Week 2. Linear Regression and Gradient Descent

Dr. Shuo Wang

Recall: Type of Machine Learning

- This week's focus is supervised learning
- Classification and regression
- Regression means learning a function that captures the "trend" between input and output.
- The output is a continuous value.
- This function is used to predict the target values for new inputs.

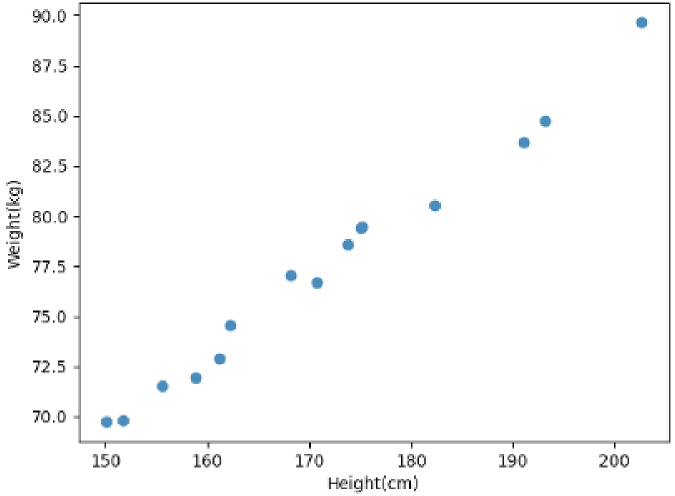
Overview

- Linear Regression – a ML algorithm for regression problems
- Gradient descent – an optimisation technique used to in ML algorithms.

Example of a regression problem

- Can we predict people's weight from their height?

Height(cm)	Weight(kg)
150.00686	69.73347
151.64326	69.83261
155.54032	71.55730
158.80535	71.92875
161.17561	72.92118
⋮	
175.15167	79.48533
182.32900	80.52182
191.11317	83.67998
193.21947	84.72086
202.68705	89.64049



- Visually, there appears to be a trend.
- A reasonable model seems to be the class of linear functions (lines).

Univariate linear regression

- We are making our assumption on the function here.
- We have one input attribute (height) – hence the name **univariate**.

$$y = f(x; \underbrace{w_0, w_1}_{\text{free parameters}}) = w_1 x + w_0$$

dependent variable

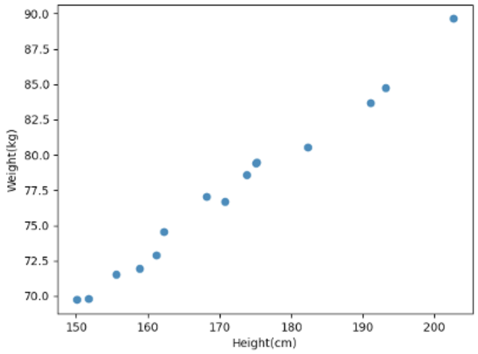
independent variable

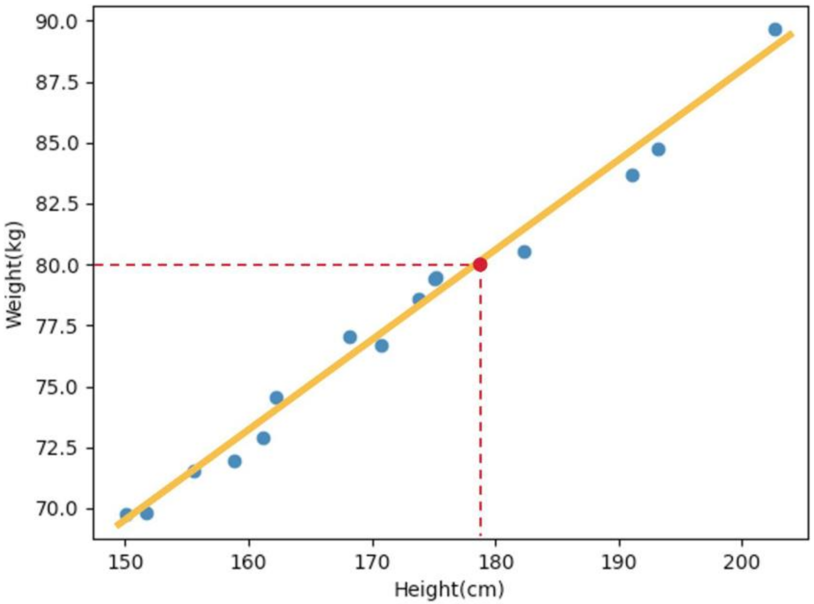
- Any line is described by this equation by specifying values for w_1 and w_0.

UNIVERSITY OF BIRMINGHAM

Check your understanding

Height(cm)	Weight(kg)
150.00686	69.73347
151.64326	69.83261
155.54032	71.55730
158.80535	71.92875
161.17561	72.92118
⋮	
175.15167	79.48533
182.32900	80.52182
191.11317	83.67998
193.21947	84.72086
202.68705	89.64049





Suppose that from historical data someone calculated the parameters of our linear model are w_0 =1.68, w_1 =0.44. A new person (James) has height x=178cm. Using our model, we can predict James' weight is 0.44 * 178 + 1.68 = 80kg.

