TensorFlow in R Programming: A Beginner's Primer

Applied Machine Learning Conference — Tom Tom Festival 2019
Presenter: Pierre DeBois, Zimana Analytics
Charlottesville, Virginia
April 11th 2019

Overview

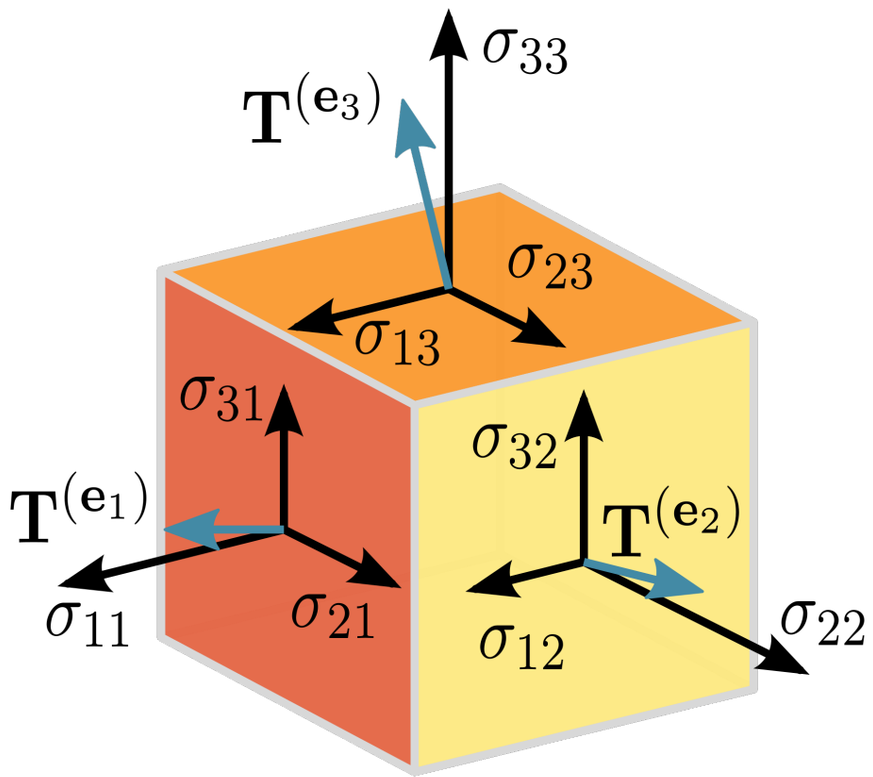
- Basics of a Tensor & TensorFlow
- Data Exploration & R Programming Syntax Basics
- Quick Demo

TensorFlow Is....

- General purpose numeric library for deep learning

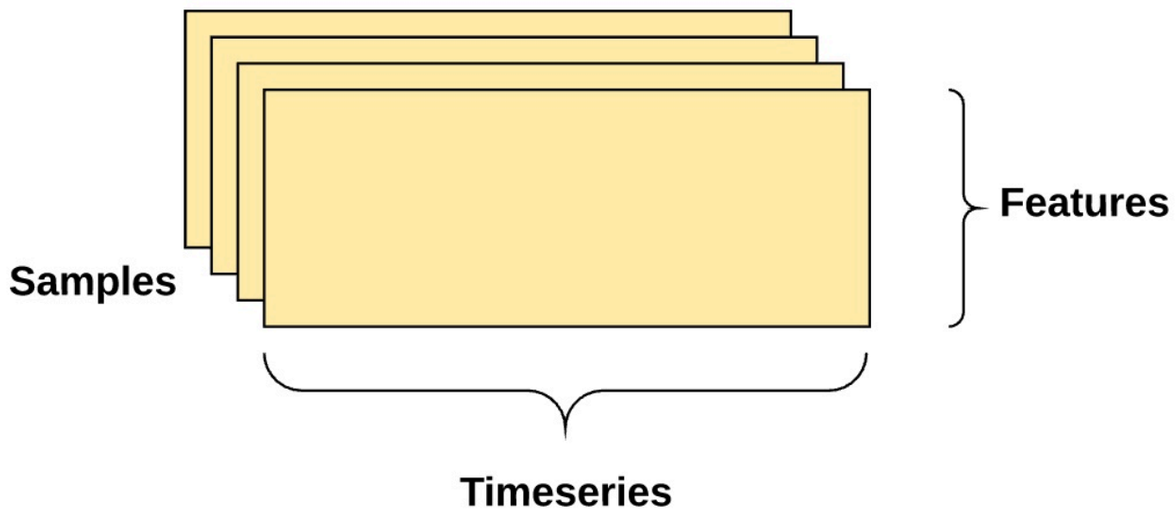
- Tensor: a multidimensional array of numbers; An epoch meant to apply a mathematical representation of independent variables

