Softmax:

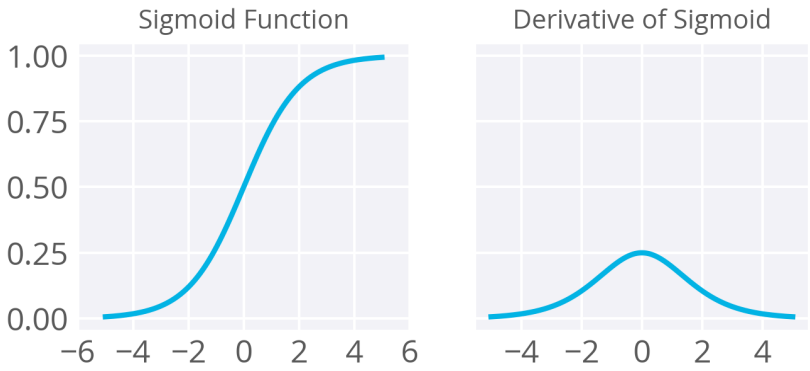
this function helps in making the data prediction into probabilities so that it can be easily managed. Through probabilities we can easily see the variance in data

```
>>> import math
>>> z = [1.0, 2.0, 3.0, 4.0, 1.0, 2.0, 3.0]
>>> z_exp = [math.exp(i) for i in z]
>>> print([round(i, 2) for i in z_exp])
[2.72, 7.39, 20.09, 54.6, 2.72, 7.39, 20.09]
>>> sum_z_exp = sum(z_exp)
>>> print(round(sum_z_exp, 2))
114.98
>>> softmax = [round(i / sum_z_exp, 3) for i in z_exp]
>>> print(softmax)
```

One hot encoding:
in this the class has the correct probability of occurrence their in array put 1 and others are zero.

RELU-> rectified linear unit function
The ReLU function is 0 for negative inputs and x for all inputs $x>0$.

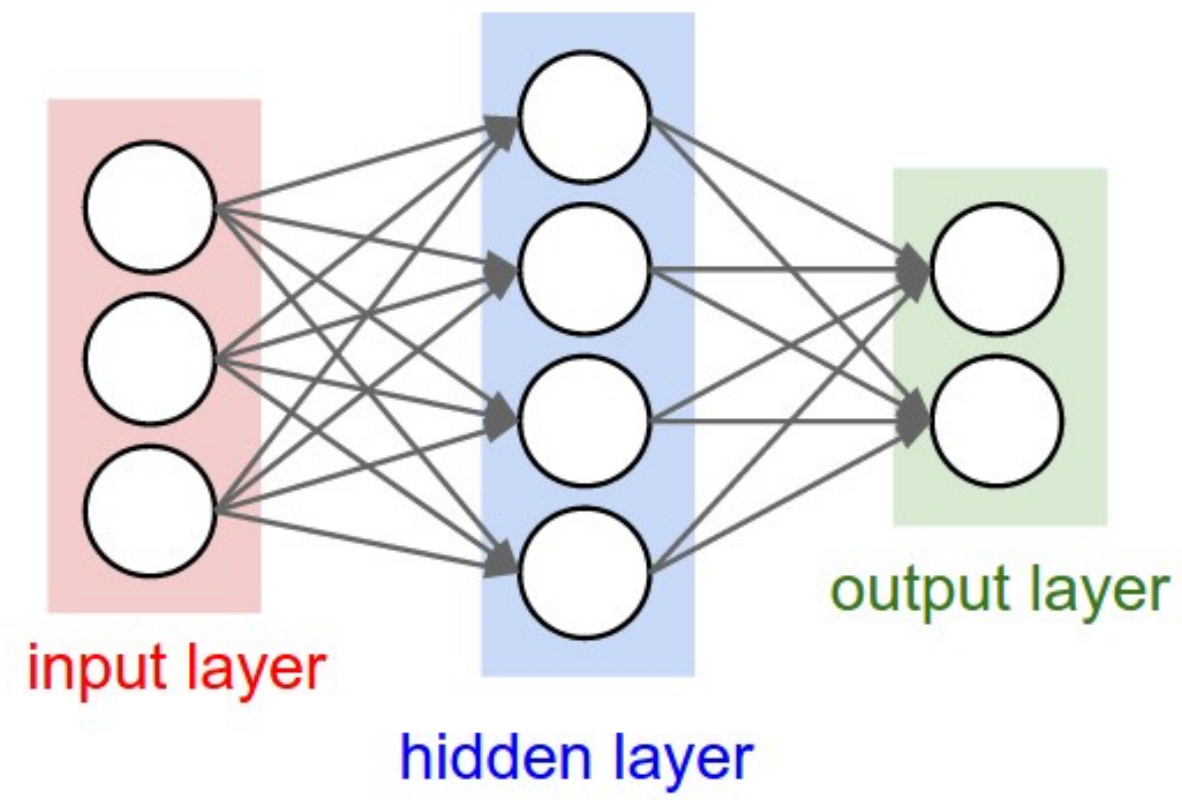
Back-Propagation
Now we can see a clear pattern. To find the gradient, you just multiply the gradients for all nodes in front of it going backwards from the cost. This is the idea behind **backpropagation**. The gradients are passed backwards through the network and used with gradient descent to update the weights and biases. If a node has multiple outgoing nodes, you just sum up the gradients from each node



