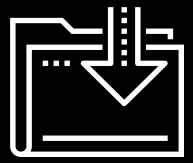
Neural Network Models in the Real World

Data Boot Camp

Lesson 21.2




W
E
L
C
O
M
E
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6
-1
0
1

Class Objectives

By the end of this lesson, you will be able to:

01

Implement deep neural network models using TensorFlow.

02

Explain how different neural network structures change algorithm performance.

03

Save trained TensorFlow models for later use.

Everyone Do:

Over the Moon on Basic Neural Networks

When input data is not linearly separable, our model struggle to classify our data points.

How do we address that?

Expanding our Neural Network Capabilities by Rebuilding our Basic Neural Network.

Time to Code

Rebuild our Neural Network

Suggested Time:

5 minutes

Everyone Do: Over the Moon on Basic Neural Networks

<time to code>

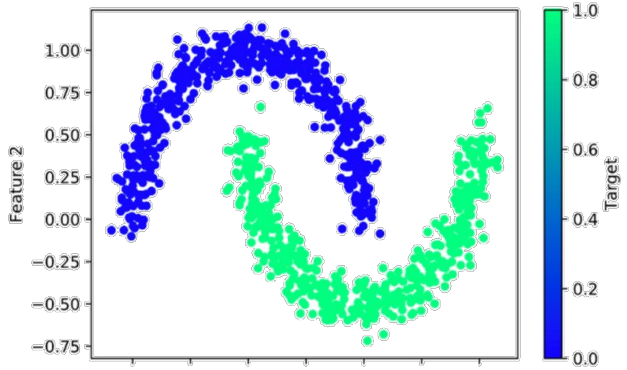
➔ **Objective:**
Rebuild our basic neural network in Jupyter notebook and apply it to a non-linear dataset.

➔ **Dataset:**
`sklearn.datasets`.make_moons

➔ **Code to produce the dataset:**

```
# Creating dummy nonlinear data
X_moons, y_moons = make_moons(n_samples=1000, noise=0.08, random_state=78)
```

➔ **Plot:**



Happy Coding!

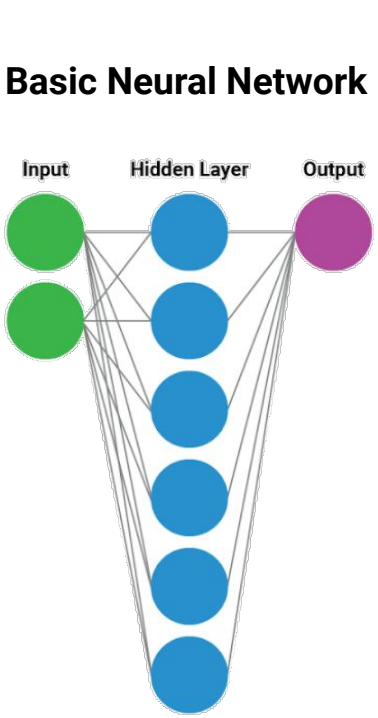
Instructor Demonstration

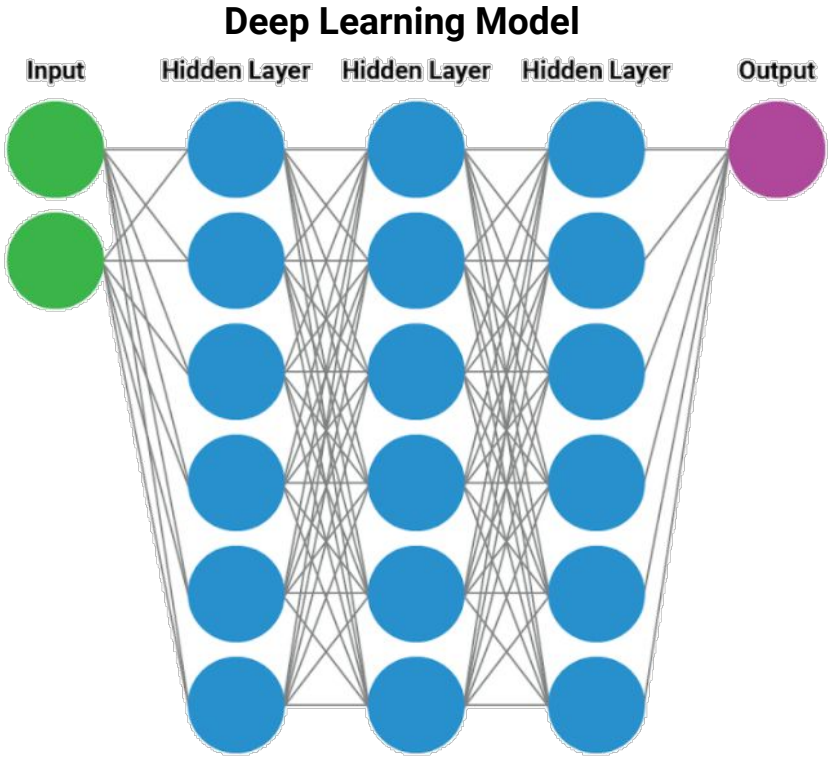
Getting Deep with Deep Learning Model

Why basic neural network models are limited?

They are engineered to evaluate input values only once before they are used in an output classification, limiting to interpret only simple linear relationships, and data with few confounding factors, or factors that have hidden effects on more than one variable.

Instructor Do: Getting Deep with Deep Learning Models

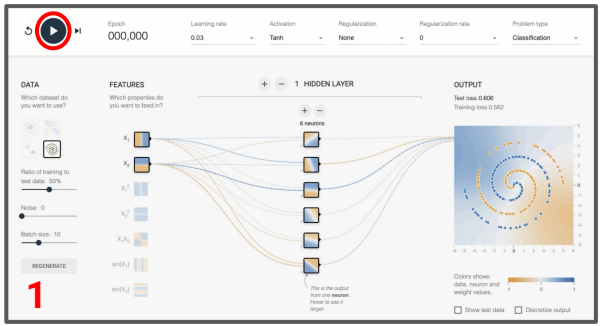


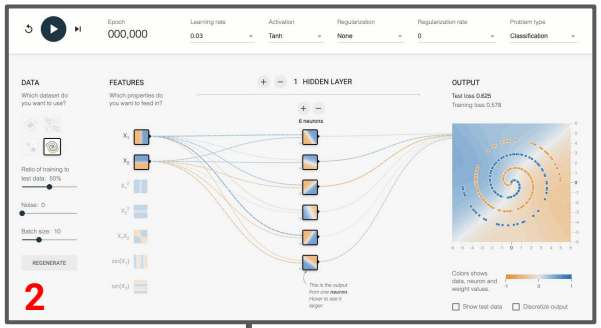


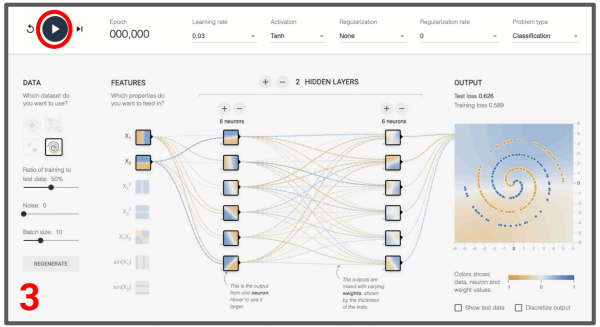
Back to Playground

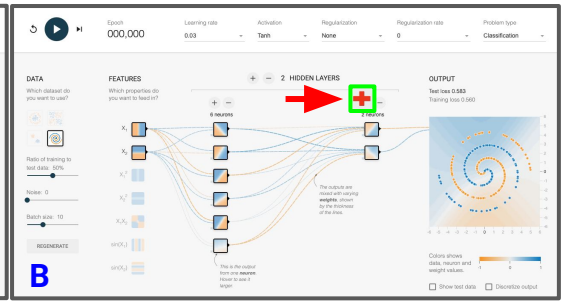
Instructor Do: Getting Deep with Deep Learning Models

- The easiest way to conceptualize the performance differences between basic neural network and deep learning models is to return to the TensorFlow Playground.









Activity: Back to the Moon

In this activity, you will try to build a deep learning classification model that can adequately predict the class from our moons dummy dataset.

