Deep Learning on Big Data Sets in the Cloud with Apache Spark and Google TensorFlow

December 9, 2016

Patrick GLAUNER and Radu STATE

first.last@uni.lu

SEDAN Lab,
SnT - Interdisciplinary Centre for Security, Reliability and Trust,
University of Luxembourg

Biography

- PhD Student at the University of Luxembourg
- Adjunct Lecturer of Artificial Intelligence at Karlsruhe University of Applied Sciences
- MSc in Machine Learning from Imperial College London
- Previously worked at CERN and SAP

Motivation

SnT
securityandtrust.lu

Definition (Artificial Intelligence)

"AI is the science of knowing what to do when you don't know what to do." (Peter Norvig)[a]

[a] `http://www.youtube.com/watch?v=rtmQ3xlt-4A4m45`

Definition (Machine Learning)

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

Motivation

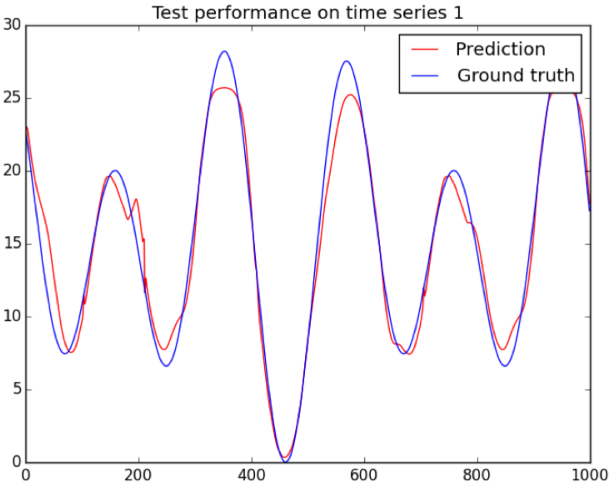
Goal: recognition of characters

`notMNIST` examples[1].

[1] `http://yaroslavvb.blogspot.lu/2011/09/notmnist-dataset.html`

Motivation

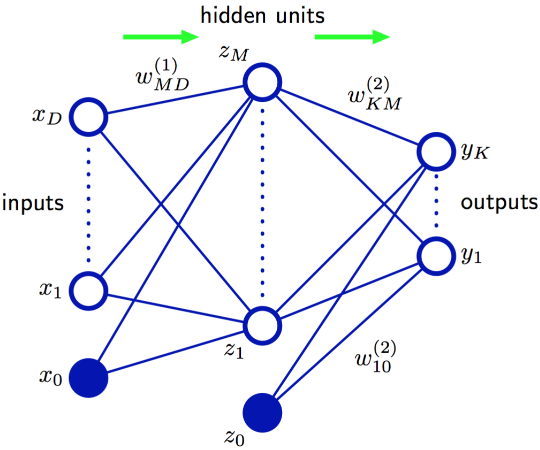
Goal: forecasting of time series



Agenda

1. Neural networks
2. Deep Learning
3. TensorFlow
4. Distributed computing
5. Example: character recognition
6. Example: time series forecasting
7. Rise of the machines?
8. Conclusions and outreach

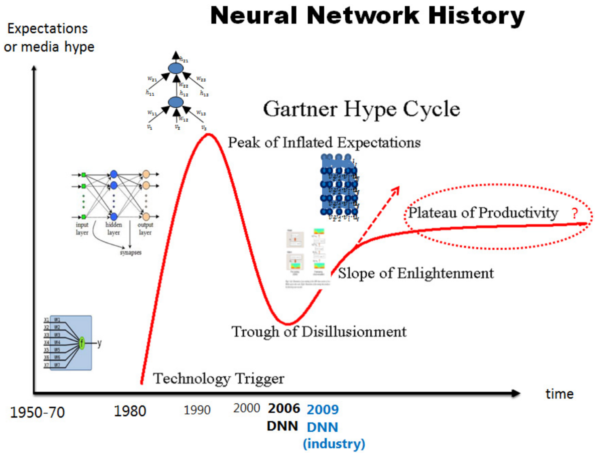
Neural networks



Neural network with two input and output units[2].

[2] Christopher M. Bishop, "Pattern Recognition and Machine Learning", Springer, 2007.