Disadvantage of sigmoid:

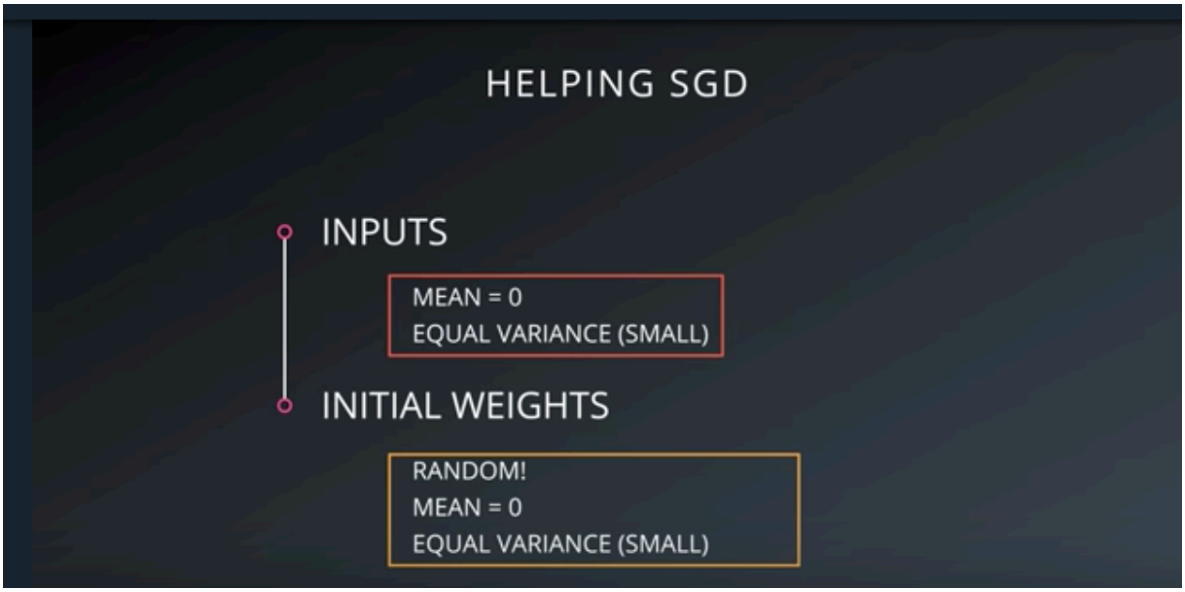
As noted in the backpropagation material, the derivative of the sigmoid maxes out at 0.25 (see above). This means when you're performing backpropagation with sigmoid units, the errors going back into the network will be shrunk by at least 75% at every layer. For layers close to the input layer, the weight updates will be tiny if you have a lot of layers and those weights will take a really long time to train. Due to this, sigmoids have fallen out of favor as activations on hidden units.