Suggested Time:

20 minutes

Instructions:

Activity: Back to the Moon

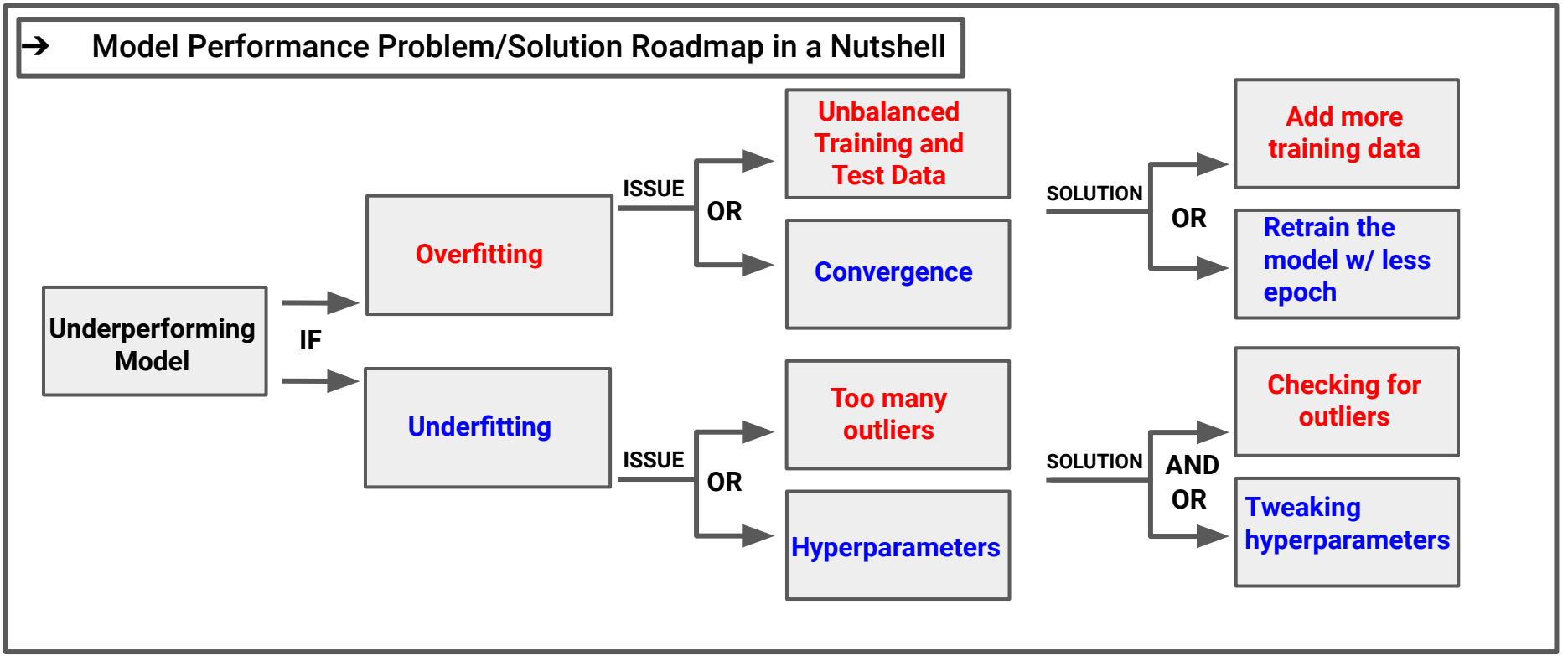
- Upload the starter notebook to Google Colab and run the cells to recreate the moons dummy dataset.
- Create your Keras Sequential model and add more than one Dense hidden layer to create a deep learning model.
    - **Notes**:
        - Only your first Dense layer uses the *input_dim* parameter.
        - All of your hidden layers should use the "relu" activation function.
- Compile your model and train the deep learning model on at least 100 epochs.
- Evaluate the performance of your model by calculating the loss and predictive accuracy of the model on your test dataset.
- **Bonus**:
    - If time permits, try recreating your deep learning model with a different set of hidden layers and neurons, and then evaluate the performance of your new model. Are you able to achieve 100% predictive accuracy?

Time’s Up! Let’s Review.