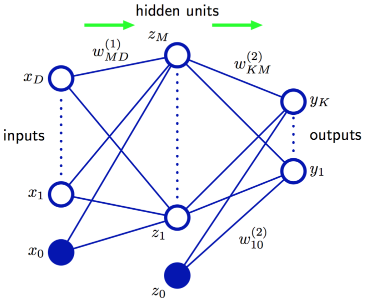
Neural networks



History of neural networks[3].

[3] Li Deng and Dong Yu, “Deep Learning Methods and Applications", Foundations and Trends in Signal Processing, vol. 7 issues 3-4, pp. 197-387, 2014.

Neural networks



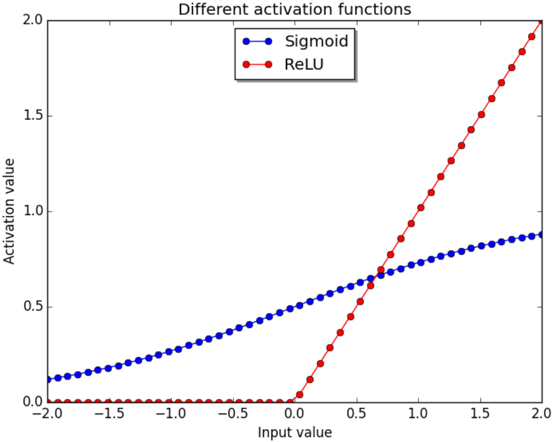
Neural network with two input and output units.

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

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

$$a_i^{(j+1)} = g\left(z_i^{(j+1)}\right) \tag{2}$$

Deep Learning: activation functions



Sigmoid and rectified linear unit (ReLU) activation functions.

Neural networks: parameter optimization

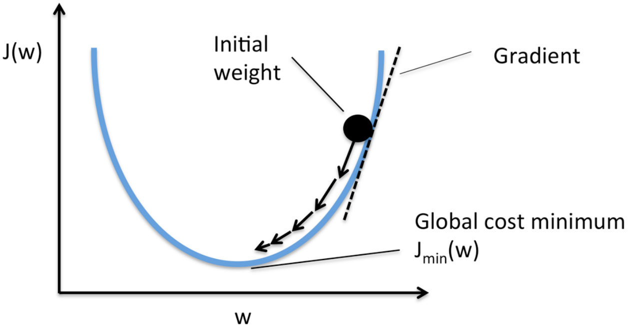
securityandtrust.lu

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

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \tag{3}$$

Deep Learning: parameter optimization

How to optimize the weights?



Visualization for one parameter[4].

[4] http://sebastianraschka.com/faq/docs/closed-form-vs-gd.html

Neural networks: parameter optimization

securityandtrust.lu

Algorithm 1 Batch gradient descent: training size m, learning rate α

repeat

$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$ (simultaneously for all j)

until convergence

Neural networks: parameter optimization

Algorithm 2 Stochastic gradient descent: training size m, learning rate α.

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: backpropagation

How to compute the partial derivatives?

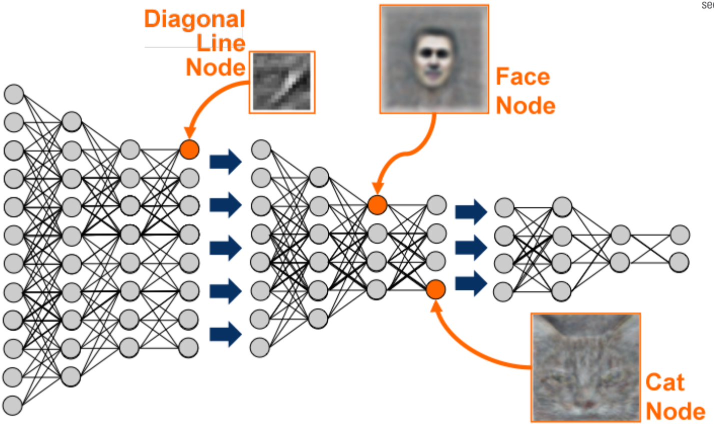
Neural networks: backpropagation

SnT
securityandtrust.lu

Algorithm 3 Backpropagation: training size m

$\Theta_{ij}^{(l)} \leftarrow rand(-\varepsilon, \varepsilon)$ (for all l, i, j)
$\Delta_{ij}^{(l)} \leftarrow 0$ (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 $\delta^{(L)} = a^{(L)} - y^{(i)}$ ▷ "error"
  Compute $\delta^{(L-1)}, \delta^{(L-2)}, ..., \delta^{(2)}$: $\delta^{(l)} = (\Theta^{(l)})^T \delta^{(l+1)} \circ g'(z^{(l)})$
  $\Delta^{(l)} \leftarrow \Delta^{(l)} + \delta^{(l+1)}(a^{(l)})^T$ ▷ 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



Deep neural network layers learning complex feature hierarchies[5].

[5] The Analytics Store, “Deep Learning", http://theanalyticsstore.com/deep-learning/, retrieved: March 1, 2015.

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

Google DeepMind[6].

[6] `http://deepmind.com/`, retrieved: March 2, 2016.

TensorFlow

TensorFlow[7] 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
- Code snippets available from Udacity class[8]

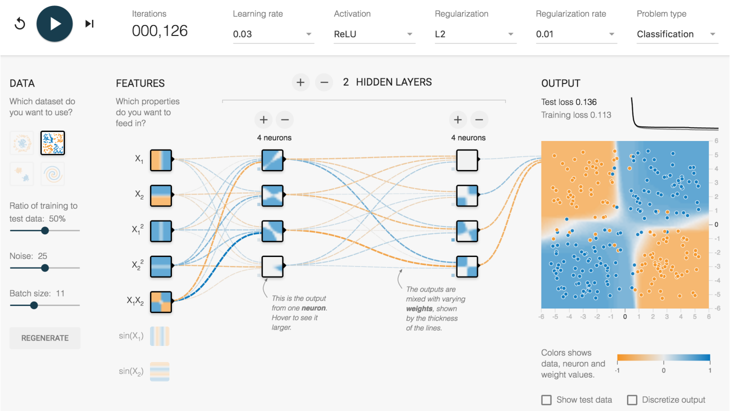
[7] J. Dean, R. Monga et al.: TensorFlow, “Large-Scale Machine Learning on Heterogeneous Distributed Systems", 2015.

[8] http://www.udacity.com/course/deep-learning--ud730

TensorFlow Playground

Let us experiment together with this playground for the next 20 minutes to get a better understanding of neural networks:

`http://playground.tensorflow.org`



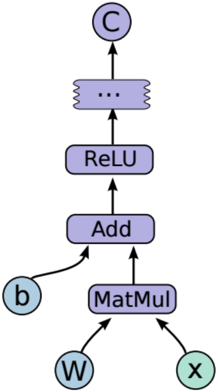
TensorFlow: graph and execution

securityandtrust.lu

- 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-differentation of the graph to compute partial derivatives used in stochastic gradient descent (SGD)

TensorFlow: graph and execution

securityandtrust.lu



Sample computation graph[9].

[9] J. Dean, R. Monga et al., “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems", 2015.

TensorFlow: installation

Great documentation[10].

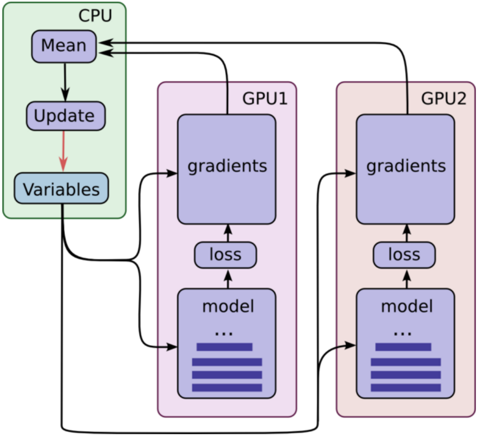
```
# Anaconda
$ sudo conda install \
   -c http://conda.anaconda.org/jjhelmus tensorflow
```

Support for Linux and Mac platforms, virtuelenv and Docker[11]. : time series

[10] http://www.tensorflow.org/versions/0.6.0/get_started

[11] http://www.tensorflow.org/versions/0.6.0/get_started/os_setup.html#pip_install

