**Classification Problems**

**Logistic Classifier:**

Function that will allow you to take inputs and output the classification of the item.

Ex. Input image of letter, output what letter it is.

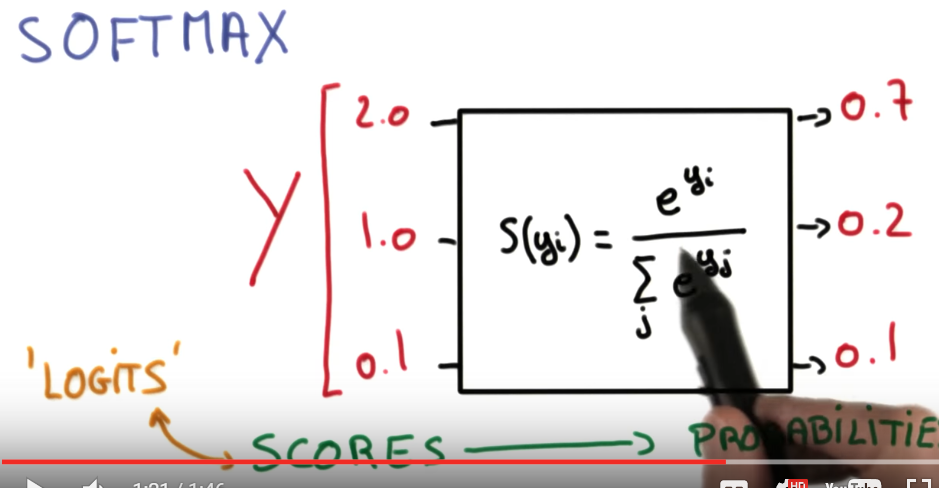
WX + b = Y

Y is the output classification that we want. X is our input. W and b are the weight and the bias, they are adjusted through machine learning to match our desired input and output mapping.

The values for W and b are trained to get good predictions.

You want the values coming out of the logistic classifier (Scores) to represent different classification labels. Therefore you need to map the output Y to some probabilities where the probabilities are closest to 1 for the correct classifications. This is due to each input only having one and only one possible label.

Turning scores (or Logits) into probabilities is done with the Softmax function.



High probabilities for high scores, and vice-versa.

TensorFlow uses the logistic classifier Y = xW + b instead of Y = Wx + b.

In order for the matrix dimensions to work you need to transpose W, x, and b.

The values for the variables W and b need to be adjustable, and thus need to be Variables in TensorFlow.

They need initial values, which are traditionally given by random numbers from a normal distribution.

