CIS 635 - Knowledge Discovery & Data Mining

ML Model training: Introduction to Gradient descent

Key concepts summary review!

Vector Space

Key concepts summary review!

Vector Space

Distance metric

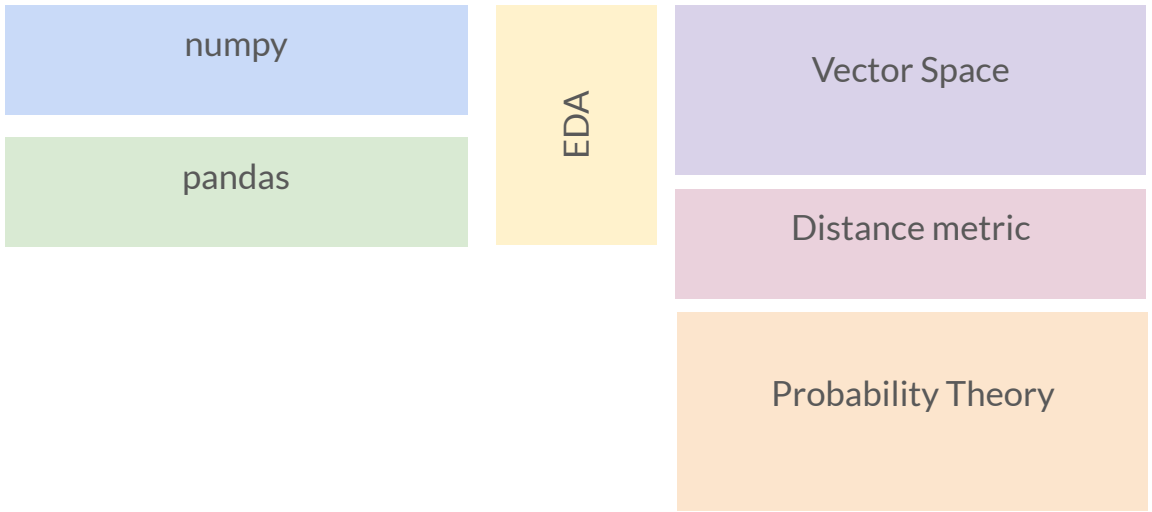
Key concepts summary review!

Vector Space

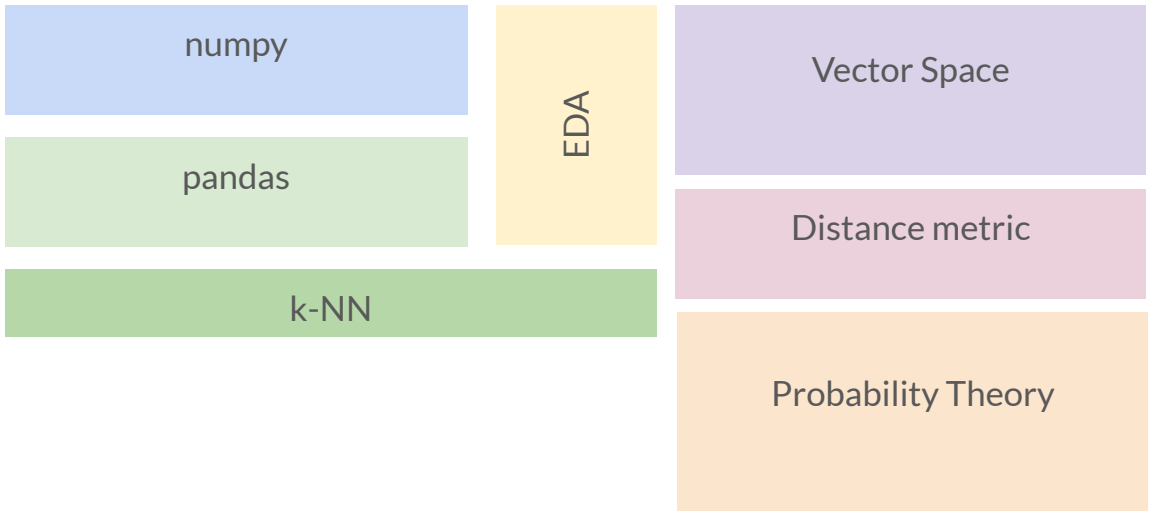
Distance metric

Probability Theory

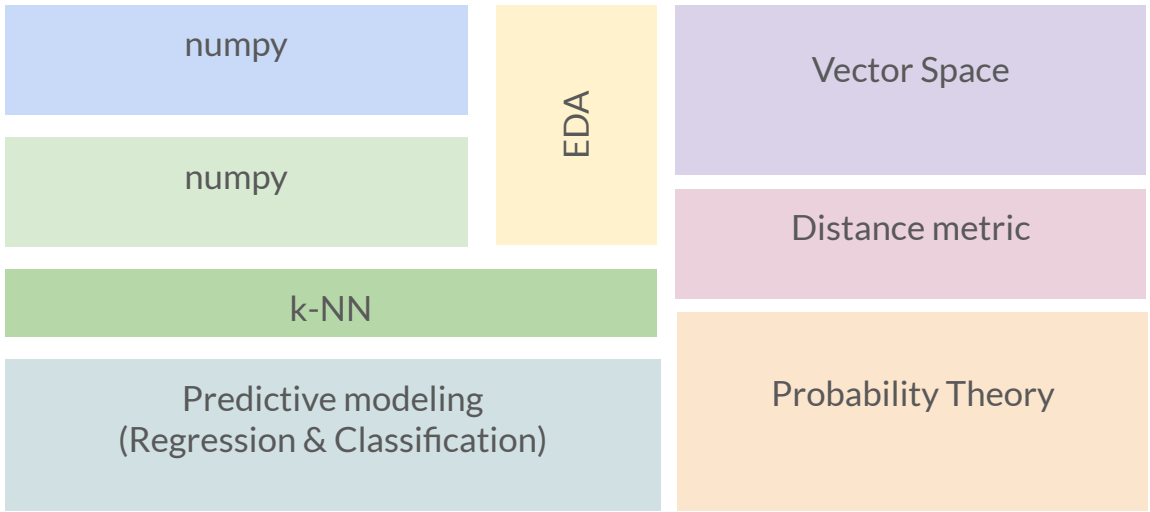
Key concepts summary review!



