Load Forecasting with Artificial Intelligence on Big Data

October 9, 2016

Patrick GLAUNER and Radu STATE
SnT - Interdisciplinary Centre for Security, Reliability and Trust,
University of Luxembourg
Biography

- PhD student at the University of Luxembourg
- Collaboration with Choice Technologies Holding on detection of non-technical losses (NTL)
- MSc in Machine Learning from Imperial College London
- Previously worked at CERN and SAP
Motivation

• **Artificial Intelligence**: "AI is the science of knowing what to do when you don't know what to do." (Peter Norvig, [www.youtube.com/watch?v=rtmQ3xlt-4A4m45](https://www.youtube.com/watch?v=rtmQ3xlt-4A4m45))

• **Machine Learning** is the field of study that gives computers the ability to learn without being explicitly programmed.
Motivation

Data:  

Label/target:  

2
Motivation

- Goal: Predict time series of load
Agenda

1. Neural networks
2. Deep Learning
3. TensorFlow
4. Load forecasting
5. Conclusions and outreach
Neural networks

Neural networks

Neural networks

The activation of unit $i$ of layer $j+1$ can be calculated:

$$z_{i}^{(j+1)} = \sum_{k=0}^{s_j} \Theta_{ik}^{(j)} x_k$$

$$a_i^{(j+1)} = g \left( z_i^{(j+1)} \right)$$
Neural networks

Different activation functions

- Sigmoid
- ReLU

Activation value vs Input value graph
Neural networks

Cost function for $m$ examples, hypothesis $h_\theta$ and target values $y^{(i)}$:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$
Neural networks

How to optimize the weights?

Neural networks

Algorithm 2.1 Batch gradient descent: training size $m$, learning rate $\alpha$

repeat
\[ \theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \] (simultaneously for all $j$)
until convergence
Neural networks

Algorithm 2.2 Stochastic gradient descent: training size $m$, learning rate $\alpha$

Randomly shuffle data set

repeat

for $i = 1$ to $m$ do

$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta, (x^{(i)}, y^{(i)}))$ (simultaneously for all $j$)

end for

until convergence
Neural networks

How to compute the partial derivatives?
Neural networks

**Algorithm 3** Backpropagation: training size $m$

$$\Theta_{ij}^{(l)} \leftarrow \text{rand}(-\varepsilon, \varepsilon) \; \text{(for all } l, i, j)$$

$$\Delta_{ij}^{(l)} \leftarrow 0 \; \text{(for all } l, i, j)$$

**for** $i = 1$ **to** $m$ **do**

$$a^{(1)} \leftarrow x^{(i)}$$

Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, ..., L$

Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)} \quad \triangleright \text{"error"}$

Compute $\delta^{(L-1)}, \delta^{(L-2)}, ..., \delta^{(2)}$: $\delta^{(l)} = (\Theta^{(l)})^T \delta^{(l+1)} \circ g'(z^{(l)})$

$$\Delta^{(l)} \leftarrow \Delta^{(l)} + \delta^{(l+1)}(a^{(l)})^T \quad \triangleright \text{Matrix of errors for units of a layer}$$

**end for**

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) \leftarrow \frac{1}{m} \Delta_{ij}^{(l)}$$
Deep Learning

The Analytics Store, “Deep Learning”,
Deep Learning: DeepMind

• Founded in 2010 in London
• Created a neural network that learns how to play video games in a similar fashion to humans
• Acquired by Google in 2014, estimates range from USD 400 million to over GBP 500 million
• Now being used in Google's search engine
• AlphaGo played the game of Go at super-human performance
TensorFlow

TensorFlow (J. Dean, R. Monga et al., `TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems`, 2015.) is used by Google for most of its Deep Learning products:

- Offers neural networks (NN), convolutional neural networks (CNN), recurrent neural networks (RNN) and long-short term memories (LSTM)
- Computations are expressed as a data flow graph
- Can be used for research and production
- Python and C++ interfaces
TensorFlow

• Code snippets available from Udacity class: https://www.udacity.com/course/deep-learning--ud730

• iPython notebooks: https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/udacity
TensorFlow: Playground

• Let us use the playground together: http://playground.tensorflow.org
TensorFlow

• A Tensor is a typed multi-dimensional array
• Nodes in the graph are called ops
• An op takes zero or more Tensors, performs some computation, and produces zero or more Tensors
• Two phases:
  – Construction phase, that assembles a graph
  – Execution phase that uses a session to execute ops in the graph
• Auto-differentiation of the graph to compute partial derivatives used in stochastic gradient descent (SGD)
TensorFlow

```
C
  ...
  ReLU
  Add
  b
  MatMul
  W
  X
```
TensorFlow: GPU acceleration
TensorFlow: GPU acceleration

http://www.nvidia.com/object/tesla-servers.html
TensorFlow

• Great documentation: https://www.tensorflow.org/versions/0.6.0/get_started
• Installation: https://www.tensorflow.org/versions/0.6.0/get_started/os_setup.html
Load forecasting

• Goal: Predict time series of load

Test performance on time series 1

- Red: Prediction
- Blue: Ground truth
Load forecasting

• Feed-forward networks lack the ability to handle temporal data
• Recurrent neural networks (RNN) have cycles in the graph structure, allowing them to keep temporal information
Load forecasting

- LSTM cells can be put together in a modular structure to build complex recurrent neural networks
- LSTMs have been reported to outperform regular RNNs and Hidden Markov Models in classification and time series prediction tasks (N. Srivastava, E. Mansimov and R. Salakhutdinov, "Unsupervised Learning of Video Representations using LSTMs", University of Toronto, 2015.)
Load forecasting

• Source code: https://github.com/pглаuner/ISGT_Europe_2016_Tutorial

• Simplified example, as time series is synthetic and harmonic

• More complex task will follow later
Load forecasting

- Training on two time series at the same time
- Input values of each time series: value, derivative, second-order derivative
- Training data must be sufficiently long
Load forecasting

Training performance on time series 1

- Red: Prediction
- Blue: Ground truth
Load forecasting

Training performance on time series 1

- Red: Prediction
- Blue: Ground truth
Load forecasting

Training performance on time series 1

Prediction
Ground truth
Load forecasting

Test performance on time series 1

- Prediction
- Ground truth
Load forecasting

```python
# Input layer for 6 inputs, batch size 1
input_layer = tf.placeholder(tf.float32, [1, INPUT_DIM * 3])

# Initialization of LSTM layer
lstm_layer = rnn_cell.BasicLSTMCell(INPUT_DIM * 3)

# LSTM state, initialized to 0
lstm_state = tf.Variable(tf.zeros([1, lstm_layer.state_size]))

# Connect input layer to LSTM
lstm_outputs, lstm_state_output1 = lstm_layer(input_layer, lstm_state)

# Update of LSTM state
lstm_update = lstm_state.assign(lstm_state_output1)
```
Load forecasting

```python
# Regression output layer
# Weights and biases
output_W = tf.Variable(tf.truncated_normal([[INPUT_DIM * 3, INPUT_DIM]]))
output_b = tf.Variable(tf.zeros([INPUT_DIM]))
output_layer = tf.matmul(lstm_output, output_W) + output_b

# Input for correct output (for training)
output_ground_truth = tf.placeholder(tf.float32, [1, INPUT_DIM])

# Sum of squared error terms
error = tf.pow(tf.sub(output_layer, output_ground_truth), 2)

# Adam optimizer
optimizer = tf.train.AdamOptimizer(0.0006).minimize(error)
```
Load forecasting
Load forecasting: Outreach

• Add some noise for more realistic synthetic data
• Models can be applied to other regression problems or time series classification (e.g. for detection of non-technical losses)
• Usually more features need to be added
• Model selection in order to tweak hyper parameters (architecture, learning rate, etc.)
Conclusions and outreach

- Deep neural networks can learn complex feature hierarchies
- Significant speedup of training due to GPU acceleration
- TensorFlow is a easy-to-use Deep Learning framework
- Interfaces for Python and C++
- Offers rich functionality and advanced features, such as LSTMs
- Udacity class and lots of documentation and examples available