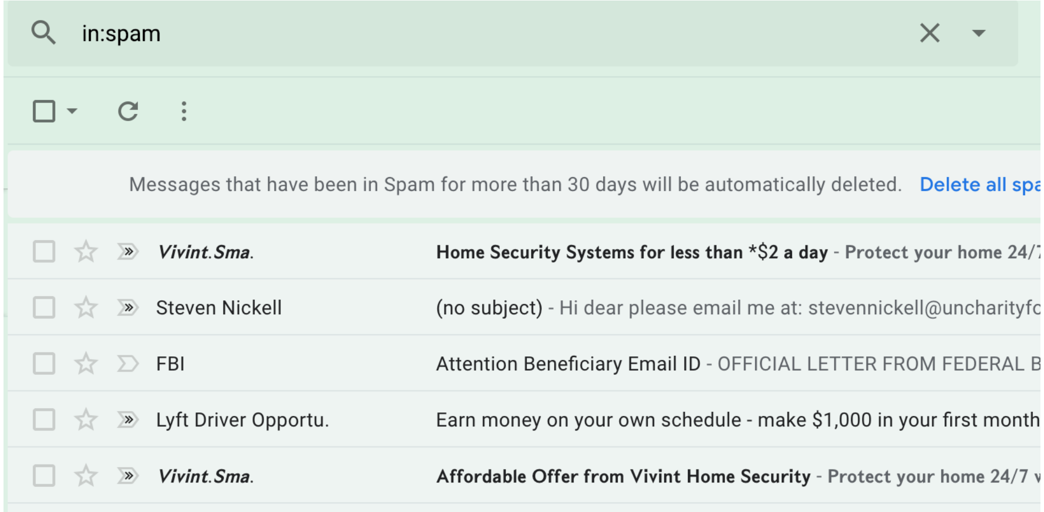
1 Linear Regression and Gradient Descent

1.1 Introduction

//TODO add references to Coursera ML and Machine Learning for Humans book We come across machine learning all the time

//TODO: add overall outline and what we are intending to cover

It's why I don't have to sift through these emails in my inbox

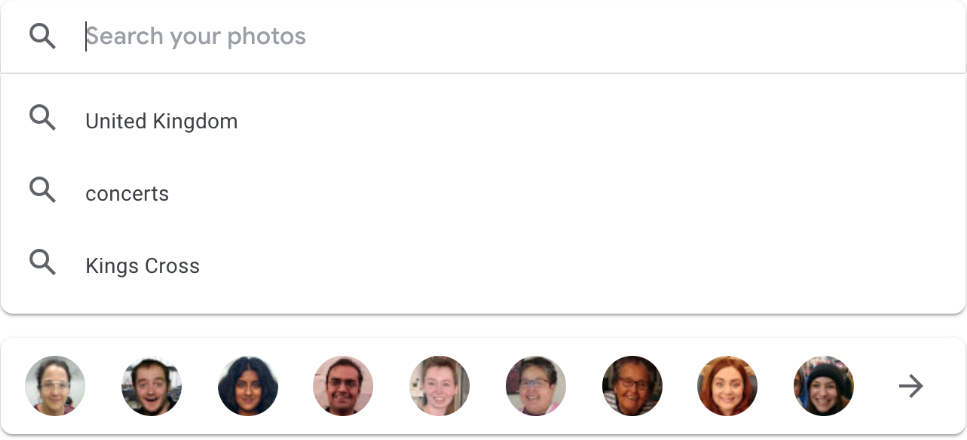


in:spam

Messages that have been in Spam for more than 30 days will be automatically deleted. Delete all spa

Vivint.Sma.	Home Security Systems for less than *$2 a day - Protect your home 24/7
Steven Nickell	(no subject) - Hi dear please email me at: stevennickell@uncharityfc
FBI	Attention Beneficiary Email ID - OFFICIAL LETTER FROM FEDERAL B
Lyft Driver Opportu.	Earn money on your own schedule - make $1,000 in your first month
Vivint.Sma.	Affordable Offer from Vivint Home Security - Protect your home 24/7 v

It's why I can search my photos, by photos



Search your photos

United Kingdom

concerts

Kings Cross

1.2 What is Machine Learning?

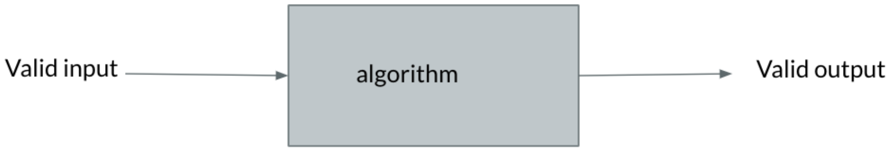
According to wikipedia: Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead.

