✏️

Multiple Linear Regression

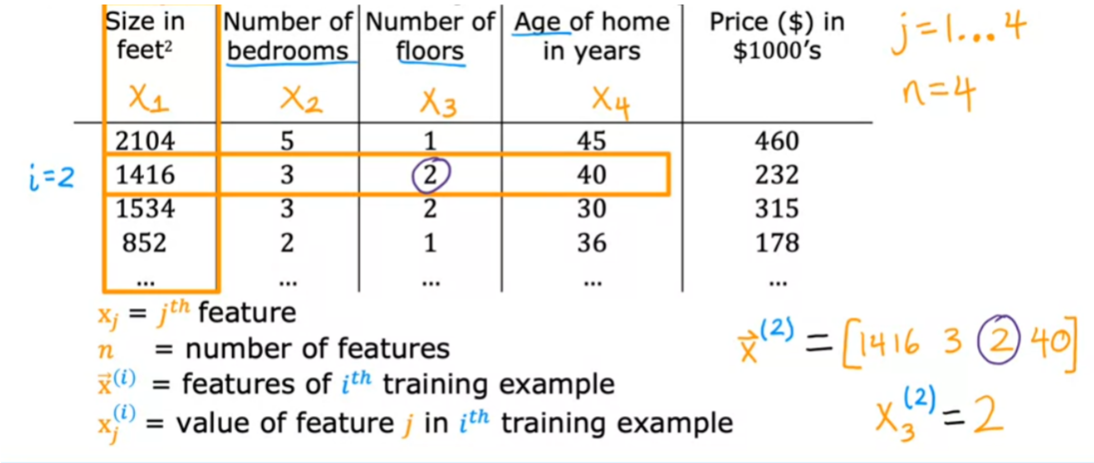
Course	
Topic	
Type	Lecture
Done	☑
Date	@July 17, 2022
Materials	https://drive.google.com/file/d/1nJT4W5YT7dK_6JI6uvikMsLkWsy_OK6z/view?usp=sharing

▼ Summary

Introduction:

- In Multiple Linear Regression**, the output variable 'y' is dependent** on more than 1 input variable/feature.

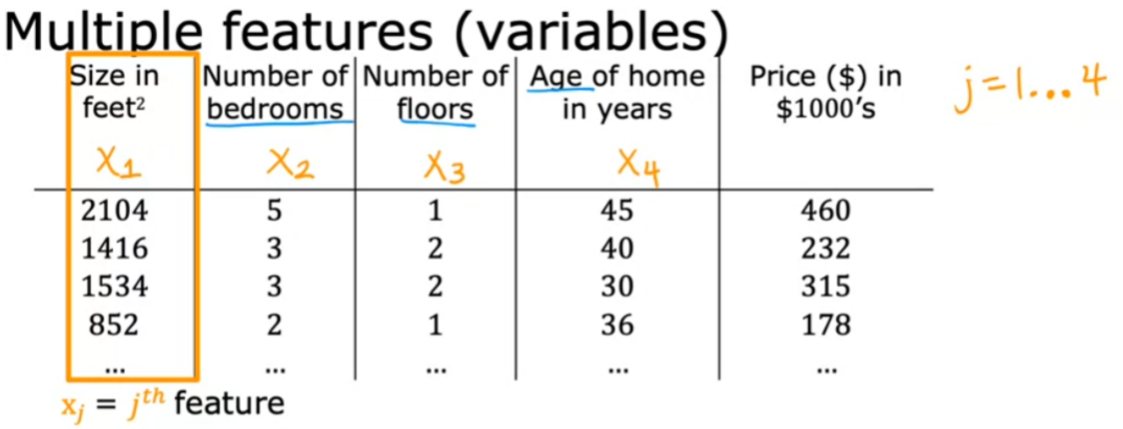
For example, what if you had the size of the house, the number of bedrooms, the number of floors and the age of the home in years to be able to predict the house price? It will give us a lot of new information to work with.