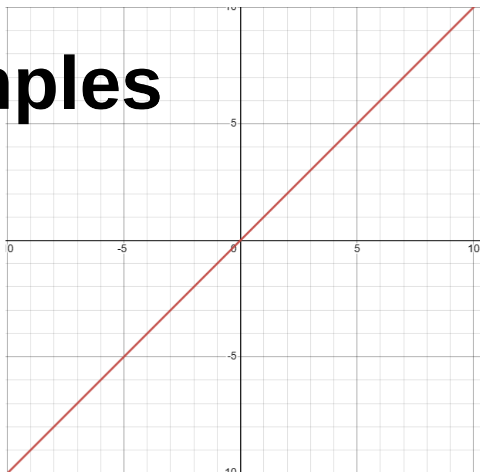
UNIVERSITY OF BIRMINGHAM

Play around with linear functions

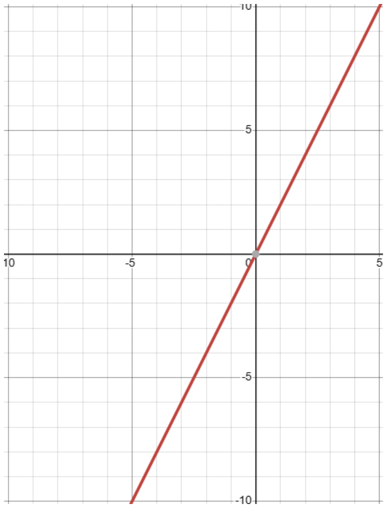
- Go to https://www.desmos.com/calculator
- Type: y = w_1 x + w_0
- Plug in some values for the free parameters, or set up the slider to see the effect of changing w_0 and w_1 .
- What is the role of the free parameters?
- w_0 is the intercept with the y-axis
- w_1 is the slope of the line – which is also the gradient function of the linear function $f(x; w_0, w_1) = w_1 x + w_0$ (recall last week!)
- Fixing concrete numbers for these parameters gives you specific lines.

