coursera   [ ] **Navigate**   [ ] **Lab Files**   (?) **Help**

⌂ **Home**   /notebooks/C2_W2_Derivatives.ipynb   **COPY**   **Go**

Optional Lab - Derivatives

This lab will give you a more intuitive understanding of derivatives. It will show you a simple way of calculating derivatives arithmetically. It will also introduce you to a handy Python library that allows you to calculate derivatives symbolically.

In [ ]:
```python
from sympy import symbols, diff
```

Informal definition of derivatives

The formal definition of derivatives can be a bit daunting with limits and values 'going to zero'. The idea is really much simpler.

The derivative of a function describes how the output of a function changes when there is a small change in an input variable.

Let's use the cost function $J(w)$ as an example. The cost J is the output and w is the input variable.
Let's give a 'small change' a name *epsilon* or ϵ. We use these Greek letters because it is traditional in mathematics to use *epsilon*(ϵ) or *delta* (Δ) to represent a small value. You can think of it as representing 0.001 or some other small value.

$$\text{if } w \uparrow \epsilon \text{ causes } J(w) \uparrow \text{ by } k \times \epsilon \text{ then} \tag{1}$$

$$\frac{\partial J(w)}{\partial w} = k$$

This just says if you change the input to the function $J(w)$ by a little bit and the output changes by k times that little bit, then the derivative of $J(w)$ is equal to k.

Let's try this out. Let's look at the derivative of the function $J(w) = w^2$ at the point $w = 3$ and $\epsilon = 0.001$

In [ ]:
```python
J = (3)**2
J_epsilon = (3 + 0.001)**2
k = (J_epsilon - J)/0.001     # difference divided by epsilon
print(f"J = {J}, J_epsilon = {J_epsilon}, dJ_dw ~= k = {k:0.6f} ")
```

We have increased the input value a little bit (0.001), causing the output to change from 9 to 9.006001, an increase of 6 times the input increase. Referencing (1) above, this says that $k = 6$, so $\frac{\partial J(w)}{\partial w} \approx 6$. If you are familiar with calculus, you know, written symbolically, $\frac{\partial J(w)}{\partial w} = 2w$. With $w = 3$ this is 6. Our calculation above is not exactly 6 because to be exactly correct ϵ would need to be [infinitesimally small (https://www.dictionary.com/browse/infinitesimally)](https://www.dictionary.com/browse/infinitesimally) or really, really small. That is why we use the symbols $\approx$ or ~= rather than =. Let's see what happens if we make ϵ smaller.

In [ ]:
```python
J = (3)**2
J_epsilon = (3 + 0.000000001)**2
k = (J_epsilon - J)/0.000000001
print(f"J = {J}, J_epsilon = {J_epsilon}, dJ_dw ~= k = {k} ")
```

The value gets close to exactly 6 as we reduce the size of ϵ. Feel free to try reducing the value further.

Finding symbolic derivatives

In backprop it is useful to know the derivative of simple functions at any input value. Put another way, we would like to know the 'symbolic' derivative rather than the 'arithmetic' derivative. An example of a symbolic derivative is, $\frac{\partial J(w)}{\partial w} = 2w$, the derivative of $J(w) = w^2$ above. With the symbolic derivative you can find the value of the derivative at any input value w.

If you have taken a calculus course, you are familiar with the many [differentiation rules (https://en.wikipedia.org/wiki/Differentiation_rules#Power_laws,_polynomials,_quotients,_and_reciprocals)](https://en.wikipedia.org/wiki/Differentiation_rules#Power_laws,_polynomials,_quotients,_and_reciprocals) that mathematicians have developed to solve for a derivative given an expression. Well, it turns out this process has been automated with symbolic differentiation programs. An example of this in python is the [SymPy (https://www.sympy.org/en/index.html)](https://www.sympy.org/en/index.html) library. Let's take a look at how to use this.

$J = w^2$

Define the python variables and their symbolic names.

In [ ]:
```python
J, w = symbols('J, w')
```

Define and print the expression. Note SymPy produces a [latex (https://en.wikibooks.org/wiki/LaTeX/Mathematics)](https://en.wikibooks.org/wiki/LaTeX/Mathematics) string which generates a nicely readable equation.

In [ ]:
```python
J=w**2
J
```