Using TensorFlow and R

LONDONR
2018-03-27

Andrie de Vries
Solutions Engineer, RStudio
@RevoAndrie
Overview

- TensorFlow using R
- Worked example of keras in R
- Demo
- Supporting tools
- Learning more

Slides at https://speakerdeck.com/andrie/londonr-tensorflow
What is TensorFlow
What is TensorFlow

- Originally developed by researchers and engineers working on the Google Brain Team for the purposes of conducting machine learning and deep neural networks research.
- Open source software (Apache v2.0 license)
- Hardware independent
  - CPU (via Eigen and BLAS)
  - GPU (via CUDA and cuDNN)
  - TPU (Tensor Processing Unit)
- Supports automatic differentiation
- Distributed execution and large datasets
What is a tensor?

• Spoiler alert: it’s an array

<table>
<thead>
<tr>
<th>Tensor dimensionality</th>
<th>R object class</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Vector of length one</td>
<td>Point value</td>
</tr>
<tr>
<td>1</td>
<td>Vector</td>
<td>Weights</td>
</tr>
<tr>
<td>2</td>
<td>Matrix</td>
<td>Time series</td>
</tr>
<tr>
<td>3</td>
<td>Array</td>
<td>Grey scale image</td>
</tr>
<tr>
<td>4</td>
<td>Array</td>
<td>Colour images</td>
</tr>
<tr>
<td>5</td>
<td>Array</td>
<td>Video</td>
</tr>
</tbody>
</table>

Note that the first dimension is always used for the observations, thus “adding” a dimension
What is tensor flow?

- You define the graph in R
- Graph is compiled and optimized
- Graph is executed on devices
- Nodes represent computations
- Data (tensors) flows between them
Why a dataflow graph?

• Major gains in performance, scalability, and portability
  • Parallelism
    • System runs operations in parallel.
  • Distributed execution
    • Graph is partitioned across multiple devices.
  • Compilation
    • Use the information in your dataflow graph to generate faster code (e.g. fusing operations)
  • Portability
    • Dataflow graph is a language-independent representation of the code in your model (deploy
Uses of TensorFlow

- Image classification
- Time series forecasting
- Classifying peptides for cancer immunotherapy
- Credit card fraud detection using an autoencoder
- Classifying duplicate questions from Quora
- Predicting customer churn
- Learning word embeddings for Amazon reviews

https://tensorflow.rstudio.com/gallery/
What is deep learning
What is deep learning?

- Input to output via layers of representation
What are layers?

• Data transformation functions parameterized by weights
  • A layer is a geometric transformation function on the data that goes through it (transformations must be differentiable for stochastic gradient descent)
  • Weights determine the data transformation behavior of a layer
library(keras)
model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32, kernel_size = c(3,3), activation = 'relu',
  input_shape = c(28,28,1)) %>%
  layer_conv_2d(filters = 64, kernel_size = c(3,3), activation = 'relu') %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_flatten() %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dense(units = 10, activation = 'softmax')
MNIST layers of representation
Geometric interpretation

• Deep-learning models are mathematical machines for uncrumpling complicated manifolds of high-dimensional data.

• Deep learning is turning meaning into vectors, into geometric spaces, and then incrementally learning complex geometric transformations that map one space to another.
How can we do this?

- How can we do this with simple parametric models trained with gradient descent?
- We just need
  - Sufficiently large parametric models,
  - trained with gradient descent on
  - sufficiently many examples
Sufficiently large parametric models

- Simple grayscale digit recognizer model has > 1 million parameters

```
Summary(model)

Layer (type)                  Output Shape                  Param #
======================================================================
conv2d_3 (Conv2D)            (None, 26, 26, 32)               320
____________________________________________________________________
conv2d_4 (Conv2D)            (None, 24, 24, 64)               18496
____________________________________________________________________
max_pooling2d_2 (MaxPooling2D) (None, 12, 12, 64)              0
____________________________________________________________________
flatten_2 (Flatten)          (None, 9216)                    0
____________________________________________________________________
dense_3 (Dense)             (None, 128)                     1179776
____________________________________________________________________
dense_4 (Dense)             (None, 10)                      1290
======================================================================
Total params: 1,199,882
Trainable params: 1,199,882
Non-trainable params: 0
```
TensorFlow using R
Why should R users care about TensorFlow?

- A new **general purpose** numerical computing library
  - Hardware independent
  - Distributed execution
  - Large datasets
  - Automatic differentiation
- **Not all data has to be in RAM**
  - Highly general optimization, e.g. SGD, Adam
- **Robust foundation** for machine and deep learning
- TensorFlow models can be **deployed with C++ runtime**
- R has a lot to offer as an **interface language**
R interface to Tensorflow

• [https://tensorflow.rstudio.com](https://tensorflow.rstudio.com)

• High-level R interfaces for neural nets and traditional models
• Low-level interface to enable new applications (e.g. Greta)
• Tools to facilitate productive workflow / experiment management
• Straightforward access to GPUs for training models
• Breadth and depth of educational resources
library(keras)

model <- keras_model_sequential()
layer_dense(units = 128, activation = 'relu', input_shape = c(784))
layer_dropout(rate = 0.4)
layer_dense(units = 128, activation = 'relu')
layer_dropout(rate = 0.3)
layer_dense(units = 10, activation = 'softmax')
TensorFlow APIs

- Distinct interfaces for various tasks and levels of abstraction

Keras API
The Keras API for TensorFlow provides a high-level interface for neural networks, with a focus on enabling fast experimentation.