UNIVERSITY OF BIRMINGHAM

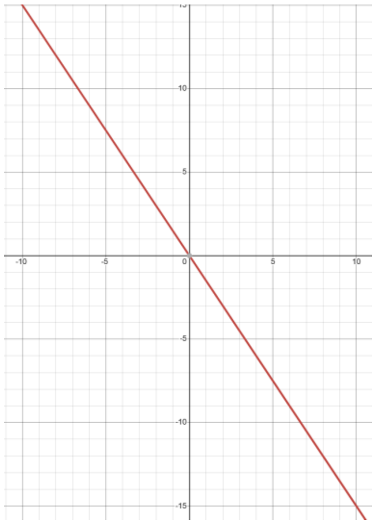
Examples



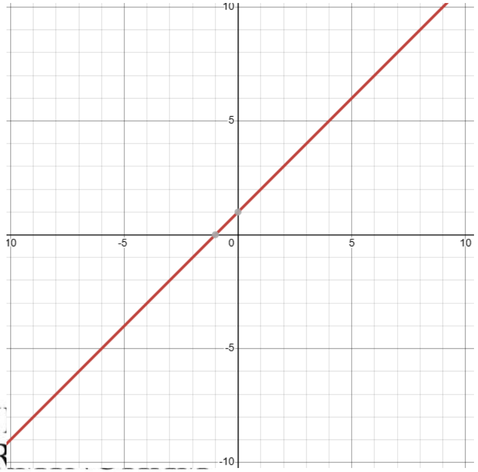
10
5
-5
-10
0
-5
0
5
10



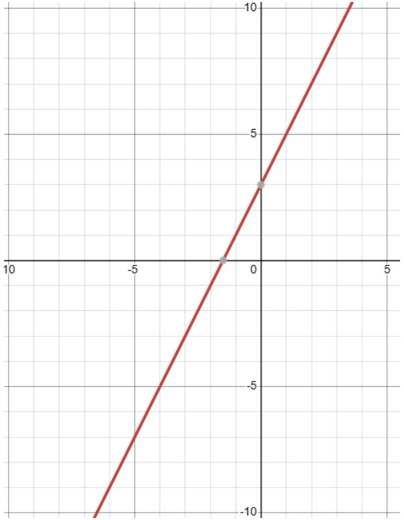
10
5
-5
-10
10
-5
0
5



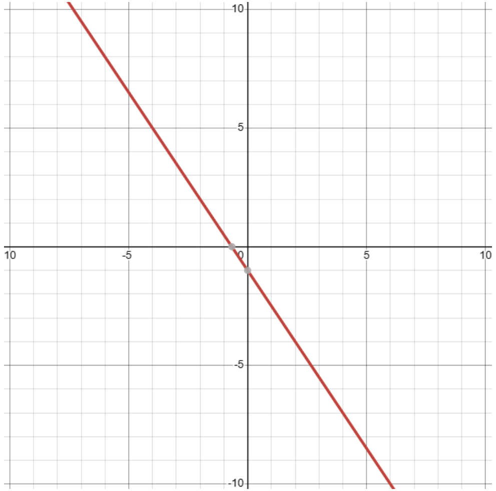
15
10
5
-5
-10
-15
-10
-5
0
5
10



10
5
-5
-10
10
-5
0
5
10

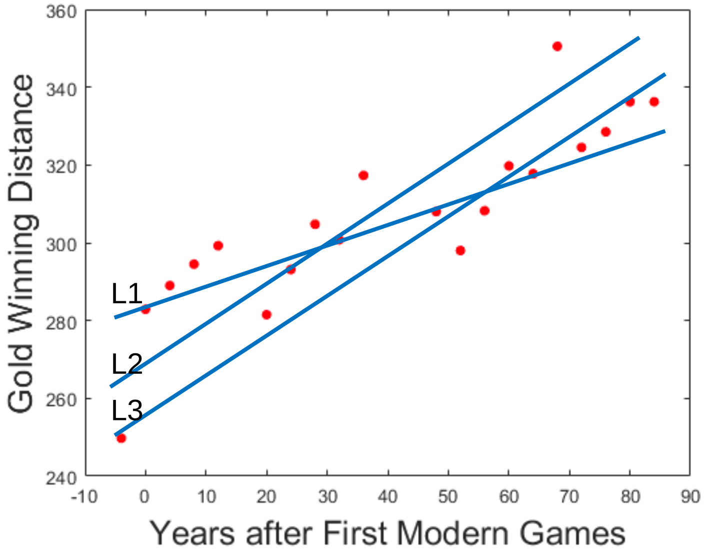


10
5
-5
-10
10
-5
0
5



10
5
-5
-10
10
-5
0
5
10

Our goal: find the "best" line (function)



- Which is the "best" line/function? That captures the trend in the data.
- Determine the "best" values for w_0 and w_1.

UNIVERSITY OF BIRMINGHAM

Loss/cost functions

- We need a criterion that tells us how good/bad that line is.
- Such criterion is called a loss function. It is a function of the free parameters ($w_{0,} w_1$).

Terminology

- Loss function = cost function = loss = cost = error function

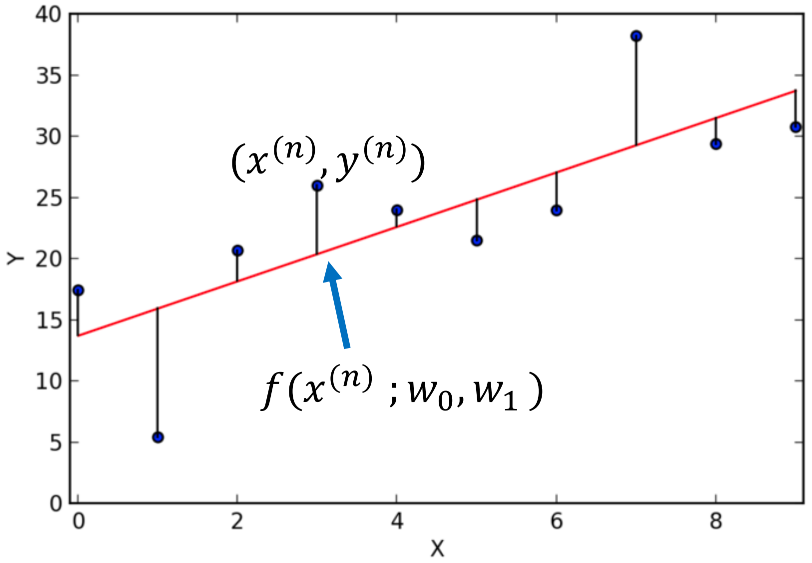
UNIVERSITY OF BIRMINGHAM

We average the losses on all training examples

- For each training example (point) n = 1,…, N,

  The loss on the n-th point is the mismatch/distance between the output of the model for this point $f\left(x^{(n)}; w_0, w_1\right)$ and the observed target $y^{(n)}$.

- Average these losses.



Loss function

- The loss expresses an error, so it must be always non-negative.
- Absolute value loss (L1 loss):

$$L1 = |f(x) - y|$$

- Mean squared error loss (L2 loss):

$$L2 = (f(x) - y)^2$$

$$\boxed{g(w_0, w_1) = \frac{1}{N} \sum_{n=1}^{N} \underbrace{\left(f\left(x^{(n)}; w_0, w_1\right) - y^{(n)}\right)^2}_{\text{Loss for the n-th training example}}}$$

$Empirical\ loss$
$used\ by\ LR$

- 0/1 loss:

$$L_{0/1} = 0\ if\ f(x) = y, else\ 1$$

UNIVERSITY OF BIRMINGHAM

Check your understanding

- Suppose a linear function with parameters w_0=*0.5*, w_1 =*0.5*
- Compute the MSE value at the training example (1,3).

