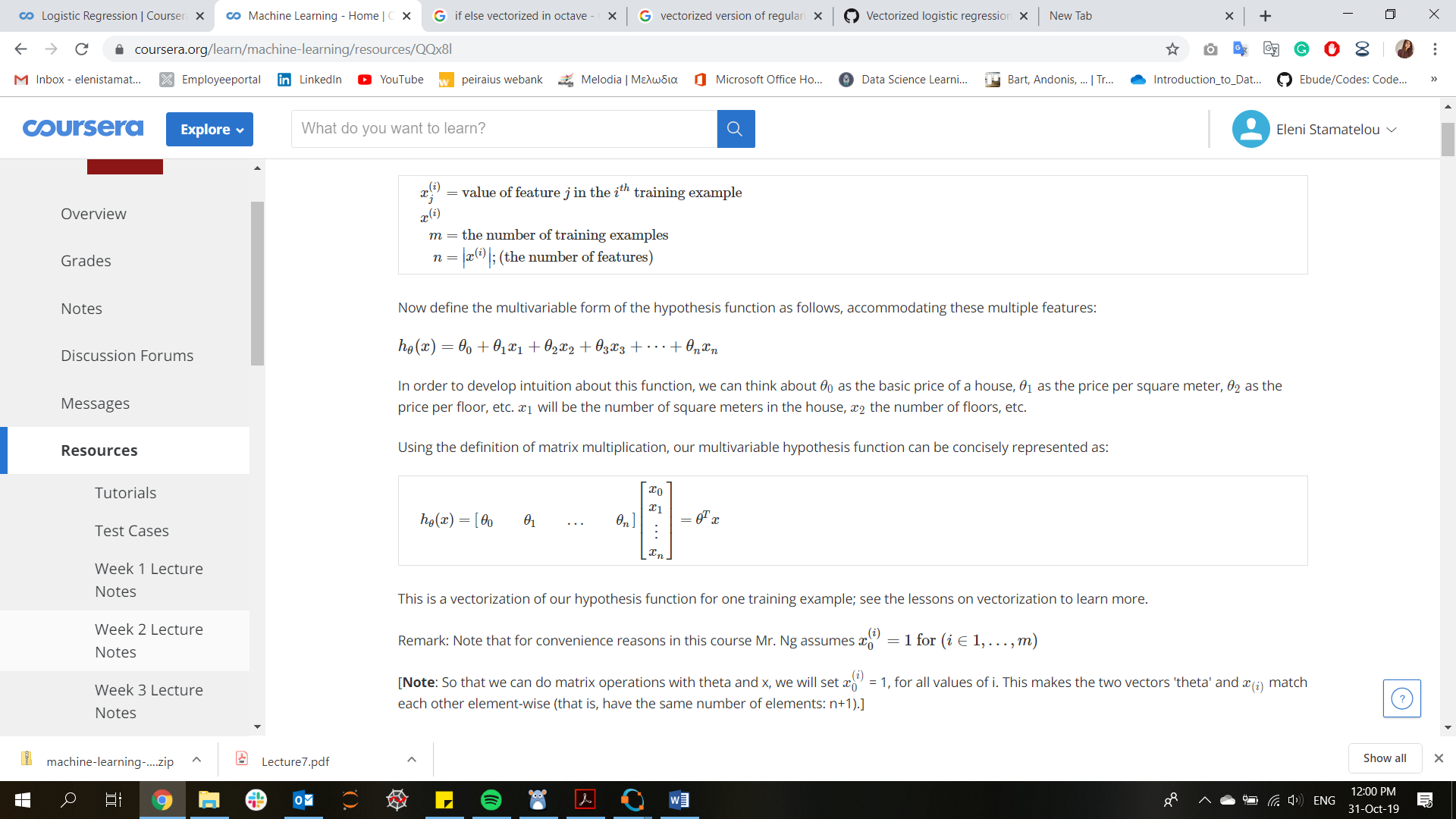
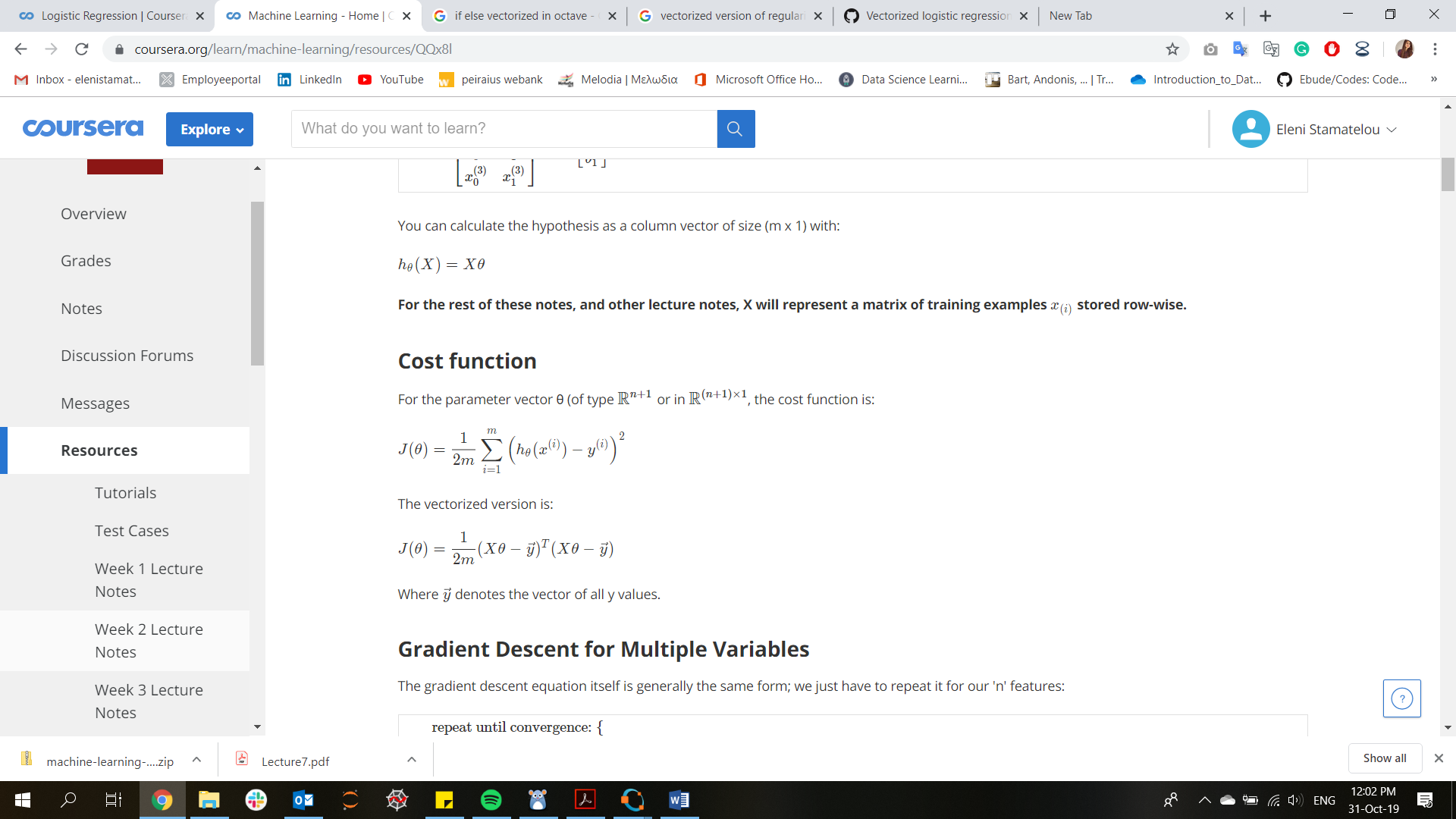
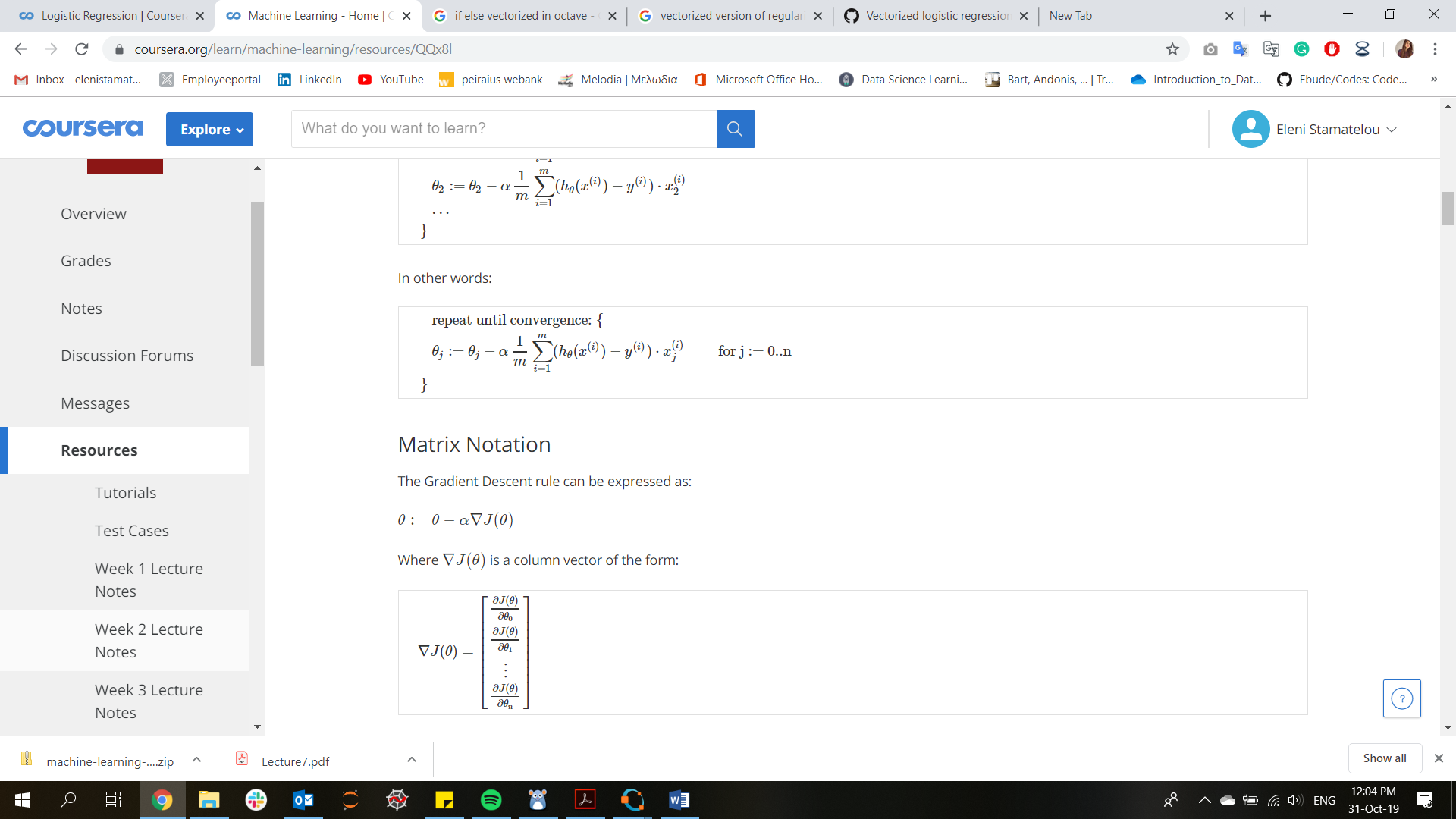
**Linear Regression**

