Tensorflow 2.0

Loïc Roldán Waals

Table of Contents

Knowledge Check

- Data science basics
- Data prep and cleaning
- Neural net basics
- Mathematics behind:
  - Optimisation functions
  - Activation functions
  - Back- and forward propagation
  - Regularisation
- Evaluation metrics

How to build an actual Network?

- Raw matrix multiplications?
- Computing backprop of complicated networks?
- Applying dropout (and then do backwards prop)?

For an example check the file called “Building your Deep Neural Network - Step by Step.ipynb” that was taken from Andrew Ng’s Coursera course that can be found here: https://www.coursera.org/specializations/deep-learning

Way to much effort for experimenting quickly. Hence...

Deep Learning Frameworks

TensorFlow

Keras

PyTorch

Caffe2

Deep Learning Frameworks

TensorFlow

Keras

PyTorch

Caffe2

Building a Model - Setup

1. Build a skeleton (you need this for any model)

```
import tensorflow as tf

# create model
model = tf.keras.Sequential()

# specify the first hidden layer
model.add(tf.keras.layers.Dense(64, activation='relu', input_shape=[10,]))
### ADD MORE LAYERS HERE ###

# compile the model
model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])
```

2. Design the architecture of your model
    a. How many layers with how many nodes?
    b. What type of layers
    c. What hyperparameters do I choose?

Building a Model - Layers

Documentation is very well written: https://keras.io/layers/core/

Most commonly used layers:

- Dense; the standard fully connected layer
- Dropout; a separate layer responsible for dropout regularisation
- BatchNormalisation; to normalise the previous layer's activations
- Flatten; for when you need to reduce dimensions in your network
- Convolutional layers; for when your input data is an image or text
- Embedding; especially used in NLP

Building a Model - Tips

- For activations just use ‘relu’ for the hidden layers and either ‘softmax’ or ‘sigmoid’ on the output layers
- For the optimiser ‘Adam’ usually works fine
- Start with a small network to see if everything works
- Use ModelCheckpoint callbacks to save your models to disk (prevents losing hours worth of training)
- Install livelossplot to see your model train in real time (pip install livelossplot); be sure to import it as: `from livelossplot.tf_keras import PlotLossesCallback`
- If you’re using dropout in your model be sure to test it with the *evaluate* method! Otherwise your model will not use all the nodes.
- Most of the errors you will encounter will be due to incompatible dimensions between layers

Practice