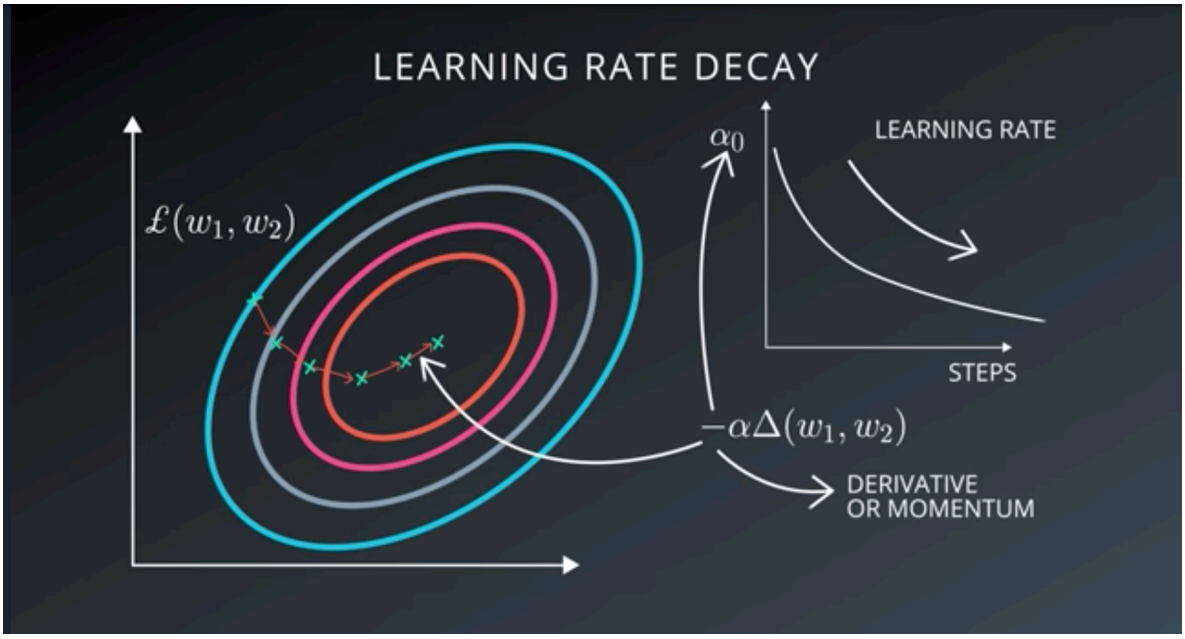
LOGITS: Logistic Regression of Digits



- The first network (left) has 4 + 2 = 6 neurons (not counting the inputs), [3 x 4] + [4 x 2] = 20 weights and 4 + 2 = 6 biases, for a total of 26 learnable parameters.
- The second network (right) has 4 + 4 + 1 = 9 neurons, [3 x 4] + [4 x 4] + [4 x 1] = 12 + 16 + 4 = 32 weights and 4 + 4 + 1 = 9 biases, for a total of 41 learnable parameters.

Stochastic Gradient Descent





Dimensionality CNN (Find output node dimension by given input

dimensions)

Setup

H = height, W = width, D = depth

- We have an input of shape 32x32x3 (HxWxD)
- 20 filters of shape 8x8x3 (HxWxD)
- A stride of 2 for both the height and width (S)
- With padding of size 1 (P)

Recall the formula for calculating the new height or width:
new_height = (input_height - filter_height + 2 * P)/S + 1
new_width = (input_width - filter_width + 2 * P)/S + 1

Padding:
Valid padding=no padding
Same padding=zeroes padding

stride for each dimension **(batch_size, height, width, depth)**
weight (for Filter or kernel) is **(height, width, input_depth, output_depth)**
bias is **(output_depth,)**
Input (for Image) is **(batch_size, height, width, depth)**
max_pool=x,ksize=[1,k,k,1],strides=[1,k,k,1],padding='SAME'

Remember in max pool k-size and strides are same

The CIFAR-10 dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

Better Upsampling(AUTOENCODER-MINIMISING SIZE OR DENOISING IMAGE)

1. **One approach is to make sure you use a kernel size that is divided by your stride, avoiding the overlap issue.**

2. **Another approach is to separate out upsampling to a higher resolution from convolution to compute features. For example, you might resize the image (using nearest-neighbor interpolation or bilinear interpolation)**

   Code
   Resize-convolution layers can be easily implemented in TensorFlow using tf.image.resize_images(). For best results, use tf.pad() before doing convolution with tf.nn.conv2d() to avoid boundary artifacts.

Transfer Learning

In an AlexNet, this would compute a 4096-D vector for every image that contains the activations of the hidden layer immediately before the classifier. We call these features **CNN codes**.

utils.load_image crops the input images for us, from the center

COLLABPRATIVE FILTERING:

- *Item-Item Collaborative Filtering*: "Users who liked this item also liked …"
- *User-Item Collaborative Filtering*: "Users who are similar to you also liked …"

Regularization techniques for preventing the overfilling of the data.

L2 regularization:

A simple hyper parammente is added to the loss to prevent our analogy to skinny jeans model which mean hardhsips sin training the small datasets.
beta(1/2(summ(w)^2))

Dropout:

Suppose if we have different layers but passing the randomly some data to the next layer as of multiplying the other ones with zeros.
So network will never rely on any given input.
It will be done in different epochs and remember ti drip data randomly.

keep_prb must be 1 when validationg the dataset.

AutoEncoder (For unsupervised learning)

To predict the data from image which is unlabelled.

Auto encoder de construct the data using hidden representation and reconstruct it sequentially using those representation.(encode and decode data)

It is a data compression also and it is bit lossy. As in the process of encoding the data again or reconstruction it again somme data get lost.

It is used for data demonising in which it reconstruct the data from the corrupted input version of it.

Example:
Variational Auto Encoder.

Total loss in this is :

Reconstruction loss + regularising loss (to see the data global means if we have handwritten images there must not be any different clusters for two writing or overfitting of the data. So keep the data global using regularising and prevent overfitting.)

We have to do reparamaterized to do back propagation for stochastic gradient descent i.e create randomness in input data for backprop.

In auto encoder there are encoder and decoder for image
So for encoding we will follow steps Like convolution and maxpooling for reducing the size of images and increasing the depth of images.,

For decoding we will do upsampling of images and decreasing the depth of images with convolution.

RNN:

let them no.s be from 1-12 we can pass it exactly like that but it is better to split it into two halves. Called as no. of seq.

No. Of sequences: so 2 no. of seq are passed 1-6 and 7-12 .

Batch Size: it is the no of seq we are using so it is 2.

Sequence length: Length of seq to feed in the network lets suppose 3 data in each batch so called mini-sequence.

Batch Numbers: