

## Without vectorization

```
for j in range(0,16):
    f = f + w[j] * x[j]
```

$t_0$   $f + w[0] * x[0]$

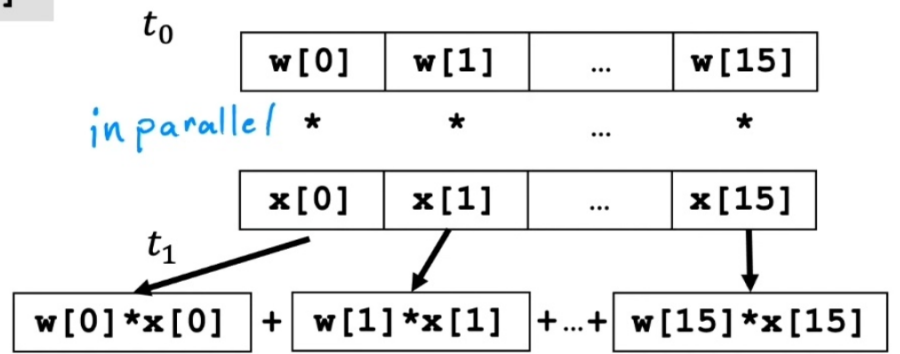
$t_1$   $f + w[1] * x[1]$

...

$t_{15}$   $f + w[15] * x[15]$

## Vectorization

```
np.dot(w,x)
```



efficient → scale to large datasets

→ It is the difference between vector.  
vs without vectorization computation.

→ Vectorization works in parallel  
so, it takes less time to compute.

Gradient descent  $\vec{w} = (w_1 \ w_2 \ \dots \ w_{16})$  ~~b~~ parameters

derivatives  $\vec{d} = (d_1 \ d_2 \ \dots \ d_{16})$

```
w = np.array([0.5, 1.3, ... 3.4])
```

```
d = np.array([0.3, 0.2, ... 0.4])
```

compute  $w_j = w_j - \underbrace{0.1}_{\text{learning rate } \alpha} d_j$  for  $j = 1 \dots 16$

### Without vectorization

$w_1 = w_1 - 0.1d_1$

$w_2 = w_2 - 0.1d_2$

$\vdots$

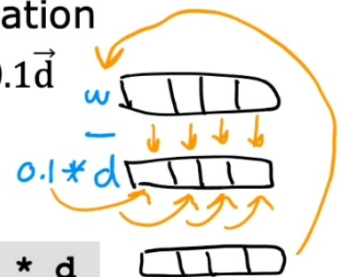
$w_{16} = w_{16} - 0.1d_{16}$

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

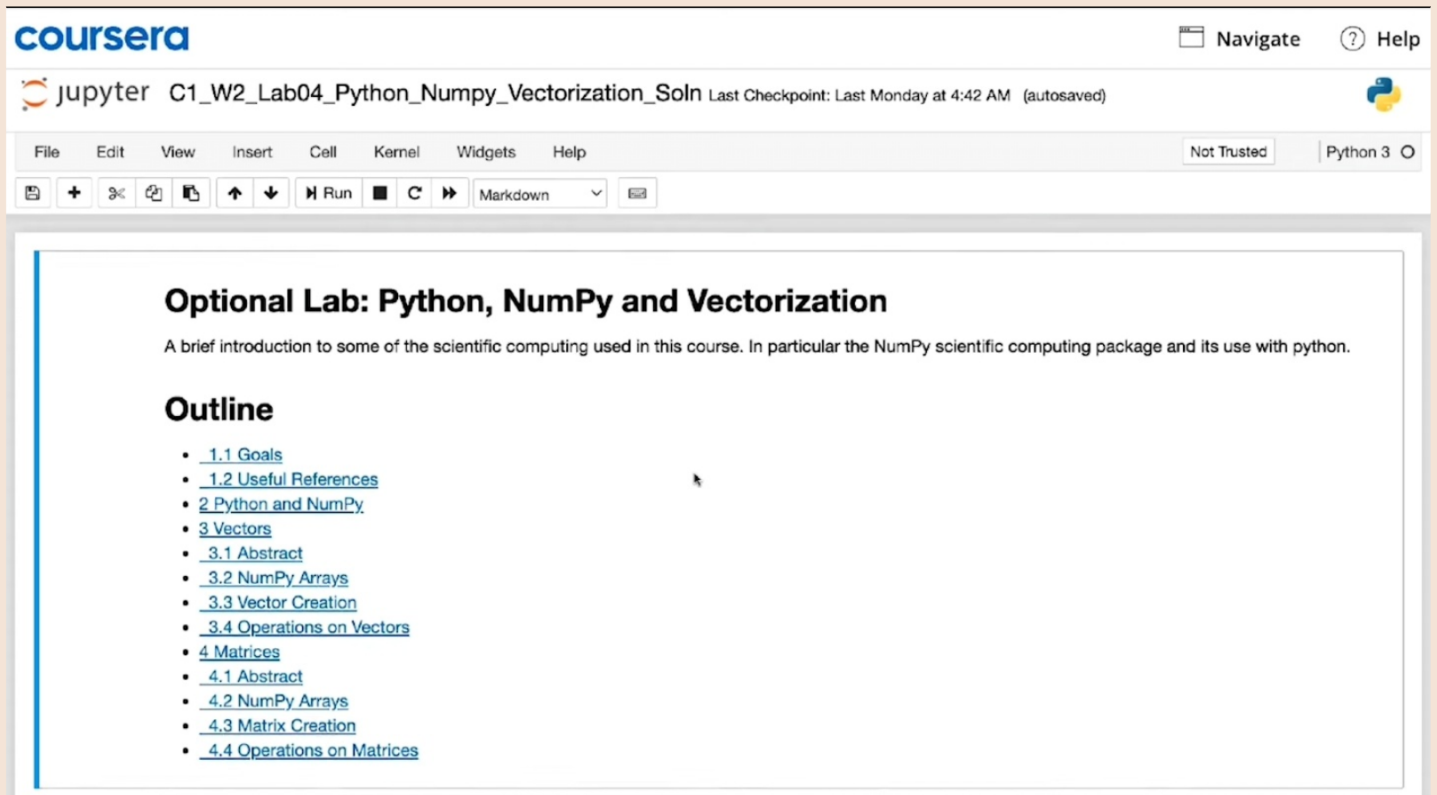
### With vectorization

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

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



→ Here how vectorization takes less time working in parallel in the gradient descent algorithm.



The screenshot shows a JupyterLab interface within a Coursera browser window. The top navigation bar includes the Coursera logo, a 'Navigate' button, and a 'Help' button. Below this, the JupyterLab header displays the file name 'C1\_W2\_Lab04\_Python\_Numpy\_Vectorization\_Soln', the last checkpoint time 'Last Monday at 4:42 AM', and an 'autosaved' status. The main menu bar contains 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. On the right side of the menu bar, there are buttons for 'Not Trusted' and 'Python 3'. The toolbar below the menu bar includes icons for file operations (new, open, save, print), navigation (up, down), execution (run, interrupt, restart), and a 'Markdown' dropdown menu. The main content area displays the title 'Optional Lab: Python, NumPy and Vectorization' followed by a brief introduction: 'A brief introduction to some of the scientific computing used in this course. In particular the NumPy scientific computing package and its use with python.' Below the introduction is an 'Outline' section with a list of links: '1.1 Goals', '1.2 Useful References', '2 Python and NumPy', '3 Vectors', '3.1 Abstract', '3.2 NumPy Arrays', '3.3 Vector Creation', '3.4 Operations on Vectors', '4 Matrices', '4.1 Abstract', '4.2 NumPy Arrays', '4.3 Matrix Creation', and '4.4 Operations on Matrices'.

**Optional Lab: Python, NumPy and Vectorization**

A brief introduction to some of the scientific computing used in this course. In particular the NumPy scientific computing package and its use with python.

**Outline**

- [1.1 Goals](#)
- [1.2 Useful References](#)
- [2 Python and NumPy](#)
- [3 Vectors](#)
- [3.1 Abstract](#)
- [3.2 NumPy Arrays](#)
- [3.3 Vector Creation](#)
- [3.4 Operations on Vectors](#)
- [4 Matrices](#)
- [4.1 Abstract](#)
- [4.2 NumPy Arrays](#)
- [4.3 Matrix Creation](#)
- [4.4 Operations on Matrices](#)

→ optional lab numpy vectorization practice.