How a neural network works?

A neural network has many layers. Each layer performs a specific function.
The purest form of a neural network has three layers:

1. The input layer: Input variables, sometimes called the visible layer.
2. The hidden layer: Layers of nodes between the input and output layers. There may be one or more of these layers.
3. The output layer: A layer of nodes that produce the output variables.

Each of these layers has a specific purpose. These layers are made up of nodes. There can be multiple hidden layers in a neural network according to the requirements. The input layer picks up the input signals and transfers them to the next layer. It gathers the data from the outside world.
We will have to train a neural network with some training data as well, before provide it with a particular problem.

What is node in neural network?

A node, also called a neuron, is a unit that has one or more weighted input connections, a transfer function that combines the inputs in some way, and an output connection. Nodes are then organized into layers to comprise a network.

In this model, three layers of nodes is used. The number of nodes in each layer differ in each try, so we can reach the minimum loss and the maximum validation accuracy. The first layer perform as a input layer that picks up the input data and transfers to the next layer.

This model is built using Keras.

What is Keras?

Keras is deep learning API developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy.

Keras models

There are two main types of models available in Keras:

the Sequential model, and the Model class used with the functional API.

Keras Sequential model is used for building this model. The Sequential model is a linear stack of layers. In this example, classifier.add() is used to define three dense layers in a Sequential model. Every dense layer has a units property. It is necessary to know what it is and how it works for each layer. **Units** are the number of neurons contained in each layer. It's a property of each layer.

Forth Try model has minimum loss and maximum accuracy. In this model the number of units for the first layer is set to 11, second layer has 8 units and the last layer has 1 units.

After defining our model and stacking the layers, we have to configure our model. We do this configuration process in the compilation phase.
Before training the model we need to compile it and define the loss function, optimizers, and metrics for prediction. Optimizer, loss, and metrics are the necessary arguments.

Here is how these necessary arguments selected:

Keras Model Optimization

```
optimizer='adam'
```

Optimizer that implements the Adam algorithm. Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments.

Loss Function in Keras

The purpose of loss functions is to compute the quantity that a model should seek to minimize during training.

```
loss='binary_crossentropy'
```

Binary crossentropy is one of Probabilistic losses That Computes the cross-entropy loss between true labels and predicted labels.

Keras Metrics

A metric is a function that is used to judge the performance of your model. Metric functions are similar to loss functions, except that the results from evaluating a metric are not used when training the model.

```
metrics=['accuracy']
```

Calculates how often predictions equal labels.

Model training is the next step.

We used `classifier.fit()`That trains the model for a fixed number of epochs.

In this step, there is two important Hyperparameter: batch size and epoch.

At first, let’s see what batch size and epoch is and what is the difference between a model parameter and a model hyperparameter.

What is a Model Parameter?

A model parameter is a configuration variable that is internal to the model and whose value can be estimated from data.

- They are required by the model when making predictions.
- They values define the skill of the model on your problem.
- They are estimated or learned from data.
- They are often not set manually by the practitioner.
- They are often saved as part of the learned model.

Parameters are key to machine learning algorithms. They are the part of the model that is learned from historical training data.

What is a Model Hyperparameter?

A model hyperparameter is a configuration that is external to the model and whose value cannot be estimated from data.

- They are often used in processes to help estimate model parameters.
- They are often specified by the practitioner.
- They can often be set using heuristics.
- They are often tuned for a given predictive modeling problem.

We cannot know the best value for a model hyperparameter on a given problem. We may use rules of thumb, copy values used on other problems, or search for the best value by trial and error.

What Is a Batch?

The batch size is a hyperparameter that defines the number of samples (rows of data) to work through before updating the internal model parameters.

Think of a batch as a for-loop iterating over one or more samples and making predictions. At the end of the batch, the predictions are compared to the expected output variables and an error is calculated. From this error, the update algorithm is used to improve the model, e.g. move down along the error gradient.

A training dataset can be divided into one or more batches.

When all training samples are used to create one batch, the learning algorithm is called batch gradient descent. When the batch is the size of one sample, the learning algorithm is called stochastic gradient descent. When the batch size is more than one sample and less than the size of the training dataset, the learning algorithm is called mini-batch gradient descent.

- **Batch Gradient Descent**. Batch Size = Size of Training Set
- **Stochastic Gradient Descent**. Batch Size = 1
- **Mini-Batch Gradient Descent**. 1 < Batch Size < Size of Training Set

In the case of mini-batch gradient descent, popular batch sizes include 32, 64, and 128 samples.

What Is an Epoch?

