[How the backpropagation algorithm works](http://neuralnetworksanddeeplearning.com/chap2.html)

Training a neural network means minimizing an associated "error" function wrt the networks weights. Now there are optimization methods that use only function values (Simplex method of Nelder and Mead, Hooke and Jeeves, etc), methods that in addition use first derivatives (steepest descend, quasi Newton, conjugate gradient) and Newton methods using second derivatives as well. So if you want to use a derivative method, you have to calculate the derivatives of the error function, which in return involves the derivatives of the transfer or activation function. Back propagation is just a nice algorithm to calculate the derivatives, and nothing more.

The other answers here have explained the math which makes it pretty clear that a derivative term will appear in your calculations while you are trying to update the weights. **But all of those calculations are being done in order to implement Back-propagation, which is just one of the ways of updating weights!** Now read on...

You are correct in assuming that at the end of the day, all a neural network tries to do is update its weights to fit the data you feed into it. Within this statement lies your answer too. What you are getting confused with here is the idea of the Back-propagation algorithm. Many textbooks use backprop to update neural nets by default but do not mention that there are other ways to update weights too. This leads to the confusion that neural nets and backprop are the same thing and are inherently connected. This also leads to the false belief that neural nets need backprop to train.

Please remember that Back-propagation is just ONE of the ways out there to train your neural network (although it is the most famous one). Now, you must have seen the math involved in backprop, and hence you can see where the derivative term comes in from (some other answers have also explained that). It is possible that other training methods won't need the derivatives, although most of them do. Read on to find out why....

Think about this intuitively, we are talking about CHANGING weights, the direct mathematical operation related to change is a derivative, makes sense that you should need to evaluate derivatives to change weights.

Do let me know if you are still confused and I'll try to modify my answer to make it better. Just as a parting piece of information, another common misconception is that gradient descent is a part of backprop, just like it is assumed that backprop is a part of neural nets. Gradient descent is just one way to minimize your cost function, there are plenty of others you can use. One of the answers above makes this wrong assumption too when it says "Specifically Gradient Descent". This is factually incorrect.