Distributed computing: GPUs

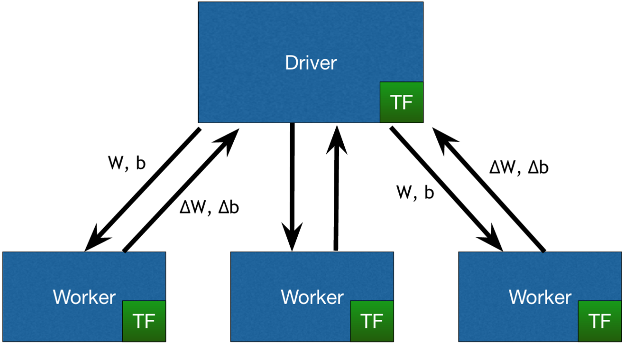


Parallel execution on multiple units[12].

[12] J. Dean, R. Monga et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems", 2015.

Distributed computing: Spark

Use of Spark for distributed computation of gradients:



Distributed computation of gradients[13].

[13] http://arimo.com/machine-learning/deep-learning/2016/arimo-distributed-tensorflow-on-spark/

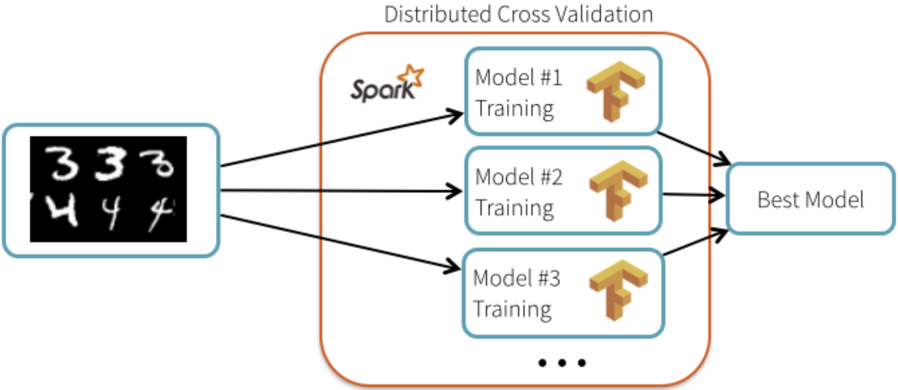
Model selection

The process of optimizing various hyper parameters, including:

- Number of layers
- Size of a layer
- Learning rate
- Regularization
- ...

Distributed computing: Spark

Use of Spark for distributed computation of model selection:



Distributed model selection on a single node[14].

[14] http://databricks.com/blog/2016/01/25/deep-learning-with-apache-spark-and-tensorflow.html

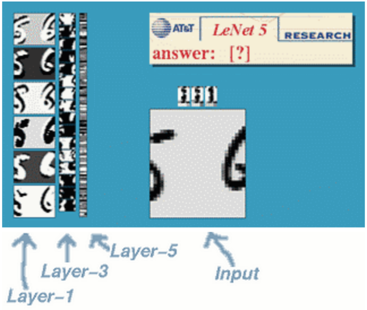
Distributed computing: cloud

SnT
securityandtrust.lu

Google Cloud Machine Learning: `https://cloud.google.com/ml/`

Example: character recognition

MNIST:



Hand-written digit recognition learned by a neural network[15].

[15] Yann LeCun et al.: LeNet-5, convolutional neural networks. `http://yann.lecun.com/exdb/lenet/`. Retrieved: April 22, 2015.

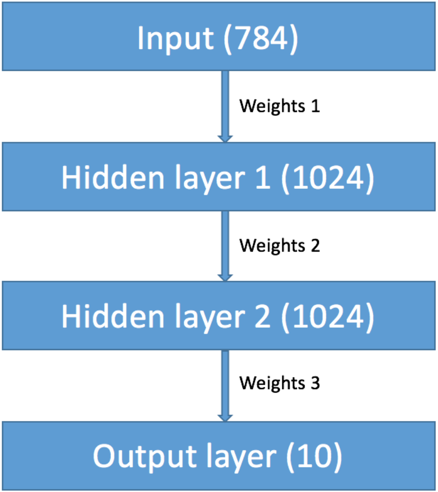
Example: character recognition

notMNIST: letters A-J.

notMNIST examples[16].

