Examples on tensorflow.rstudio.com/gallery – check it for image classification and NLP advice

AWS RKeras lecture – “Machine Learning with TensorFlow and R”

Speaker – JJ Allaire, CEO of Rstudio

Tensorflow hardware options include CPU, GPU via CUDA or cuDNN, new thing Google’s making called TPU (Tensor)

Supports automatic differentiation

Built for scale, great for distrib comp and large datasets

V general optimization algorithms, which *don’t* require that all your data is in RAM! So, does work with R

C++ runtime – faster than pure R would be.

Vector’s a 1D tensor, matrix is 2D tensor, R supports n-D tensors. Tensors are just arrays.

Vector data – 2D tensors of shape (samples, features)

Time series – 3D of (samples, timesteps, features)

Convention: samples is always the first dim. Each observation in your test set is a sample.

Images – 4D of (samples, height, width, color channels)

The tensor “flow”: consider a graph, where each node is an operation. Model tensorflow operations as data flowing from node to node in this graph.

Keras automatically generates and displays graph of your model when you use it in R! V helpful.

Why dataflow graph? Performance, scalability, and portability that you don’t get with a lower-level script.

Graph can be partitioned across multiple devices

Use info on your dataflow graph for e.g. fusing operations, generates faster code

Dataflow graph is a language-independent rep of the code in your model (well, it’s ultimately in

C++…)

Greta – package for defining stats models in MCMC and then using a Tensorflow graph to fit them with MCMC. Trains fast, parallelizes well.

Can think of an nnet layer as a data transformation fct parameterized by weights. Has to be differentiable for SGD (though RELUs, lol)

One way to think of learning features is coming up with new coordinate systems that better represent your data.

Layers of representation create filters that remove irrelevant information, check for learned commonalities.

Deep learning achievements: near-human image classification, speech recog, handwriting transcription, autonomous driving. Improved machine translation, text-to-speech, search results. Ability to answer natural language qs. Superhuman Go playing.

Chollet – Nnets are chains of differentiable, parameterized functions. He argues they’re not truly neural nor truly networks (I’d quibble re RNNs and sparsely connected nets but yeah fair cop for fully connected feed-forward)

Sufficiently large parametric models trained w/ grad des on sufficiently many examples work really well.

A simple MNIST digit recognizer can have 1m+ parameters!

VGG has 138m params, Imagenet has 14m images in its training set

Deep learning frontiers right now: computer vision problems that were beyond horizion; NLP esp translation; time series; biomedical

Conventional time series tools are beating deep learning so far, except w/ v noisy data. Convnets (incl pooling layers) shows promise there. LSTM stacked autoencoders for financial time series is being tried out (this is not going to work I think), using wavelet transforms to address noise

Deep learning for “fundamental biological processes” – ooh, sounds promising for making systems biology actually work.

DL for electronic health records – learns reps of key factors and interactions from the data itself, rather than from highly curated (labor-intensive, cleaned up) as previous work on trying to predict conditions from EHRs as done

One paper showed this beating traditional predictive models! Caveat – Google funded, elite team, may not replicate w/o same effort and funding

# Keras

tfestimators and tfdatasets packages – useful for implementations of common model types (like regressors and classifiers) and big datasets to use.

Tfestimators is a lot less granular than Keras! Take a look at it.

Keras is pulling away from other interfaces to Keras as Tensorflow is pulling away from other frameworks.

## Sample MNIST Keras example.

Preprocessing: you need to turn your raw data into tensors of some kind.

Model definition: here are the layers you’re gonna use, and figuring out right hyperparams will take experimentation. (Find that auto hyperparameter search thing)

Models are modified in-place. Object semantics are not by-value as is conventional in R

i.e., running data %>% fct() changes data, even if you don’t do data <- data %>% fct()

Model training: traverse the input dataset *epochs* times, hold out data for validation (validation\_split=20 gives you a validation set that’s 20% of your training set), batch\_size of minibatches

Evaluation and prediction step: final test data, etc.

Real-time graph of test vs validation, which is great for stopping the model and trying something else when it starts to overfit.

## API

65 layer types built in, you can define your own.

layer\_dense() is classic fully connected layers. Most complex nets have a few of these at the bottom!

Dropout, batch normalization, 2d and 3d convolutional

layer\_conv\_2d()

Recurrent layers: maintain state based on previously seen data

Layer\_simple\_rnn

Layer\_gru (gated recurrent unit, similar to lstm)

Layer\_lstm

Learn a vector of state information, use it to condition input to next layer

Embedding layers – these make a word embedding for NLP. Vectorization of text that reflects semantic relationships b/w words. Layer\_embedding

Can learn the embeddings jointly with your task, or load word2vec or GloVe

Compiling models: prepares the model for training by

1. Converting layers into a TF graph
2. Applies the loss fct and optimizer
3. (missed)

Lists the predefined loss fcts, optimizers, and error metrics

See the Keras for R cheatsheet

Image classification on small datasets – can get 97% classification accuracy on dogs vs cats trained on only 2000 input images! How? Cheating! We’re doing transfer learning from a huge-ass imagenet model.

Credit card fraud detection example – logreg struggles with this bc imbalanced classes really severely, deep learning does well

Customer churn example has good example of how to use Shiny with Keras – check it out.

Explainability – you can visualize activations with

Imagenet\_decode\_predictions()

Heatmap based on activations in intermediate layers of the model to tell what parts of the model led to the prediction

You can also use LIME! R implementation github.com/thomasp85/lime

Tensorflow.rstudio.com has thing on how to use GPUs – your comp may have it, if not will run you through how to cloud it out. There’s an Rstudio server with tensorflow that runs through AWS that recommend using.

The cloudml package also recommended, good w/ Keras

Tfruns package – use to manage your experiments. Training\_run() fct that collects data on every run, I presume you can feed it to that Gaussian Process hyperparameter operator. (If not, figure out how!) Great for model tuning.

Tfdeploy - use it to deploy R to production! Servers, embedded devices, mobile phones…

Also look up TensorFlow Lite for mobile models

And Keras.js lets you deploy Keras models in browser (do I need this when have Shiny?)

Goodfellow, Bengio, and COurville Deep Learning book is all the concepts and math behind deep learning. Strongly recommended.

## Slides are at https://rstd.io/ml-with-tensorflow-and-r

# Questions

Keras Applications – pretrained models (mostly machine vision) that can plug in, like Caffe and Mxnet model zoos.

For multimodal learning – Keras has a concept of a “multi-input” model. You can build a graph of any level of complexity, not just a simple FF