Size in feet² x_1	Number of bedrooms x_2	Number of floors x_3	Age of home in years x_4	Price ($) in $1000's
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

Figure 1: A simple example of multiple linear regression.

- We're going to use the variables X_1, X_2, X_3 and X_4, to denote the four features. Thus for any feature j, X_j is used to denote the list of features. For example, X_1 represents size in feet.



Multiple features (variables)

Size in feet² x_1	Number of bedrooms x_2	Number of floors x_3	Age of home in years x_4	Price ($) in $1000's
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

Figure 2: The yellow rectangle(size of feet) represents X_1 feature

- $\vec{x}^{(i)}$ is used to denote all the features of the i^{th}training example. For example, column 2 [1416 3 2 40] is represented by $\vec{x}^{(2)} = [1416 \ 3 \ 2 \ 40]$
- To refer to a specific feature in the i^{th} training example, we will write $\vec{X}_j{}^{(i)}$, so for example, $\vec{X}_3^{(2)}$ will be the number of floors in the second training example. So, $\vec{X}_3^{(2)}$= 2.

Representing a multiple linear regression model

- Previously, for linear regression in one variable, the predicted value of a input feature was:

$$f_{w,b}(x) = wx + b$$

Now for multiple linear regression,

$$\mathbf{f_{w,b}(x) = w_1x_1 + w_2x_2 + w_3x_3 + \dots\dots\dots\dots\dots\dots + w_nx_n + b}$$

- We can represent $w_1, w_2, w_3 \dots\dots\dots w_n$ as a single vector $\vec{w}$:

  $\vec{w} = \begin{bmatrix} w1 & w2 & w3 & w4 & \dots\dots\dots\dots & w_n \end{bmatrix}$

- We can represent $x_1, x_2, x_3 \dots\dots\dots x_n$(the features of model like size of house, no of bathrooms etc) as a single vector $\vec{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & \dots\dots\dots\dots & x_n \end{bmatrix}$

- $w_1x_1 + w_2x_2 + w_3x_3 + \dots\dots\dots\dots\dots + w_nx_n$ can be represented by the dot product of $\vec{w}$ and $\vec{x}$ i.e $\vec{w} \cdot \vec{x}$