Machine learning algorithms can learn to do a particular task without being explicitly programmed by building a mathematical model based on sample data, known as "training data". Then, that model can be applied to new data not previously used to build the model.

1.3 What is an algorithm?

An algorithm is often described as a set of steps to accomplish a particular task. You could describe an algorithm for brushing your teeth, or making a grilled cheese sandwich

But it's a bit more general than just a set of steps



An algorithm is a way to solve a computational problem, and a computational problem just specifies valid input and desired output.

For example, for the problem of spam filtering the input would be an email message, and the output would be a classification (spam/not spam). The algorithm could be a set of steps. We could compose a series of regular expressions to the message that are based on previous messages that we know have turned out to be spam.

We could also train a model based on a dataset of emails and classifications, to learn patterns of what spam messages look like without explicitly writing spam identification rules. Then, we could use the model for new data without aclassification. That's the approach we're more interested in here, but both approaches are algorithms.

1.4 What is a machine learning algorithm?

The ML Coursera course defines a *Well Posed Machine Learning Problem*

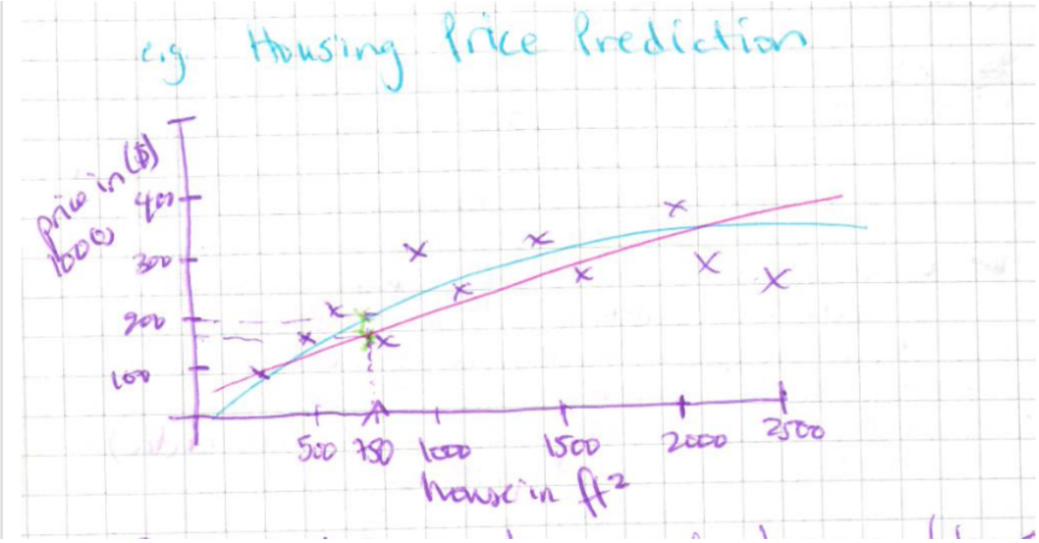
A computer program is said to learn from experience E with respect to some task T and some performance measure P if its performance on T, as measured by P improves with experience E

So, we can see that the rule based spam filtering approach wouldn't be a machine learning based approach because having more labelled data would not help us to classify spam any more accurately.

1.5 Supervised Learning

Here is an example of a supervised learning problem:

Suppose we have a dataset of houses that have sold containing how much they have sold for, and the area of the property in square feet.



Then, we can use this data to build a model so that we can make a prediction like:

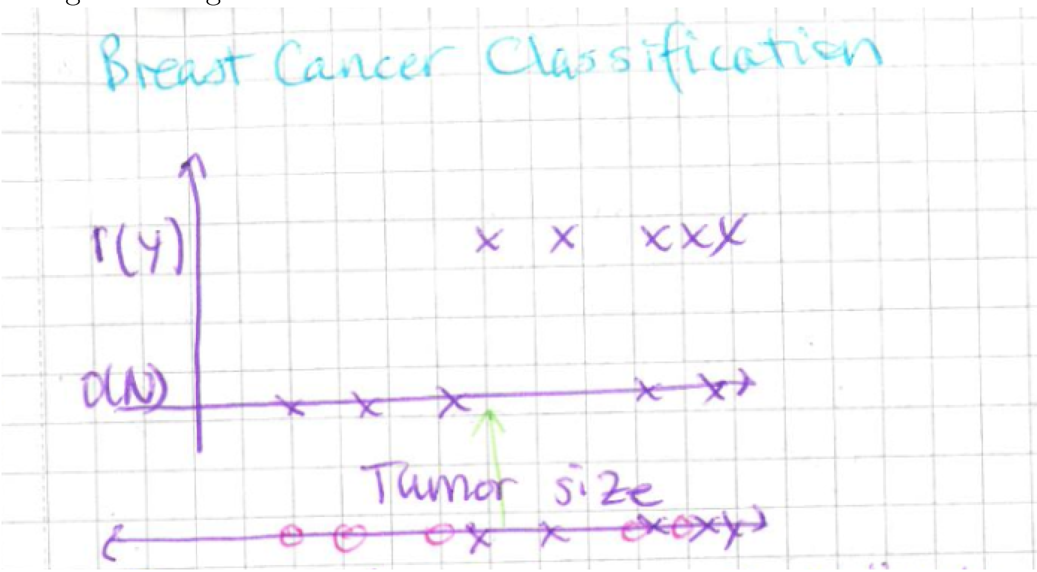
For this new 750 ft^2 property, how much will it sell for?

This is a *Supervised Learning* problem because existing labelled data (with the right answers) to work with and to test how accurate our model is.

This is also a *Linear Regression* problem because the prediction is a continuous valued output.

Another example of a supervised learning problem would be:

Suppose we want to predict whether a tumour is benign or malignant, and we have a dataset that contains tumours, their size and whether they are benign or malignant.



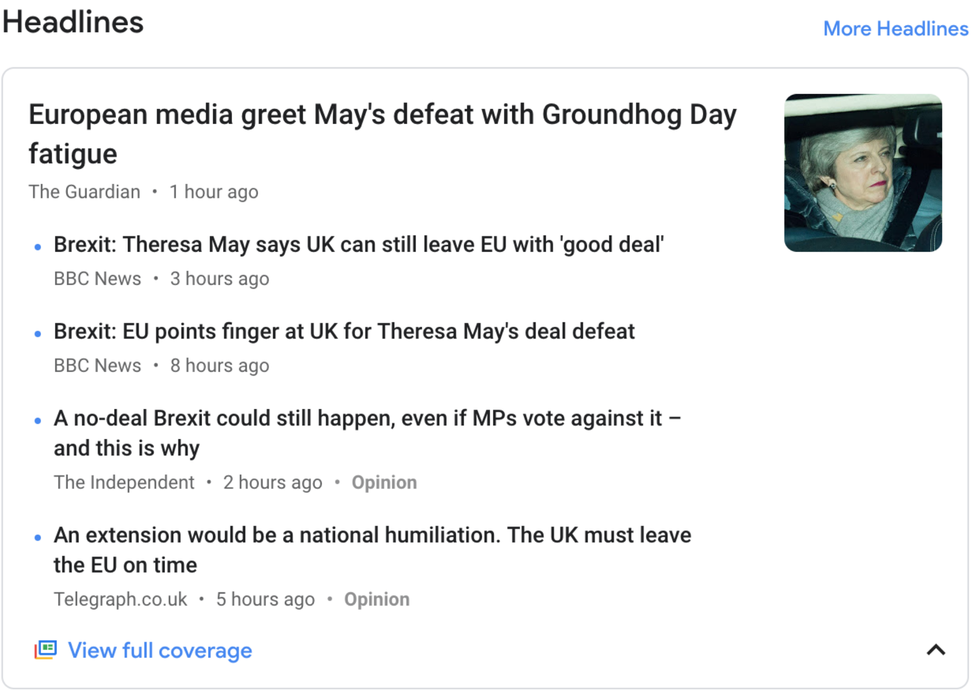
This would be a *classification* problem, because the prediction is a discrete valued output (in this case, either benign or malignant).

These might seem like toy examples so far. Cancer researchers will collect all sorts of data about tumours and patients that could all be useful in our model. But, we will aim to introduce concepts with a simplified view of the data to help our understanding, then move on to being able to use more features/data and more complicated techniques.

1.6 Unsupervised Learning

There is another area of Machine Learning, unsupervised learning that doesn't use labelled data to build a model. We won't cover unsupervised learning here, but let's introduce a couple unsupervised learning problems so that we'll get a feel for what these problems are like, and why they are different from supervised learning.

Here is an example of an unsupervised learning problem:



Headlines

More Headlines

European media greet May's defeat with Groundhog Day fatigue

The Guardian • 1 hour ago

- Brexit: Theresa May says UK can still leave EU with 'good deal'

  BBC News • 3 hours ago

- Brexit: EU points finger at UK for Theresa May's deal defeat

  BBC News • 8 hours ago

- A no-deal Brexit could still happen, even if MPs vote against it – and this is why

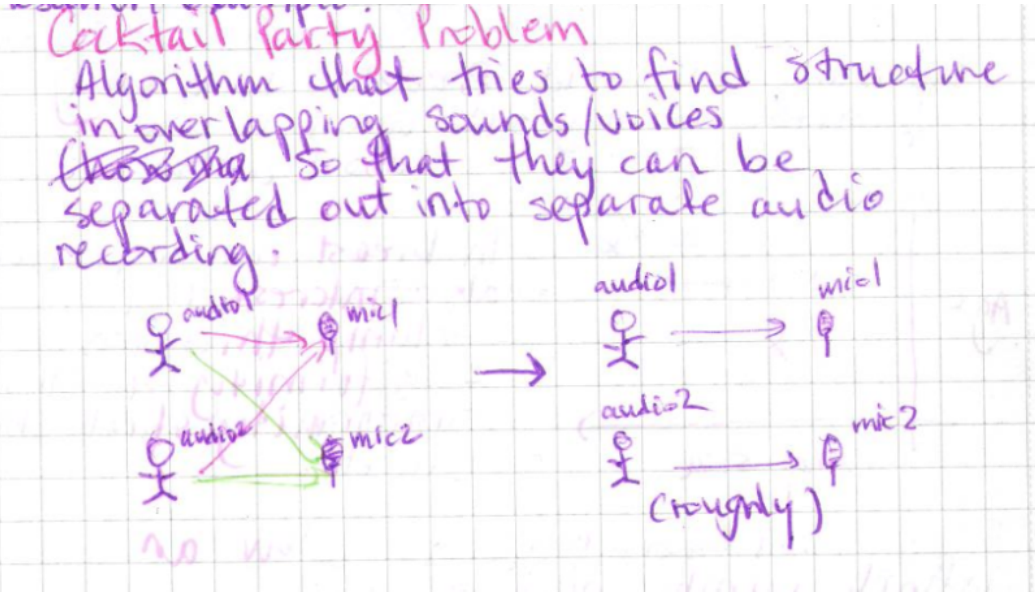
  The Independent • 2 hours ago • Opinion

- An extension would be a national humiliation. The UK must leave the EU on time

  Telegraph.co.uk • 5 hours ago • Opinion

News aggregators don't have a predefined list of topics, but will find an underlying structure in news articles to be able to group them together, as seen above.

Here is another example of an unsupervised learning problem:



Cocktail Party Problem

Algorithm that tries to find structure in overlapping sounds/voices so that they can be separated out into separate audio recording.

Given fixed mics in a room, and conversations happening around them, the mics will collect audio data for all of the conversations mixed together. If you want to be able to follow one of the conversations, an algorithm that identifies the underlying structure to be able to extract just the parts of the audio that belong to one conversation is an unsupervised learning problem.

1.7 Linear Regression

Let's move on to working through our first machine learning problem.