- Model output: $f(x; 0.5, 0.5) = 0.5 \times 1 + 0.5 = 1$
- Actual target: 3
- MSE: $(1\text{-}3)^{2=}4$

Univariate linear regression

- Given training data

$$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots\ (x^{(N)}, y^{(N)})$$

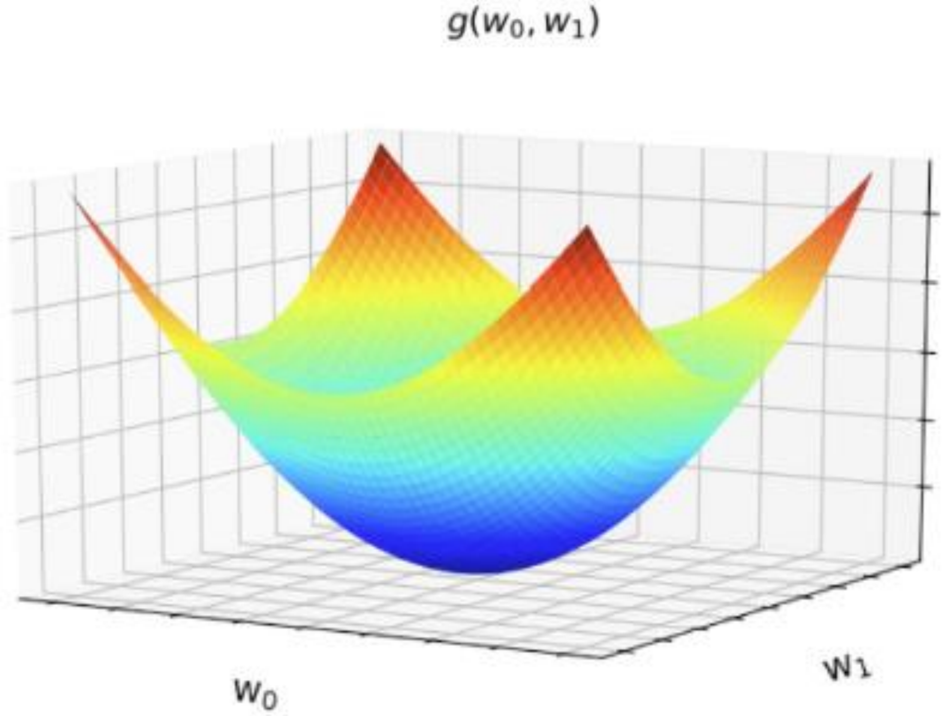
- Fit the model

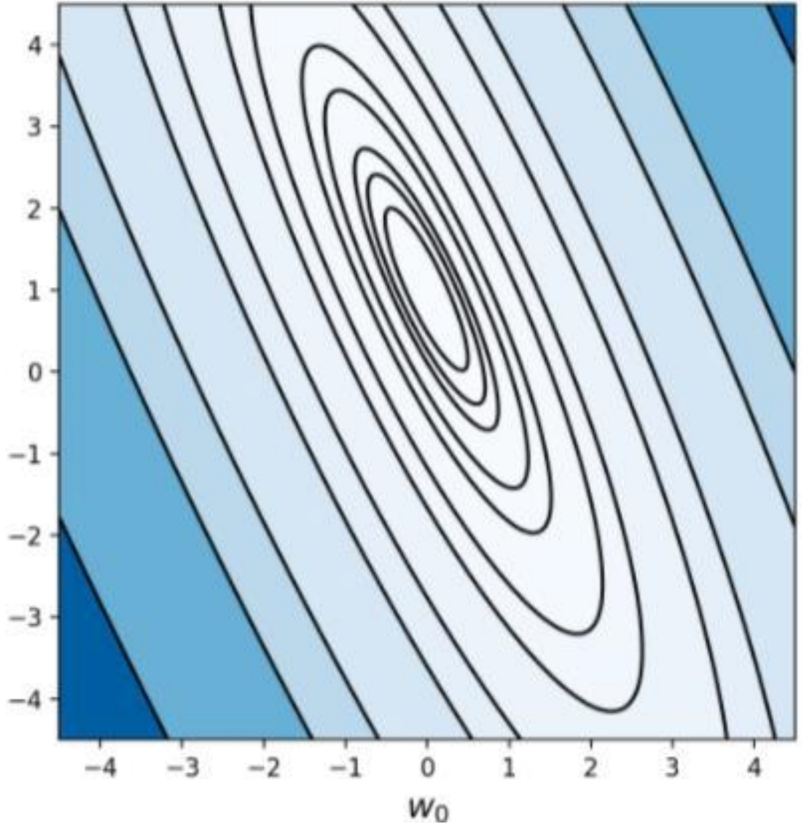
$$y = f(x; w_0, w_1) = w_1 x + w_0$$

- By minimizing the cost function

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

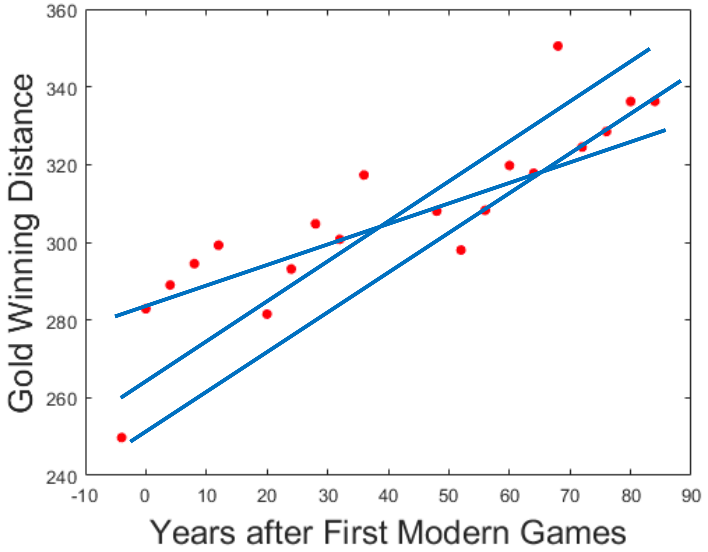
Cost function depends on the free parameter