- IRL Tensor arrays reflect observations of product/service/ events



Tensor Array

- 0D - Scalar (ex. 5 meters, 55 mph)
- 1D - Scale and direction (5 meters north, 55 mph south)
- 2D
- 3D
- 4D
- Vector data (sample, feature1, feature2, ….)



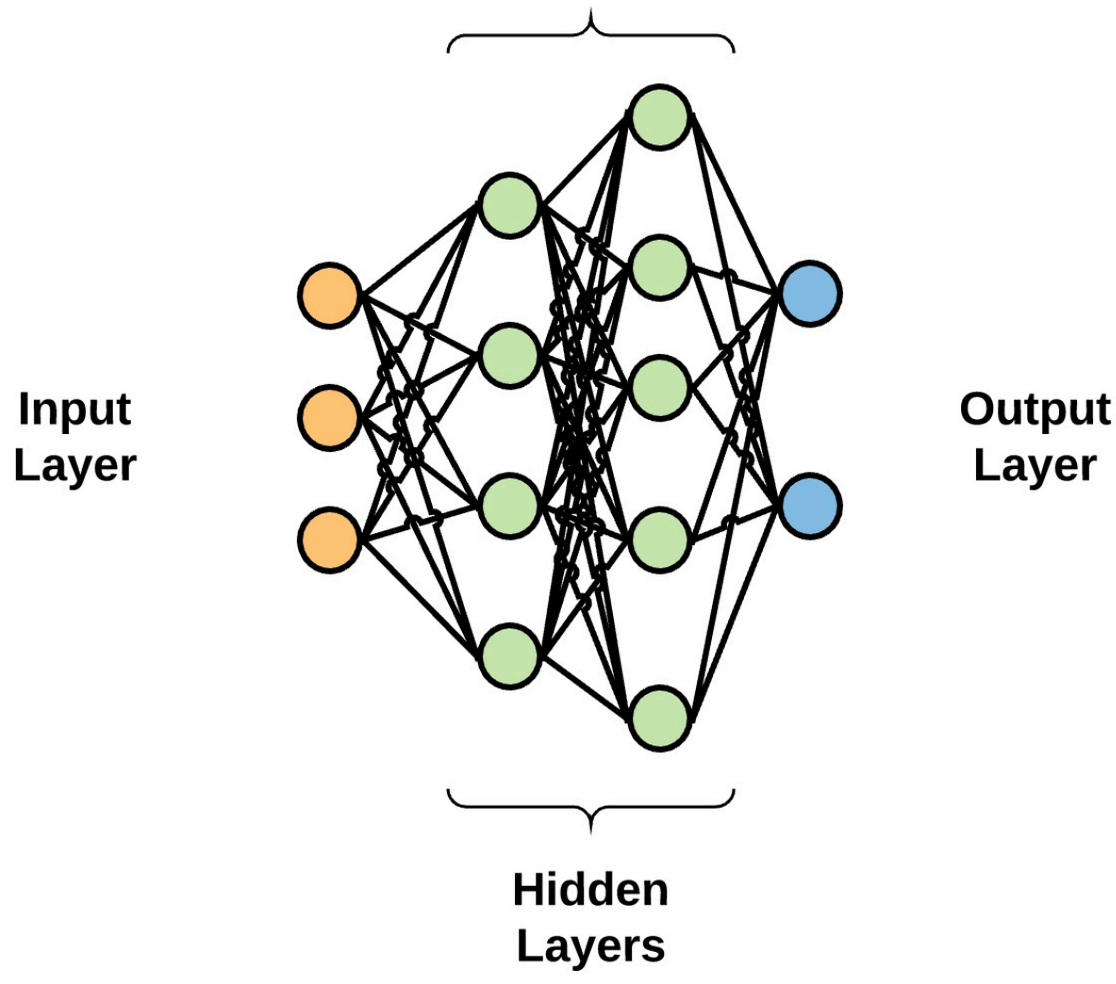
TensorFlow Network

- Input layer: Data

- Hidden layers: Geometric transformation of the input layer data until it approximates the output.

- Output layer: Predicted output from the hidden layers

- **Continuous** (i.e. price), **Binary** (yes/no), or **Categorical** (identified image or sound)



Applications

- Time Series Prediction
- Optimize Features of a product/service/event
- Customer churn predictions
- Image Recognition
- Voice Recognition
- Prediction of complex spatial or sequence dependencies

TensorFlow & Programming

- Usually with Keras library for running neural networks at high level (other backends - Theano & CNTK)

- JavaScript version for running data at the browser (script or npm)

- R (and Python) can apply statistical models, with R allowing graphic techniques

- Useful for programmers familiar with the subject behind the data

- Gives developers Tensorflow / Keras functions with the vast libraries available for R

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3
                  carb
Mazda RX4            4
Mazda RX4 Wag        4
Datsun 710           1
Hornet 4 Drive       1
Hornet Sportabout    2
Valiant              1
>
```