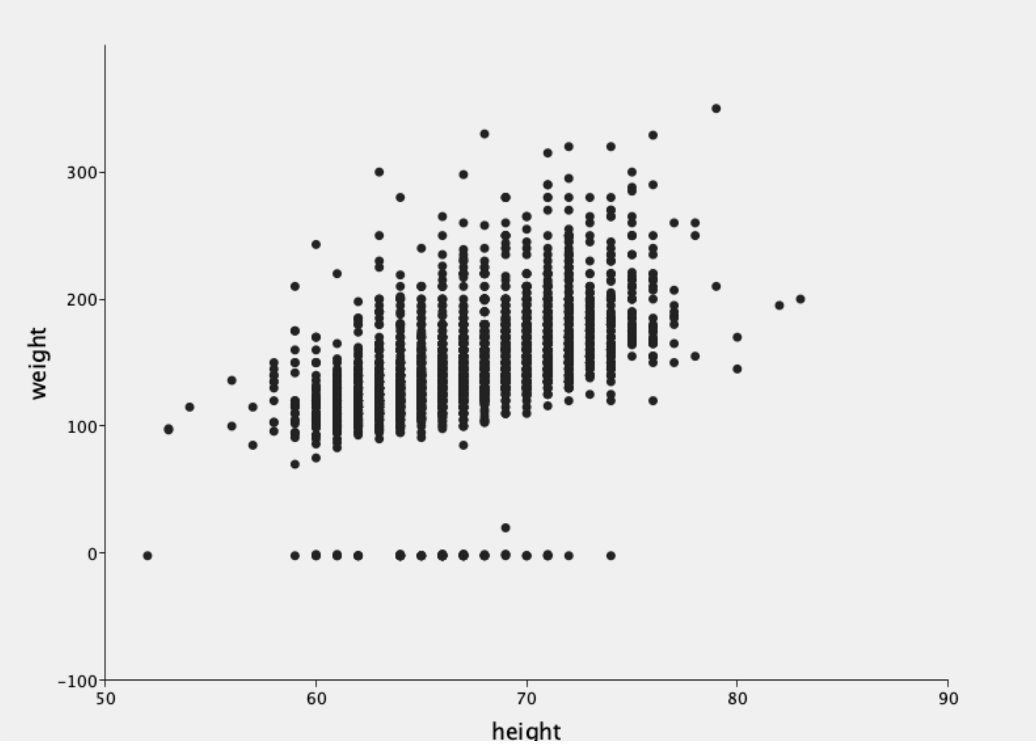
Suppose we want to predict a person's weight, given their height. This isn't something terribly interesting, but hey, it's two variables that are clearly dependent on each other.

Here is a sample from a dataset of heights and weights:

Let's plot height vs weight for the dataset.

Table 1: Sample from a dataset of heights and weights

Age	Height (inches)	Weight (lbs)	How would you describe your weight?
18	67	150	Slightly overweight
17	67	140	About the right weight
16	67	100	Slightly underweight
18	62	185	Slightly overweight
17	62	140	Slightly overweight



It does seem reasonable to presume that height and weight have a linear relationship.

Let's begin setting up our model.

But first:

1.7.1 Training and Test Sets

We can't use all of our data to build our model. We will portion our data into two parts:

1. Training data: data to help build our model

2. Test data: data to test the accuracy of our model, once it's already trained

So now that we have a separate training set, here is some notation for how we will refer to the data:

Table 2: Sample from a dataset of heights and weights

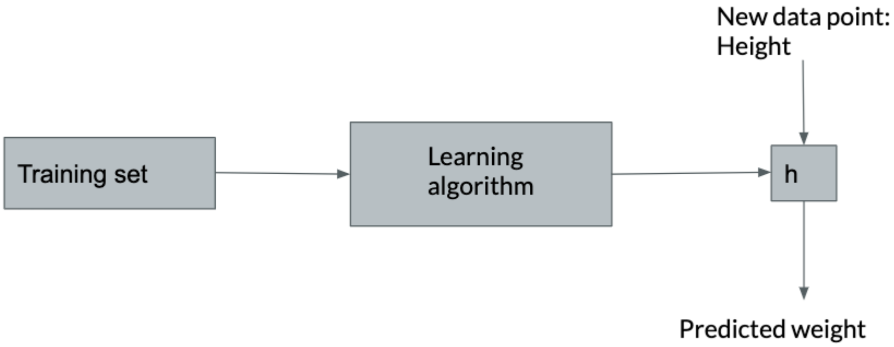
notation	what is it
m	the number of training examples
the xs e.g. x_1	input variables/features
the y	output variables/feature that we are trying to predict
(x_1^i, y^i)	The ith training example