Therefore,

$$\mathbf{f_{w,b}(\vec{x}) = \vec{w} \cdot \vec{x} + b}$$

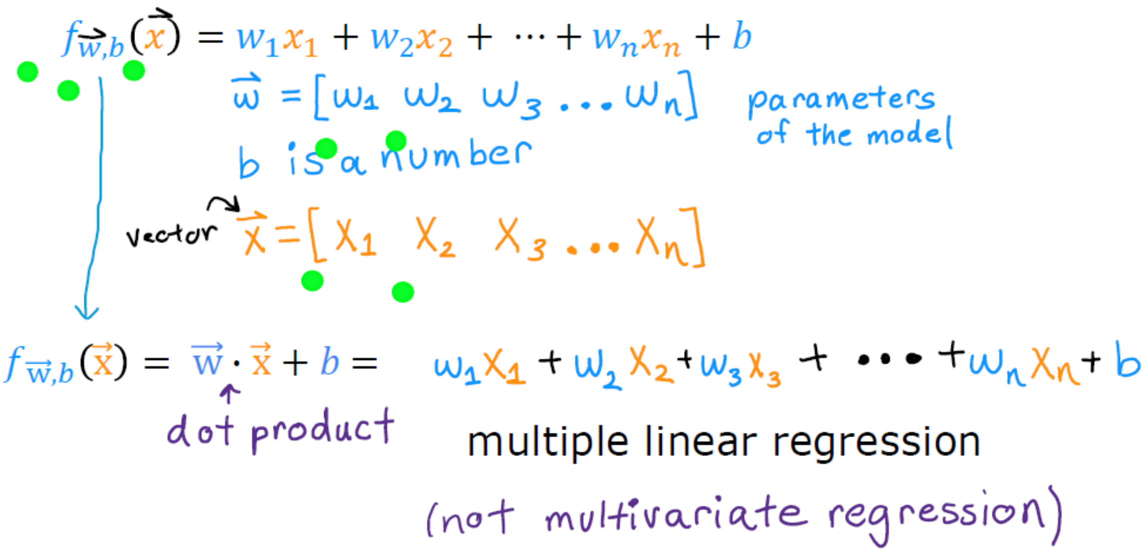


Figure 3: Representing a multiple linear regression model

Vectorization