```
TimeSeriesTrial.R | ClusterTrial.R* | R data sets | R data sets
Source on Save                                   Run   Source
 9 #cluster and factoextra are a must
10 library(cluster)
11 library(factoextra)
12 #start with importing data and viewing for anomallies
13 data("mtcars")
14 head(mtcars)
15 tail(mtcars)
16 describe(mtcars)
```

Examine Data Structure

- Unstructured data is often in data lakes and other sources
- Be prepared to examine
- Move categories out of the the datafields
- Remove special characters
- Rely on the knowledge of your data

Sample	Category	Numerical
1	Corvette	1
2	Corvette	1
3	F-series	2
4	Passport	3
5	Jetta	4
6	Passport	3
7	Jetta	4

Sample	Corvette	F-series	Passport	Jetta
1	1	0	0	0
2	1	0	0	0
3	0	2	0	0
4	0	0	3	0
5	0	0	0	4
6	0	0	3	0
7	0	0	0	4

1. Transform Data for Input Layer

- Set Training & Test Data
- Reshape array table
- Rescale data
  - standardized (mean 0, s.v. 1)
  - normalized (Range 0–1)

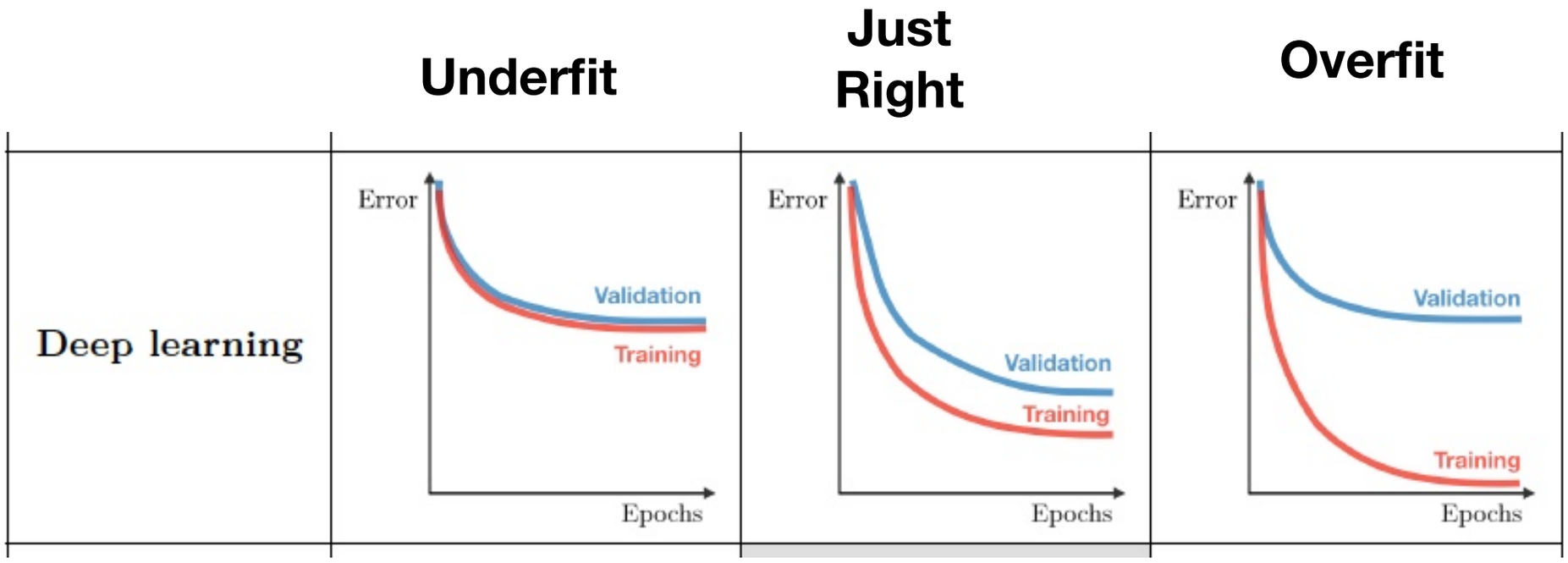
```r
library(keras)
#mnist - image dataset provided via keras
mnist <- dataset_mnist()
x_train <- mnist$train$x
y_train <- mnist$train$y
x_test <- mnist$test$x
y_test <- mnist$test$y
#
# reshape - 28 column , so 28x28 = 784
#array_reshape maps array to a "layer"
#
x_train <- array_reshape(x_train, c(nrow(x_train), 784))
x_test <- array_reshape(x_test, c(nrow(x_test), 784))
#
# rescale - turns training data into 0-1; Grayscale
# values of images is 255
#
x_train <- x_train / 255
x_test <- x_test / 255
#
#one hot encode - turns a class into a binary class
#
y_train <- to_categorical(y_train, 10)
y_test <- to_categorical(y_test, 10)
#
```

2. Select Core Layers, Then Compile

- Core layers (Dense)
- Convolutional Layers (image, voice)
- Pooling layers
- Activation Layers
- Dropout Layers (optimization)