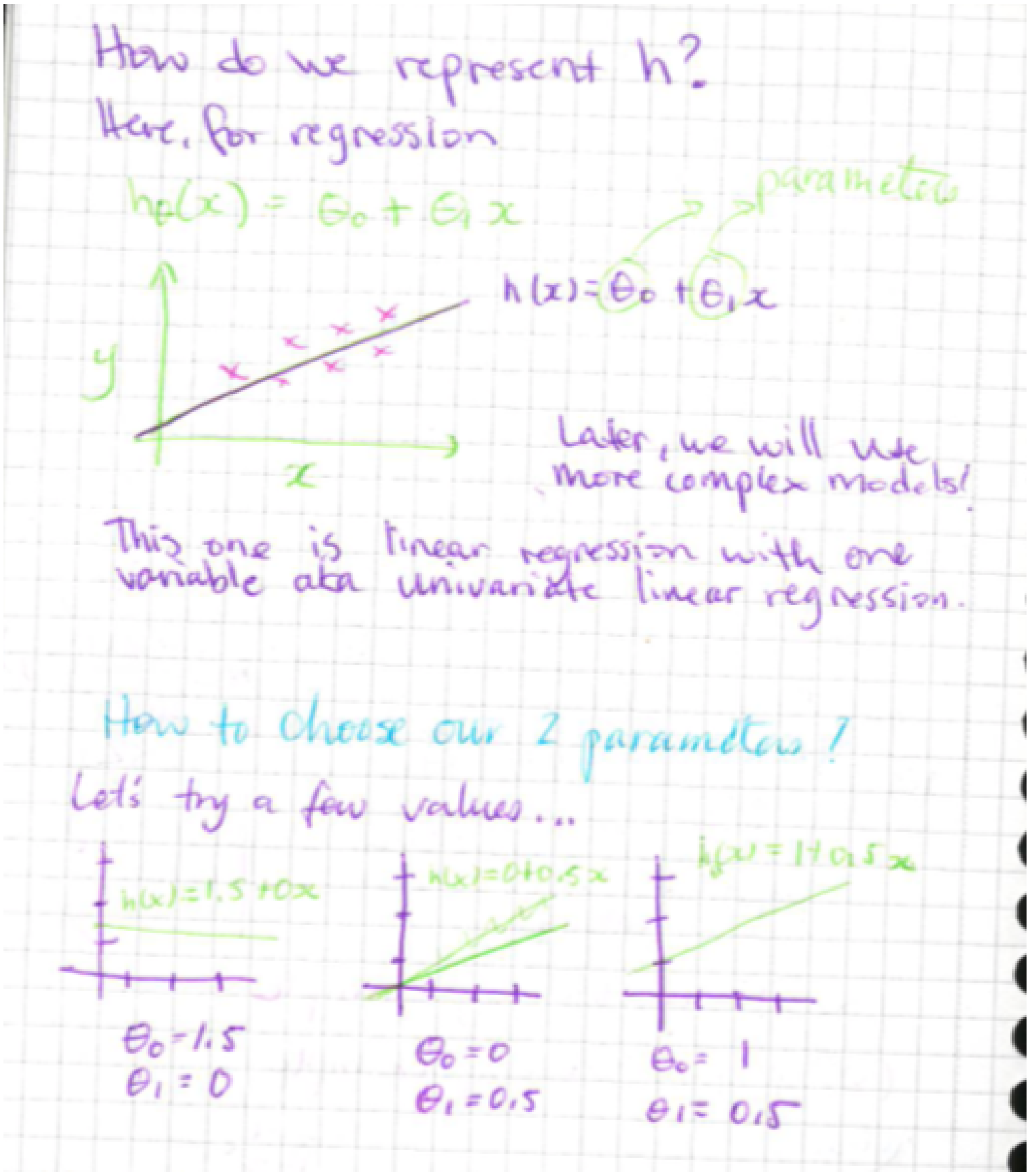
So, in considering our weight prediction problem, weight is y and any other variables in the dataset that we might use to train our model are the xs.

Table 3: Sample from a dataset of heights and weights

x_1	x_2	x_3	y
Age	Height (inches)	How would you describe your weight?	Weight
18	67	Slightly overweight	150
17	67	About the right weight	140
16	67	Slightly underweight	100
18	62	Slightly overweight	185
17	62	Slightly overweight	140

Here, the *learning algorithm* will, learning from our training data, produce a way to map from previously unseen xs to predicted ys.


New data point:
Height
Training set
Learning algorithm
h
Predicted weight



How do we represent h?

Here, for regression

$h_\theta(x) = \theta_0 + \theta_1 x$

parameters

$h(x) = \theta_0 + \theta_1 x$

Later, we will use more complex models!

This one is linear regression with one variable aka univariate linear regression.

How to choose our 2 parameters?