- Using vectorization will both make our code shorter and also make it run much more efficiently.

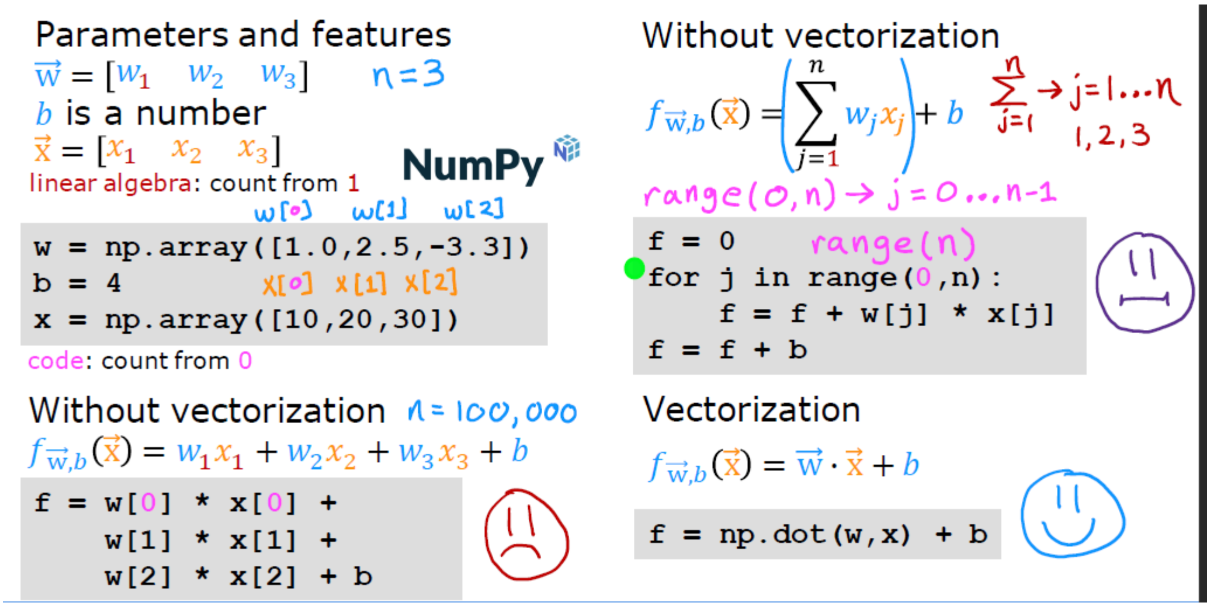
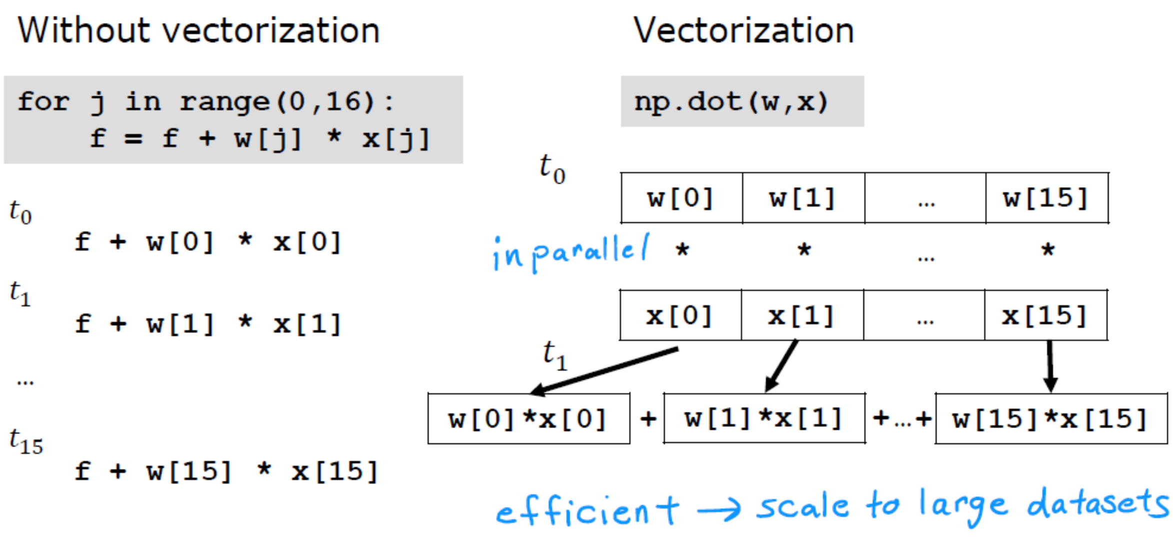


