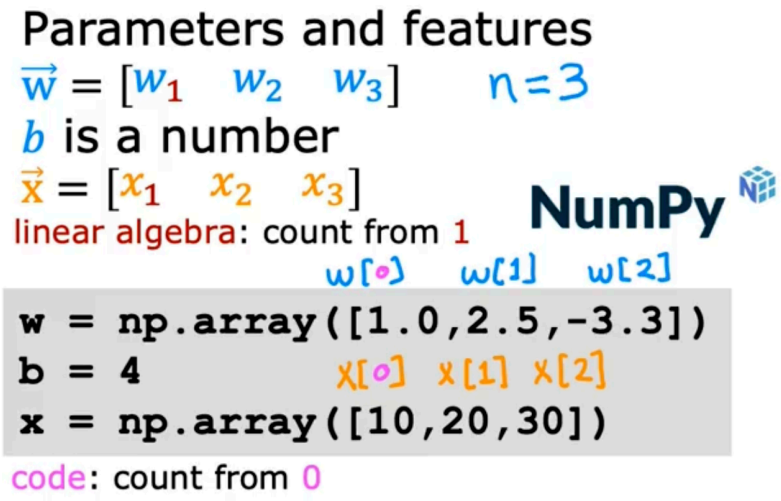
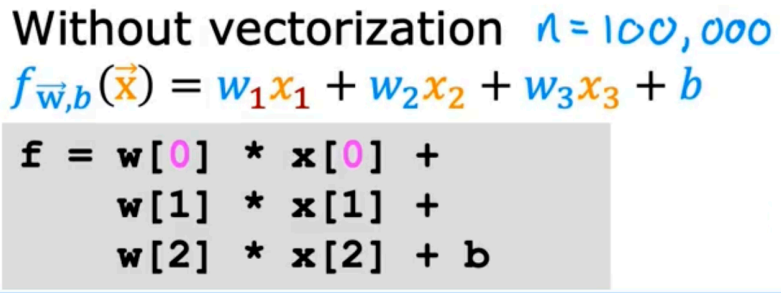
B. Vectorization

Friday, 5 September 2025 12:11 PM

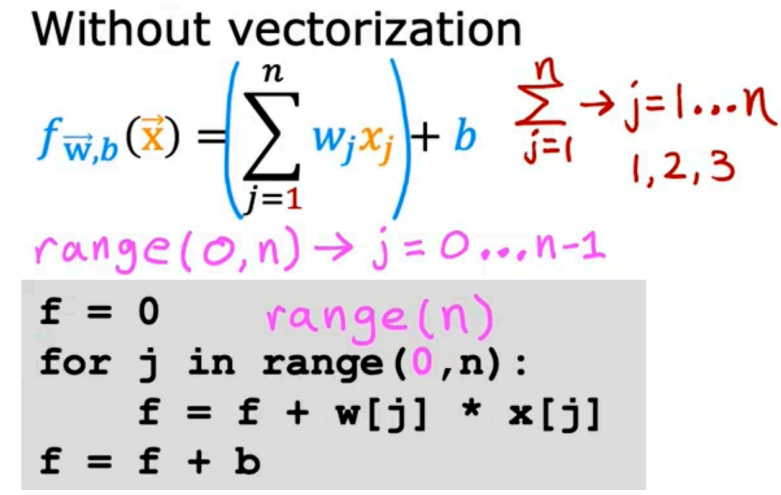
1. **Introduction:**
   When implementing a learning algorithm, using vectorization makes code shorter and also makes it run much more efficiently. It will allow to make use of modern numerical linear algebra libraries and as well as GPU, since it turns out, it can help execute code much more quickly.



