Multi-node Restricted Boltzmann Machines for Big Data

Nikolay Manchev

18 November 2016

The team

Nikolay Manchev

IBM Big Data Technical Team – Europe; Interests – Machine Learning, Big Data

Organiser – London Machine Learning Study Group

M.Sc. Computer Science, M.Sc. Data Science

`https://twitter.com/nikolaymanchev`

Dr. Tillman Weyde

Senior Lecturer at the Department of Computing at City University London

PhD Systematic Musicology (Music Technology), University of Osnabrck, 2002

MSc Computer Science, University of Osnabrck, 1999

MSc Mathematics, Music, Philosophy & Pedagogy, University of Osnabrck, 1994

`http://www.city.ac.uk/people/academics/tillman-weyde`

What is a Neural Network [1/2]

Formal definition [Haykin, 1998]

A **neural network** is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experimental knowledge and making it available for use.

- Knowledge is acquired from the environment through a **learning process**
- Interneuron connection strengths, known as **synaptic weights**, are used to store the acquired knowledge.

What is a Neural Network [2/2]

Inspired by the human brain

- Nonlinearity
- Massively parallel
- Fault tolerant
- Learning from examples

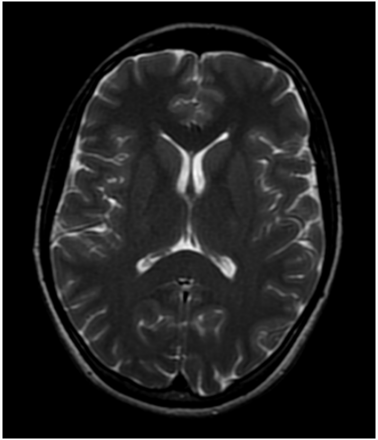


Image of Normal axial T2-weighted MR image of the brain, Author: Sean Novak, Creative Commons Attribution-Share Alike 4.0 International licence.

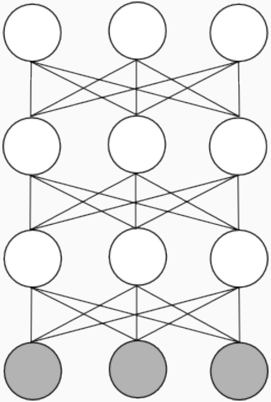
Neural Networks applications

Machine Learning algorithms

- Function approximation
- Regression analysis
- Classification
- Compression (associative memory)
- Blind signal separation
- Clustering
- Robotics

Architecture

- Massive interconnection of simple computing elements
- The processing units of the network are referred to as “neurons”
- Can be software- or hardware-based



Types of Neural Networks

1. Multilayer Perceptron
2. Kohonen Self-Organising Networks
3. Recurrent Neural Networks
4. Hopfield Model
5. Boltzmann Machines
6. Restricted Boltzmann Machines
7. Radial Basis Function Networks
8. Adaptive Resonance Memory
9. Associative Memory
10. Support Vector Network
11. ... many more ...

Boltzmann Machine

Definitions

- Energy function
  $E = -\left(\sum_{i,j} w_{ij} s_i s_j + \sum_i \theta_i s_i \right)$
- Energy gap
  $\Delta E_i = E_{i=off} - E_{i=on}$
- Probability of transition to a lower energy state
  $p_{i=on} = \frac{1}{1+e^{-\frac{\Delta E_i}{T}}}$
- The machine is "run" by sequentially updating the units until reaching a thermal equilibrium

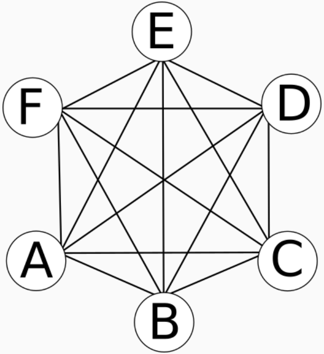


Figure 1: A graphical representation of a Boltzmann Machine with six symmetrically connected units.