Instructor Demonstration

Getting Hands on With Model Optimization

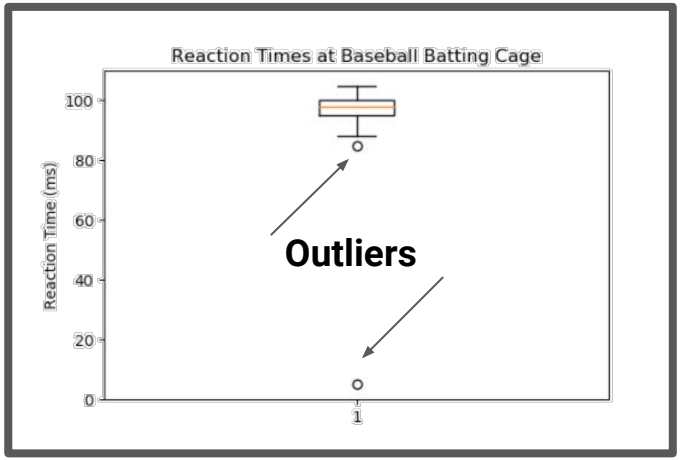
Instructor Do: Getting Hands on With Model Optimization



Instructor Do: Getting Hands on With Model Optimization

- Checking for outliers and adjusting hyperparameters.

➔ The easiest way to check for outliers is qualitatively using a **boxplot**, or quantitatively by applying the **1.5*IQR rule**.



Hyperparameter Adjustment	Pros	Cons
Add more neurons	Speeds up model, may reduce loss	More computational resources required
Add more layers	Considers more interactions between variables	More computational resources required
Change Activation Function	Drastically changes how a model interprets inputs	Not all new interpretations are effective
Add more epochs	Increase likelihood that model will achieve optimal weight coefficients	Increased risk of overfitting

➔ List of model optimizations technique.

Instructor Demonstration

Take the Guesswork out of Model Optimization

What is Keras Tuner?

The Keras Tuner is a library that helps you pick the optimal set of hyperparameters for your model. The process of selecting the right set of hyperparameters for your machine learning (ML) application is called *hyperparameter tuning* or *hypertuning*.

Lets see some Code using Google Colab

Activity: Giving Your Model Building a Tune-Up

In this activity, you will use keras-tuner to create a model that can adequately predict Scikit-learn's *make_circles* dataset.

Suggested Time:

20 minutes

Instructions:

Activity: Giving Your Model Building a Tune-Up

- Run the the starter code provided in Google Colab to create the circles dummy dataset.
- Convert the circles dataset to a dataframe and plot the circles dataset using Pandas.
- Create a method that creates and compiles a new Sequential deep learning model with hyperparameter options. Be sure to include the following features:
    - Allow kerastuner to select between relu and tanh activation functions for each hidden layer.
    - Allow kerastuner to decide from 1 to 30 neurons in the first dense layer.
        - **Note**: To limit the tuner runtime, increase your *step* argument to at least 5.
    - Allow kerastuner to decide from 1 to 5 hidden layers and 1 to 30 neurons in each dense layer.
- Import the kerastuner library and create a Hyperband tuner instance. Your tuner instance should use the following parameters:
    - The *objective* is "val_accuracy"
    - *max_epochs* equal to 20
    - *hyperband_iterations* equal to two.
- Run the kerastuner search for best hyperparameters over 20 epochs.
- Retrieve the top 3 model hyperparameters from the tuner search and print the values.
- Retrieve the top 3 models from the tuner search and compare their predictive accuracy against the test dataset.

Time’s Up! Let’s Review.

Countdown timer

15:00

(with alarm)

Instructor Demonstration

Getting Real with Neural Network Datasets

When building machine learning models, most of the design effort is not writing code to build the complex model. Rather, most of the effort is preprocessing and cleaning up the input data.

What is one-hot encoding?

Instructor Do: Getting Real with Neural Network Models

- One-hot encoding method.
  - It transform our categorical into vectors of zeros and ones. The length of these vectors is equal to the number of classes or categories that our model is expected to classify.

1. Labels are transformed or encoded into vectors.
2. Each of these vectors elements is equal to output categories.
3. Each index within the vector corresponds to one the 4 categories.

Eye_Color
Blue
Brown
Brown
Brown
Hazel
Green
Hazel
Brown
Blue
Brown

Blue = [1, 0, 0, 0]

Brown = [0, 1, 0, 0]

Hazel = [0, 0, 1, 0]

Green = [0, 0, 0, 1]

Eye_Color:Blue	Eye_Color:Brown	Eye_Color:Hazel	Eye_Color:Green
1	0	0	0
0	1	0	0
0	1	0	0
0	1	0	0
0	0	1	0
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0
0	1	0	0