Estimator API
The Estimator API for TensorFlow provides high-level implementations of common model types such as regressors and classifiers.

Core API
The Core TensorFlow API is a lower-level interface that provides full access to the TensorFlow computational graph.
• Low level access to TensorFlow graph operations
  https://tensorflow.rstudio.com/tensorflow

```{r}
library(tensorflow)

W <- tf$Variable(tf$random_uniform(shape(1L), -1.0, 1.0))
b <- tf$Variable(tf$zeros(shape(1L)))
y <- W * x_data + b

loss <- tf$reduce_mean((y - y_data) ^ 2)
optimizer <- tf$train$GradientDescentOptimizer(0.5)
train <- optimizer$minimize(loss)

sess = tf$Session()
sess$run(tf$global_variables_initializer())

for (step in 1:200)
  sess$run(train)
```
tfestimators

- High level API for TensorFlow models
  (https://tensorflow.rstudio.com/tfestimators/)

```r
library(tfestimators)
linear_regressor()
linear_classifier()
dnn_regressor()
dnn_classifier()
dnn_linear_combined_regressor()
dnn_linear_combined_classifier()
```
keras

• High level API for neural networks (https://tensorflow.rstudio.com/keras/ )

library(keras)

model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32, kernel_size = c(3,3), activation = 'relu',
                input_shape = input_shape) %>%
  layer_conv_2d(filters = 64, kernel_size = c(3,3), activation = 'relu') %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_dropout(rate = 0.25) %>%
  layer_flatten() %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 10, activation = 'softmax')
Worked example using keras
Steps in building a keras model

Define
- Model
- Sequential model
- Multi-GPU model

Compile
- Optimiser
- Loss
- Metrics

Fit
- Batch size
- Epochs
- Validation split

Evaluate
- Evaluate
- Plot

Predict
- classes
- probability

Keras data pre-processing

- Transform input data into tensors

```r
library(keras)

# Load MNIST images datasets (built-in to Keras)
c(c(x_train, y_train), c(x_test, y_test)) %<-% dataset_mnist()

# Flatten images and transform RGB values into [0,1] range
x_train <- array_reshape(x_train, c(nrow(x_train), 784))
x_test  <- array_reshape(x_test, c(nrow(x_test), 784))
x_train <- x_train / 255
x_test  <- x_test / 255

# Convert class vectors to binary class matrices
y_train <- to_categorical(y_train, 10)
y_test  <- to_categorical(y_test, 10)
```

Datasets are downloaded from S3 buckets and cached locally

Use %<-% to assign to multiple objects

TensorFlow expects row-primary tensors. Use `array_reshape()` to convert from (column-primary) R arrays

Normalize to [-1; 1] range for best results

Ensure your data is numeric only, e.g. by using one-hot encoding
Model definition

model <- keras_model_sequential()
  %>% layer_dense(units = 256, activation = 'relu', input_shape = c(784))
  %>% layer_dropout(rate = 0.4)
  %>% layer_dense(units = 128, activation = 'relu')
  %>% layer_dropout(rate = 0.3)
  %>% layer_dense(units = 10, activation = 'softmax')

model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(),
  metrics = c('accuracy'))
Note: Models are modified in-place

- Object semantics are not by-value! (as is conventional in R)
  - Keras models are directed acyclic graphs of layers whose state is updated during training.
  - Keras layers can be shared by multiple parts of a Keras model.

```r
# Modify model object in place (note that it is not assigned back to)

model %>% compile(
  optimizer = 'rmsprop',
  loss = 'binary_crossentropy',
  metrics = c('accuracy')
)
```

In the compile() step, do not assign the result, i.e. modify in place
Keras: Model training

• Feeding mini-batches of data to the model thousands of times

```r
history <- model %>% fit(
  x_train, y_train,
  batch_size = 128, 
  epochs = 10, 
  validation_split = 0.2
)
```