```
#
model <- keras_model_sequential()
model %>%
  layer_dense(units = 256, activation = 'relu', input_s
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 10, activation = 'softmax')
summary(model)
#compile the model with loss function
model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(),
  metrics = c('accuracy')
)
```

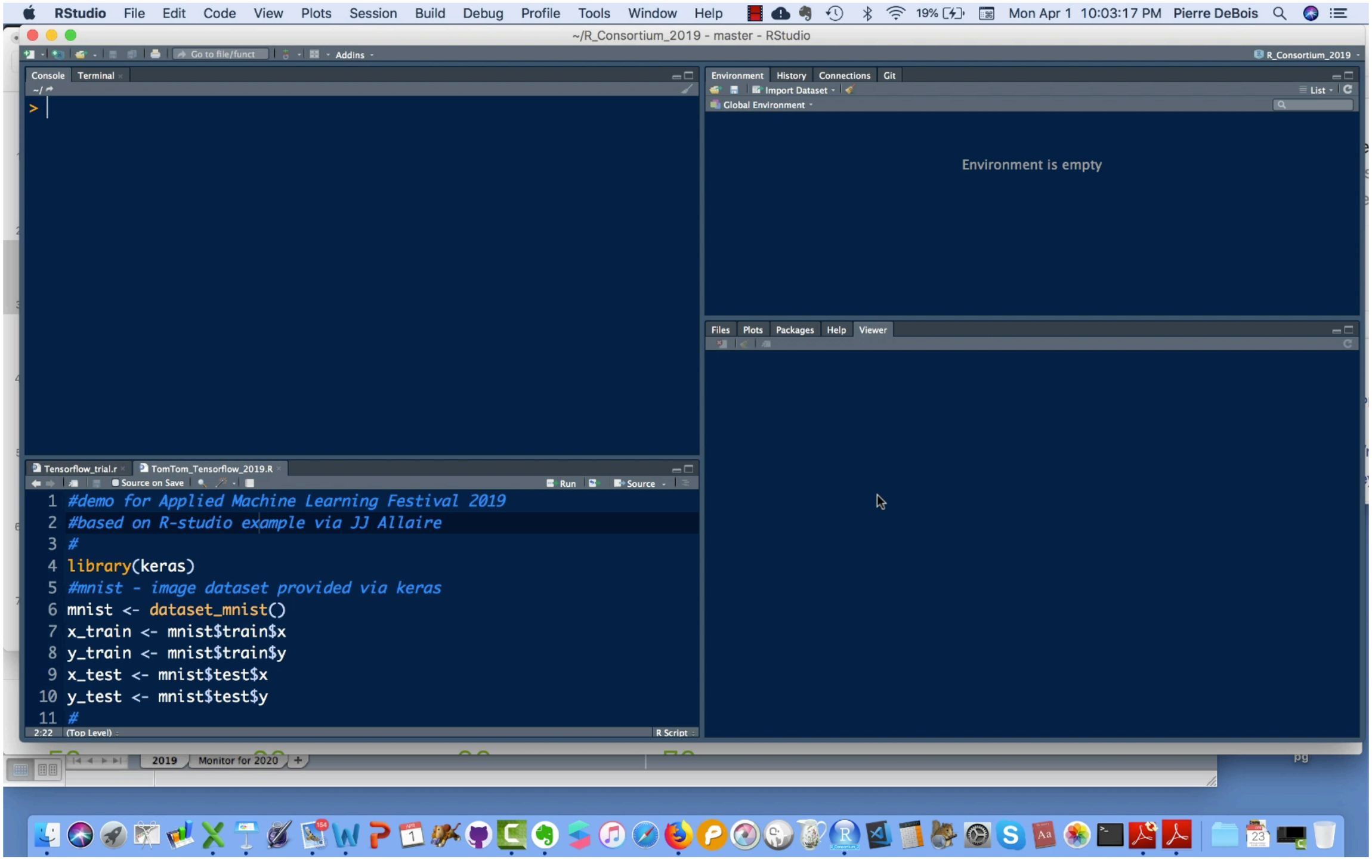
3. Train & Inspect fit



- Check Accuracy & loss to evaluate number of epochs needed

- Adjust nodes and rerun to see if acc/loss improves

Machine Learning in Action



Takeaways

- Examine dataset to plan variables, training and test data (80/20 split)
- Transform Data,
- Select Layers
- Train & Inspect Fit
- Experiment to determine right epochs that assure accuracy
- TensorFlow can help you develop a predictive model for image, voice, and numeric arrays

Thank You

pdebois@zimana.com @zimanaanalytics
www.zimana.com

Resources

- https://www.tensorflow.org/tutorials
- tensorflow.rstudio.com/tools/gpu
- Studio Deep Learning with Keras cheatsheet - functions
- JJ Allaire - RStudio introduction of TensorFlow
- CS 229 Machine Learning Tips (Stanford University) https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-deep-learning

Additional Libraries

- cloudml - R interface to Google Cloud ML

- tfuns - tracks and records data on each iteration

- tfdeploy - exports model to CloudML, RStudio Connect, TensorFlow Serving, Shiny app

- Neuralnet — maps neural network within R

- Tensorboard - sharing accuracy & loss graphics