Key concepts summary review!



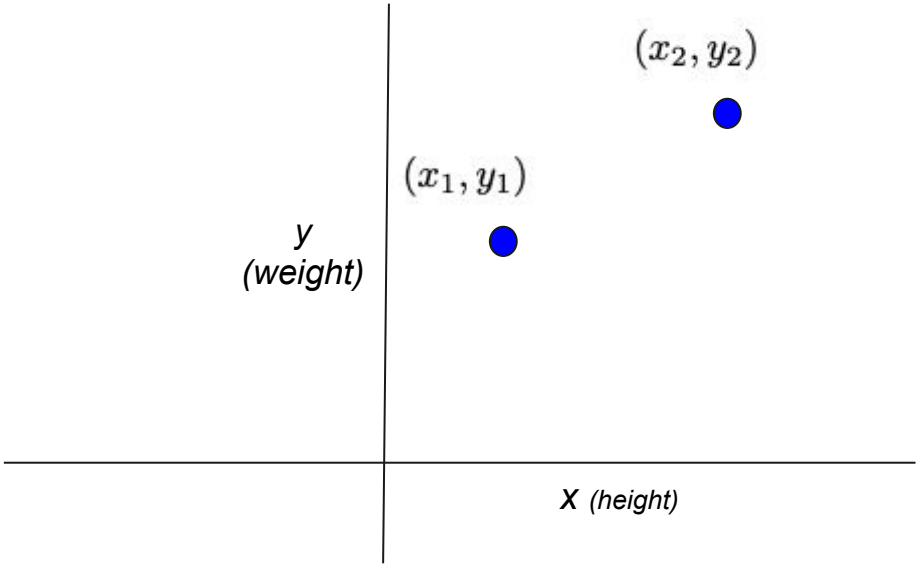
Key concepts summary review!



What we'd like to accomplish today

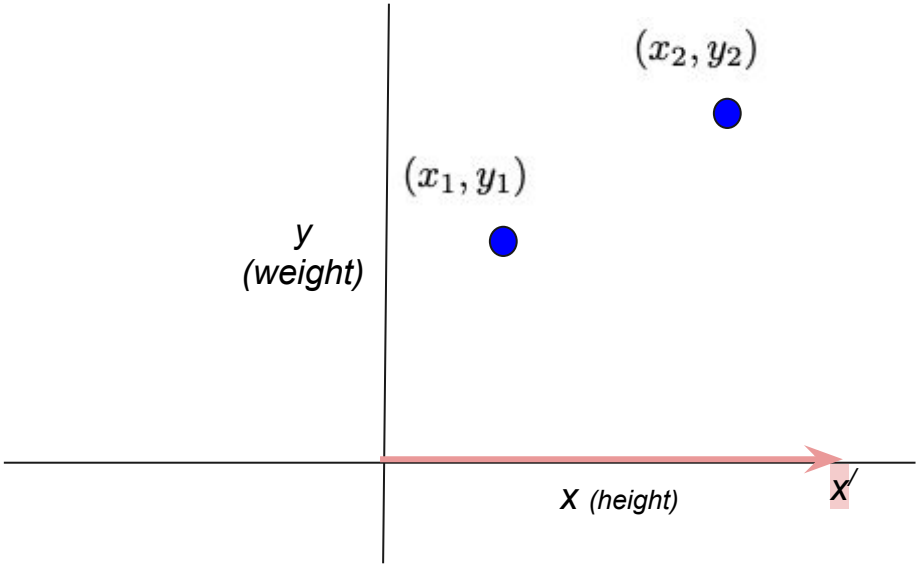
- Model training using Gradient descent
  - ➜ Refreshing some high school maths: **linear equation**
  - ➜ A simple two parameter **linear regression** model
  - ➜ The **Gradient descent algorithm**
- Hands on **Notebook implementation**
- QA