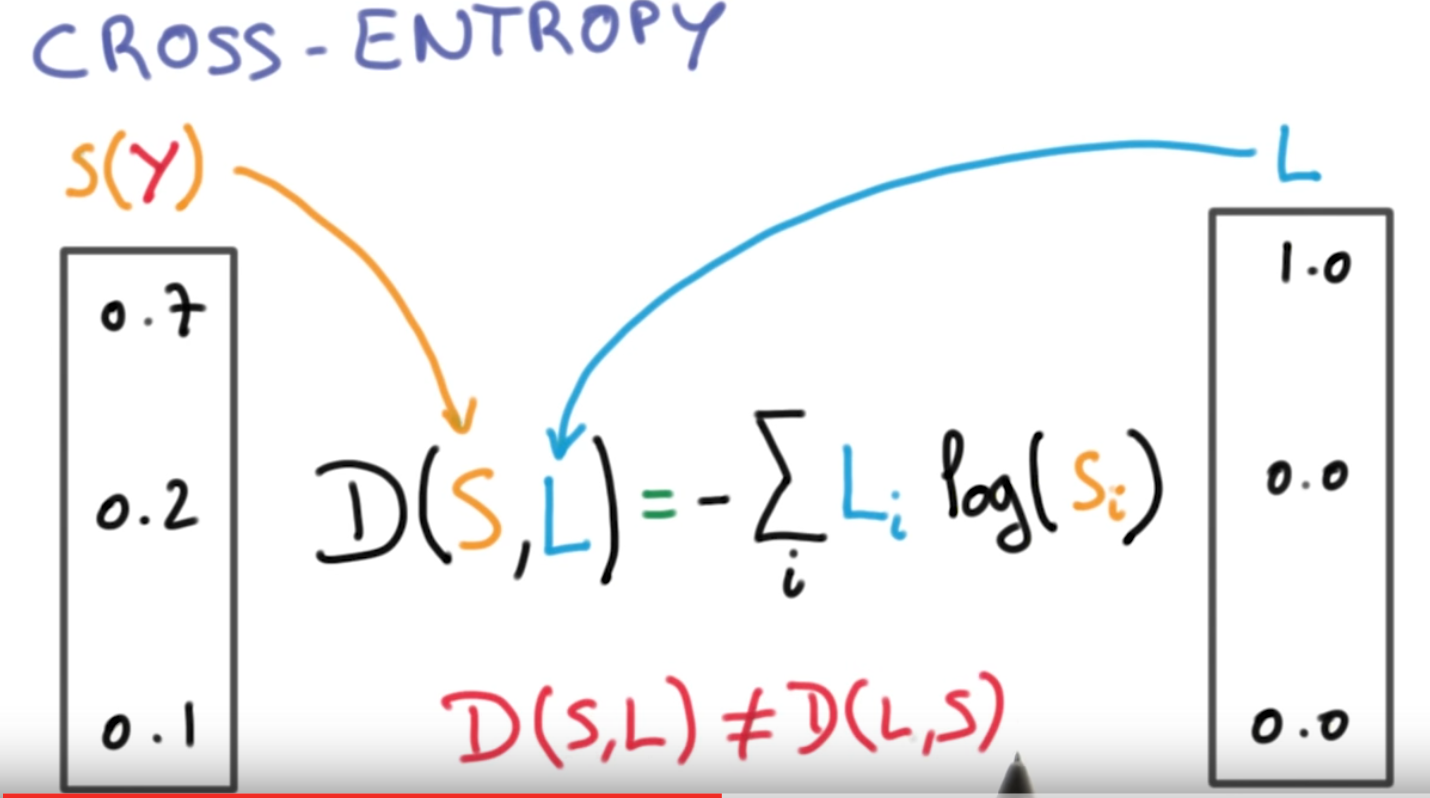
Randomizing helps the model from becoming stuck in the same place every time you train it, and normal distribution prevents any one weight from overwhelming other weights.

Since the weights are already randomized to prevent the model from getting stuck, you don’t need to randomize the bias. Just initialize it as zero.

With the probabilities acquired, we want to use one-hot encoding to place a 1 in the column for the correct value.

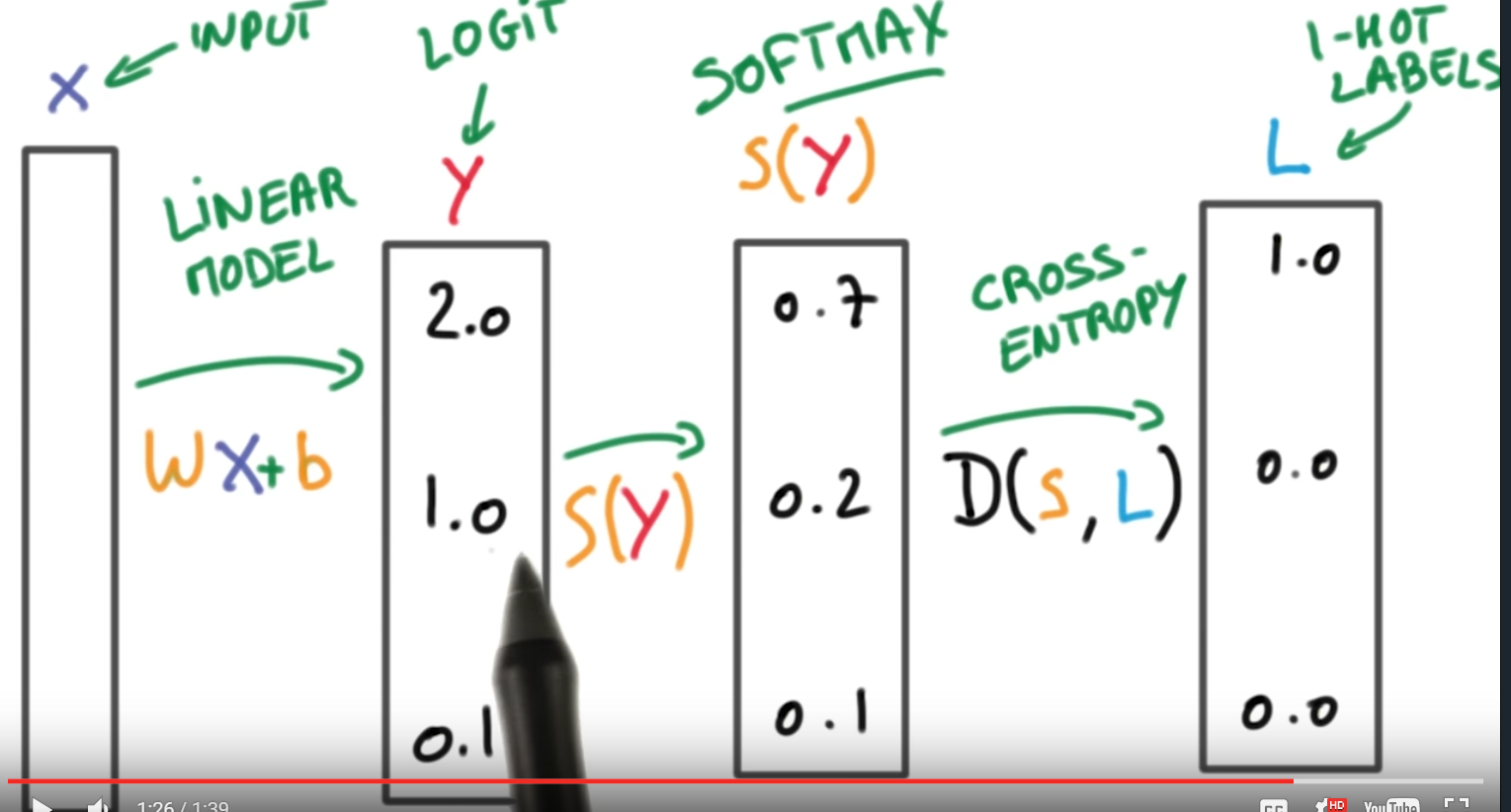
It just maps the class with the highest probability with a 1 as the encoding and the rest of the values as 0’s.

