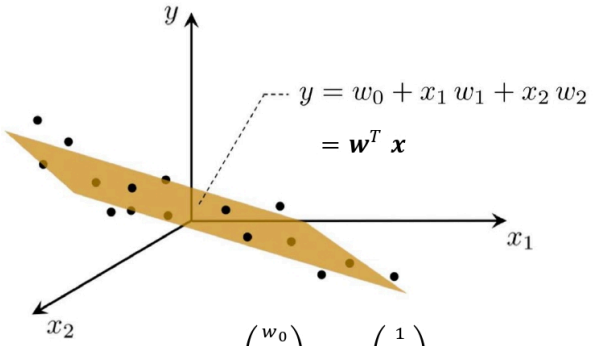


UNIVERSITY OF BIRMINGHAM

Multivariate linear regression



$\boldsymbol{w} = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}, \boldsymbol{x} = \begin{pmatrix} 1 \\ x \end{pmatrix}$



$\boldsymbol{w} = \begin{pmatrix} w_0 \\ w_1 \\ W_2 \end{pmatrix}, \boldsymbol{x} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$

UNIVERSITY OF BIRMINGHAM

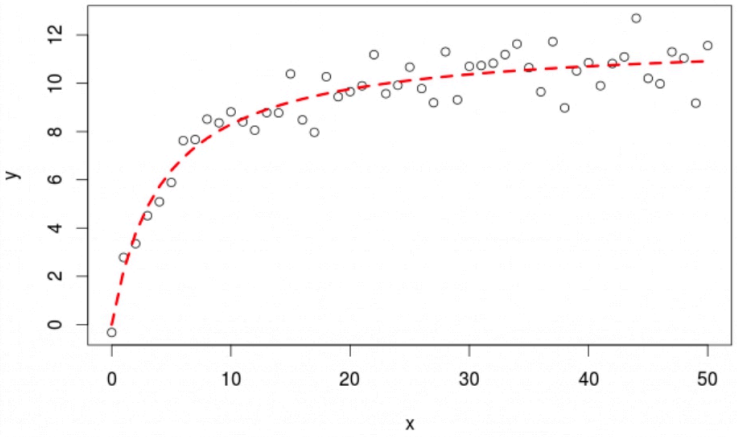
Left shows univariate linear regression and right shows multivariate linear regression in order to highlight the difference

The w^T is needed to allow multiplication of the vectors.

Without it you'd have a mismatch of dimensions

Univariate nonlinear regression

$$y = w_0 + w_1 x$$



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

This is an m-th order polynomial regression model.

Advantages of vector notation

- Vector notation is more concise.
- With the vectors $\boldsymbol{w}$ 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:

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

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

UNIVERSITY OF BIRMINGHAM