The number of epochs is a hyperparameter that defines the number times that the learning algorithm will work through the entire training dataset.

One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters. An epoch is comprised of one or more batches. For example, an epoch that has one batch is called the batch gradient descent learning algorithm.

You can think of a for-loop over the number of epochs where each loop proceeds over the training dataset. Within this for-loop is another nested for-loop that iterates over each batch of samples, where one batch has the specified “batch size” number of samples.

The number of epochs is traditionally large, often hundreds or thousands, allowing the learning algorithm to run until the error from the model has been sufficiently minimized.

What Is the Difference Between Batch and Epoch?

The batch size is a number of samples processed before the model is updated.

The number of epochs is the number of complete passes through the training dataset.

The size of a batch must be more than or equal to one and less than or equal to the number of samples in the training dataset.

The number of epochs can be set to an integer value between one and infinity. You can run the algorithm for as long as you like and even stop it using other criteria besides a fixed number of epochs, such as a change (or lack of change) in model error over time.

They are both integer values and they are both hyperparameters for the learning algorithm, e.g. parameters for the learning process, not internal model parameters found by the learning process.

There are no magic rules for how to configure these parameters. You must try different values and see what works best for your problem.

There is seven model with different number of units, batch size and epochs.

Used numbers in forth try

Finally, let’s make this concrete with given dataset.

We have a dataset with 8000 samples (rows of data) and you choose a batch size of 20 and 200 epochs. This means that the dataset will be divided into 400 batches, each with 20 samples (rows of data). The model weights will be updated after each batch of 20 samples. This also means that one epoch will involve 400 batches or 400 updates to the model. With 200 epochs, the model will be exposed to or pass through the whole dataset 200 times.

What is TenserBoard?

It is a machine learning toolkit that helps us visualize metrics such as loss and accuracy in training and validation data, model graphs, weights and biases.

In this Google Colab link, you can see 7 different model with different hyperparameters that lead us to the best machine learning model which has the minimum loss and maximum validation accuracy.

https://colab.research.google.com/drive/1xNt_sGRp9KIR4dI08ER6WrpdphyC2KYE#scrollTo=rlxrqwjFUXIo

as you can see in the link above, the best machine learning model for given database is forth one. Which has the following features:

First layer : 11 units

Second layer : 8 units

Output layer : 1 units

Batch size : 20

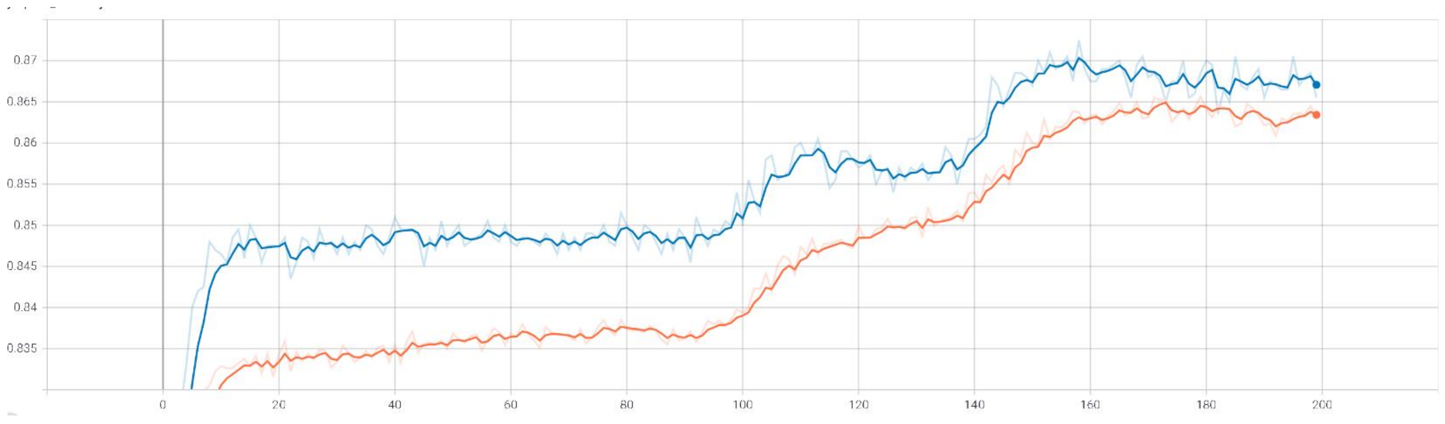
Epochs : 200

Accuracy : 0.8695

Validation accuracy : 0.8655

Loss : 0.3240

Validation loss : 0.3286



Author: Yasamin Rahnavard (Yas.rhd77@gmail.com)