k-NN Regression



Given two known data points (x_1,y_1), and (x_2,y_2), and

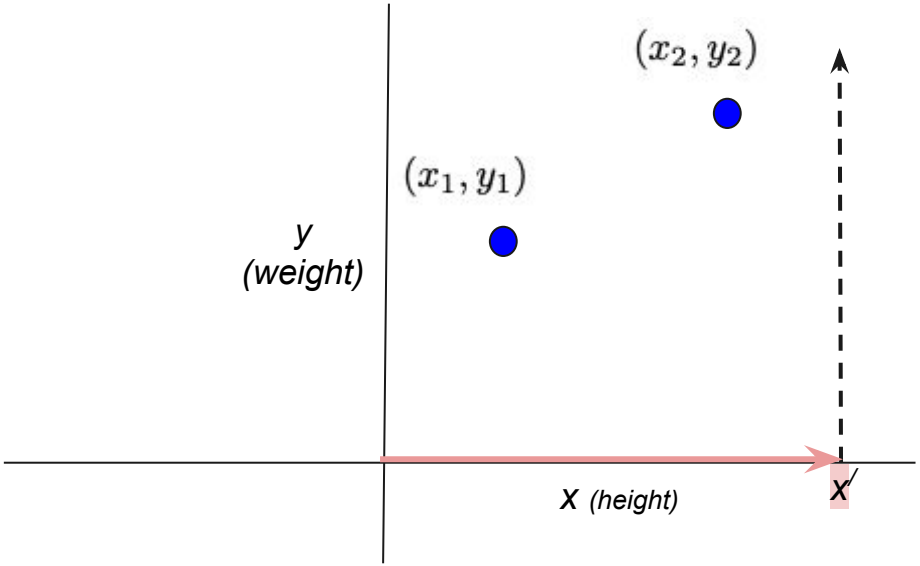
k-NN Regression



Given two known data points (x_1,y_1), and (x_2,y_2), and

- for test input $x^{/}$, you have to predict $y(x^{/})$.
- I.e. you have to plot ($x^{/}$, ?)

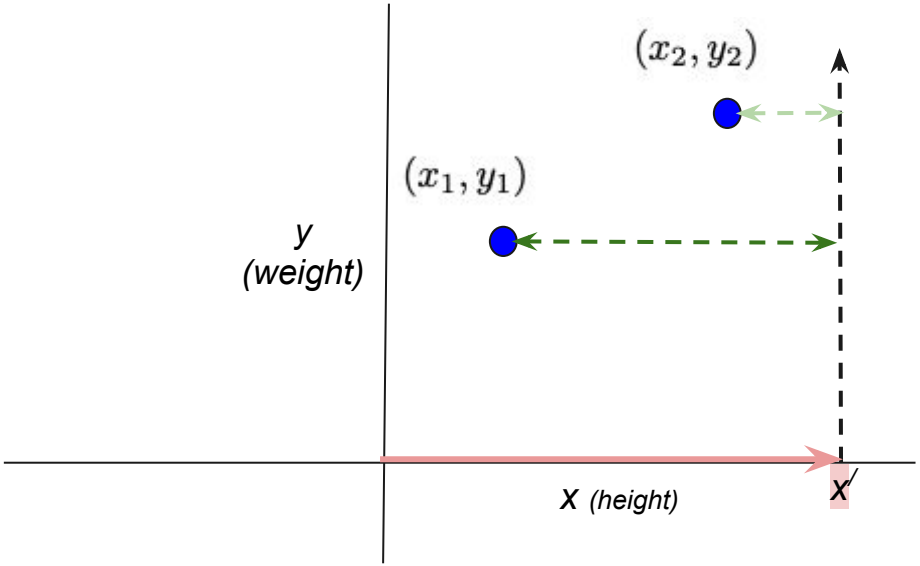
k-NN Regression



Given two known data points (x_1,y_1), and (x_2,y_2), and

- for test input x', you have to predict $y(x')$.
- I.e. you have to plot (x', ?)
- To estimate the distances let's draw the vertical line

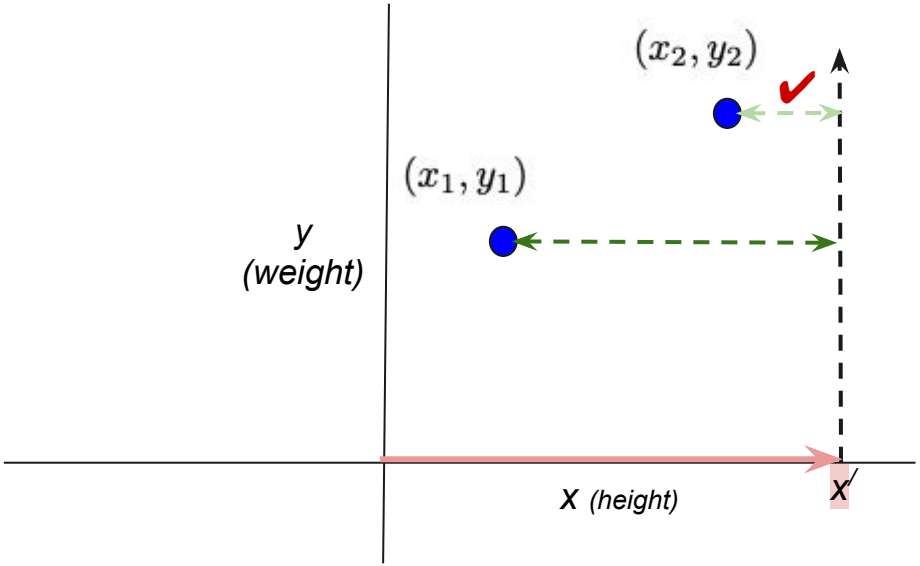
k-NN Regression



Given two known data points (x_1,y_1), and (x_2,y_2), and

- for test input x', you have to predict $y(x')$.
- I.e. you have to plot (x', ?)
- To estimate the distances let's draw the vertical line
- Horizontal dotted lines show the point distances (L1)

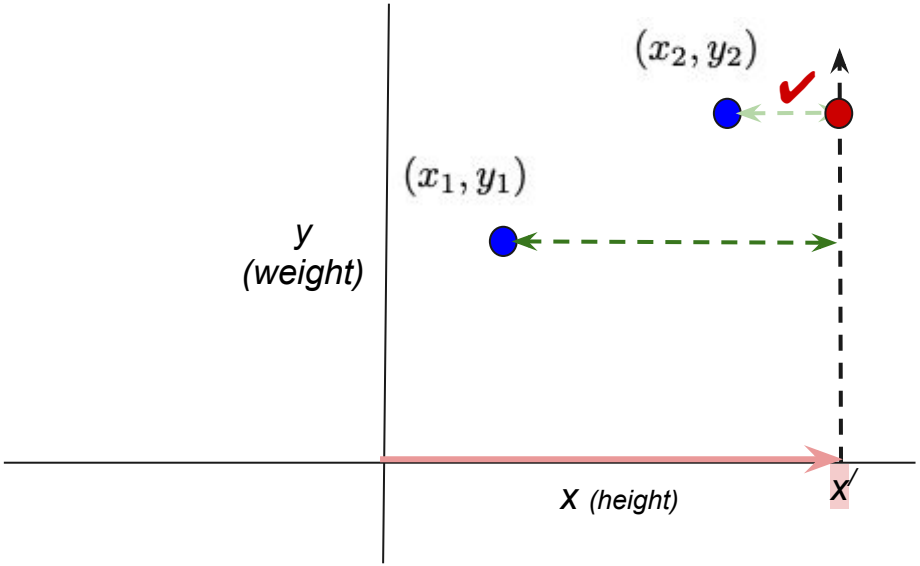
k-NN Regression



Given two known data points (x_1,y_1), and (x_2,y_2), and

- for test input $x^{/}$, you have to predict $y(x^{/})$.
- I.e. you have to plot ($x^{/}$, ?)
- To estimate the distances let's draw the vertical line
- Horizontal dotted lines show the point distances (L1)
- We find the lighter green on is the closest one [k(1)-NN]
-

k-NN Regression



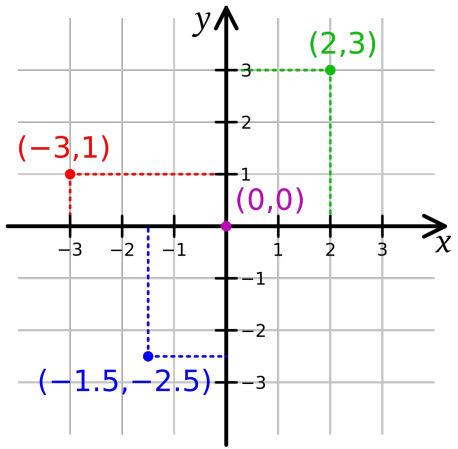
Given two known data points (x_1,y_1), and (x_2,y_2), and

- for test input $x^/$, you have to predict $y(x^/)$.
- I.e. you have to plot ($x^/$, ?)
- To estimate the distances let's draw the vertical line
- Horizontal dotted lines show the point distances (L1)
- We find the lighter green on is the closest one [k(1)-NN]
- We propagate the associated label(s), i.e.

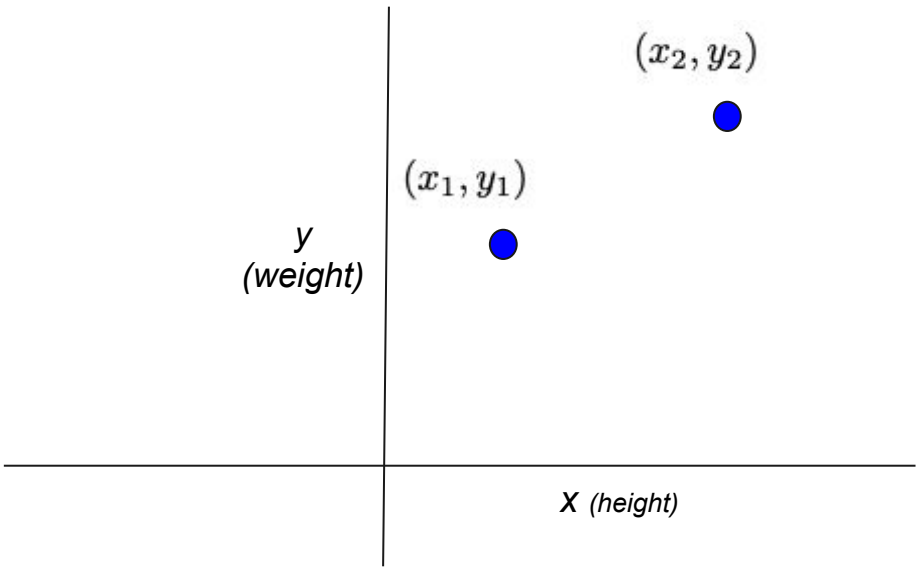
  $y(x^/) = y_2$
- If we have more data points we may go for a higher k, and take the average

Recall, we said k-NN is non parametric

- K-nearest neighbors (k-NN)
  - Supervised learning
  - **Non parametric**
- Based on what data (features are available) and on distance measures.

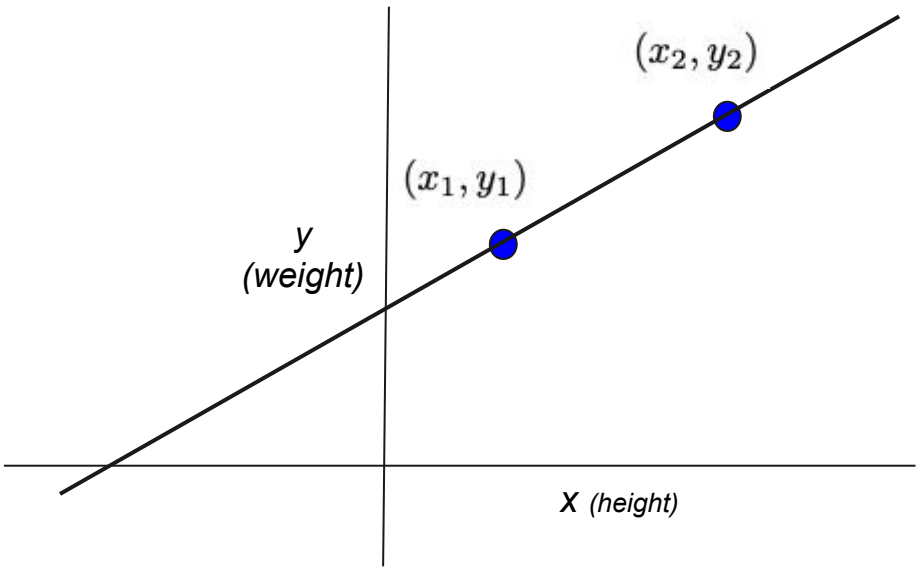


Linear equation, a quick review



Given two data points

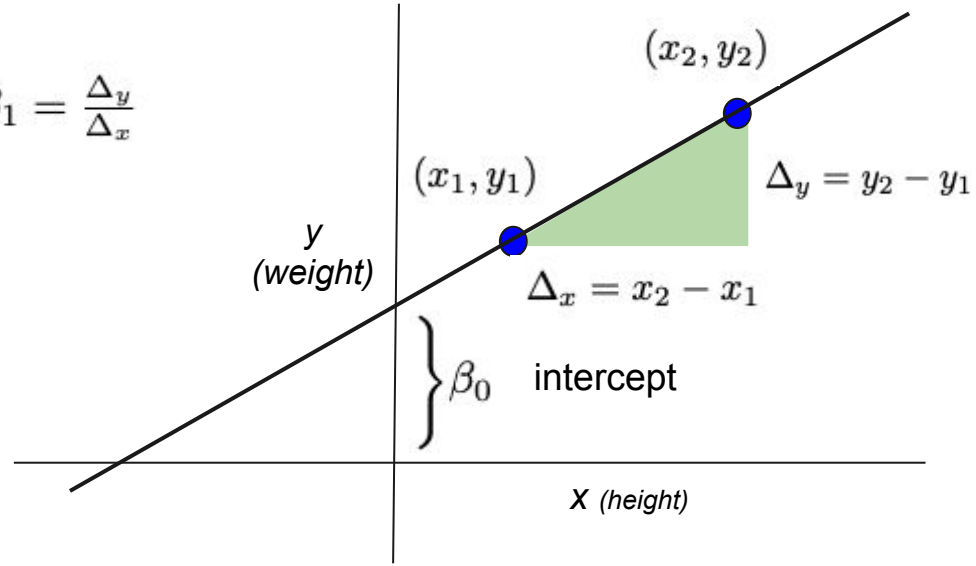
Linear equation, a quick review



We can fit a linear equation

Linear equation to a linear function, a quick review

slope $\beta_1 = \frac{\Delta_y}{\Delta_x}$



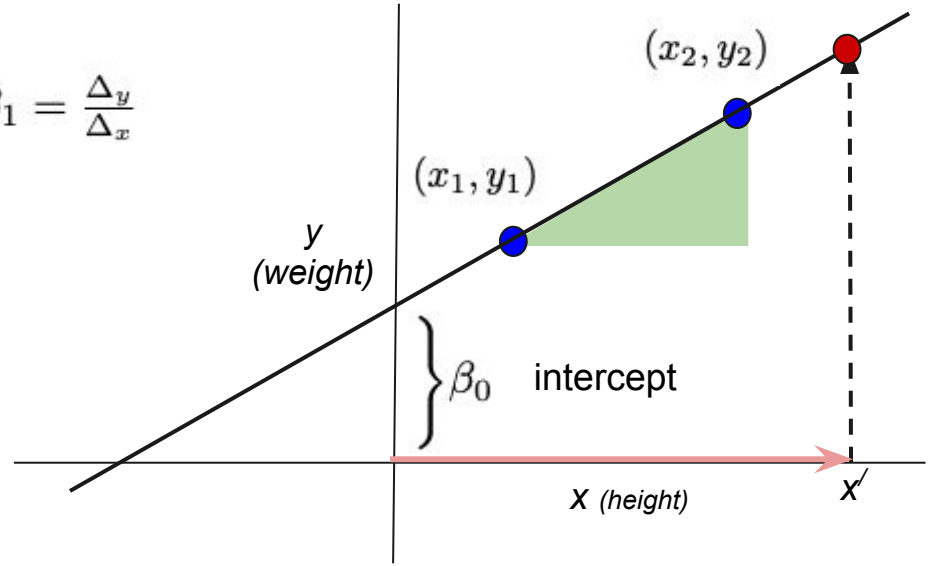
- This linear equation can be used to explain the relationship between the two axes ***x*** (independent variable) vs ***y*** (dependent variable) - as

$$y = \beta_0 + \beta_1 x$$

- A simple model with parameters: **slope**, and **intercept**
- For any given x', this model can predict $y(x')$ using the above equation.

Linear equation to a linear function, a quick review

slope $\beta_1 = \frac{\Delta_y}{\Delta_x}$

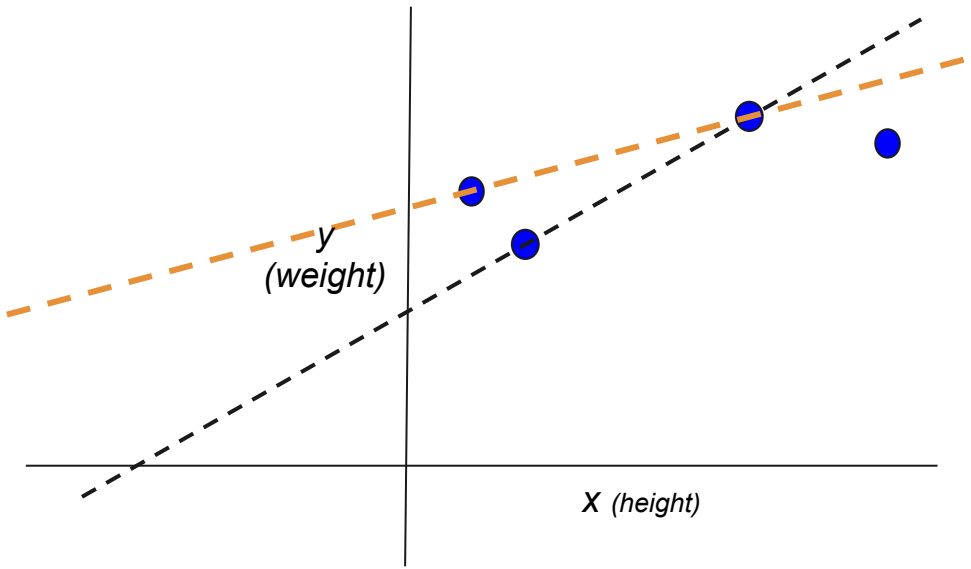


- This linear equation can be used to explain the relationship between the two axes ***x*** (independent variable) vs ***y*** (dependent variable) - as

$$y = \beta_0 + \beta_1 x$$

- A simple model with parameters: **slope**, and **intercept**
- For any given x', this model can predict $y(x')$ using the above equation.

Fitting a Linear function, a quick review



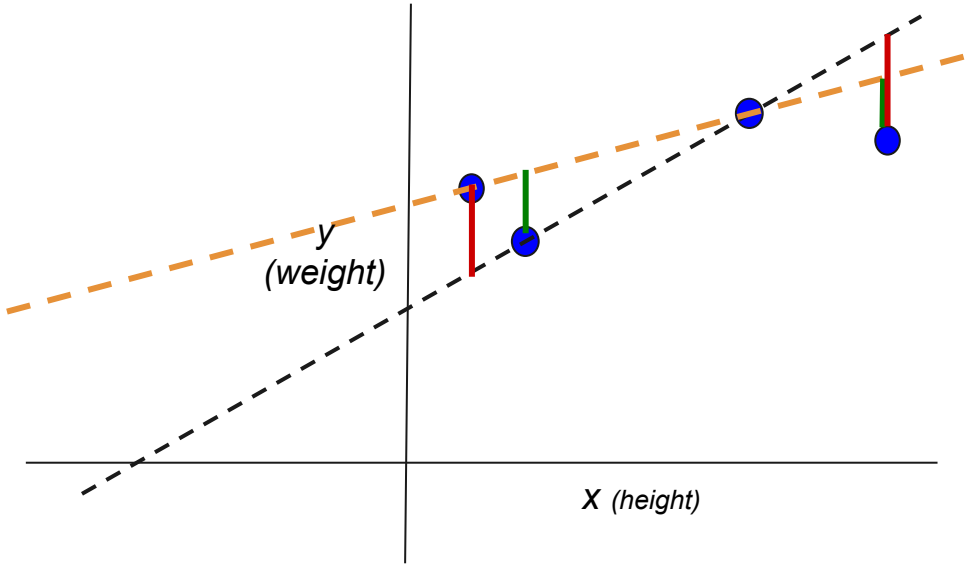
- Example of linear relationship: height vs weight of a person, marketing expense vs sales
- > 2 data points are unlikely fit perfectly on a straight line, which a straight line (2 param model) cannot fit
- We need some approximations
- Let's examine the two models for the 4 data points on the left

model β_0, β_1

model: β_0, β_1

- Both model perfectly fits 2 points each
- Orange (visual screening) model seems to be a better fit, but why?

Fitting a Linear function/function, a quick review



Based on Error is higher than **Error**

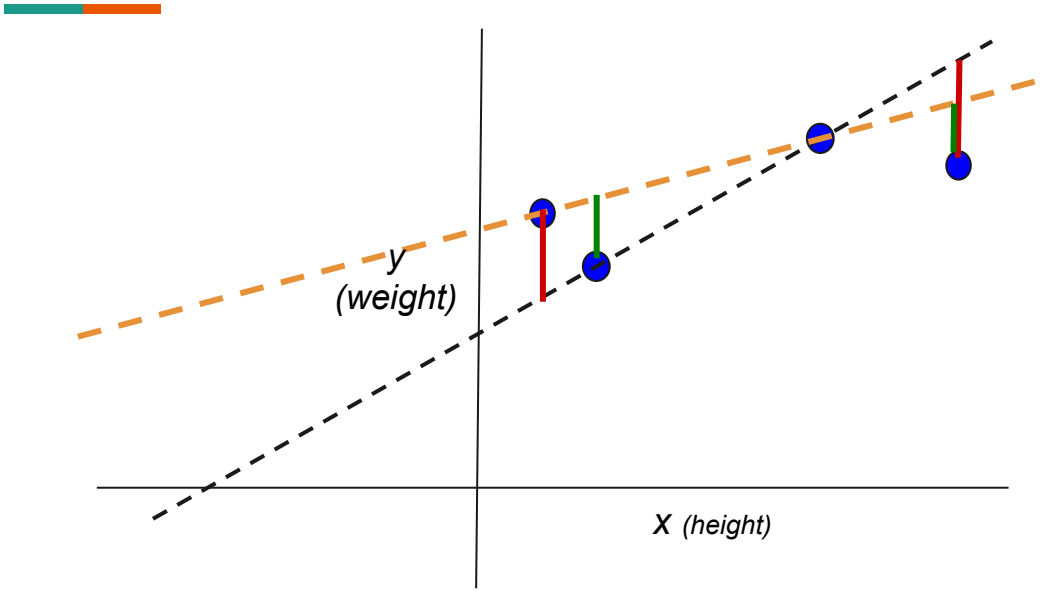
- Example of linear relationship: height vs weight of a person, marketing expense vs sales
- > 2 data points are unlikely fit perfectly on a straight line, which a straight line (2 param model) cannot fit
- We need some approximations
- Let's examine the two models for the 4 data points on the left

model β_0, β_1

model: β_0, β_1

- Both model perfectly fits 2 points each
- Orange (visual screening) model seems to be a better fit, but why?

Fitting a linear function/model



Model

$$\hat{y} = \beta_0 + \beta_1 x$$

$$\Theta = \{\beta_0, \beta_1\}$$

Fitting Error

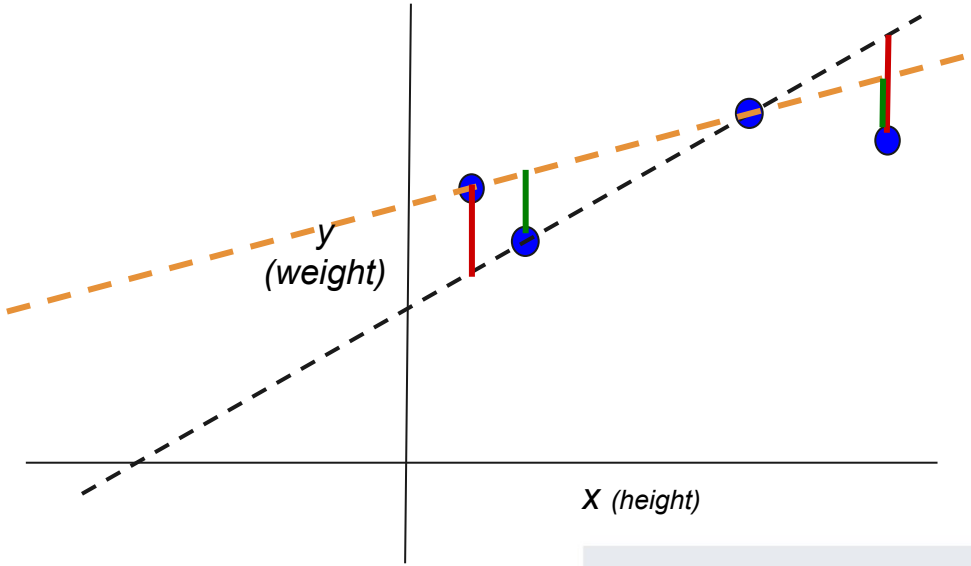
$$\epsilon = |\hat{y} - y|$$

Optimization function

$$E_\Theta = \frac{1}{2}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

$$\Theta^* = \text{argmin}_\Theta \, E\{(x_i, y_i)\}_{i=1,\cdots,N}$$

Fitting a linear function/model



Ordinary Least Squares(OLS)

$$\hat{\beta} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y},$$

Out of scope today ….

Model

$$\hat{y} = \beta_0 + \beta_1 x$$

$$\Theta = \{\beta_0, \beta_1\}$$

Fitting Error

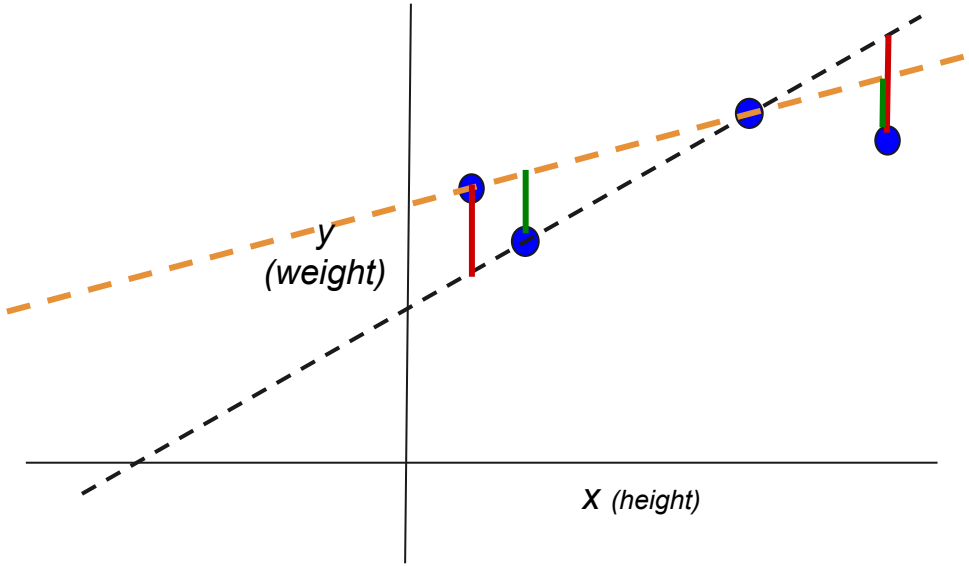
$$\epsilon = |\hat{y} - y|$$

Optimization function

$$E_\Theta = \frac{1}{2}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

$$\Theta^* = \text{argmin}_\Theta \, E\{(x_i, y_i)\}_{i=1,\cdots,N}$$

Gradient descent (- ascent)



Model

$$\hat{y} = \beta_0 + \beta_1 x$$

$$\Theta = \{\beta_0, \beta_1\}$$

Fitting Error

$$\epsilon = |\hat{y} - y|$$

Optimization function

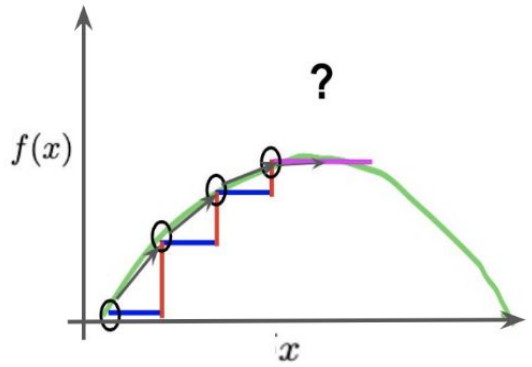
$$E_\Theta = \frac{1}{2}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

$$\Theta^* = \text{argmin}_\Theta \, E\{(x_i, y_i)\}_{i=1,\cdots,N}$$

Gradient descent (- ascent)

- We have decided to build some stairs as we need to visit the top of a mountain frequently. We want
  - To produce **equal** width stairs (**step size**)
  - To make the stair corners **touch the periphery** of the mountain (maybe someday we want recreate the shape of the mountain)

- For such a **convex** function ***f(x)***, we will have varying height stairs, and at the **top of the mountain** stairs height will be **close to zero**, an indicator that we are at the top.
- Stair **height/width** is called the **gradient** of *f(x)*, the arrowhead denoting ts **direction (towards an increasing value)**.
- We have to choose the **step size** sensibly: larger step size may miss the peak while smaller step size will take too much efforts to reach to the top



$step\ size:\ \delta x$
$gradient:\ \frac{\delta}{\delta x} f(x)$

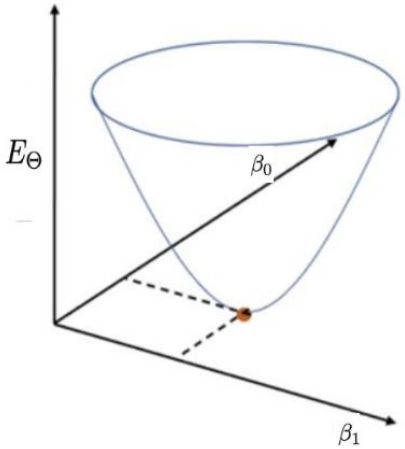
Gradient descent (- ascent)

1. Start with an initial ***(β_0, β_1)*** and a ***learning rate (L)*** , a scalar, which controls the gradient step.
2. For ***N*** training data points, estimate the **model loss**
3. Estimate the gradient (vector of partial derivatives): $\nabla E_\Theta = \left[\frac{\delta}{\delta\beta_0}(E_\Theta), \frac{\delta}{\delta\beta_1}(E_\Theta)\right]$
4. Update parameters

$$\beta_0 \leftarrow \beta_0 - L * \frac{\delta}{\delta\beta_0}(E_\Theta)$$

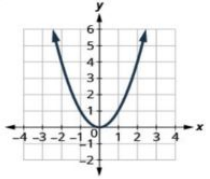
$$\beta_1 \leftarrow \beta_1 - L * \frac{\delta}{\delta\beta_1}(E_\Theta)$$

4. Go to step 2) and iterate until the model loss reaches a predefined **threshold** or a certain **number of iterations** are executed.



Objective function

$$\Theta^* = \text{argmin}_\Theta \, E\{(x_i, y_i)\}_{i=1,\cdots,N}$$



Partial derivatives

$$\frac{\delta}{\delta\beta_0}(E_\Theta) = \sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)$$

$$\frac{\delta}{\delta\beta_1}(E_\Theta) = \sum_{i=1}^{N}(\beta_0 + \beta_1 x_i - y_i)x_i$$

Variants of gradient descent (- ascent)

- **Batch gradient descent**: Gradient based on chunk of data
- **Stochastic gradient descent**: chunk size is **1**

Notebook presentation

- Notebook github

What we have discussed today

- Model training using Gradient descent
  - ➜ Refreshing some high school maths: **linear equation**
  - ➜ A simple two parameter **linear regression** model
  - ➜ The **Gradient descent algorithm**
- Hands on **Notebook implementation**
- **QA …….**

Merci Beaucoup!!

GPT

Our model today

Regression vs Classification!

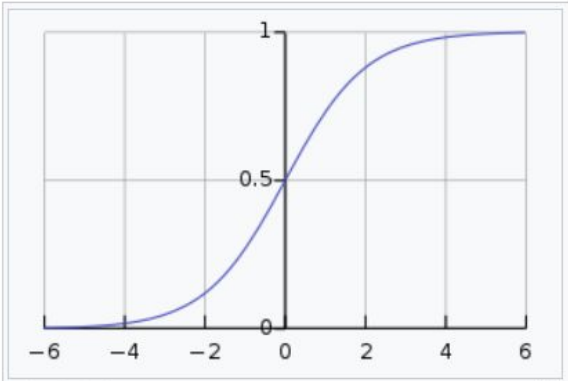
Whiteboarding!

Classification

- General Idea (two steps process)
  - LR (Bias Only)
  - LR (general)

Classification

- General Idea (two steps process)
  - LR (Bias Only)
  - LR (general)
- **Logistic Regression (one single step, but with probability distribution)**
  - **The famous sigmoid function**
  - **In NN they call it activation function**



The logistic curve

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$