[16] http://yaroslavvb.blogspot.lu/2011/09/notmnist-dataset.html

Example: character recognition



Architecture of network (biases omitted).

Example: character recognition

SnT
securityandtrust.lu

- Source code: `http://github.com/pglauner/UCC_2016_Tutorial`
- Run `create_notmnist.py` once to get and convert the data
- Run `notminst_classifier.py` for the experiments

Example: character recognition

SnT
securityandtrust.lu

```
weights1 = tf.Variable(
  tf.truncated_normal([image_size * image_size, 1024]))
biases1 = tf.Variable(tf.zeros([1024]))

weights2 = tf.Variable(
  tf.truncated_normal([1024, 1024]))
biases2 = tf.Variable(tf.zeros([1024]))

weights3 = tf.Variable(
  tf.truncated_normal([1024, num_labels]))
biases3 = tf.Variable(tf.zeros([num_labels]))
```

Example: character recognition

```
[...]
def model(data, train=False):
  hidden1 = tf.nn.relu(
    tf.matmul(data, weights1) + biases1)
  if train:
    hidden1 = tf.nn.dropout(hidden1, 0.7, seed=SEED)

  hidden2 = tf.nn.relu(
    tf.matmul(hidden1, weights2) + biases2)
  if train:
    hidden2 = tf.nn.dropout(hidden2, 0.7, seed=SEED)

  return tf.matmul(hidden2, weights3) + biases3
[...]
```

Example: character recognition

```
logits = model(tf_train_dataset, True)

loss = tf.reduce_mean(
  tf.nn.softmax_cross_entropy_with_logits(
    logits, tf_train_labels))

# L2 regularization for the fully connected parameters
regularizers = (tf.nn.l2_loss(weights1)
                + tf.nn.l2_loss(biases1)
                + tf.nn.l2_loss(weights2)
                + tf.nn.l2_loss(biases2)
                + tf.nn.l2_loss(weights3)
                + tf.nn.l2_loss(biases3))
loss += 5e-4 * regularizers
```

Example: character recognition

```
Training set (200000, 784) (200000, 10)
Validation set (10000, 784) (10000, 10)
Test set (10000, 784) (10000, 10)

Initialized
Minibatch loss at step 0: 13926.021484
Minibatch accuracy: 7.8%
Validation accuracy: 25.4%
Minibatch loss at step 500: 839.786133
Minibatch accuracy: 76.6%
Validation accuracy: 81.2%
[...]
Minibatch loss at step 2500: 515.079651
Minibatch accuracy: 78.9%
Validation accuracy: 80.4%
Minibatch loss at step 3000: 503.497894
Minibatch accuracy: 66.4%
Validation accuracy: 80.1%
Test accuracy: 87.2%
```

Example: character recognition

Goal: become invariant to translation and rotation

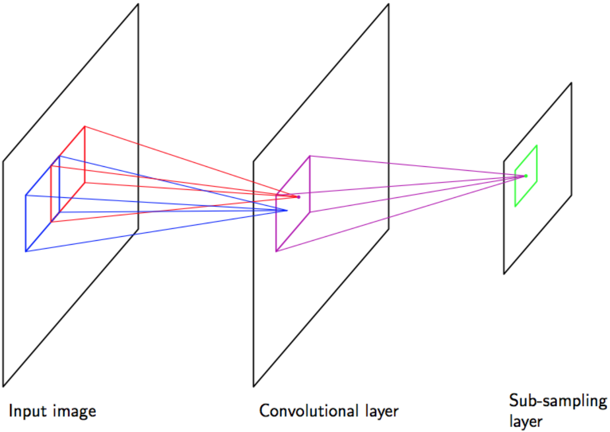


Illustration of a Convolutional Neural Network (CNN)[17].

[17] C. M. Bishop, "Pattern Recognition and Machine Learning", Springer, 2007.