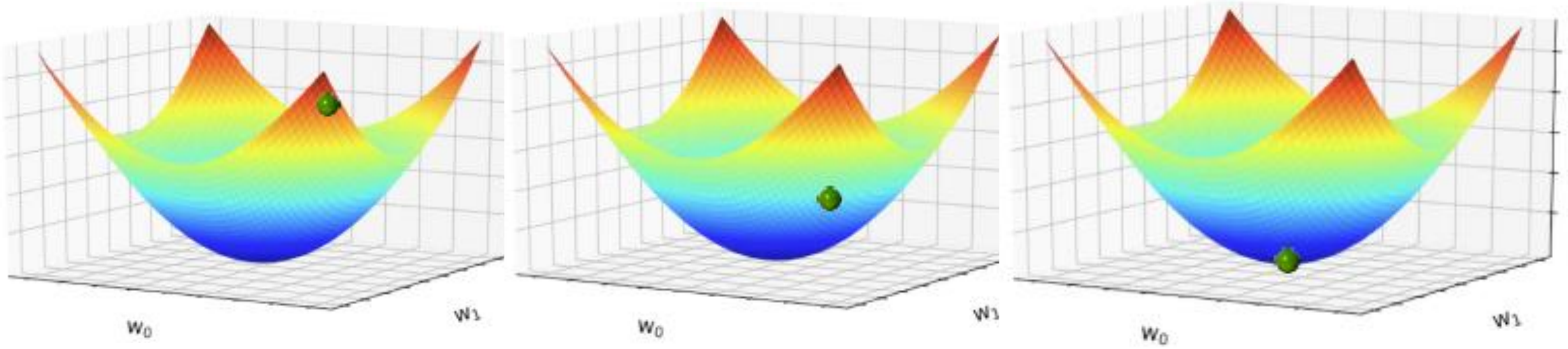




Univariate linear regression

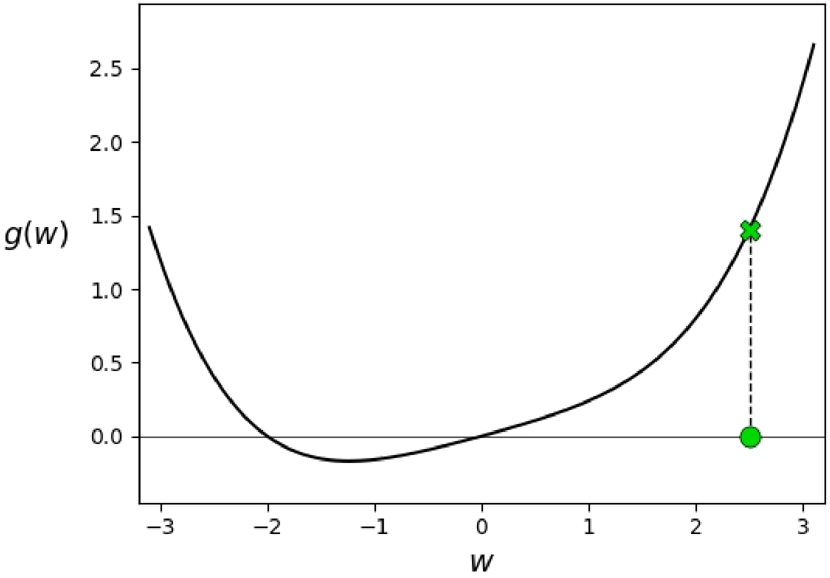
- Every combination of (w_0, w_1) has an associated cost.
- Key training task: find the 'best' values of (w_0, w_1) such that the cost is minimum.





Gradient Descent

Demo Example for Gradient Descent



machine_learning_refined/notes/3_First_order_methods/Introduction.html at gh-pages · jermwatt/machine_learning_refined · GitHub

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
Return w_0, w_1.

Gradient Descent – More General…

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

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

In a multi-dimension space:

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

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

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

- 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 $(\widehat{w}_0, \widehat{w}_1)$ gives us the direction of the steepest descent from that location.
- We take a small step in that direction – using the learning rate α.