Figure 5: Vectorization is faster than normal calculations

Why is Vectorization faster?

- Without vectorization,
- If j ranges from 0 to say 15,code performs operations one after another for 15 times.
- On the first timestamp t_0, it first operates on the values at index 0.
- At the next time-step, it calculates values corresponding to index 1 and so on until the 15th step, where it computes that.
- In other words, it calculates these computations one step at a time, one step after another.

- With vectorization, In contrast, np.dot() is implemented in the computer hardware with vectorization.
- The computer can get all values of the vectors w and x, and in a single-step, it multiplies each pair of w and x with each other all at the same time in parallel.
- Then after that, the computer takes these 16 numbers and uses specialized hardware to add them altogether very efficiently, rather than needing to carry out distinct additions one after another to add up these 16 numbers. This is all done in one step.-
- This means that codes with vectorization can perform calculations in much less time than codes without vectorization. This matters more when you're running algorithms on large data sets or trying to train large models, which is often the case with machine learning. That's why being able to vectorize implementations of learning algorithms, has been a key step to getting learning algorithms to run efficiently, and therefore scale well to large datasets that many modern machine learning algorithms now have to operate on



Fugure 6: Explaning why vectorization is faster

Gradient Descent for multiple linear regression

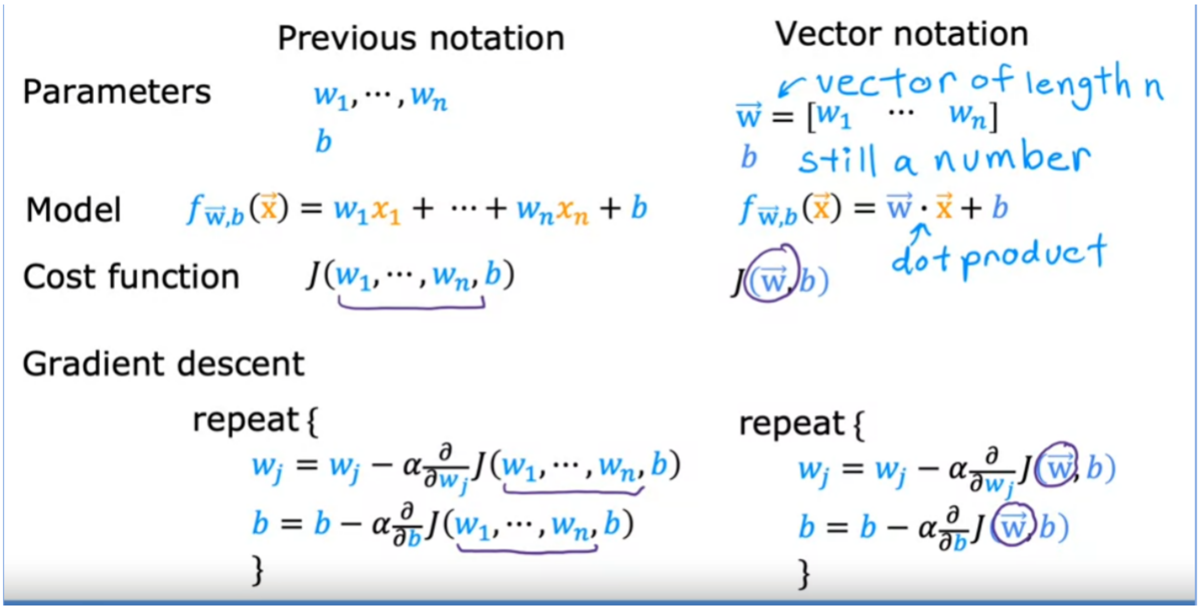


Figure 7: How gradient descent changes for multiple linear regression

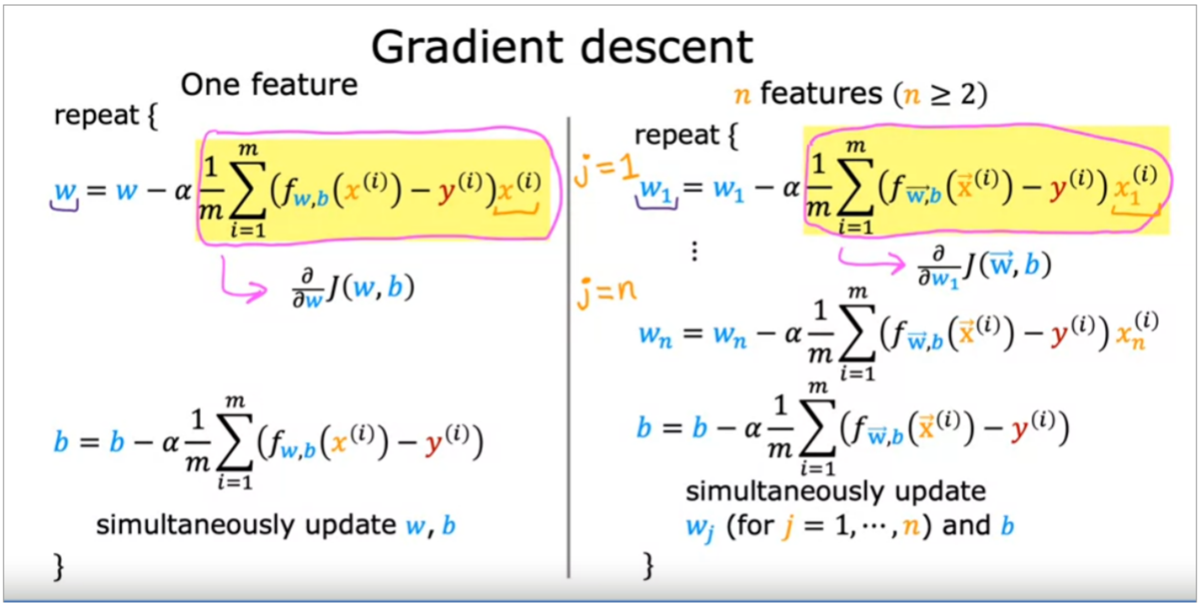


Figure 8: How gradient descent changes for multiple linear regression

Alternative to Gradient Descent

Normal equation

- An alternative way for finding w and b for linear regression is the normal equation
- Only for linear regression
- Solve for w and b all in one goal without iterations.
- Normal equation method may be used in machine learning libraries that implement linear regression.

Disadvantages

- Doesn't generalize to other learning algorithms.
- Slow when number of features is large (> 10,000)

💡 **Gradient descent is the recommended method for finding parameters w,b.**