• Feed 128 samples at a time to the model \( (\text{batch\_size} = 128) \)
• Traverse the input dataset 10 times \( (\text{epochs} = 10) \)
• Hold out 20% of the data for validation \( (\text{validation\_split} = 0.2) \)
Evaluation and prediction

```r
model %>% evaluate(x_test, y_test)

$loss
[1] 0.1078904

$acc
[1] 0.9815

model %>% predict_classes(x_test[1:100,])

[1]  7  2  1  0  4  1  4  9  5  9  0  6  9  0  1  5  9  7  3  4  9  6  6  5  4  0  7  4  0  1  3  1  3  4  7
[36]  2  7  1  2  1  1  7  4  2  3  5  1  2  4  4  6  3  5  5  6  0  4  1  9  5  7  8  9  3  7  4  6  4  3  0
[71]  7  0  2  9  1  7  3  2  9  7  7  6  2  7  8  4  7  3  6  1  3  6  9  3  1  4  1  7  6  9
```
Easy plotting of fitting history

plot(history)
Demo
```r
library(keras)

# Load MNIST images datasets (built in to Keras)
c(c(x_train, y_train), c(x_test, y_test)) %<-% dataset_mnist()

# Flatten images and transform RGB values into [0,1] range
x_train <- array_reshape(x_train, c(nrow(x_train), 784))
x_test <- array_reshape(x_test, c(nrow(x_test), 784))
x_train <- x_train / 255
x_test <- x_test / 255

# Convert class vectors to binary class matrices
y_train <- to_categorical(y_train, 10)
y_test <- to_categorical(y_test, 10)

# Define the model
model <- keras_model_sequential()
layer_dense(units = 256, activation = 'relu', input_shape = c(784))
layer_dropout(rate = 0.4)
layer_dense(units = 128, activation = 'relu')
layer_dropout(rate = 0.3)
layer_dense(units = 10, activation = 'softmax')

# Compile the model
model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(),
  metrics = c('accuracy')
)

# Print a summary
summary(model)

# Fit the model
history <- model %>% fit(
x_train, y_train,
batch_size = 128,
epochs = 10,
validation_split = 0.2
)

# Plot the training history
plot(history)
```
Supporting tools
tfruns

- [https://tensorflow.rstudio.com/tools/tfruns/](https://tensorflow.rstudio.com/tools/tfruns/)
- Successful deep learning requires a huge amount of experimentation.
- This requires a systematic approach to conducting and tracking the results of experiments.
- The `training_run()` function is like the `source()` function, but it automatically tracks and records output and metadata for the execution of the script:

```r
library(tfruns)
training_run("mnist_mlp.R")
```
cloudml

- [https://tensorflow.rstudio.com/tools/cloudml/](https://tensorflow.rstudio.com/tools/cloudml/)
- Scalable training of models built with the keras, tfestimators, and tensorflow R packages.
- On-demand access to training on GPUs, including Tesla P100 GPUs from NVIDIA®.
- Hyperparameter tuning to optimize key attributes of model architectures in order to maximize predictive accuracy.
tfdeploy

- [https://tensorflow.rstudio.com/tools/tfdeploy/](https://tensorflow.rstudio.com/tools/tfdeploy/)
- TensorFlow was built from the ground up to enable deployment using a low-latency C++ runtime.
- Deploying TensorFlow models requires no runtime R or Python code.
- Key enabler for this is the TensorFlow SavedModel format:
  - a language-neutral format
  - enables higher-level tools to **produce, consume and transform** models.
- TensorFlow models can be deployed to servers, embedded devices, mobile phones, and even to a web browser!
Resources
Recommended reading

Chollet and Allaire

Goodfellow, Bengio & Courville
R examples in the gallery

- [https://tensorflow.rstudio.com/gallery/](https://tensorflow.rstudio.com/gallery/)

- Image classification on small datasets
- Time series forecasting with recurrent networks
- Deep learning for cancer immunotherapy
- Credit card fraud detection using an autoencoder
- Classifying duplicate questions from Quora
- Deep learning to predict customer churn
- Learning word embeddings for Amazon reviews
- Work on explainability of predictions
Keras for R cheat sheet

rstudio::conf videos

- Keynote: Machine Learning with TensorFlow and R
Summary
## Summary

### TensorFlow APIs

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>keras</strong></td>
<td>Interface for neural networks, focus on fast experimentation.</td>
</tr>
<tr>
<td><strong>tfestimators</strong></td>
<td>Implementations of common model types, e.g. regressors and classifiers.</td>
</tr>
<tr>
<td><strong>tensorflow</strong></td>
<td>Low-level interface to the TensorFlow computational graph.</td>
</tr>
</tbody>
</table>

### Supporting tools

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tfdatasets</strong></td>
<td>Scalable input pipelines for TensorFlow models.</td>
</tr>
<tr>
<td><strong>tfruns</strong></td>
<td>Track, visualize, and manage TensorFlow training runs and experiments.</td>
</tr>
<tr>
<td><strong>tfdeploy</strong></td>
<td>Tools designed to make exporting and serving TensorFlow models easy.</td>
</tr>
<tr>
<td><strong>cloudml</strong></td>
<td>R interface to Google Cloud Machine Learning Engine.</td>
</tr>
</tbody>
</table>
Summary

- TensorFlow is a new general purpose numerical computing library with lots to offer the R community.

- Deep learning has made great progress and will likely increase in importance in various fields in the coming years.

- R now has a great set of APIs and supporting tools for using TensorFlow and doing deep learning.

Slides at https://speakerdeck.com/andrie/londonr-tensorflow