Let's see what would our sample code look like without using vectorization,



But what if we had a large set of features and weights, we could use a for loop:



Let's see how we can do this using vectorization,

Vectorization

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

```
f = np.dot(w,x) + b
```

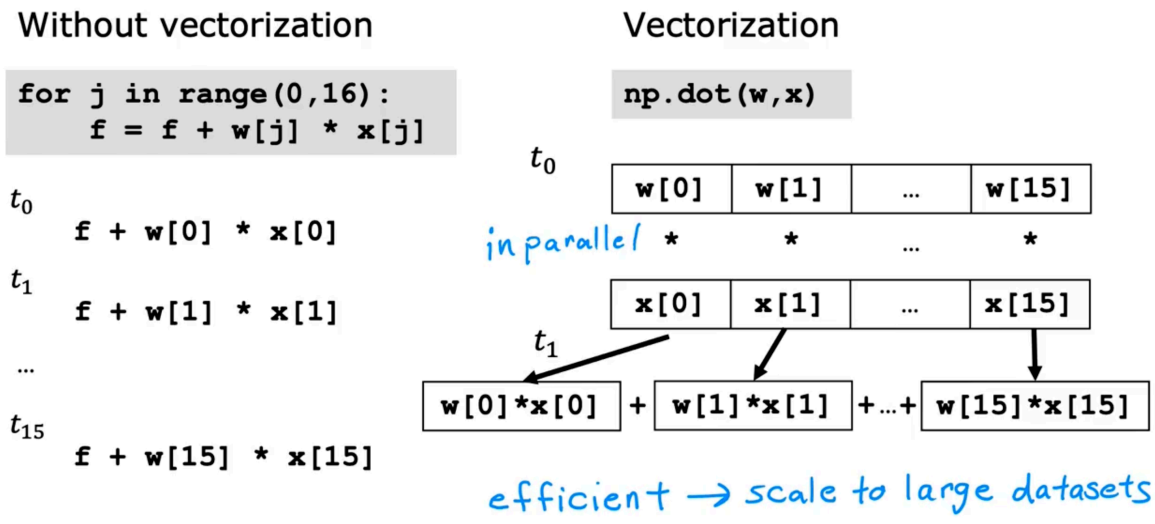
With this one line of code, the y value will be computed a lot faster for a very large n.

Vectorization has two distinct benefits:
a. It makes the code shorter
b. It runs the code faster

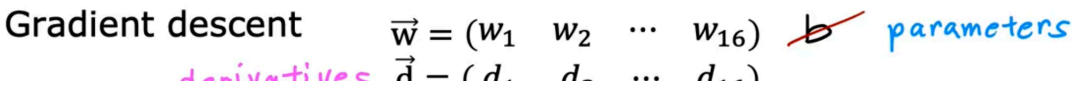
The reason the compute here is fast with Vectorization is because, the NumPy dot function is able to use parallel hardware in your computer. This is true for normal PCs or PC's with GPU as well.

B. Analysis

Let's take a look how vectorization implementation works on the computer behind the scenes.



Now, let's look at how vectorization helps in implementing Multiple Linear Regression



derivatives $d = (d_1 \quad d_2 \quad \ldots \quad d_{16})$

```
w = np.array([0.5, 1.3, … 3.4])
d = np.array([0.3, 0.2, … 0.4])
```

learning rate α

compute $w_j = w_j - 0.1d_j$ for $j = 1 \ldots 16$

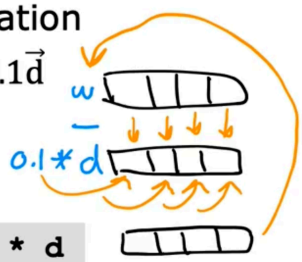
Without vectorization

$$w_1 = w_1 - 0.1d_1$$
$$w_2 = w_2 - 0.1d_2$$
$$\vdots$$
$$w_{16} = w_{16} - 0.1d_{16}$$

```
for j in range(0,16):
    w[j] = w[j] - 0.1 * d[j]
```

With vectorization

$$\vec{w} = \vec{w} - 0.1\vec{d}$$




```
w = w - 0.1 * d
```