If you have a problem that has tens of thousands or more classes to classify, one-hot encoding stops working so well. This is because when using one-hot encoding, the one-hot encoding vector is really large with all zeros values except for one.



We can use the cross-entropy function to measure how well we are doing by comparing the one hot encoded vector to our vector of probabilities from the Logistic classifier. The labels vector (one-hot encoded) and our distributions are compared, and the function isn’t linear so don’t swap the vector locations.

This whole process combined is called Multinomial Logistic Classification. D(S(WX+b),L)



The parameters W (weights) and b (bias) are developed by minimizing the average cross-entropy (error) using gradient descent. The values chosen are the ones that minimize the cost/loss function.

Initial value for the weights are random with a normal distribution and a mean of 0 and std.dev of 1.

The dataset used will also need to be normalized to have a mean of zero and std deviation of 1.

**Stochastic gradient descent:**

The problem with normal gradient descent is that the loss function is computed with the whole dataset, and a rule of thumb is that computing its gradient of your loss takes roughly 3 times the amount of floating point operations as computing the loss.

Instead of computing the loss with the whole dataset we will get an estimate (not necessarily a good estimate). We will compute the loss with the average loss of a much smaller random collection from the dataset (called the batch). It is very important that your batch is random. You compute that small sets average loss, and then compute its gradient and then pretend that it gives an accurate representation of the whole dataset.

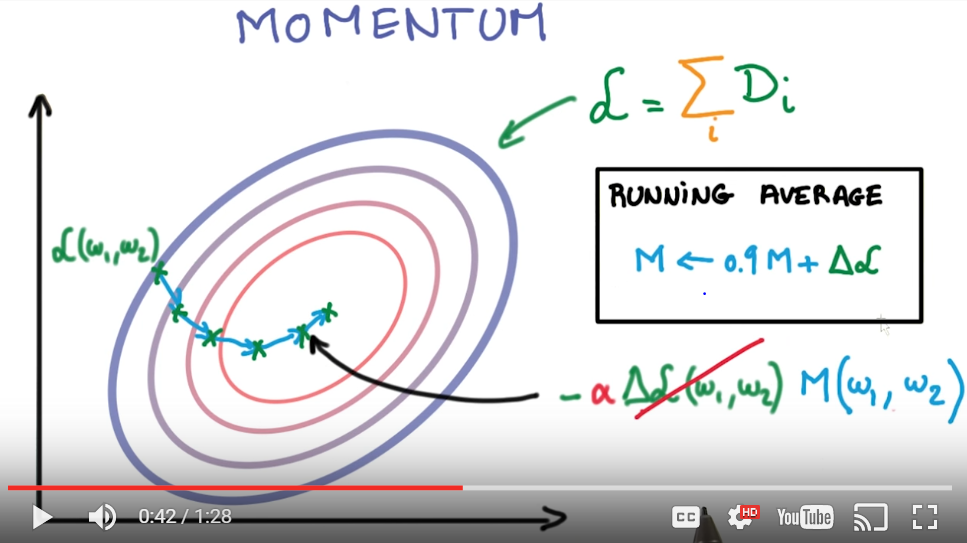
The key is you to take many smaller steps than larger steps. It is much more efficient speed and accuracy. Even if one step is in the wrong direction, on the whole, it will tend toward the minimum.