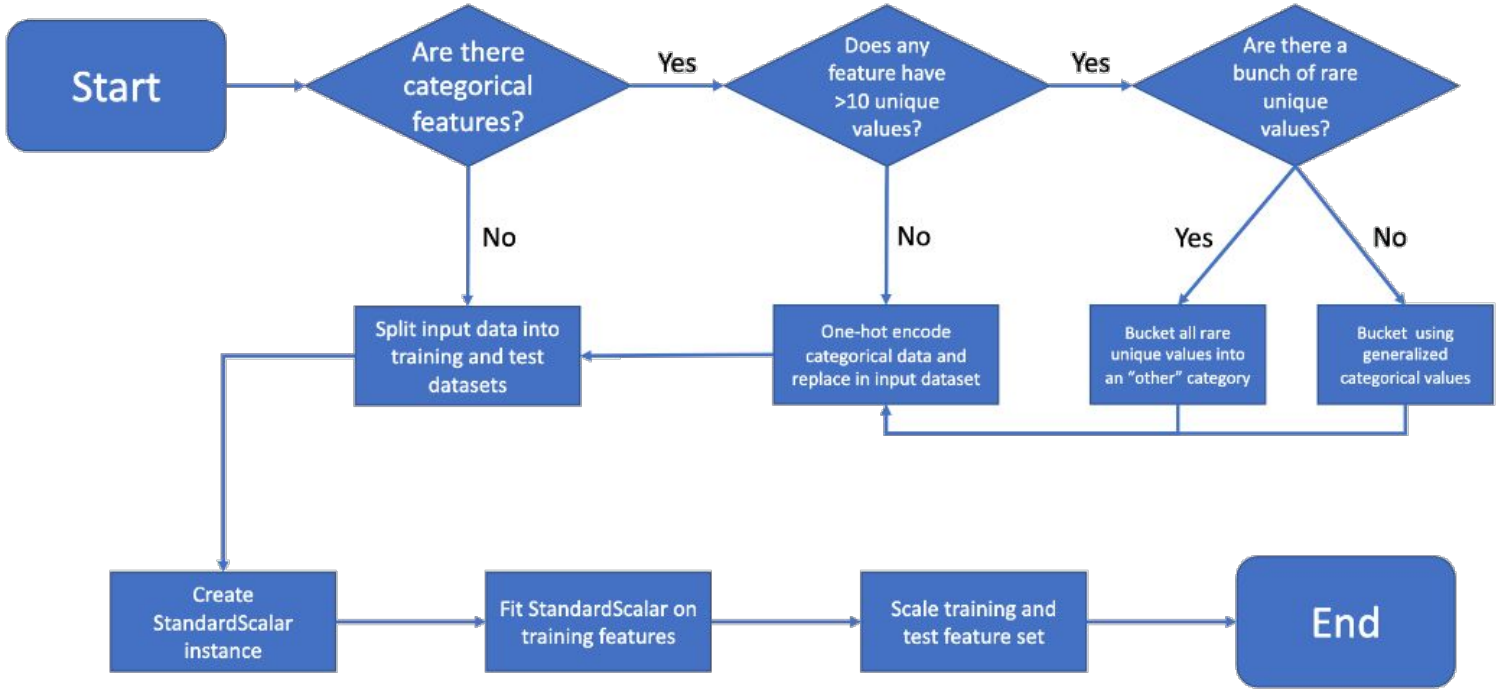
Time to Code

Getting Real with Neural Network Datasets (OneHotEnconder)

Suggested Time:

5 minutes

Instructor Do: Getting Real with Neural Network Models



Activity: Detecting Diabetes through Deep Learning

In this activity, you will preprocess a medical dataset and create a deep learning model capable of predicting whether a patient will be diagnosed with diabetes.

Suggested Time:

20 minutes

Instructions:

Activity: Detecting Diabetes through Deep Learning

- Upload the starter notebook to Google Colab, and run the code to import the dependencies and load the diabetes dataset.
- Separate the diabetes Outcome target from the other features in the dataset
- Split the features and target into training and test datasets
- Preprocess the input data accordingly:
  - If preprocessing categorical data use Scikit-learn's OneHotEncoder module.
  - If preprocessing numerical data use Scikit-learn's StandardScaler module.
- Define a deep learning model with the following features:
  - A first dense layer with 8 inputs and the "relu" activation function
  - A second dense layer with at least 8 neurons and the "relu" activation function
  - An output layer with one neuron and the "sigmoid" activation function
- Compile and train the model across no more than 100 epochs
- Evaluate the performance of the deep learning model by calculating the test loss and predictive accuracy.

Time’s Up! Let’s Review.