"Learning" a Boltzmann Machine

- Probability of the network settling in state E_i can be expressed as $\frac{1}{Z}e^{-\frac{E_i}{k_B T}}$
- Adjust the parameters in such way that the probability distribution fits the training data
- Learning occurs in two phases:
  - Positive phase
  - Negative phase
- Update rule for gradient ascent-based learning $\Delta w_{ij} = \eta \left(p_{ij}^{+} - p_{ij}^{-} \right)$
- **Boltzmann machines are difficult to train**

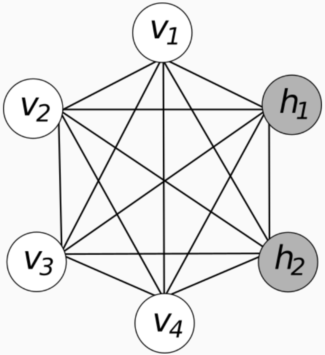


Figure 2: A Boltzmann Machine with four visible $V = \{v_1, v_2, v_3, v_4\}$ and two hidden units.

Restricted Boltzmann Machines

New topology

- Two layers of processing elements – a bipartite graph
- The restrictions applied imply that the visible variables are independent given the state of the hidden units and vice versa
- The RBM update rule can be written as $w_{ij} \leftarrow w_{ij} + \eta[\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}]$
- Sampling from the data distribution can be performed in one parallel step
- Sampling from the model still requires Gibbs sampling (we can use CD-k)

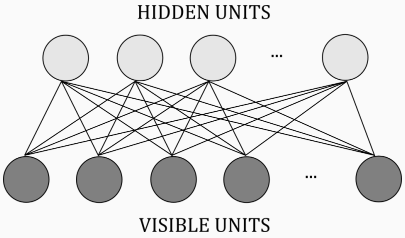


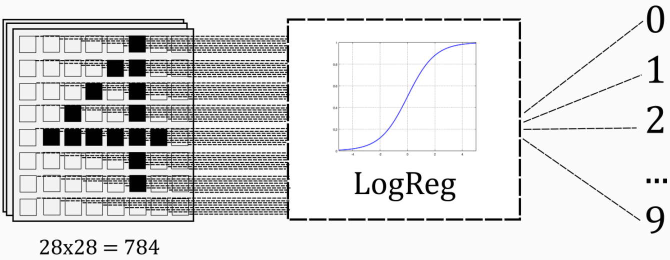
Figure 3: Restricted Boltzmann Machine

RBM Applications

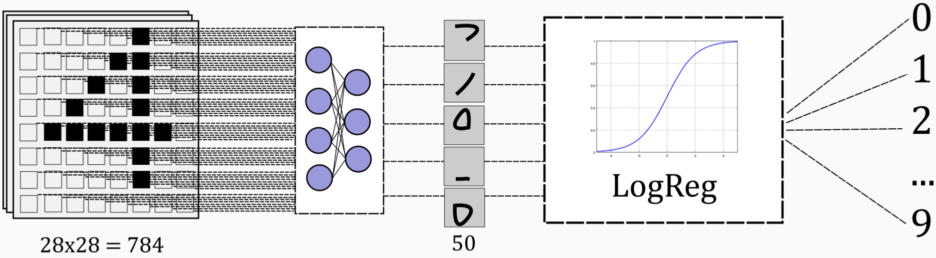
Wide range of applications

- Dimensionality reduction
- Collaborative filtering
- Classification
- Extraction of semantic document representation

RBM in an ML pipeline [1/2]



RBM in an ML pipeline [2/2]



The state of RBMs

RBMs for Big Data

- RBMs are still computationally intensive
- Big Data = millions and billions of parameters
- Parameter estimations for a conventional RBM can take weeks
- Numerous attempts to develop a parallelized model – all using GPU-based computing
    - Reduce training time from weeks to 1 day [Raina, Madhavan, Ng, 2009]
    - 99.5% of the time was spent on moving data
    - 4GB of shared memory can fit only up to 1 billion parameters
- **Can we do it in Hadoop?**

Example

Linear Regression with Normal Equations

- Simple Linear Regression
  - Dependent and independent variables $(\boldsymbol{X}, \boldsymbol{y})$
  - $h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$
- To estimate the parameters we have to minimise $J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$
- If $J(\theta)$ is convex then $\theta = (X^T X)^{-1} X^T y$
- ```
  a = t(X) %*% X + diag(lambda);
  b = t(X) %*% y;
  theta = solve(a,b);
  ```