It scales well with more data, but still has its problems.

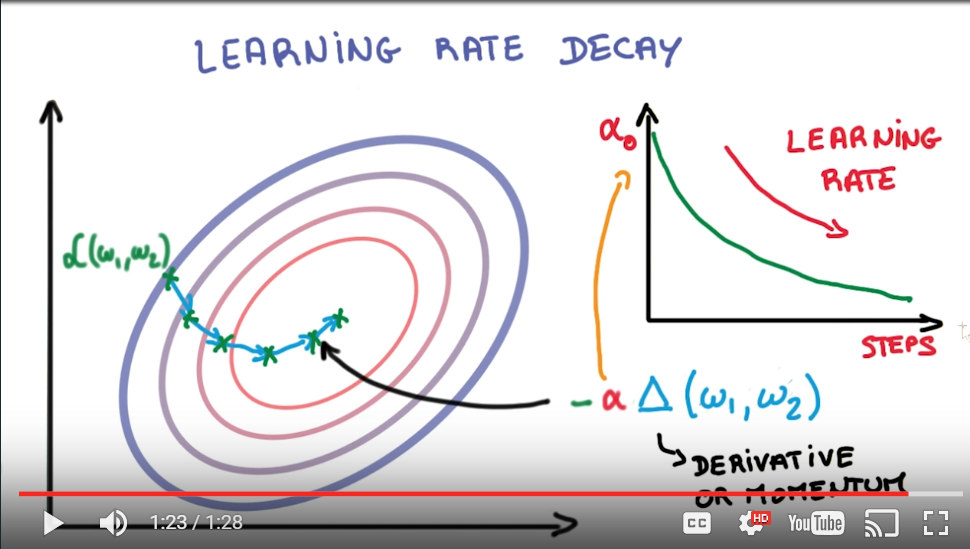
**Helping Stochastic Gradient Descent work well**

Initializing your variables with the normal distribution with equal variance and Normalizing your dataset are very important.

Momentum: Each single step takes is in a random direction, but on the whole we are making progress towards the minimum. We can use a technique called momentum to use the running average, meaning where the data is heading towards on the average so far, as a good metric of the proper direction you should be heading.



Another technique is to decay the Learning rate



Lower the learning rate with more steps. When looking at the loss from step to step, the higher learning rate may learn faster, but might end up flatting out earlier than if a lower learning rate was used.

It is important to understand that how fast you learn has nothing to do with how well you train.



There are many parameters that can be adjusted.

Finding the best parameters takes some intuition and testing. ADAGRAD is a process that automatically defines some of the parameters, but doesn’t lead to as great a performance.

Cross – Validation:

A model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set.

It is mainly used in settings where the goal is prediction, and one wants to estimate how [accurately](https://en.wikipedia.org/wiki/Accuracy) a [predictive model](https://en.wikipedia.org/wiki/Predictive_modelling) will perform in practice. In a prediction problem, a model is usually given a dataset of *known data* on which training is run (*training dataset*), and a dataset of *unknown data* (or *first seen* data) against which the model is tested (*testing dataset*).[[4]](https://en.wikipedia.org/wiki/Cross-validation_(statistics)#cite_note-Newbie_question:_Confused_about_train.2C_validation_and_test_data.21-4) The goal of cross validation is to define a dataset to "test" the model in the training phase (i.e., the [*validation dataset*](https://en.wikipedia.org/wiki/Validation_set)), in order to limit problems like [overfitting](https://en.wikipedia.org/wiki/Overfitting), give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem), etc.