Example: character recognition

- Source code: `http://github.com/pglauner/UCC_2016_Tutorial`
- Run `notminst_classifier_CNN.py` for the experiments

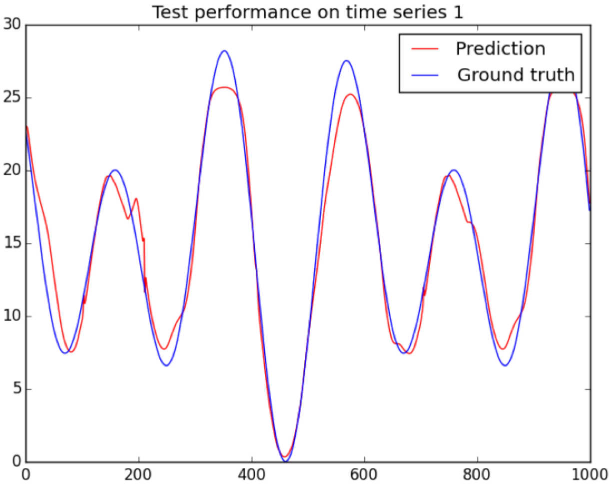
Example: character recognition

```
Training set (200000, 28, 28, 1) (200000, 10)
Validation set (10000, 28, 28, 1) (10000, 10)
Test set (10000, 28, 28, 1) (10000, 10)

Initialized
Minibatch loss at step 0: 5.747538
Minibatch accuracy: 6.2%
Validation accuracy: 10.0%
Minibatch loss at step 500: 0.642069
Minibatch accuracy: 87.5%
Validation accuracy: 81.9%
[...]
Minibatch loss at step 2500: 0.721265
Minibatch accuracy: 75.0%
Validation accuracy: 86.1%
Minibatch loss at step 3000: 0.646058
Minibatch accuracy: 87.5%
Validation accuracy: 86.5%
Test accuracy: 93.2%
```

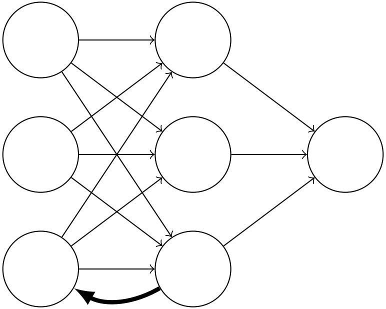
Example: time series forecasting

Goal: predict time series of electricity load



Example: time series forecasting

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



Simple RNN, current connection in **bold**.

Example: time series forecasting

- A long short-term memory (LSTM)[18] is a modular recurrent neural network composed of LSTM cell
- 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[19]

---

[18] S. Hochreiter and J. Schmidhuber, “Long short-term memory", Neural Computation, vol. 9, issue 8, pp. 1735-1780, 1997.

[19] N. Srivastava, E. Mansimov and R. Salakhutdinov, “Unsupervised Learning of Video Representations using LSTMs", University of Toronto, 2015.

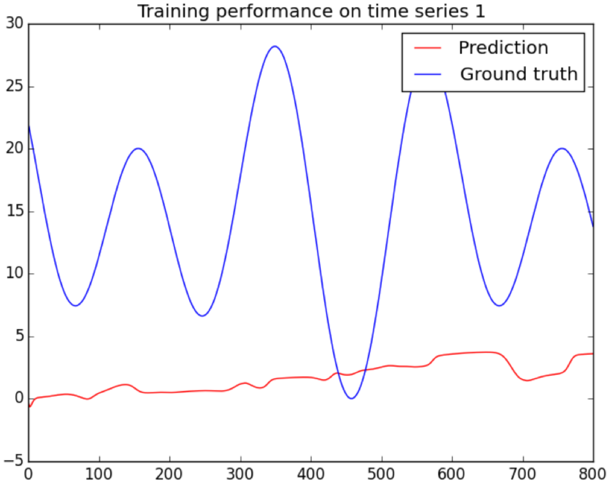
Example: time series forecasting

- Source code: `http://github.com/pglauner/UCC_2016_Tutorial`
- Run `LSTM.py` for the experiments
- Simplified example, as time series is synthetic and harmonic
- More complex task will follow later