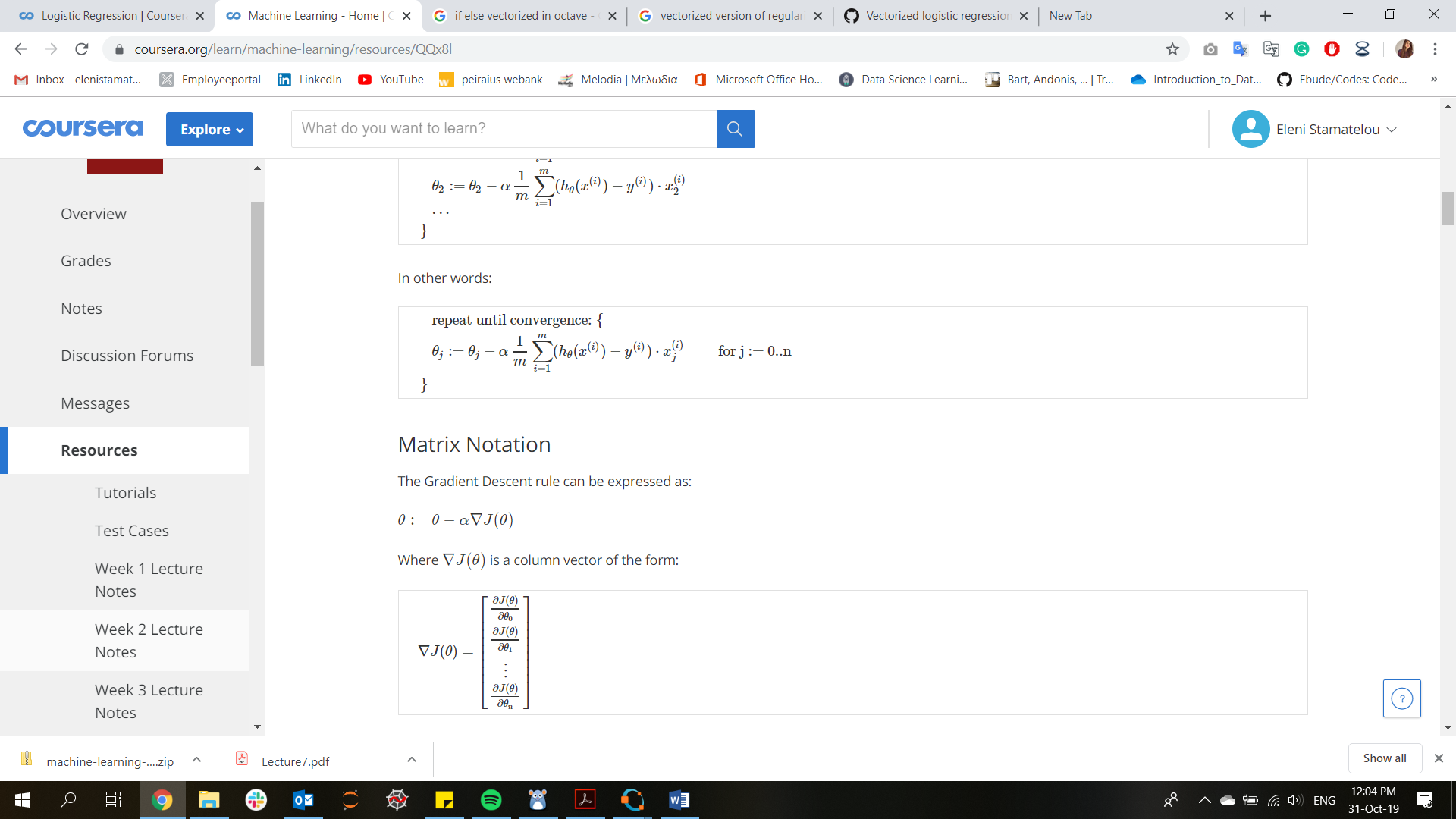
Hypothesis function:

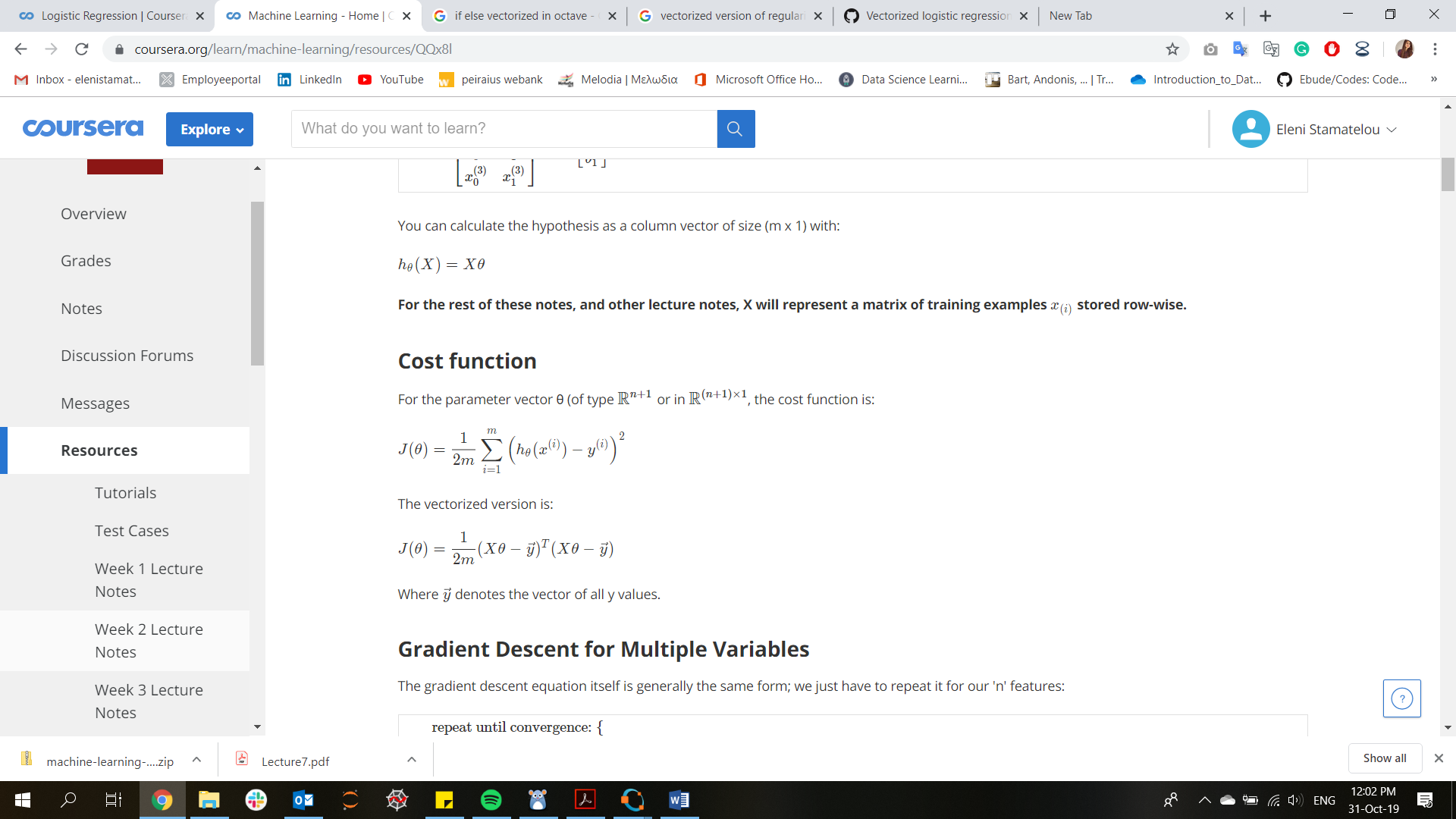
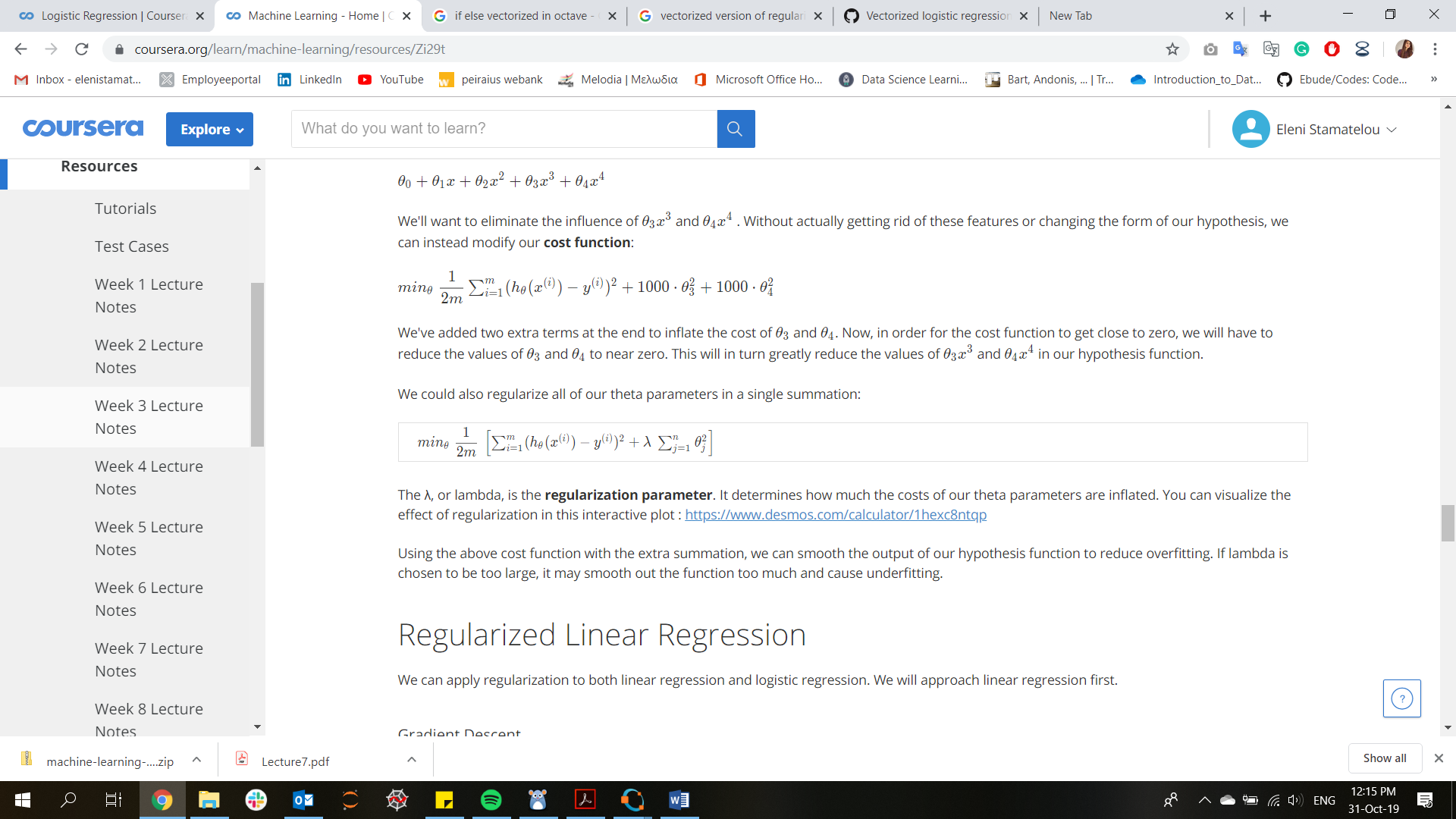


Cost function:

Gradient Descent:

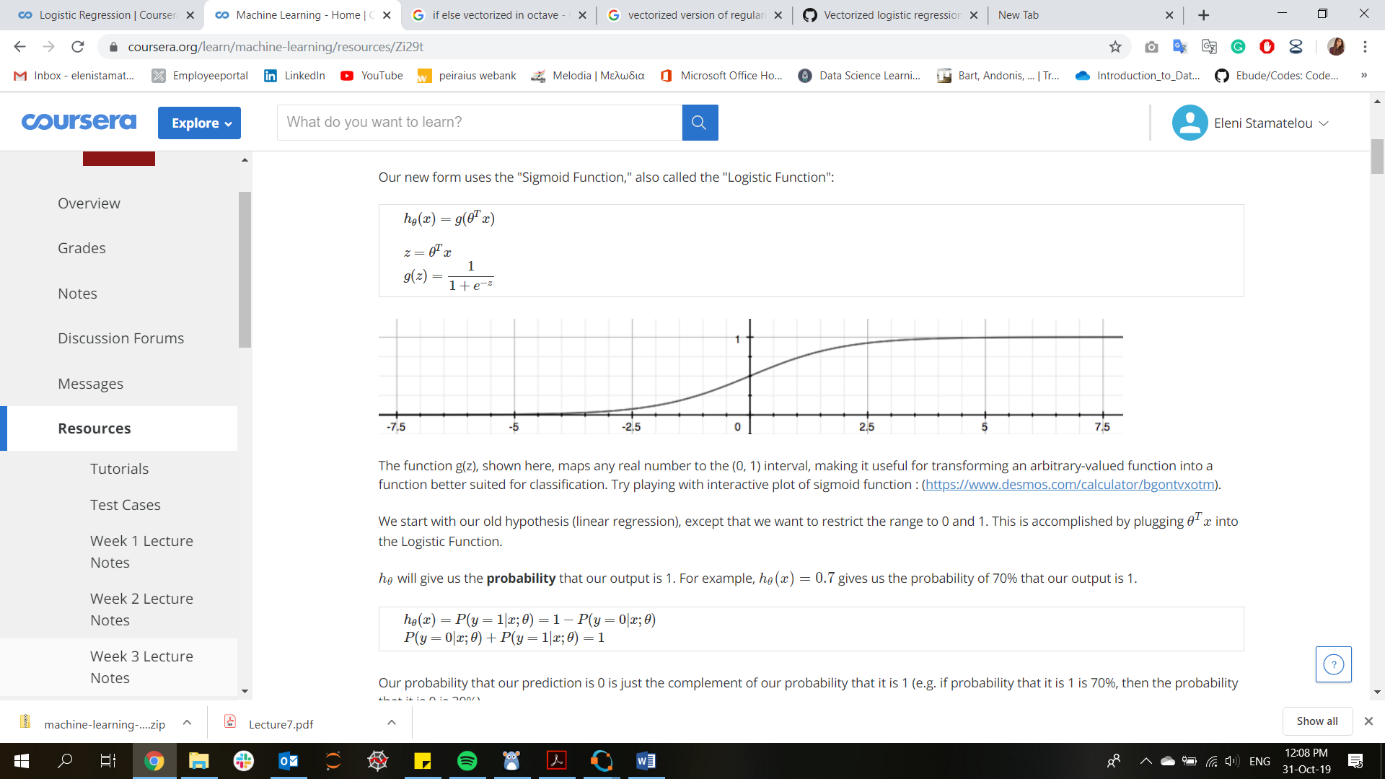
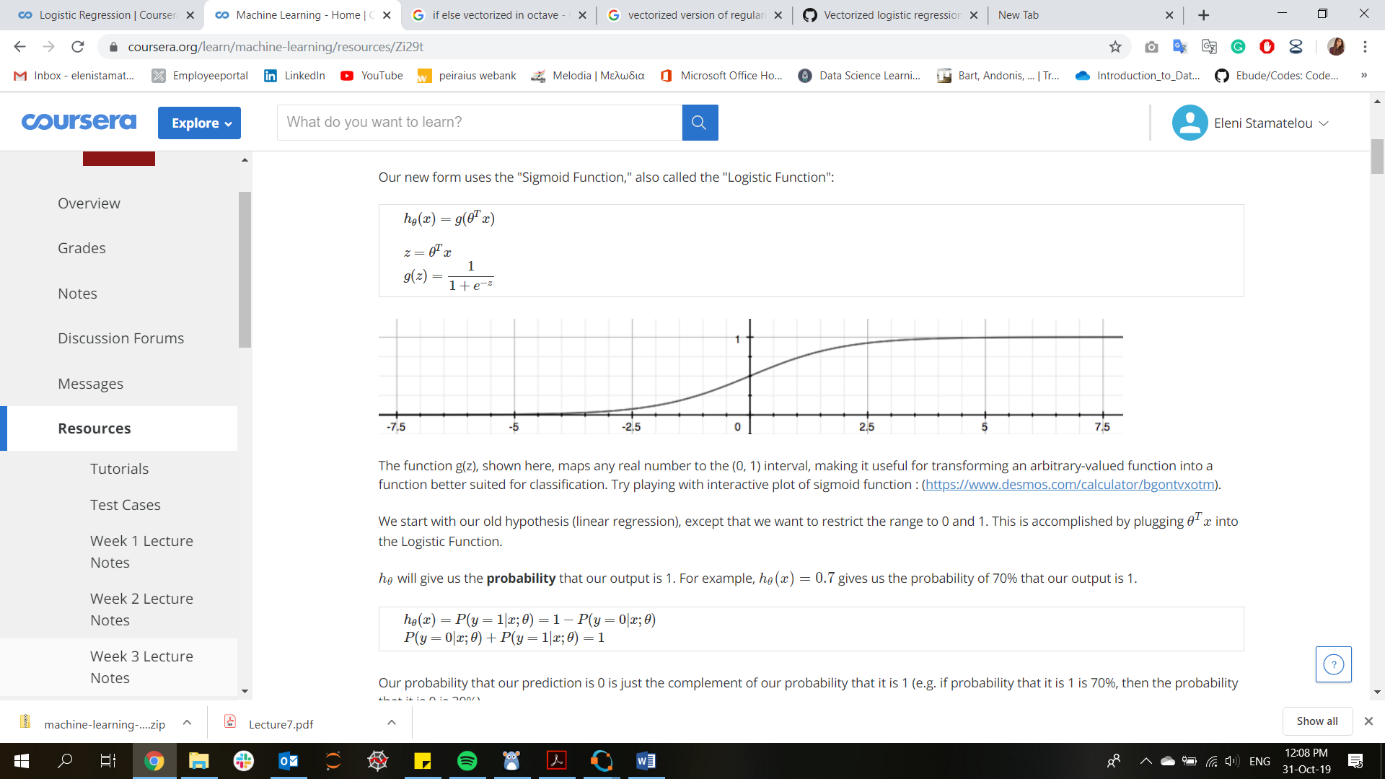