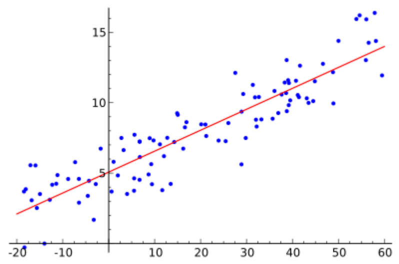
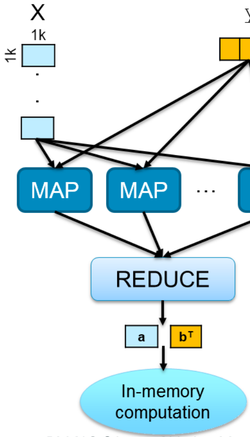
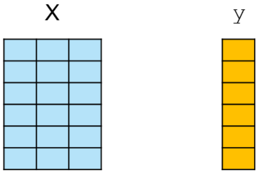


Figure 4: Simple Linear Regression
  ```

Optimal Execution Plan

```
a = t(X) %*% X + diag(lambda);
b = t(X) %*% y;
theta = solve(a,b);
```

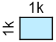




500 features
300M observations
4TB text file

300M observations
9GB text file

X^TX for each
y^TX for each



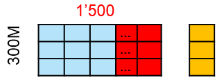
<u>(a,b) < 2 MB</u>

1. get b
2. call `solve(a,b)`

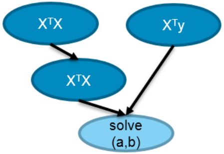
<u>Cluster Configuration</u>
3.5 GB Map Task JVM
7 GB In-memory Master JVM
128 MB HDFS block size

Plan Changes

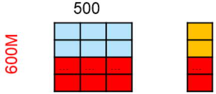
- 3 times more attributes



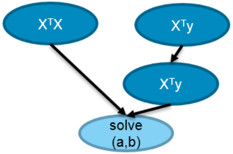
Cluster Configuration
3.5 GB Map Task JVM
7 GB In-memory Master JVM
128 MB HDFS block size



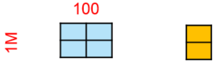
- 2 times more observations



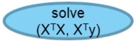
Cluster Configuration
3.5 GB Map Task JVM
7 GB In-memory Master JVM
128 MB HDFS block size



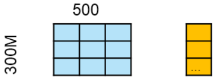
- The dataset fits in memory



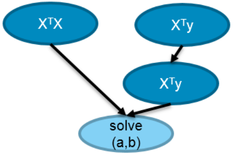
Cluster Configuration
3.5 GB Map Task JVM
7 GB In-memory Master JVM
128 MB HDFS block size



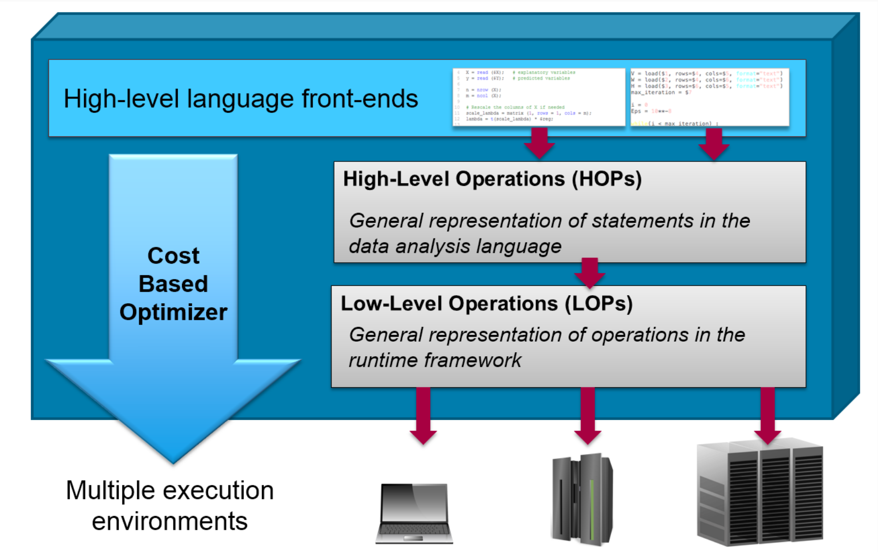
- Cluster configuration change



Cluster Configuration
1.5 GB Map Task JVM
7 GB In-memory Master JVM
128 MB HDFS block size



SystemML


High-level language front-ends
High-Level Operations (HOPs)
General representation of statements in the data analysis language
Low-Level Operations (LOPs)
General representation of operations in the runtime framework
Cost Based Optimizer
Multiple execution environments

RBM in SystemML

RBM training using DML

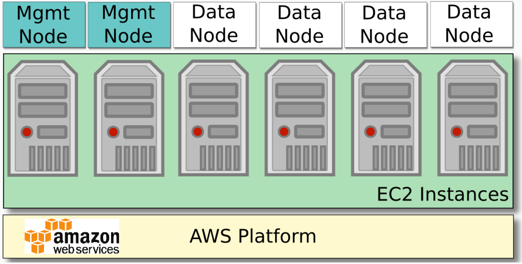
```
# POSITIVE PHASE
# Compute the probabilities P(h=1|v)
p_h1_given_v = 1.0 / (1 + exp(-(v_1 %*% w + b)))
# Sample from P(h=1|v)
h1 = p_h1_given_v > rand(rows = batch_N, cols = hidden_units_count)

# NEGATIVE PHASE
# Compute the probabilities P(v2=1|h1)
p_v2_given_h1  = 1.0 / (1 + exp(-(h1 %*% t(w) + a)))
# Compute the probabilities P(h2=1|v2)
p_h2_given_v2 = 1.0 / (1 + exp(-(p_v2_given_h1 %*% w + b)))
```

The Test Cluster

The specs

- Instance type – m4.large
- vCPU(s) – 2x 2.4 GHz Intel Xeon E5-2676v3
- Memory – 8 GB
- Storage – EBS Optimised – 40 GB magnetic
- Network performance – Moderate



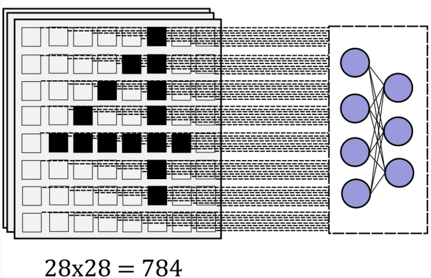
MNIST Dataset

- Database of handwritten
- 60'000 training examples
- 10'000 test examples

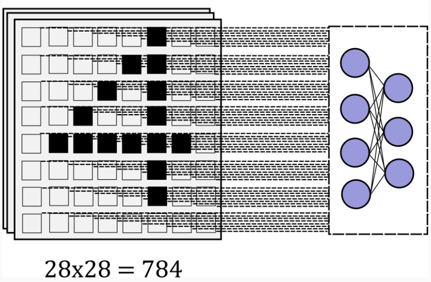
Image from "Gradient-Based Learning Applied to Document Recognition", LeCun, Y. and Bottou, L. and Bengio, Y. and Haffner, P., Proceedings of the IEEE, November 1998

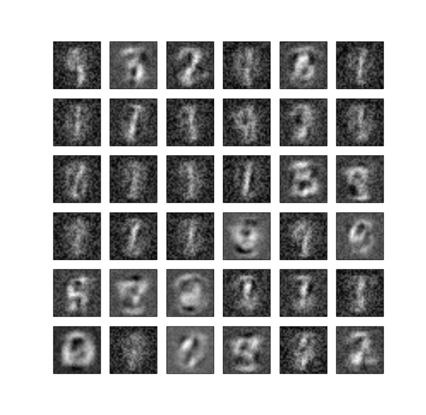
Learnt Features

200 output features



200 output features





Impact on convergence and accuracy

SVMs and Naïve Bayes

- No significant impact on accuracy ($91.55\% \rightarrow 91.49\%$, $83.65\% \rightarrow 84.35\%$)

Multinomial Logistic Regression – converges 150 times faster