Applying GD to solve univariate linear regression

- Recall: we aim to minimizing the cost function

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

- Using the chain rule, we have *:

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

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

*For a very detailed explanations of all steps watch: https://www.youtube.com/watch?v=sDv4f4s2SB8

Algorithm for univariate linear regression using GD

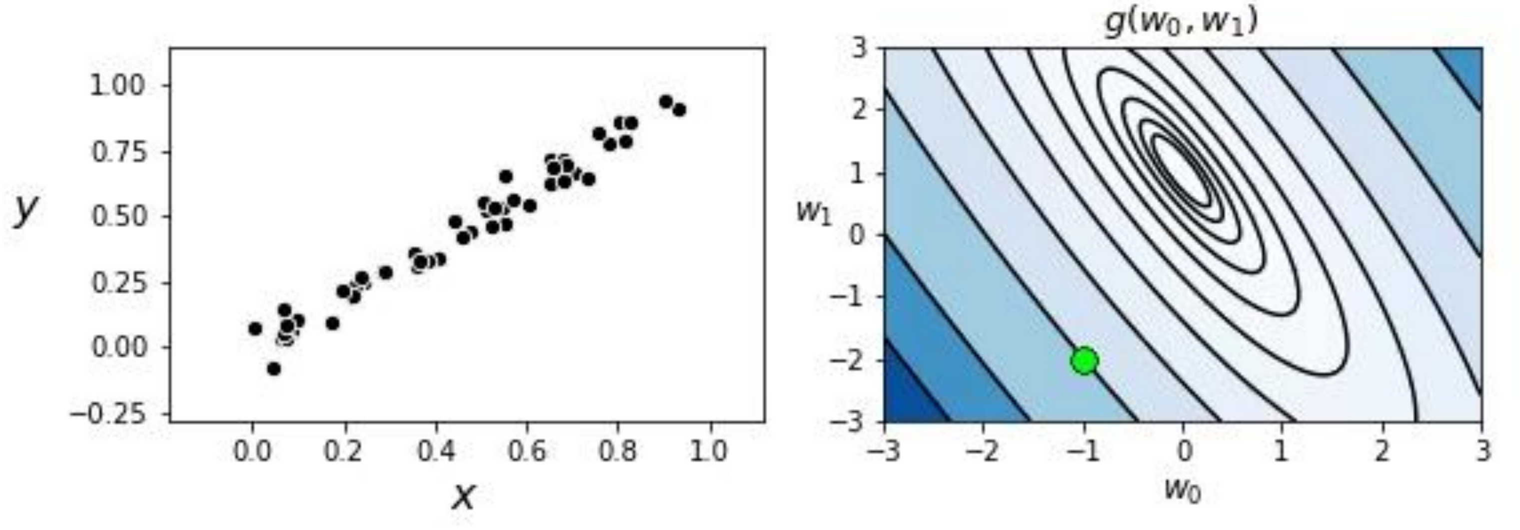
Input: $\alpha > 0$, training set $\{(x^{(n)}, y^{(n)}): n = 1,2 \dots N\}$
Initialise $w_0 = 0, w_1 = 0$
Repeat
    for n = 1, 2… N //more efficient to update after each data point
        $w_0 \coloneqq w_0 - \alpha((w_1 x^{(n)} + w_0) - y^{(n)})$
        $w_1 \coloneqq w_1 - \alpha((w_1 x^{(n)} +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

UNIVERSITY OF BIRMINGHAM

Remember the cost function: $g(w_0, w_1) = \frac{1}{N}\sum_{n=1}^{N}\left(f\left(x^{(n)}; w_0, w_1\right) - y^{(n)}\right)^2$

Univariate linear regression

- Every combination of (w_0, w_1) has an associated cost.
- Key training task: find the 'best' values of (w_0, w_1) such that the cost is minimum.

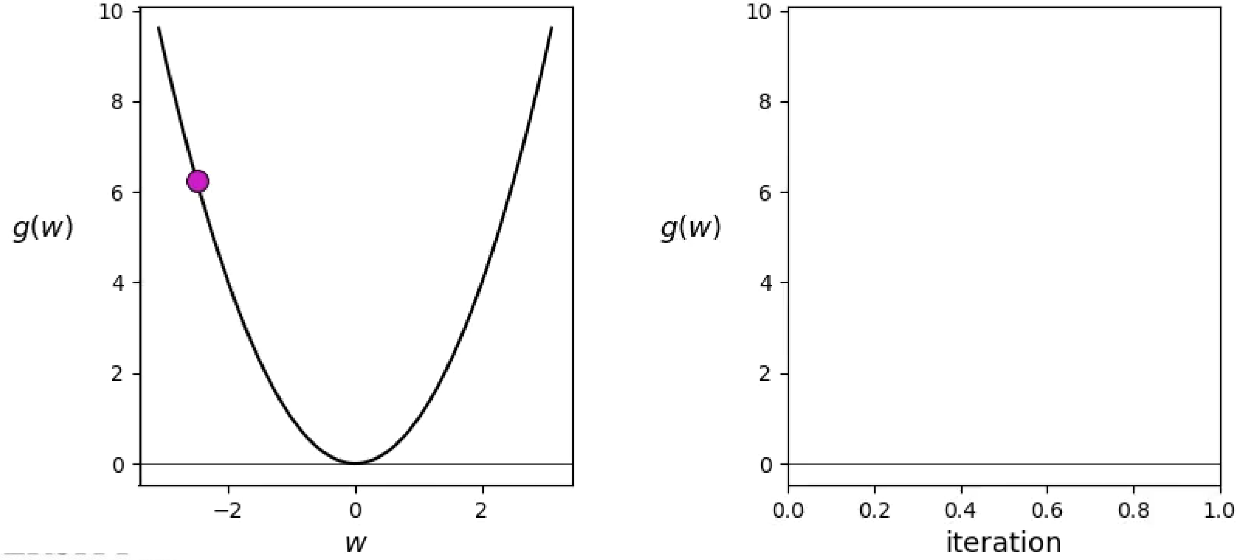


BIRMINGHAM

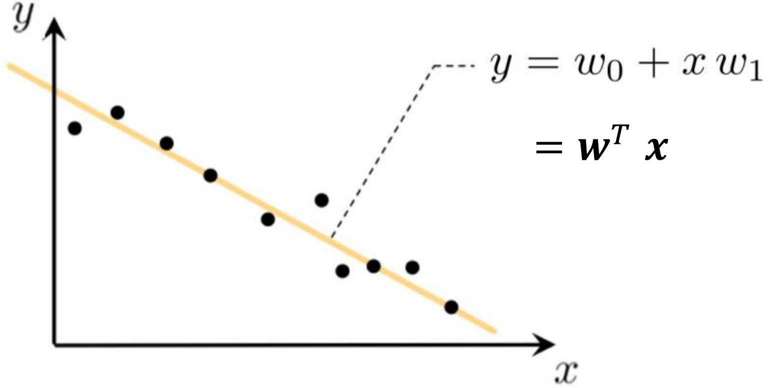
https://github.com/jermwatt/machine_learning_refined/tree/gh-pages/notes/5_Linear_regression/videos/animation_1.mp4

Effect of the learning rate

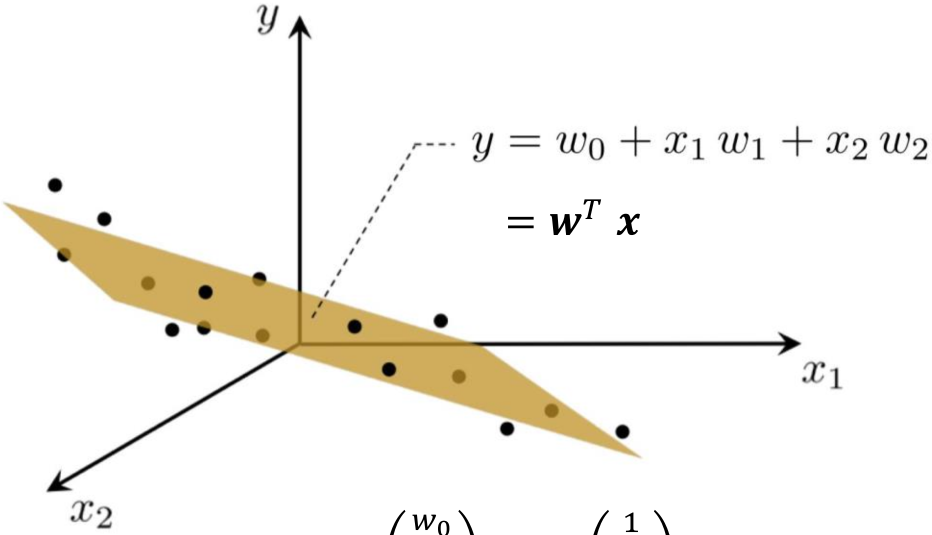
Whether or not we descend in the function when taking this step depends completely on how far along it we travel.



Multivariate linear regression



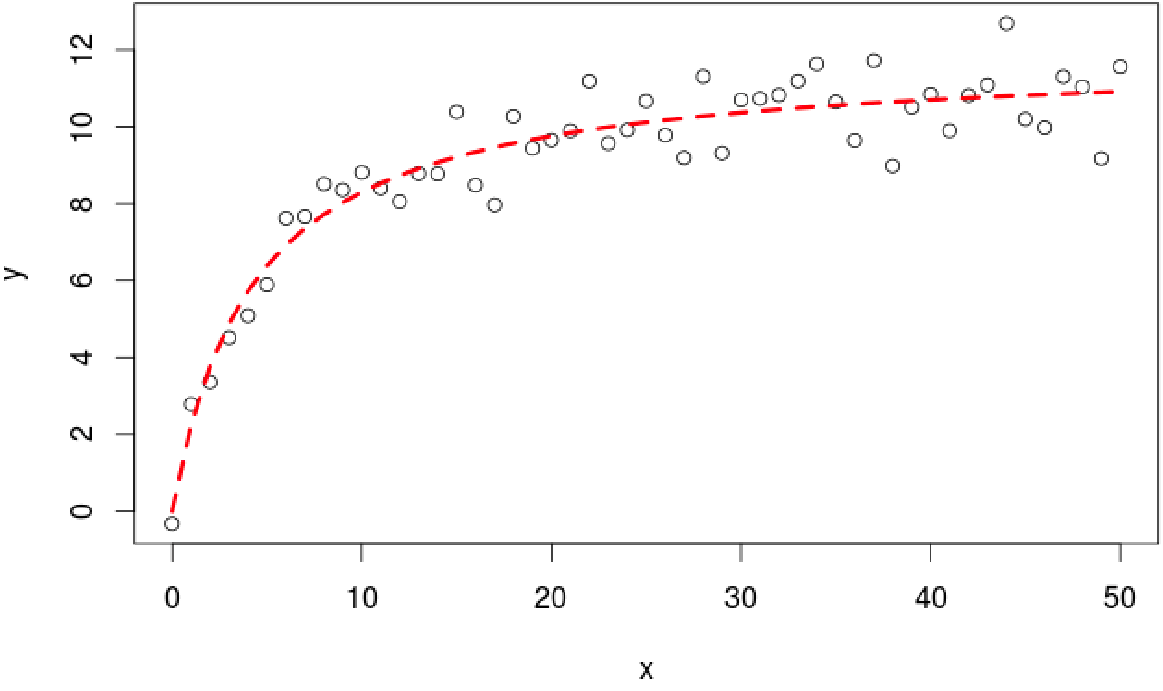
$$\boldsymbol{w} = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}, \boldsymbol{x} = \begin{pmatrix} 1 \\ x \end{pmatrix}$$



