Machine Learning and deep learning learned representation from the data like background texture, nose etc.

Machine Learning

“searching for useful representations of some input data, within a predefined space of possibilities(hypothesis space), using guidance from a feedback signal.”

Hypothetical space->Set of predefined operation or possible hypothesis

Shallow learning-> 1 or 2 layers

Deep Network-> multistage information distillation process where input information purified from successive filters(layers)

Weights->transformation implemented by a layer is *parameterized* by its weights (Weights are also sometimes called the *parameters* of a layer.)

Initially deep learning prediction is bad or loss is high as weights were assigned random values. Loss function takes prediction and true value, computes loss and do back propagation using optimizer to decrease the loss and update weights accordingly. This all done during training.

Feature engineering in machine learning->manually engineer good layers of representations for their data. But this is done automatically by deep learning that’s why deep learning is better than the machine learning

These are the two essential characteristics of how deep learning learns from data:

the *incremental, layer-by-layer way in which increasingly complex representations are developed*,

and the fact that *these intermediate incremental representations are learned jointly*

To make the network ready for training, we need to pick three more things, as part

of the *compilation* step:

 *A loss function*—How the network will be able to measure its performance on

the training data, and thus how it will be able to steer itself in the right direction.

 *An optimizer*—The mechanism through which the network will update itself

based on the data it sees and its loss function.

 *Metrics to monitor during training and testing*—Here, we’ll only care about accuracy

(the fraction of the images that were correctly classified).

3D shape explanation:

If all data at once without batch then, then 0 axis is all data or sample axis

(sample, dim, dim) -> (60000, 28, 28)

If perform batching then 0 axis is batch axis

(batchsize dim, dim) -> (128, 28, 28) (if one batch contain 128 data points or sample points)

Batch formula -> batch = train\_images[128 \* n:128 \* (n + 1)] -> Illustration given below

for first batch n=0

128\*0:128\*(0+1)

0:128

Backpropagation:

Backpropagation starts with the final loss value and works backward from the top layers to the bottomlayers, applying the chain rule to compute the contribution that each parameterhad in the loss value.

WHOLE NEURAL NETWORK PROCESS:

the network, composed of layers that are chained together, maps the input data to predictions. The loss function

then compares these predictions to the targets, producing a loss value: a measure of how well the network’s predictions match what was expected. The optimizer uses this loss value to update the network’s weights.

About different tensors and layers used as per those tensors:

For instance, simple vector data, stored in 2D tensors of shape (samples, features), is often processed by *densely connected* layers, also called *fully connected* or *dense* layers (the Dense class in Keras). Sequence data, stored in 3D tensors of shape (samples, timesteps, features), is typically processed by *recurrent* layers such as an LSTM layer. Image data, stored in 4D tensors, is usually processed by 2D convolution layers (Conv2D).

SELECTION OF LOSS FUNCTION:

Binary crossentropy for a two-class classification

Categorical crossentropy for a many-class classification problem,

Mean squared error (mse) for a regression problem,

Connectionist temporal classification (CTC) for a sequence-learning problem

SELECTION OF ACTIVATION FUNCTION:

If more than two class classification and categorized the data -> Softmax

If between two classes and find probability-> sigmoid(score between 0 and 1)

WHY ACTIVATION FUNCTION IS REQUIRED:

If no activation function, then layers will learn linear transformation of the input data(less hypothesis space).

In order to get access to a much richer hypothesis space that would benefit from

deep representations, you need a non-linearity which we get using activation function or activation function. relu is the most popular activation function in deep learning

WORKFLOW IN KERAS:

***Developing with Keras: a quick overview***

1 Define your training data: input tensors and target tensors.

2 Define a network of layers (or *model* ) that maps your inputs to your targets.

3 Configure the learning process by choosing a loss function, an optimizer, and some metrics to monitor.

4 Iterate on your training data by calling the fit() method of your model.

NORMALIZATION:

Data centered to mean 0 and std 1

If different ranges-> normalize the data

When features in the input data have values in different ranges, each feature should be scaled independently(normalized) as a preprocessing step.

Otherwise vectorise the data

If softmax->one hot encode labels

K-Fold Validation:

If data samples are smaller in size like bouston housing dataset, then it is difficult to split training data into training and validation data. To do so, we use k fold validation where it splits available training data into k partions typically *K* = 4 or 5), instantiating *K* identical models, and training each one on *K* – 1 partitions while evaluating on the remaining partition. The validation score for the model used is then the average of the *K* validation scores obtained. In terms of code, this is straightforward.