```
-- Outer Iteration 100: Had 1538 CG iterations, trust bound REACHED
   -- Obj.Reduction:  Actual = 0.39088532897949335,  Predicted = 0.38795690261300836  (A/P
   -- New Objective = 12409.243355281007,  Beta Change Norm = 0.07383856136561 43,  Gradient
Termination / Convergence condition satisfied.
230m53.441s
...
-- Outer Iteration 18: Had 168 CG iterations
   -- Obj.Reduction:  Actual = 1.145968653872842E-4,  Predicted = 1.1458312793505605E-4  (A
   -- New Objective = 15096.583584691629,  Beta Change Norm = 0.012169174253463384,  Gradie
Termination / Convergence condition satisfied.
1m29.347s
```

But does it scale?

Project Gutenberg

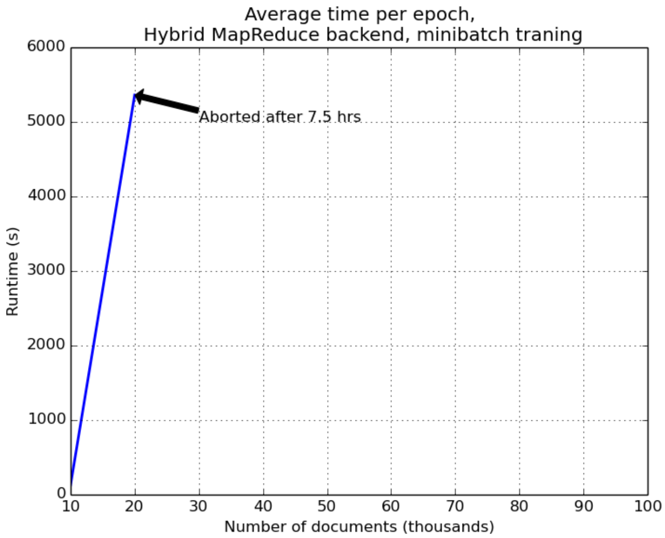
- 53'000 e-books
- About 12GB of data
- Most books have associated meta-data

$$\boldsymbol{D} = \begin{pmatrix} y_1 & x_{1,1} & \cdots & x_{1,n} \\ y_2 & x_{2,1} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ y_m & x_{m,1} & \cdots & x_{m,n} \end{pmatrix} \tag{1}$$

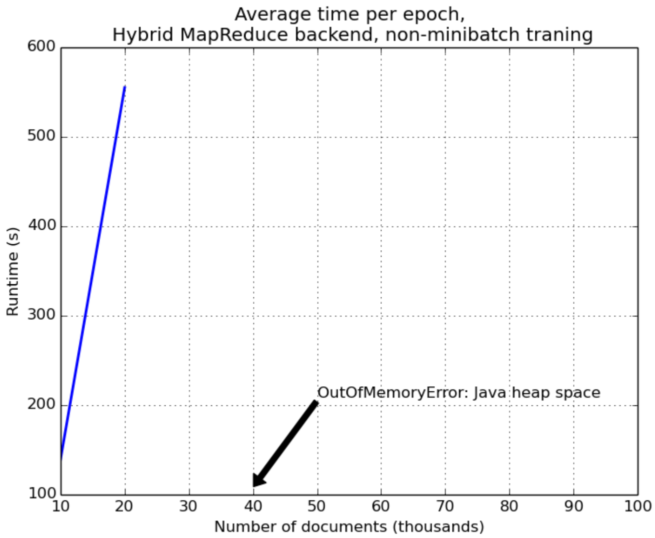
Pre-processing

- Remove invalid entries
- Reconcile books and meta-data
- Transform documents into a suitable format
- Create subsets of 10k, 20k, 40k, etc.

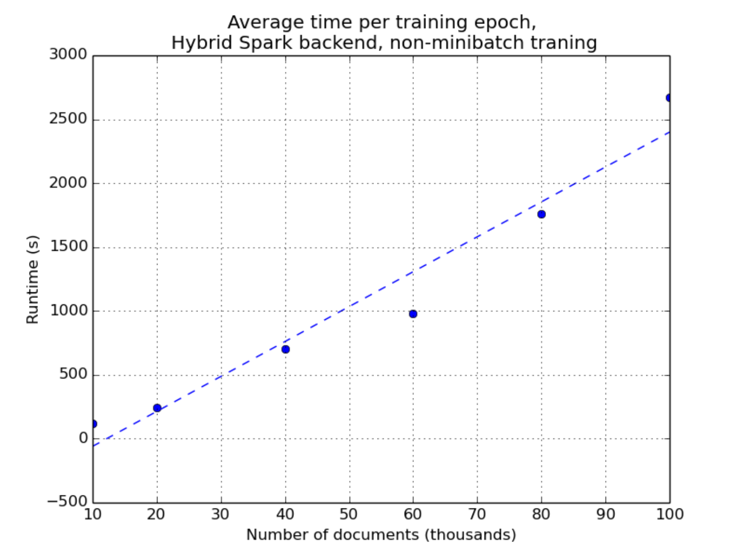
MR2, RBM mini-batches



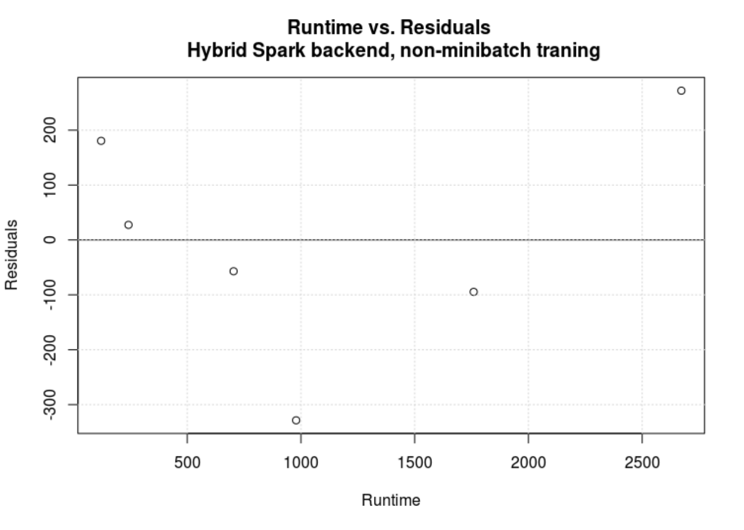
MR2, RBM no mini-batches



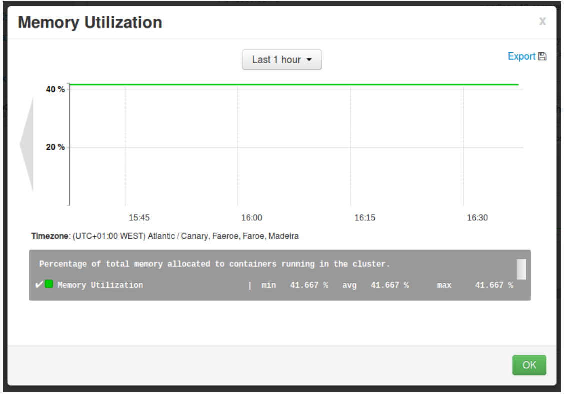
Spark, RBM mini-batches



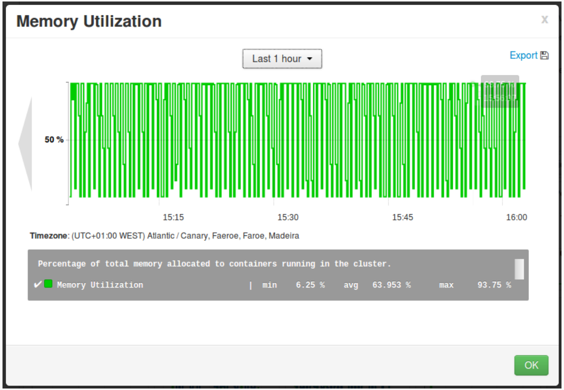
Spark, RBM mini-batches

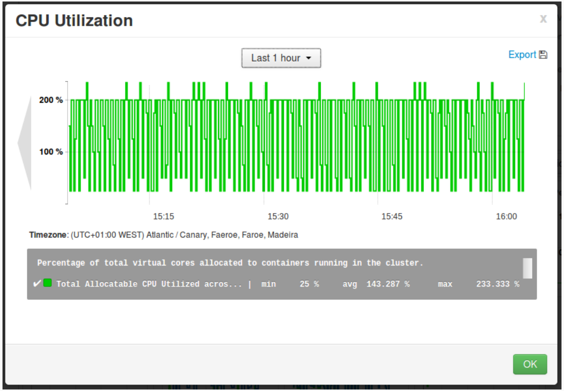


Some cluster stats – Spark back-end



Some cluster stats – MapReduce back-end





Summary

What we did

- Distributed RBM learning on Hadoop is possible
- Proved it is functional and it scales
- Looked into optimal batch sizes and learning rates
- Contributed to SystemML

Our TODO list

- Test a distributed pipeline
- Confirm the relationship type
- Tune the Hadoop cluster
- Compare to GPUs

Q&A