$$\boldsymbol{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}, \boldsymbol{x} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$$

Univariate nonlinear regression

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \cdots + w_m x^m = \boldsymbol{w}^T \boldsymbol{x}$$



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

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:

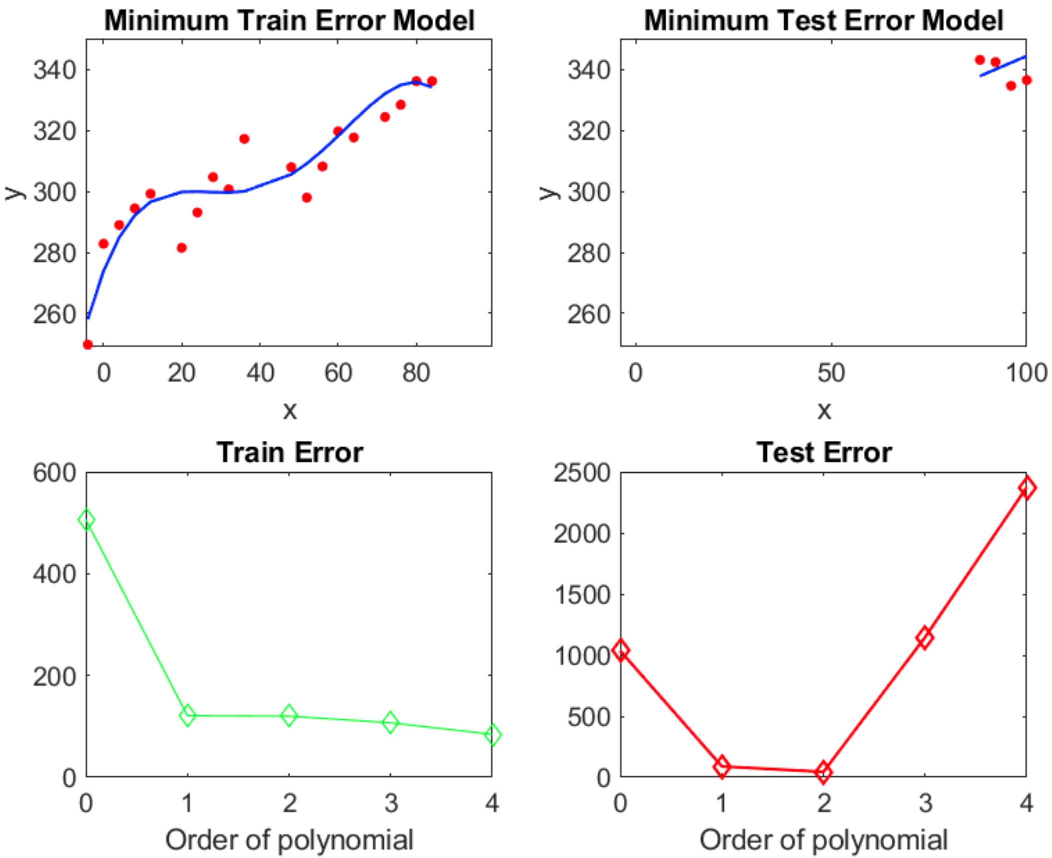
$$\nabla g(\boldsymbol{w}) = 2(\boldsymbol{w}^T \boldsymbol{x}^{(n)} - y^{(n)})\boldsymbol{x}^{(n)}$$

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

UNIVERSITY OF BIRMINGHAM

Overfitting

- Linear regression also has overfitting problems. The model can be over-complex.
- What order of polynomial would you choose?



UNIVERSITY OF BIRMINGHAM

Summary of the lecture

- The idea of linear regression
- Univariate/multivariate linear regression
- Loss/cost function and optimisation
- Gradient vector and gradient descent
- Reading: L1 and L2 loss functions

Q/A

Teams Channel for Week2
Office hours
See Canvas module homepage

Figures and animations referred to:
Jeremy Watt et al. Machine Learning Refined. Cambridge University Press, 2020.
https://github.com/jermwatt/machine_learning_refined