Let's try a few values...

$h(x) = 1.5 + 0x$

$\theta_0 = 1.5$
$\theta_1 = 0$

$h(x) = 0 + 0.5x$

$\theta_0 = 0$
$\theta_1 = 0.5$

$h(x) = 1 + 0.5x$

$\theta_0 = 1$
$\theta_1 = 0.5$

So let's move onto finding h

Remember that we need a way to take previously unseen xs and predict y. Since our weight and height data seem to have a linear relationship, we will do this by fitting a line to our training data. This means that we're going to find

$$h(x) = \theta_0 + \theta_1 x$$

such that when we evaluate some $h(x^{(i)})$, it will be as close to $y^{(i)}$ as possible.

We need a way, more formally, to say how well our $h(x)$ fits our training data. We will compare the y predicted by $h(x)$ for each data point, and sum up the differences with the following function:

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

(Recall: $h_\theta(x) = \theta_0 + \theta_1 x$)

In short, here is our model:

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$ (we think that a line will fit our data)

Parameters: θ_0, θ_1

Cost function:

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Goal: minimise $J(\theta_0, \theta_1)$. Once we have θ_0, θ_1, we can plug them into our hypothesis and make predictions!

1.8 Gradient Descent

We want to find θ_0, θ_1 that will minimise our cost function

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

.

We will use gradient descent to do that. Here is the idea: Imagine that you are in a hilly landscape blindfolded, and you want to get to the lowest elevation around. What would you do? You'd probably tap around your immediate area, and then move to the lowest elevation that you found. Then you'd repeat until you tapped around your immediate area and found that the ground around you to all be higher or on the same level. That is what gradient descent does.

Our plan:

- Pick some θ_0, θ_1 (Perhaps $\theta_0 = 1, \theta_1 = 1$, but it doesn't really matter).
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$, until we have found a minimum

More formally:
set some initial values for θ_0, θ_1
repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{\delta}{\delta \theta_0} J(\theta_0, \theta_1)$$

$$\theta_1 := \theta_1 - \alpha \frac{\delta}{\delta \theta_1} J(\theta_0, \theta_1)$$

}
We will update θ_0, θ_1 simultaneously.

α is the learning rate, and it represents the size of the step you take in each iteration. More on that later. $\frac{\delta}{\delta \theta_0} J(\theta_0, \theta_1)$ and $\frac{\delta}{\delta \theta_1} J(\theta_0, \theta_1)$ are partial derivatives for the cost function, and they represent the direction we need to go if we want $J(\theta_0, \theta_1)$ to decrease in the next iteration. Here is the algorithm with the partial derivatives for our cost function subbed in:
set some initial values for θ_0, θ_1
repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \times x^{(i)}$$