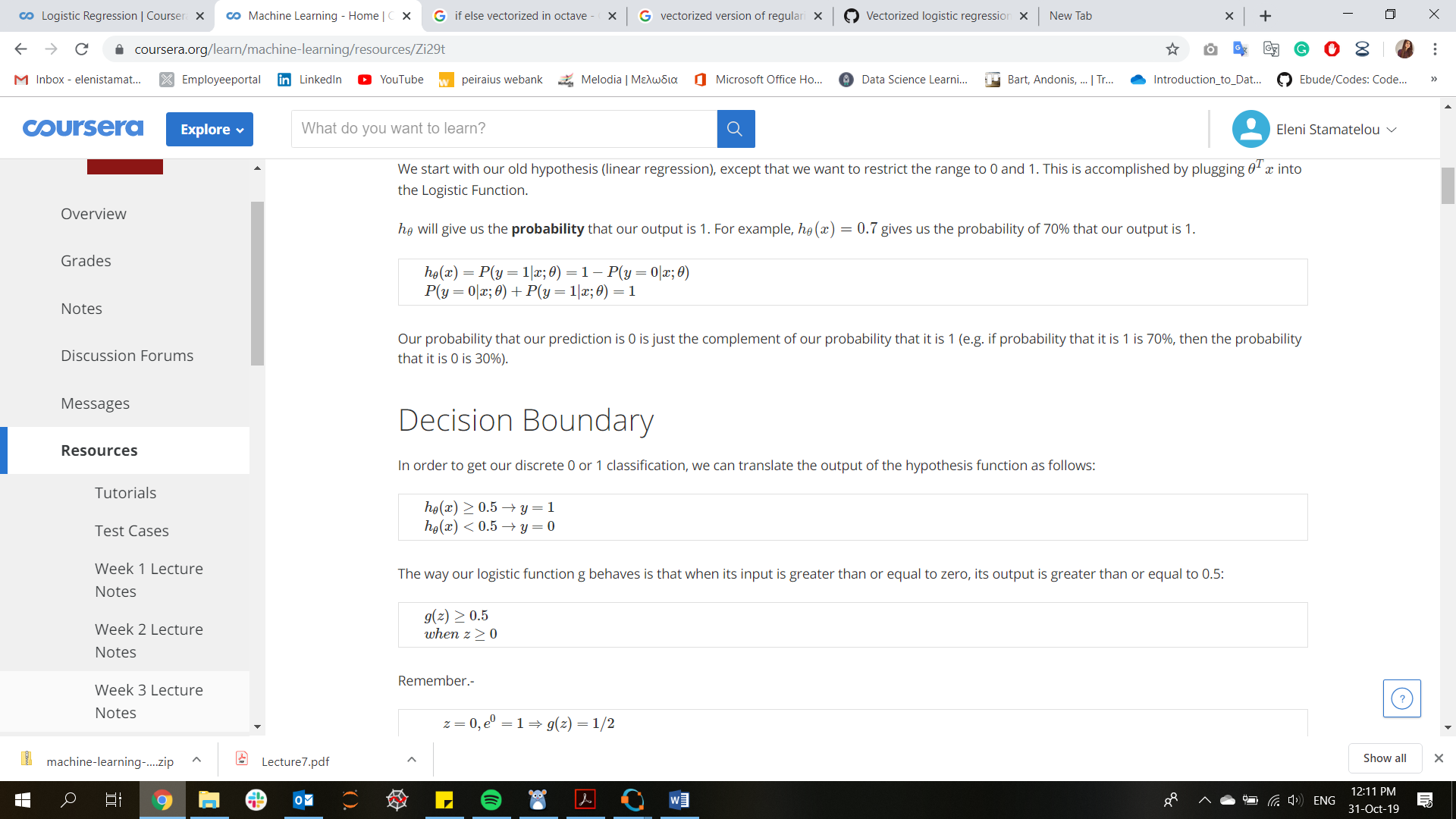


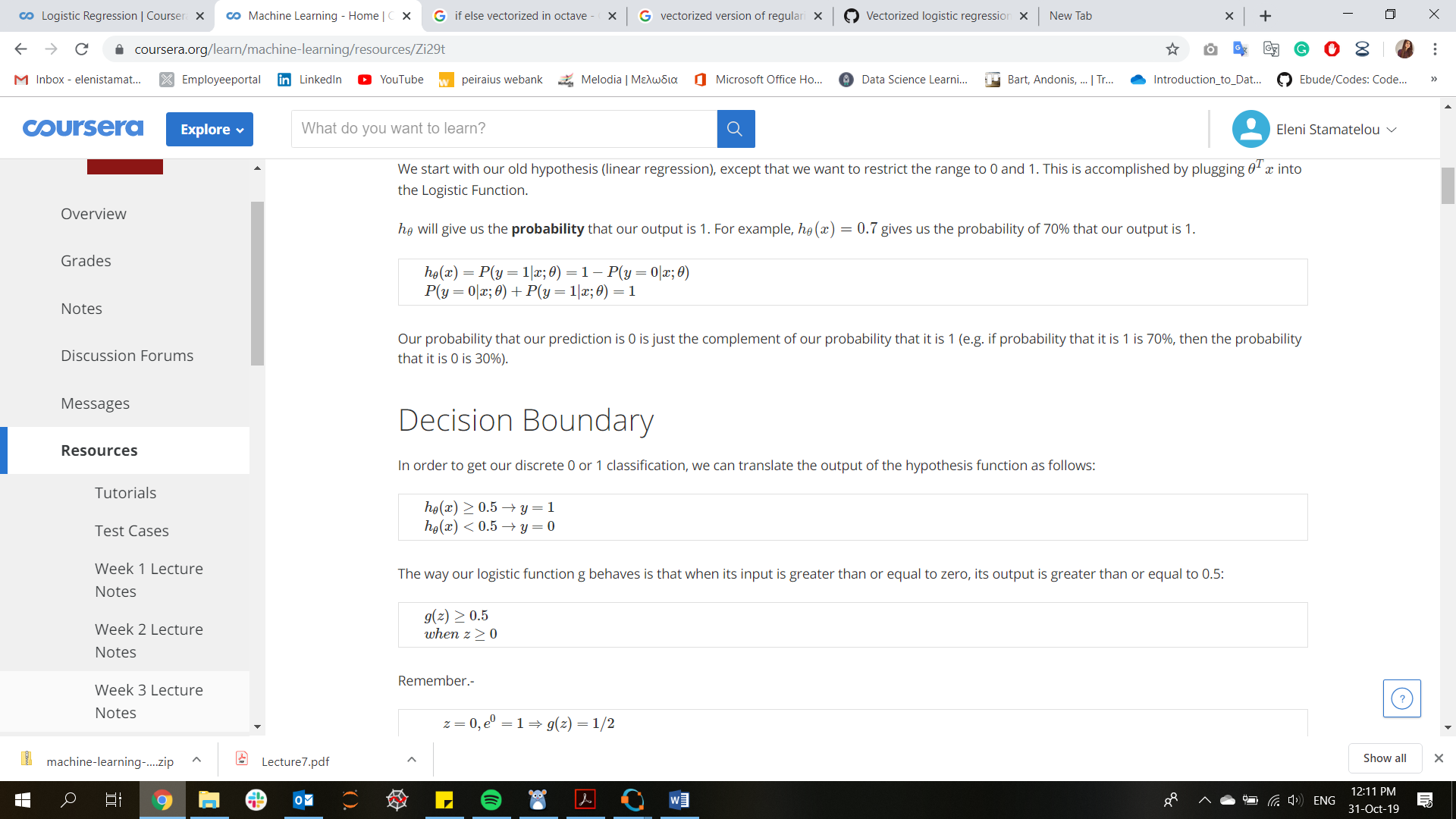
Regularization:

**Logistic Regression**

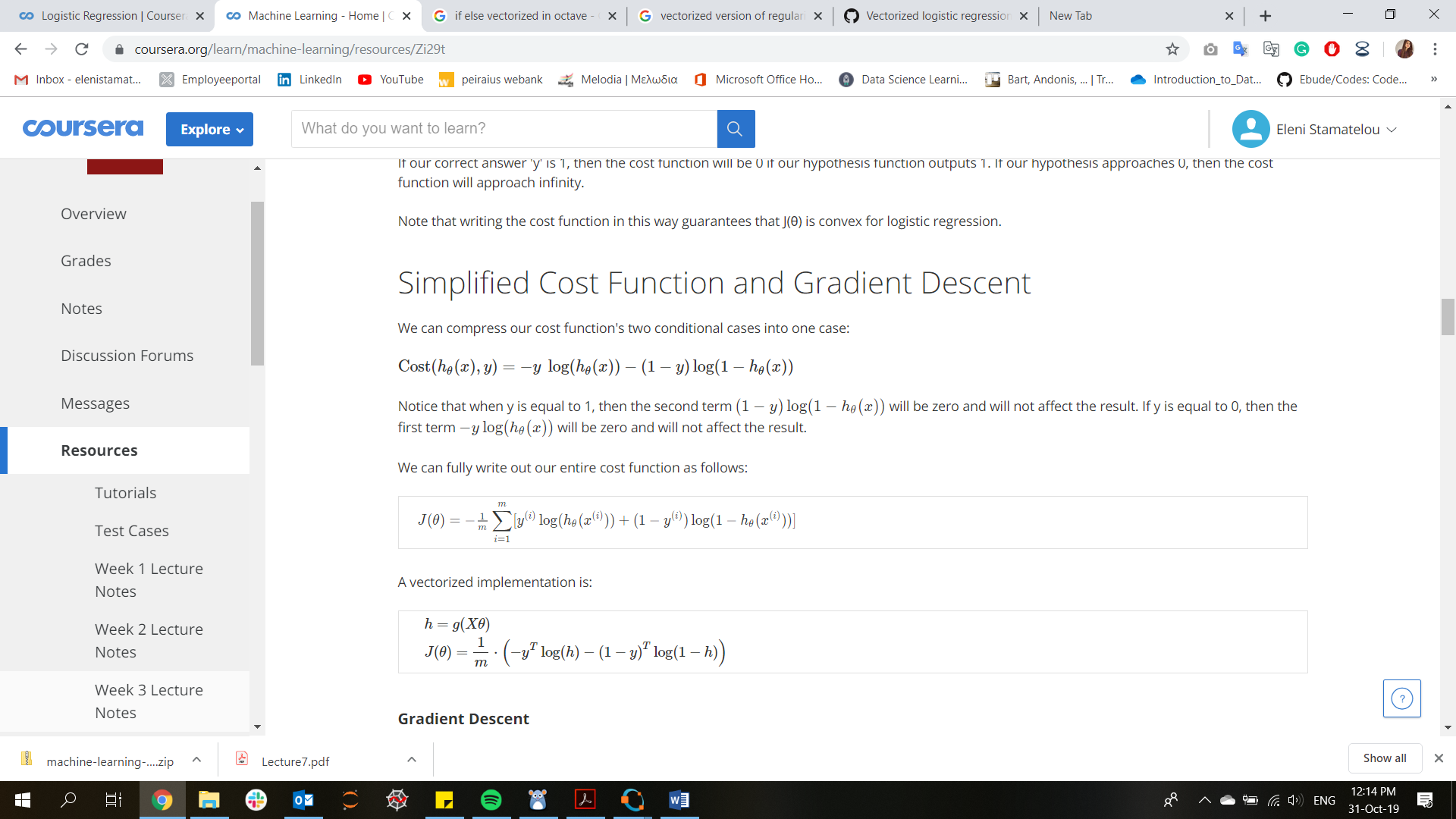
Hypothesis function:

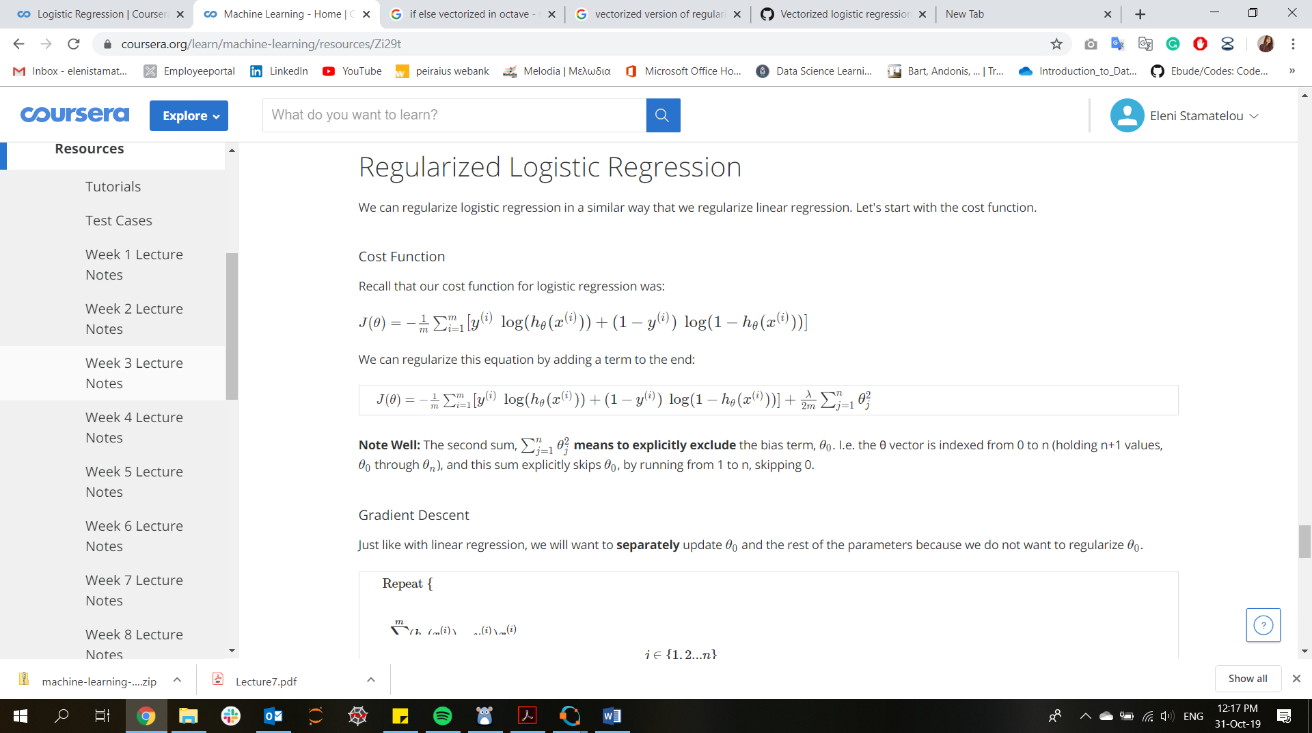






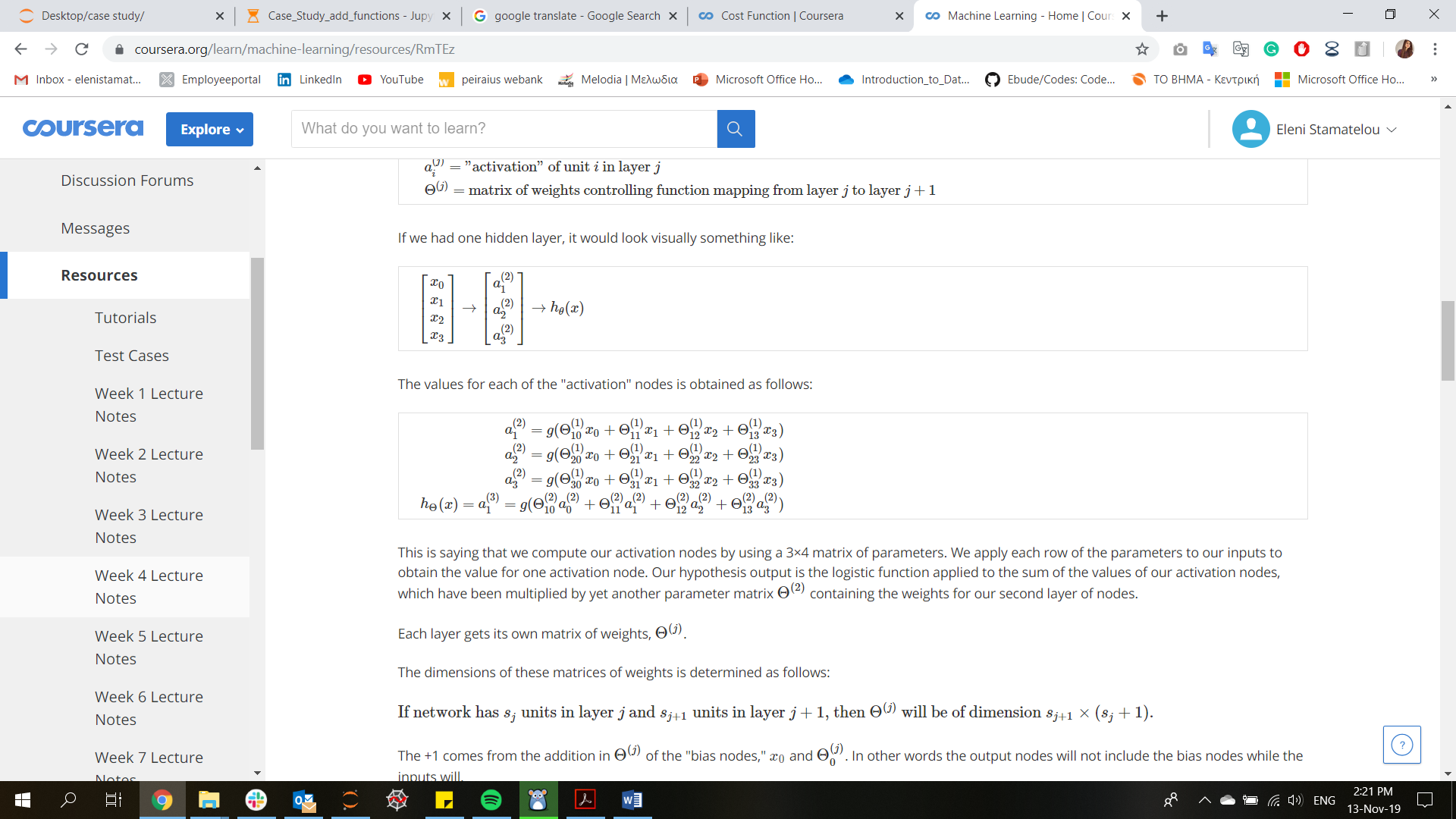
Cost function:

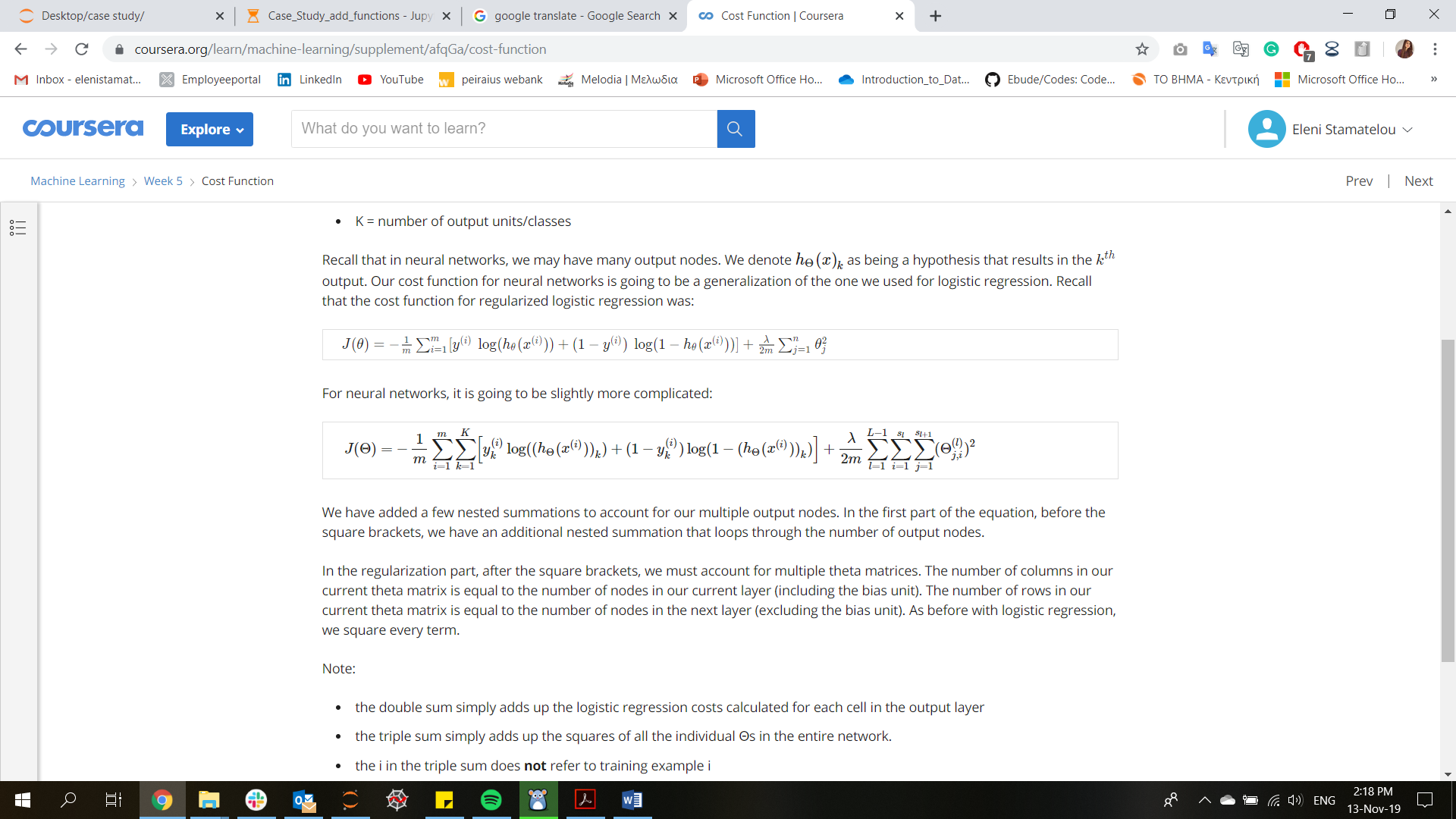


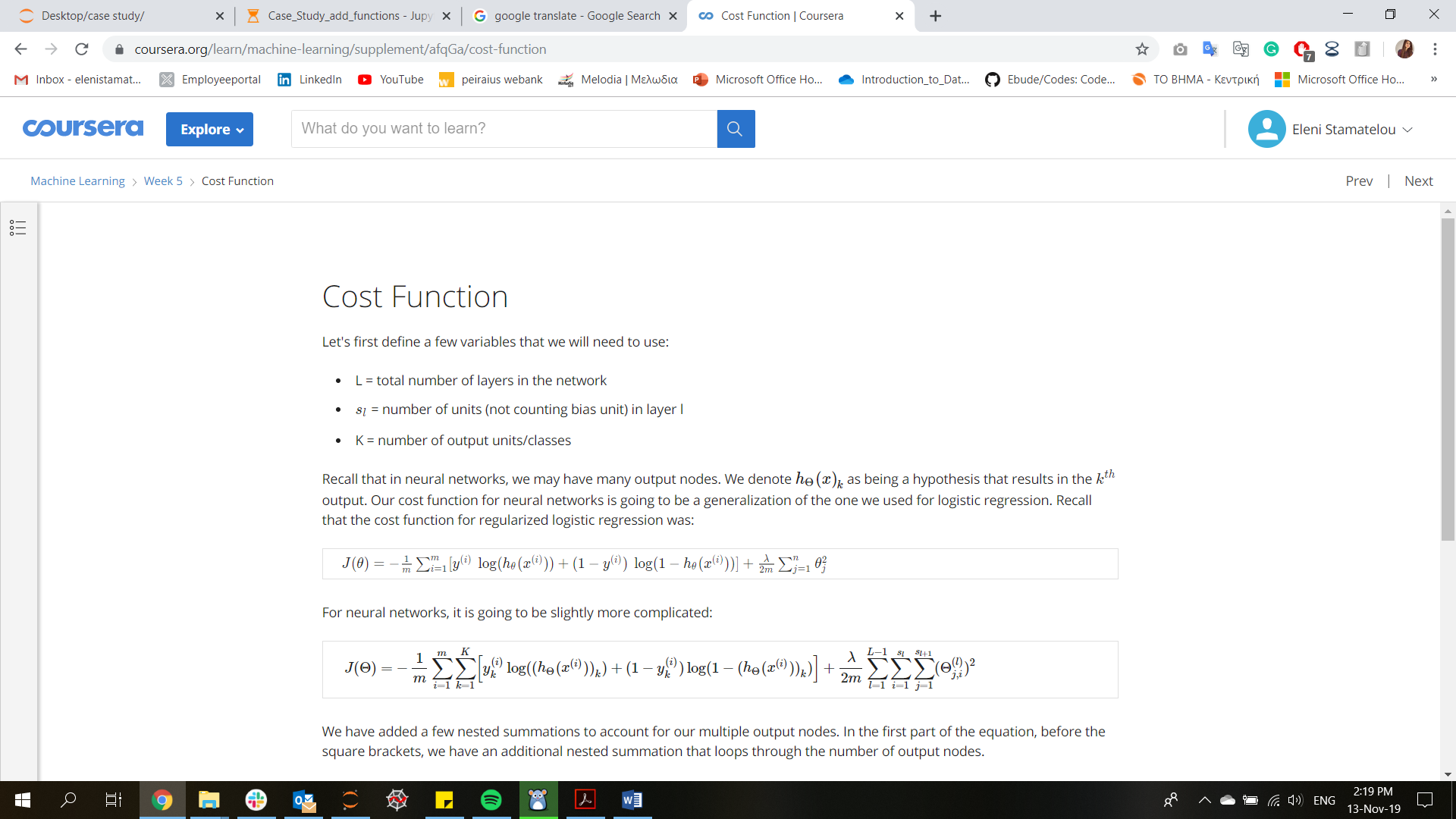
Regularization:

**Neural Networks**

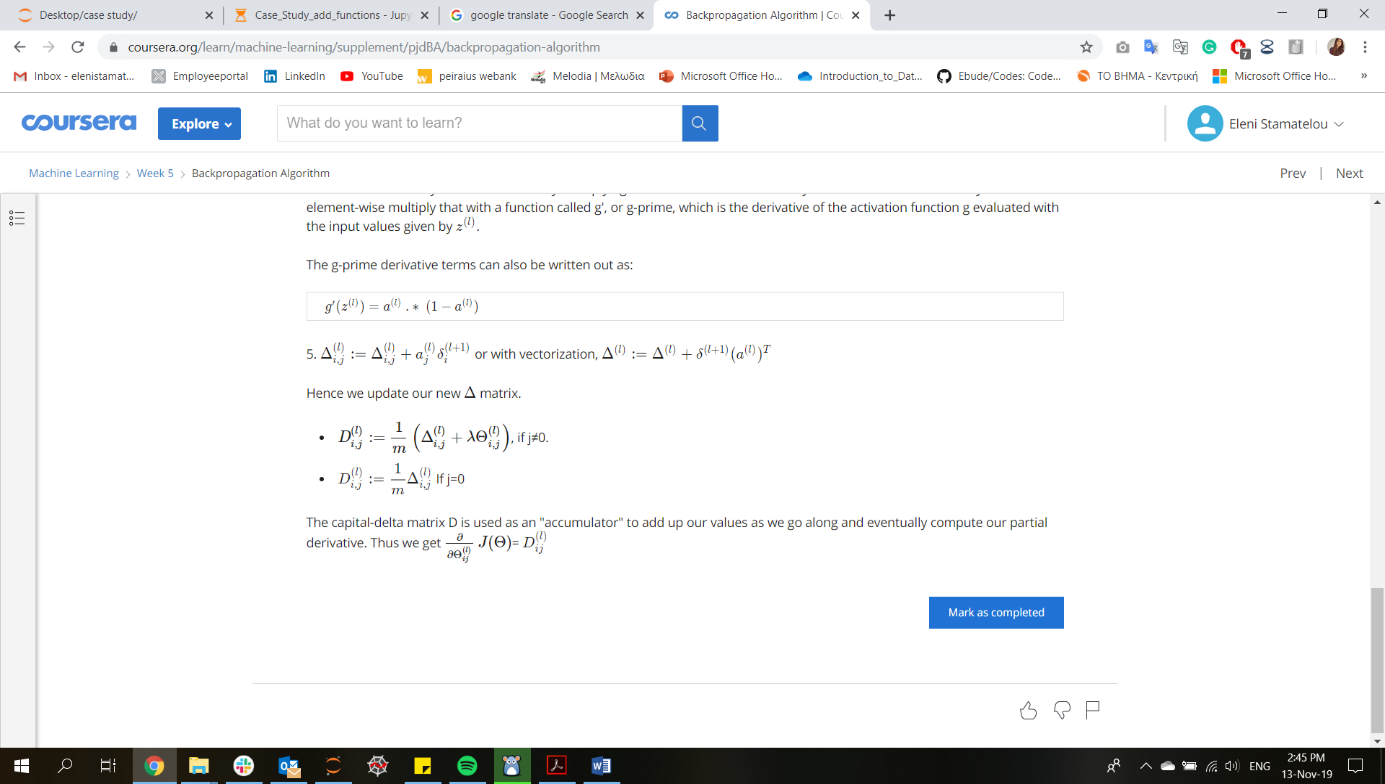
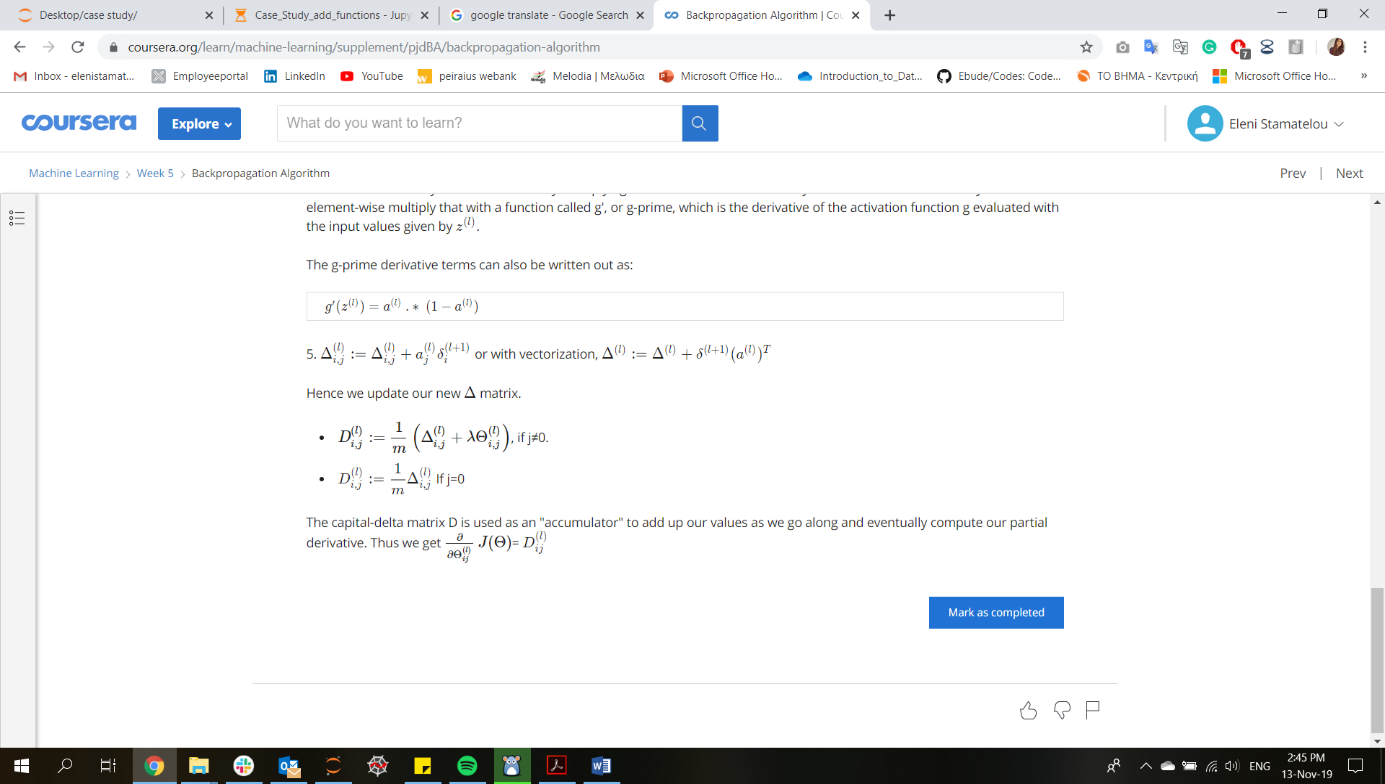
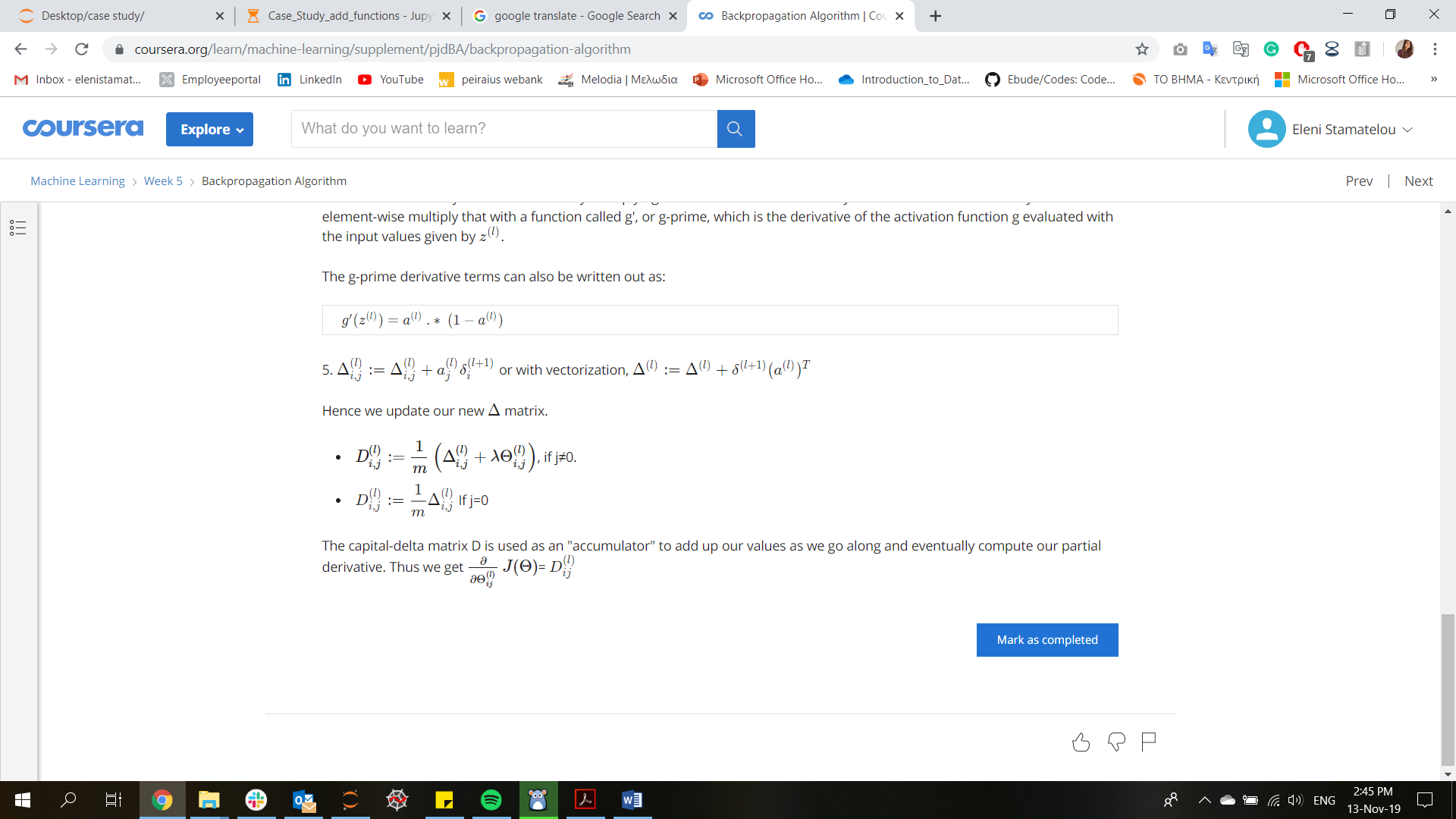
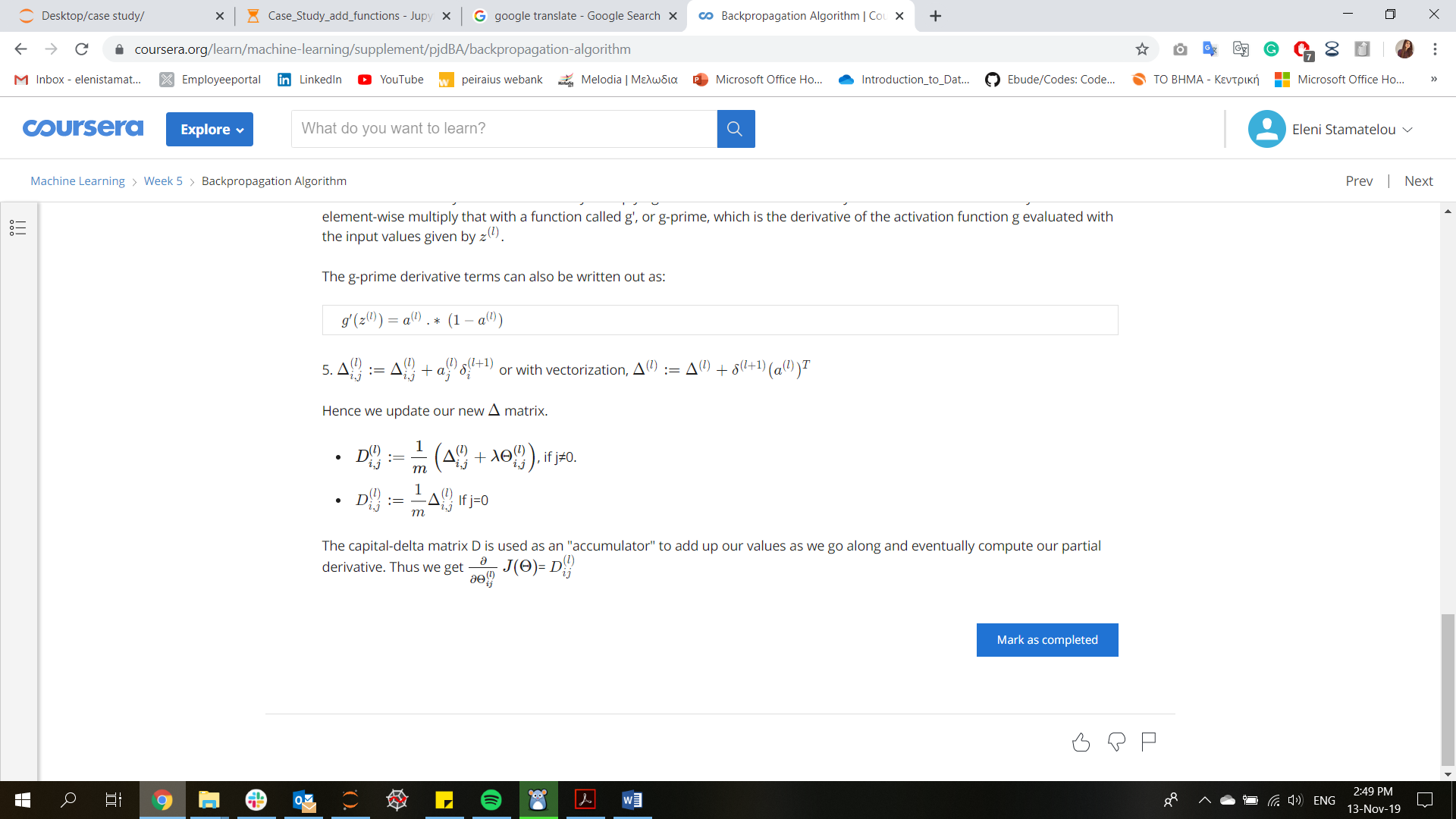
Hypothesis function – Forward propagation:

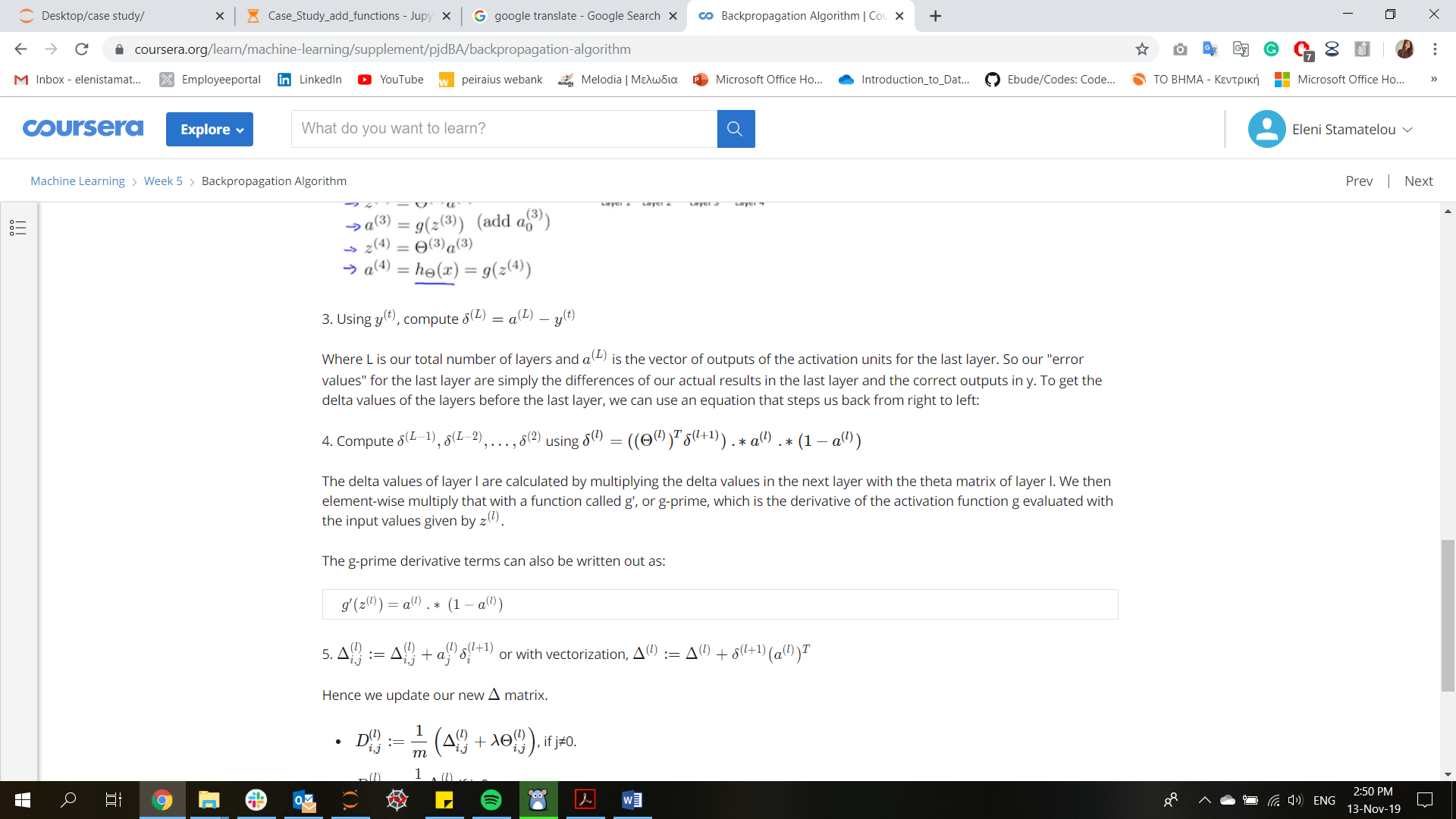


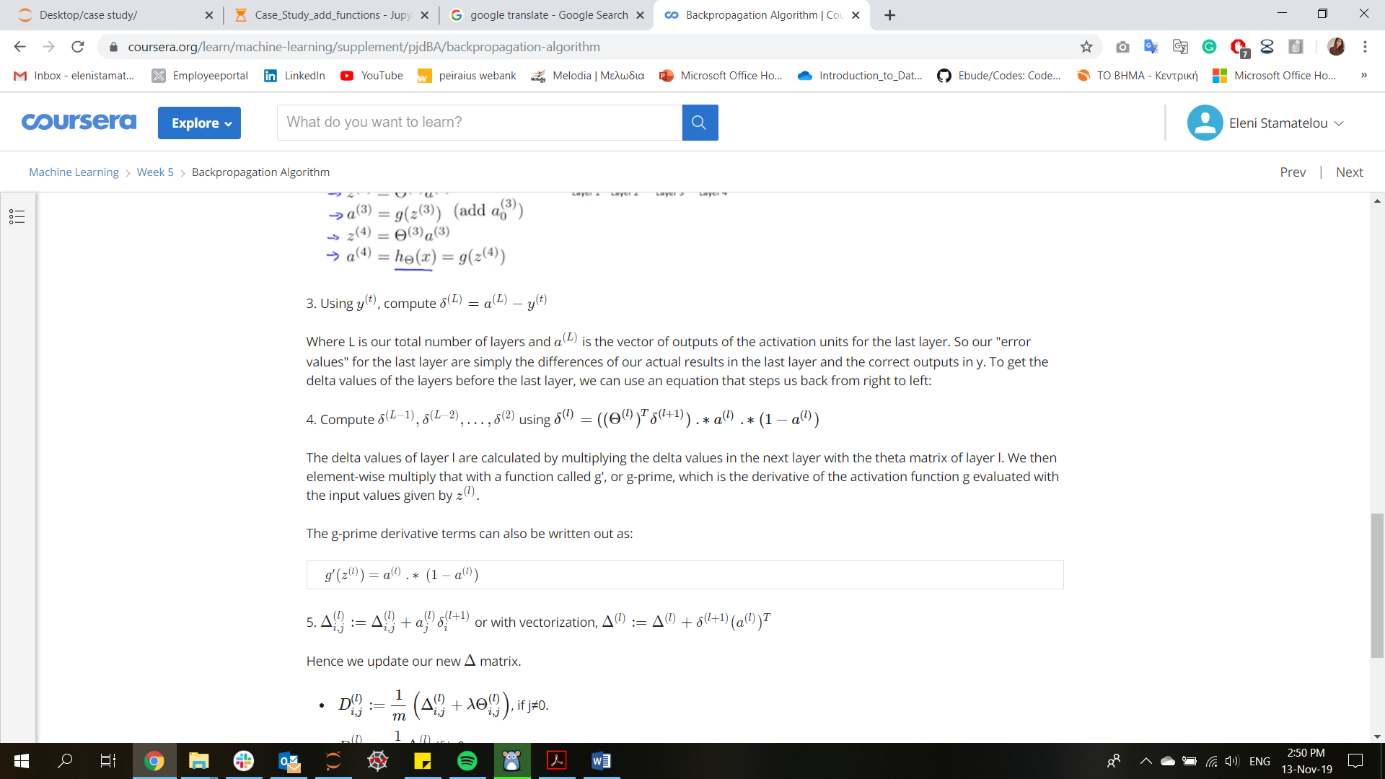
Cost Function + Regularization:



Backpropagation: (neural-network terminology to find the derivatives of cost function)

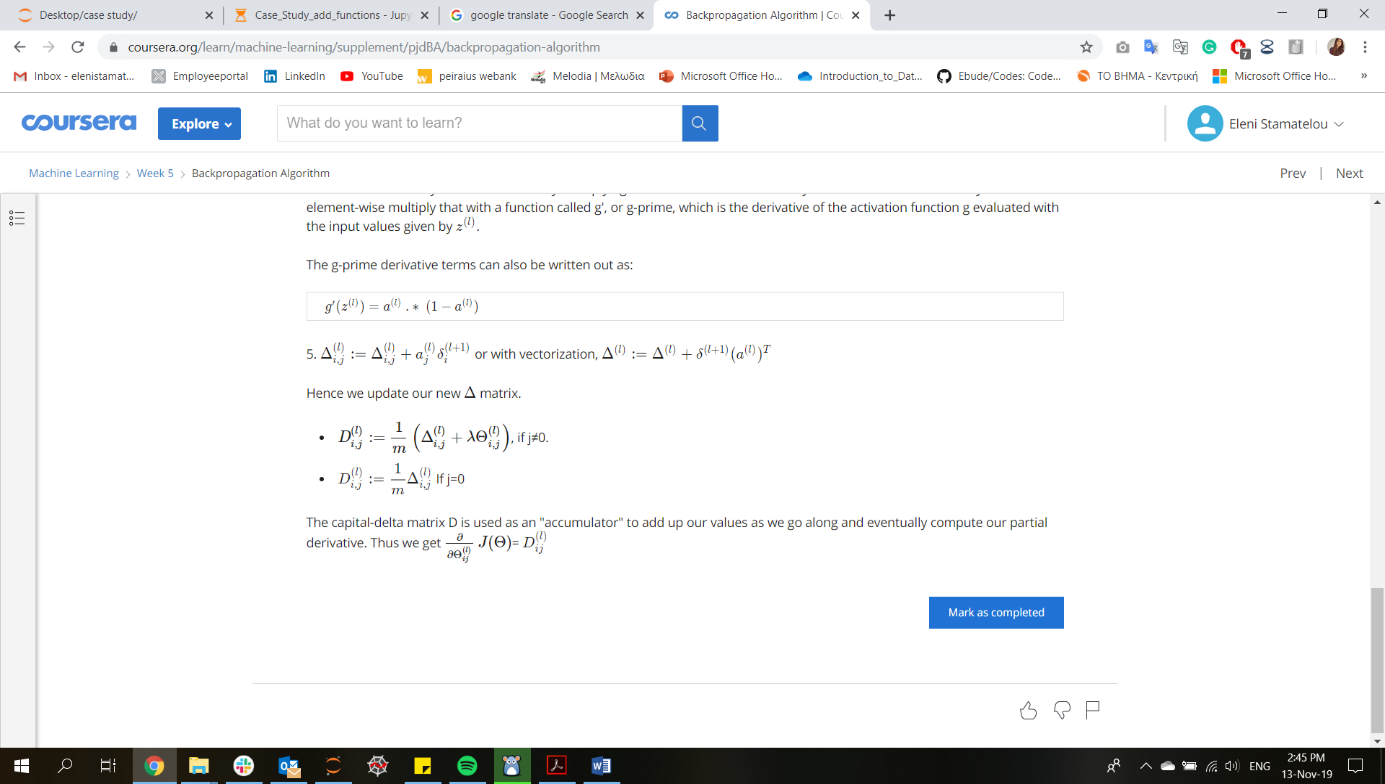


Derivatives

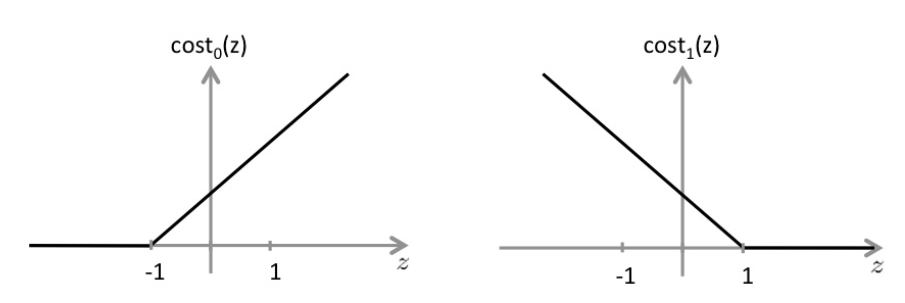
Final layer

Rest of layers

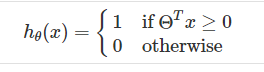
**STEPS**

1. Pick a network architecture
2. Random weight (Θ) initialization
3. Forward propagation to get hypothesis function h(x) (inside for loop/per sample)
4. Define Cost Function J(θ)
5. Backpropagation to compute partial derivatives
6. Gradient Checking to validate backprop (inside for loop/per sample)
7. Gradient descent or other optimization algorithm to minimize cost function + define weights (Θ)

Support Vector Machines

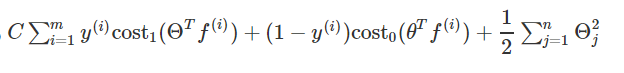
Hypothesis function: 

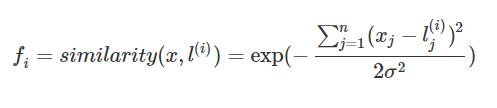


= (0, 1) for class y = 0

= (0, 1) for class y = 1

Cost function:

C:\Users\20194066\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Capture.png



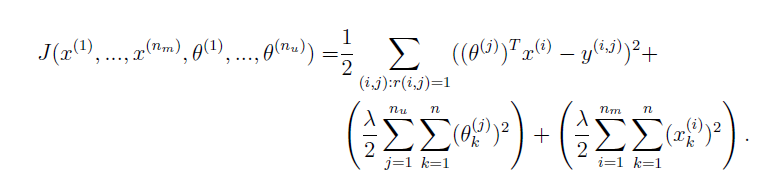
Collaborative filtering

j = user , i = movie ,

Θ = users rating for each feature

X = movies’ rating for each feature

Cost Function:



Gradient descent:

