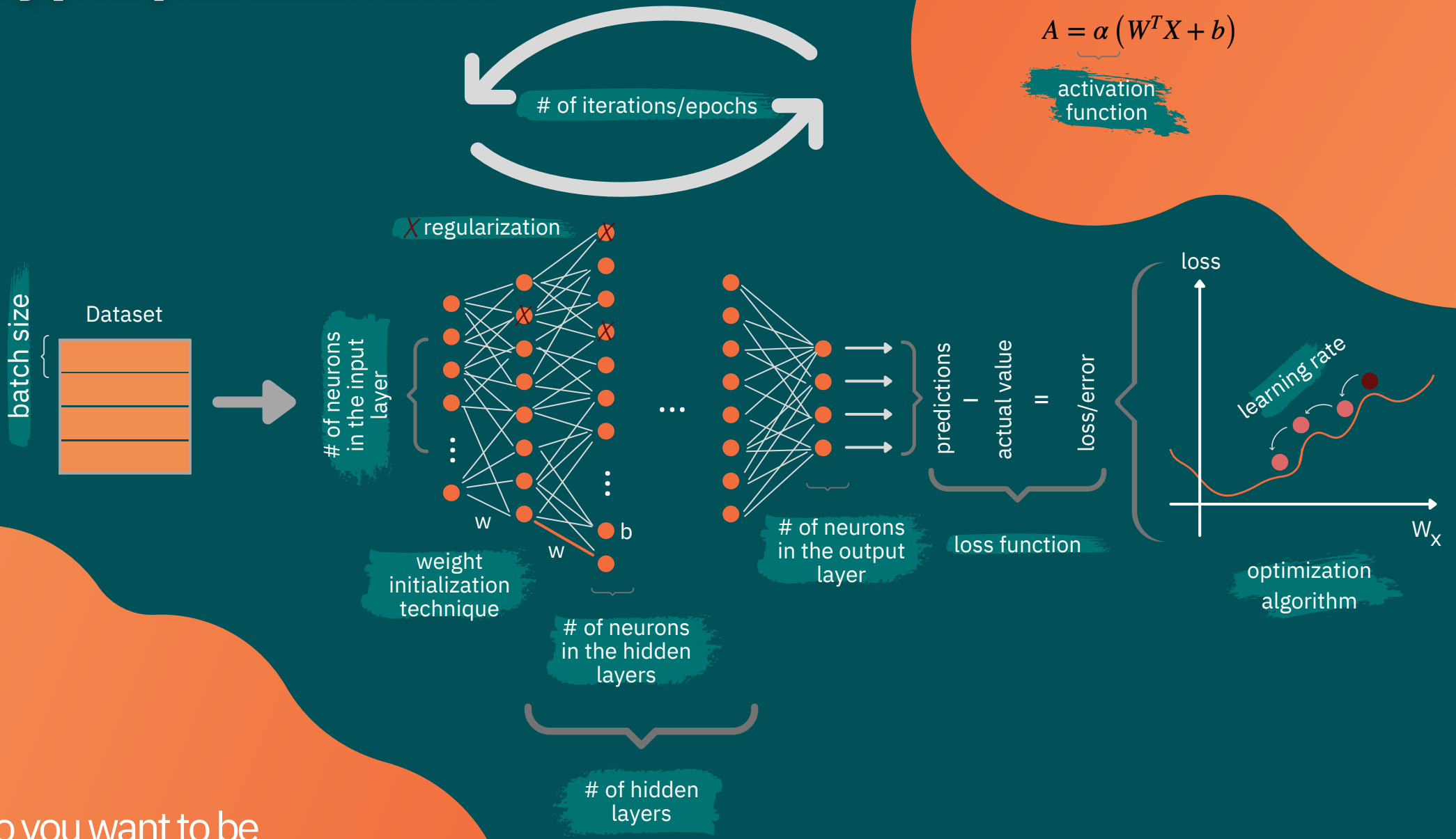
Neural Networks Hyperparameters

Calculating layer outputs:

$$A = \alpha\left(W^T X + b\right)$$

activation function



So you want to be **a data scientist?**

Pre-determined hyperparameters

Number of neurons in the input layer

This will depend on the data you are training with.

If you have n features in your dataset you will have n input neurons. If your input is a photo with nxn pixels, you will have n^2 input neurons.

Number of neurons in the output layer

This will again depend on the data you are training with.

If you are doing binary classification or regression, 1 output neuron will do. If you are doing multi-class or multi-label classification, you need as many output neurons as classes/labels.

Hyperparameters that need tuning

Number of hidden layers

The more hidden layers you have, the deeper your network is going to be. Generally, having more layers will help your network more than increasing the number of neurons in the hidden layers.

Number of neurons in the hidden layers

Each hidden layer can have a different number of neurons. This value needs to be adjusted based on how the network is performing.

Activation function

Each layer has its own activation function except the input layer. The activation function of the hidden layers cannot be linear.

Optimization algorithm

The optimization algorithm determines how the network updates its weights.

Loss function

The loss (cost) function is what we're trying to minimize during training. The cost function you use will depend on the type of problem you have.

Batch size

Batch size is the amount of data points you put in each batch while training your network.

If you use a batch size of 1, you will be doing **Stochastic gradient descent (GD)**. If you use batch size equals the number of data points in your whole dataset, that is **Batch GD**. If batch size is anything between 2 to number of data points in your whole dataset, you are doing **mini-batch gradient-descent.**

Number of epochs

This is how many times you run the whole dataset through your network. The network learns the dataset a little bit better after each epoch.

Learning rate

Learning rate determines how big of a step to take towards the direction set by gradient descent. Too small of a learning rate might slow down learning and too big of a learning rate might cause the network to keep missing the optima.

Learning rate scheduling techniques are used to dynamically decide on the learning rate value.

So you want to be
a data scientist?

Weight initialization technique

This is how the weights of the networks are initialized. Bias values can be (and often are) initialized to zero. But weights cannot be initialized to zero because if they are, they will all be updated the same way and the model won't be able to learn.

At the same time, just using normal distribution for the random initialization of the weights cause problems with the network. That's why, sometimes we need a more advanced approach.

Regularization

Regularization is a way to deal with overfitting. There are many different ways to do regularization and often these techniques come with their own hyperparameters.

Some examples are:

- L1
- L2
- Dropout

L1 and L2 regularization has a hp called alpha and dropout has a hp called the dropout rate.