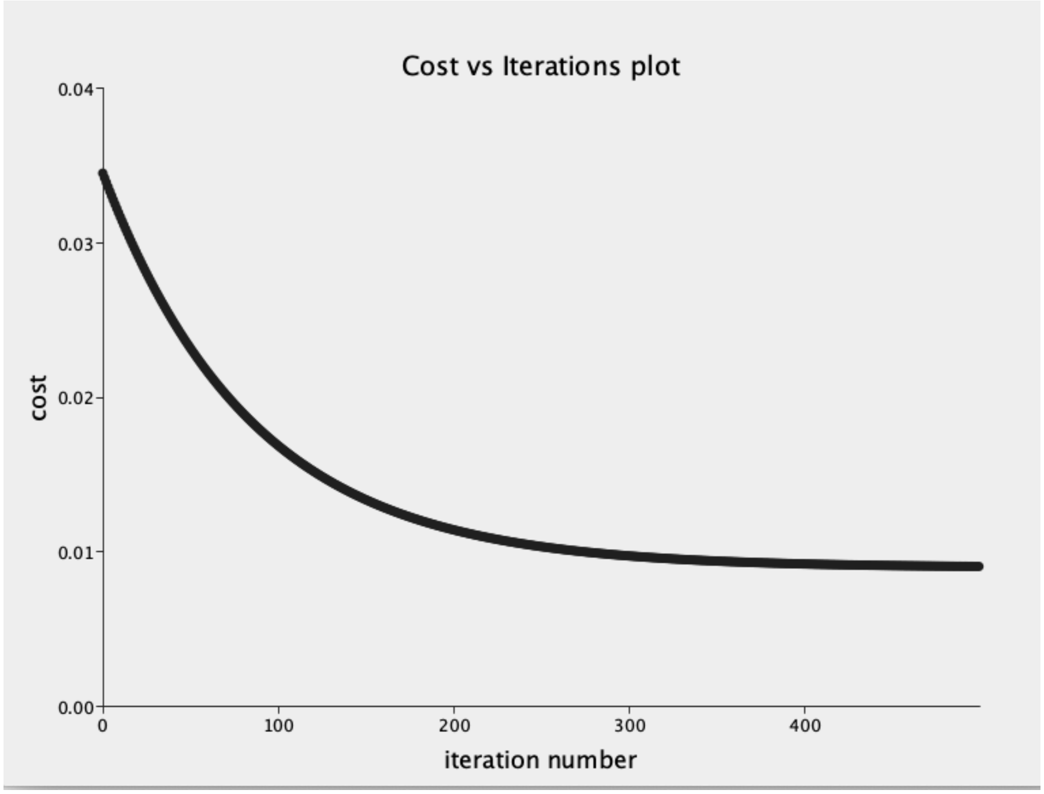
}

1.8.1 Gradient Descent - how do we detect convergence?

The number of iterations we need will vary, but note that:

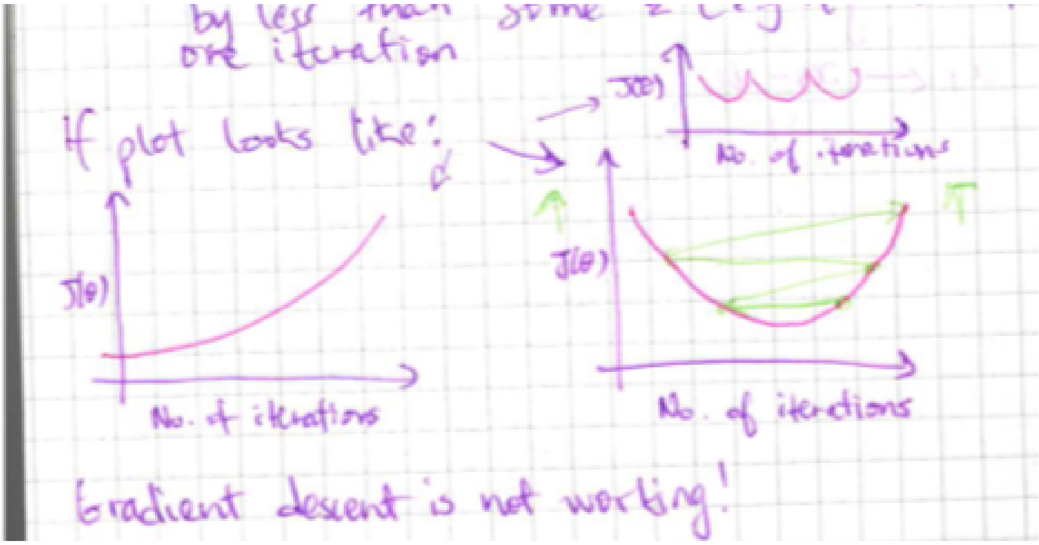
- For sufficiently small α, $J(\theta)$ should decrease on every iteration
- But if α is too small, gradient descent is *slow.* It can also encourage getting stuck in a local minima.

The best way to know when we can stop iterating is to keep track of $J(\theta)$ on each iteration, and then plot it like the following:



We can tell gradient descent is working if it decreases on each iteration, and at the point where the cost seems to level off is when we can stop iterating.

You can also use these plots to spot if something has gone wrong.



This usually means that our α is too big. We are probably overshooting the minimum.

1.8.2 Feature scaling

Imagine we are interested in predicting housing prices and we have a feature x_1 which corresponds to the number of bedrooms, and another x_2 which is square feet. The range of x_1 could be 0-5, and x_2 could be in the thousands. We can speed up gradient descent by ensuring that our data falls into similar ranges.

One common technique is *mean normalisation*

For some feature x_i, we find the mean μ_i and the range s_i (max value - min value). Then for each point j in x_i:

$$x_i^{(j)} \leftarrow \frac{x_i^{(j)} - \mu_i^{(j)}}{s_i^{(j)}}$$

and then for each $x_i^{(j)}$, $-0.5 \leq x_i^{(j)} \leq 0.5$