Example: time series forecasting

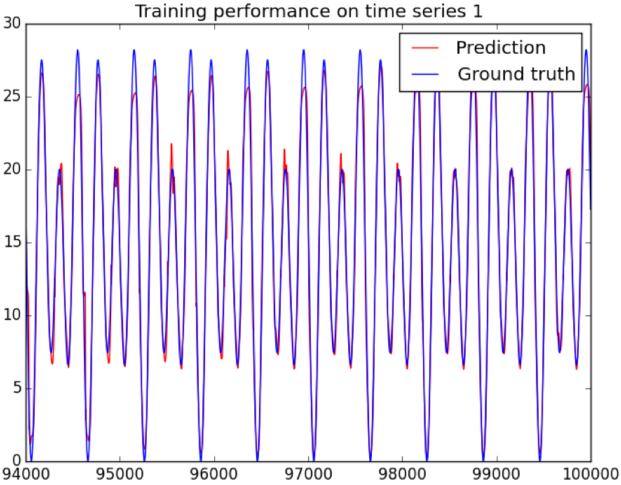
- Training on two time series at the same time
- Input values of each time series: value, derivative, second-order derivative
- Training data must be sufficiently long

Example: time series forecasting

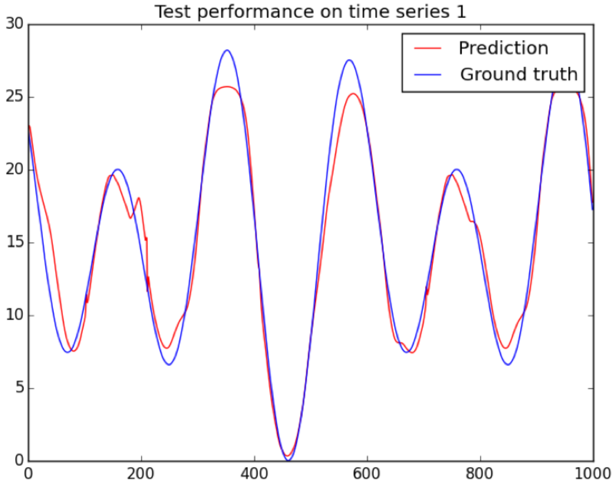


Example: time series forecasting

Example: time series forecasting



Example: time series forecasting



Example: time series forecasting

securityandtrust.lu

```
# 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_output, lstm_state_output1 = lstm_layer(
                                  input_layer, lstm_state)
# Update of LSTM state
lstm_update = lstm_state.assign(lstm_state_output1)
```

Example: time series forecasting

```
# 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)
```

Example: time series forecasting

- Add some noise for more realistic synthetic data
- Real-world load forecasting problem: `http://www.kaggle.com/c/global-energy-forecasting-competition-2012-load-forecastin`
- Models can be applied to other regression problems or time series classification (e.g. for detection of electricity theft)
- Usually more features need to be added
- Model selection in order to tweak hyper parameters (architecture, learning rate, etc.)

Rise of the machines?

Rise of the machines?

Do we have to be worried?

- Specialized AIs have made significant progress and started to outperform humans
- Do we have to be worried about machines taking over?
- When will we achieve the singularity, the point in time when machines will become more intelligent than humans?
- Fears are spread by Stephen Hawking and other researchers

Rise of the machines?

From a researcher who actually works on AI

"There's also a lot of hype, that AI will create evil robots with super-intelligence. That's an unnecessary distraction. [...] Those of us on the frontline shipping code, we're excited by AI, but we don't see a realistic path for our software to become sentient. [...] If we colonize Mars, there could be too many people there, which would be a serious pressing issue. But there's no point working on it right now, and that's why I can't productively work on not turning AI evil." (Andrew Ng)[a]

[a] http://www.theregister.co.uk/2015/03/19/andrew_ng_baidu_ai/

Rise of the machines?

SnT
securityandtrust.lu

Some thoughts

- The fear of an out-of-control AI is exaggerated
- Fears are mostly spread by people who do not work on AI, such as Stephen Hawking
- A lot of work needs to be done to work towards an **artificial general intelligence**
- Working towards simulating the brain may achieve the singularity in the late 21st century[a]
- In any case, many jobs will disappear in the next decades
- If computers only do a larger fraction of today's jobs, this will put pressure on salaries

[a] M. Shanahan, “The Technological Singularity", MIT Press, 2015.

Conclusions and outreach

- Deep neural networks can learn complex feature hierarchies
- TensorFlow is a easy-to-use Deep Learning framework
- Significant speedup of training on GPUs or Spark
- Interfaces for Python and C++
- Offers rich functionality and advanced features, such as LSTMs
- Udacity class and lots of documentation and examples available
- AI will not turn evil so soon