**Iris Data Set Classification Problem**

we will use Iris Data Set Classification Problem for this demonstration. Iris Data Set is famous dataset in the world of pattern recognition and it is considered to be “Hello World” example for machine learning classification problems. It was first introduced by [Ronald Fisher](https://en.wikipedia.org/wiki/Ronald_Fisher), British statistician and botanist, back in 1936. In his paper *The use of multiple measurements in taxonomic problems,*he used data collected for three different classes of Iris plant: *Iris setosa*, *Iris virginica,*and *Iris versicolor*.

This dataset contains 50 instances for each class. What is interesting about it is that first class is linearly separable from the other two, but the latter two are not linearly separable from each other. Each instance has five attributes:

* Sepal length in cm
* Sepal width in cm
* Petal length in cm
* Petal width in cm
* Class (*Iris setosa*, *Iris virginica, Iris versicolor*)

In next chapter we will build Neural Network using Keras, that will be able to predict the class of the Iris flower based on the provided attributes.

**Code**

Keras programs have similar to the workflow of TensorFlow programs. We are going to follow this procedure:

* Import the dataset
* Prepare data for processing
* Create the model
* Training
* Evaluate accuracy of the model
* Predict results using the model

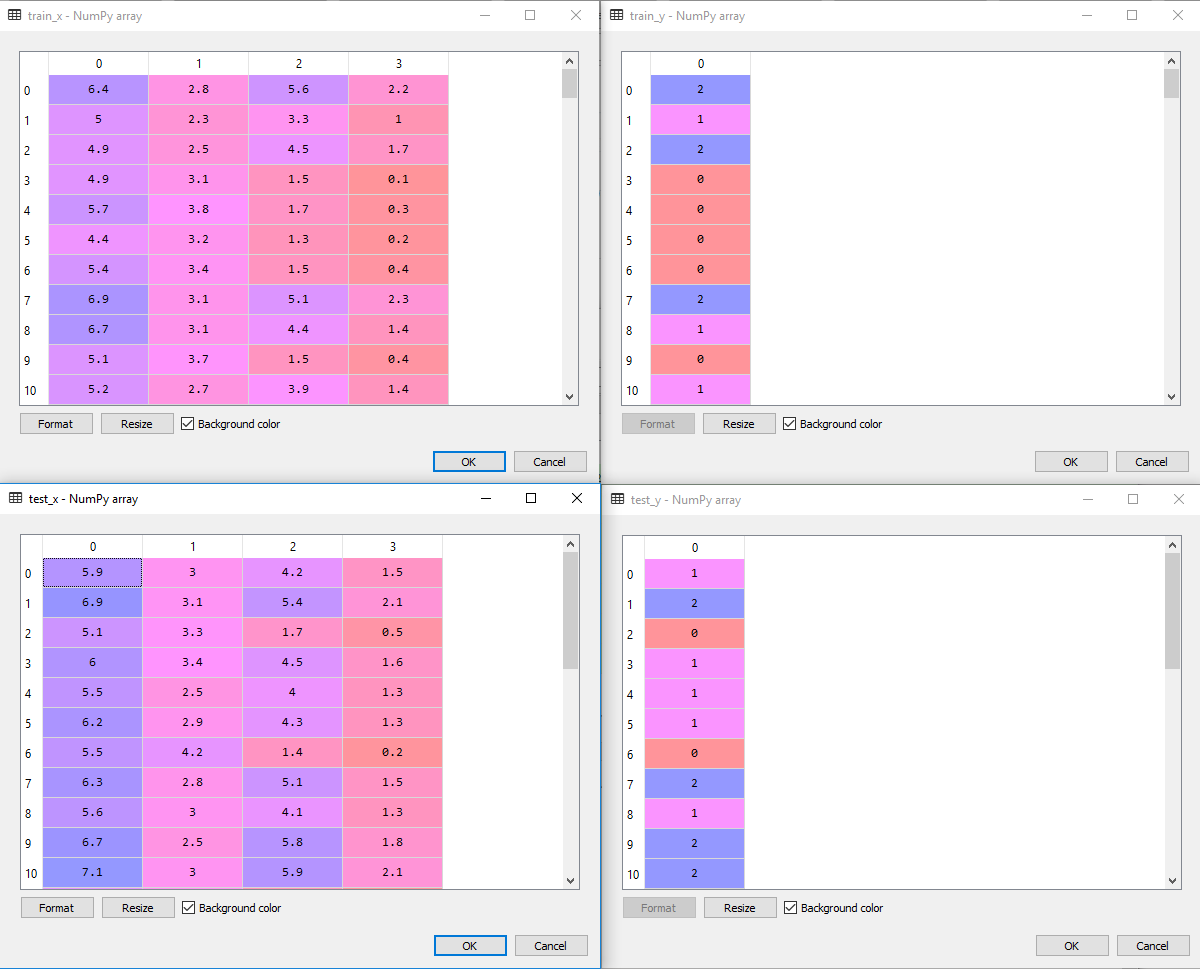
Training and evaluating processes are crucial for any Artificial Neural Network. These processes are usually done using two datasets, one for training and other for testing the accuracy of the trained network. In the real world, we will often get just one dataset and then we will split them into two separate datasets. For the training set, we usually use 80% of the data and another 20% we use to evaluate our model. Here is the list of the libraries that we need to import.

|  |  |
| --- | --- |
|  | # Importing libraries |
|  | from keras.models import Sequential |
|  | from keras.layers import Dense |
|  | from keras.utils import np\_utils |
|  | import numpy |
|  | import pandas as pd |

As you can see we are importing Keras dependencies, *NumPy*and P*andas. NumPy* is the fundamental package for scientific computing and *Pandas*provides easy to use data structures and data analysis tools.

