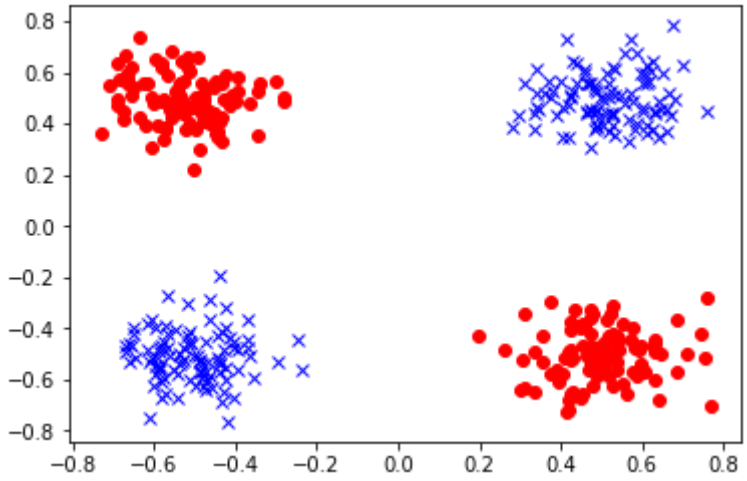
Visualizing The Non-linearity of Neural Networks

Cihan Soylu

Jul 14, 2019 · 5 min read ★

In this article I will go over a basic example demonstrating the power of non-linear activation functions in neural networks. For this purpose, I have created an artificial dataset. Each data point has two features and a class label, 0 or 1. So we have a binary classification problem. If we call the features x_1 and x_2, then the plot of the data in (x_1, x_2)-space is as follows:



Here the red points are corresponding to the negative class and the blue points are corresponding to the positive class. Notice that the data is not linearly separable, meaning there is no line that separates the blue and red points. Hence a linear classifier wouldn't be useful with the given feature representation. Now we will train a neural network with one hidden layer with two units and a non-linear `tanh` activation function and visualize the features learned by this network.

In order to create the model, I will use Tensorflow 2.0 and `tf.keras` :

```
outputs = tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)(x)
model = tf.keras.Model(inputs=inputs, outputs=outputs)

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

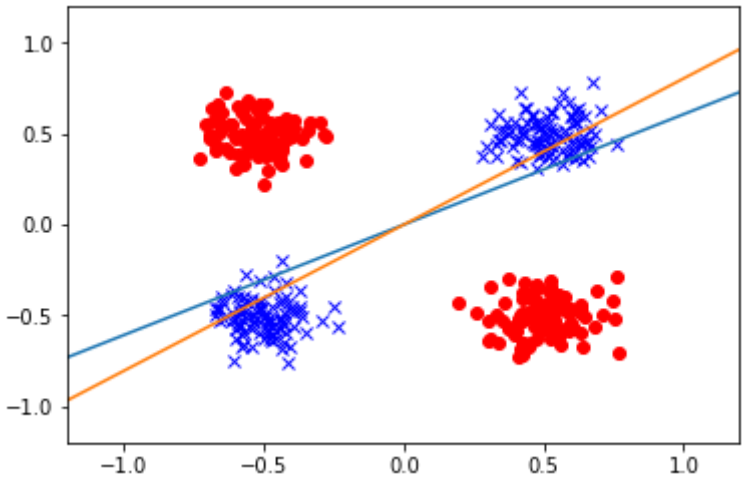
Let's denote the weights and the biases of the hidden layer as follows:

$$W^{(1)} = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \quad b^{(1)} = [b_1, b_2]$$

The initial values of the weights and biases define two lines in the (x_1, x_2)-space,

$$x_1 W_{11} + x_2 W_{21} + b_1 = 0$$
$$x_1 W_{12} + x_2 W_{22} + b_2 = 0$$

These initial lines before training together with the data are shown below:



Data and the initial lines defined by the hidden layer in (x_1,x_2)-space

Notice that these two initial lines are not dividing the given two classes nicely. Now let's train the model and see what happens to these lines.

```
model.fit(x_train, y_train, batch_size = 16, epochs=100)
```

Our training accuracy is 100% after 100 epochs so the model is classifying all points in the training data correctly.

Note that you can get the parameters of a keras model by using `model.weights` which returns a list of weights and biases:

```
[<tf.Variable 'dense/kernel:0' shape=(2, 2) dtype=float32, numpy=
array([[-3.2753847, 3.2302036], [ 3.3264563, -3.2554653]],
dtype=float32)>, <tf.Variable 'dense/bias:0' shape=(2,)
dtype=float32, numpy=array([1.5562934, 1.5492057], dtype=float32)>,
<tf.Variable 'dense_1/kernel:0' shape=(2, 1) dtype=float32, numpy=
array([[2.625529], [2.670275]], dtype=float32)>, <tf.Variable
'dense_1/bias:0' shape=(1,) dtype=float32, numpy=array([-2.0918093],
dtype=float32)>]
```

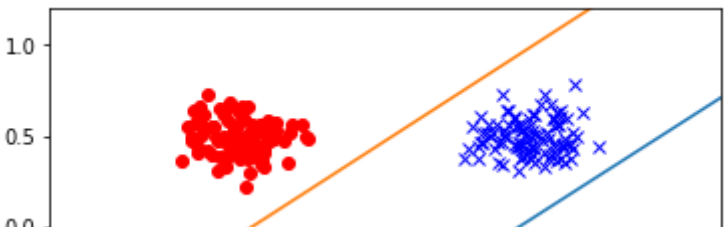
The first two elements in this list are the weights and biases of the hidden layer and the last two elements are weights and biases of the output layer. All these elements are tensors and you can get the values as numpy arrays using numpy() method. For example, the following will give you the weights of the hidden layer as a numpy array,

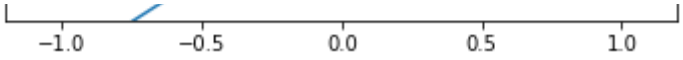
```
model.weights[0].numpy()

array([[-3.2753847,  3.2302036],        [ 3.3264563, -3.2554653]],
dtype=float32)
```

The coefficients of the first line are -3.27 and 3.32; and the coefficients of the second line are 3.23 and -3.25. You can see their biases by running `model.weights[1].numpy()`

Let's visualize the lines defined by the learned weights and biases of the hidden layer,





The data and the lines defined by the hidden layer after training in (x_1,x_2)-space

As you can see now the lines divide the space in a way that the classes are contained in separate regions. The coefficients of these lines are telling us the positive side of each line. For the blue line the positive direction is the upper side and for the orange line the positive side is the downside. So the blue points are in the positive side of both lines and the red points are on the negative side of one line and the positive side of the other.

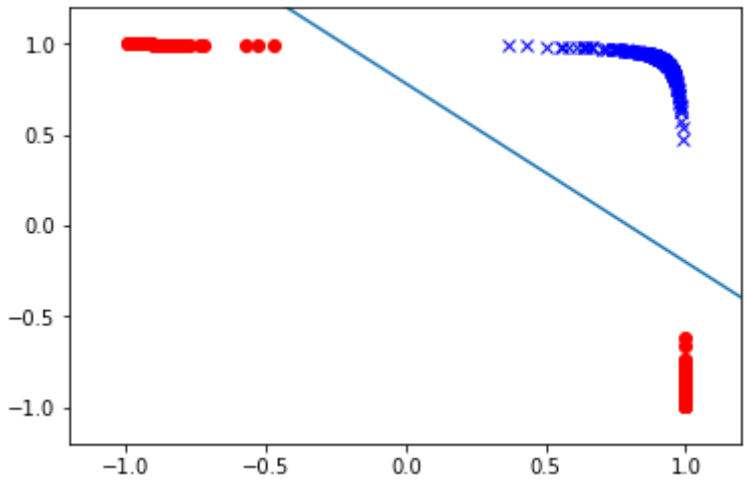
Now let's apply the non-linear activation function `tanh` and visualize the data in the new feature space (a_1, a_2). The activations of the hidden layer are computed as follows,

$$a_1 = \tanh(x_1 W_{11} + x_2 W_{21} + b_1)$$
$$a_2 = \tanh(x_1 W_{12} + x_2 W_{22} + b_2)$$

For each data point the arguments of the `tanh` are determined by the position of the data point in relation to the above lines. We can think of a_1 and a_2 as the new features and (a_1,a_2)-space as the new feature space. The weights and the bias of the output layer define a line in this new feature space given by the following equation

$$a_1 W_1 + a_2 W_2 + b = 0$$

This line together with the data in this new feature space is plotted below,



Note that in this new feature space our data becomes linearly separable and the line defined by the output layer separates the two classes. The blue points has both a_1 and a_2 coordinates positive because those points in the (x_1, x_2) space are on the positive side of both lines defined by the hidden layer parameters and after applying `tanh` both coordinates are positive. For the red points one of a_1 and a_2 is positive because in the (x_1, x_2)-space the red points are on the positive side of only one of the lines defined by the hidden layer parameters and depending on this line they have only one of their coordinates in the new feature space positive and the other is negative. This explains the data plot in (a_1,a_2)-space above.

Conclusion: Neural networks learn a new representation of the data which makes it easy to classify with respect to this new representation.

Here is a link to the google colab notebook I used for this article.

Thanks for reading.

Machine Learning Deep Learning Neural Networks Data Science