After we imported libraries, we can proceed with importing the data and preparing it for the processing. We are going to use *Pandas* for importing data:

|  |  |
| --- | --- |
|  | # Import training dataset |
|  | training\_dataset = pd.read\_csv('iris\_training.csv', names=COLUMN\_NAMES, header=0) |
|  | train\_x = training\_dataset.iloc[:, 0:4].values |
|  | train\_y = training\_dataset.iloc[:, 4].values |
|  |  |
|  | # Import testing dataset |
|  | Write code like above |
|  |  |
|  |  |

Firstly, we used *read\_csv*function to import the dataset into local variables, and then we separated inputs *(train\_x, test\_x)* and expected outputs *(train\_y, test\_y)*creating four separate matrixes. Here is how they look like:

However, our data is not prepared for processing yet. If we take a look at our expected output values, we can notice that we have three values: 0, 1 and 2. Value 0 is used to represent Iris setosa, value 1 to represent Iris versicolor and value 2 to represent virginica. The good news about these values is that we didn’t get string values in the dataset. If you end up in that situation, you would need to use some kind of encoder so you can format data to something similar as we have in our current dataset. For this purpose, one can use ***[LabelEncoder](http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html" \t "_blank)***of sklearn library. Bad news about these values in the dataset is that they are not applicable to *Sequential* model. What we want to do is reshape the expected output from a vector that contains values for each class value to a matrix with a boolean for each class value. This is called [**one-hot encoding**](https://en.wikipedia.org/wiki/One-hot). In order to achieve this, we will use *np\_utils*from the Keras library:

|  |  |
| --- | --- |
|  | # Encoding training dataset |
|  | encoding\_train\_y = np\_utils.to\_categorical(train\_y) |
|  |  |
|  | # Encoding training dataset |
|  | encoding\_test\_y = np\_utils.to\_categorical(test\_y) |

If you still have doubt what one-hot encoding is doing, observe image below. There are displayed *train\_y* variable and *encoding\_train\_y* variable. Notice that first value in *train\_y* is 2 and see the corresponding value for that row in *encoding\_train\_y.*

Once we imported and prepared the data we can create our model. We already know we need to do this by using *Sequence*and *Dense* class. So, let’s do it following tutorial given. Remember input\_dim this variable value represents the column number or features used in dataset

We need to create:

* one input layer with four nodes, because we are having four attributes in our input values
* two hidden layers with ten neurons each
* one output layer with three neurons, because we are having three output classes

In hidden layers, neurons use [**Rectifier activation function**](http://rubikscode.net/2017/11/20/common-neural-network-activation-functions/), while in output layer neurons use Softmax activation function (ensuring that output values are in the range of 0 and 1). After that, we compile our model, where we define our [**cost function**](http://rubikscode.net/2018/01/15/how-artificial-neural-networks-learn/) and optimizer. In this instance, we will use Adam [**gradient descent optimization algorithm**](http://rubikscode.net/2018/01/15/how-artificial-neural-networks-learn/) with a logarithmic cost function (called *categorical\_crossentropy*in Keras).

|  |
| --- |
|  |
|  |
|  | # Compiling model |
|  | model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy']) |

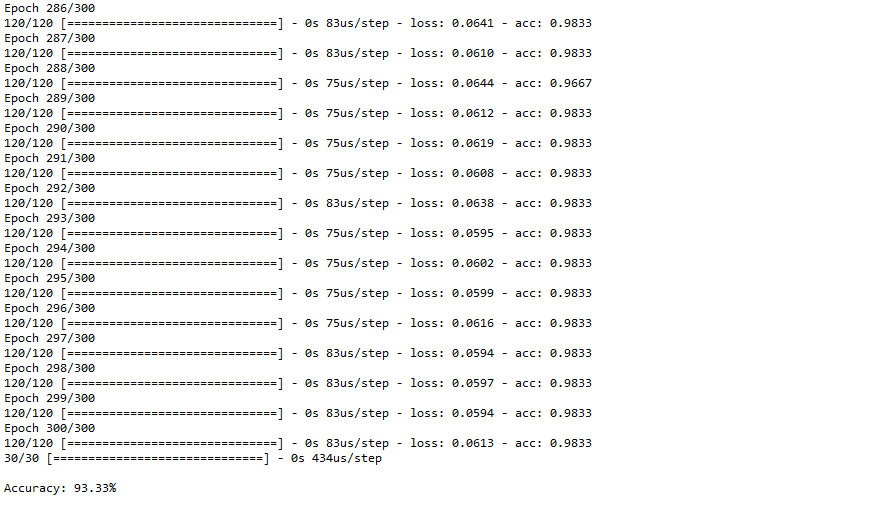
Finally, we can train our network:

|  |  |
| --- | --- |
|  | # Training a model |
|  | model.fit(use required parameters) |

And evaluate it:

|  |  |
| --- | --- |
|  | # Evaluate the model |
|  | scores = model.evaluate(use required parameters) |
|  | print("\nAccuracy: %.2f%%" % (scores[1]\*100)) |

If we run this code, we will get these results:



Assigned Task

Implement prediction model, you will provide the data for a particular flower Neural network has to tell which flower it belong, it will show flower name. You have to use model.predict(your data) to get prediction for any given data.

Hints: You have to use encoding decoding technique here, so that output can be decoded into flowers name