Deep Dive on Apache MXNet with Python

Julien Simon, AI Evangelist, EMEA
@julsimon

What to expect

- An introduction to Deep Learning and Apache MXNet
- Demos using Jupyter notebooks on Amazon SageMaker
- Resources
- Q&A

- Artificial Intelligence: design software applications which exhibit human-like behavior, e.g. speech, natural language processing, reasoning or intuition

- Machine Learning: teach machines to learn from data without being explicitly programmed

- Deep Learning: using neural networks, teach machines to learn from data where features cannot be explicitly expressed

Amazon Alexa is based on Deep Learning

Amazon AI is based on Deep Learning

Vision Services

Amazon Rekognition Image
Deep learning-based image analysis

Learn more »

Amazon Rekognition Video
Deep learning-based video analysis

Learn more »

Conversational chatbots

Amazon Lex
Build chatbots to engage customers

Learn more »

Language Services

Amazon Comprehend
Discover insights and relationships in text

Learn more »

Amazon Translate
Fluent translation of text

Learn more »

Amazon Transcribe
Automatic speech recognition

Learn more »

Amazon Polly
Natural sounding text to speech

Learn more »

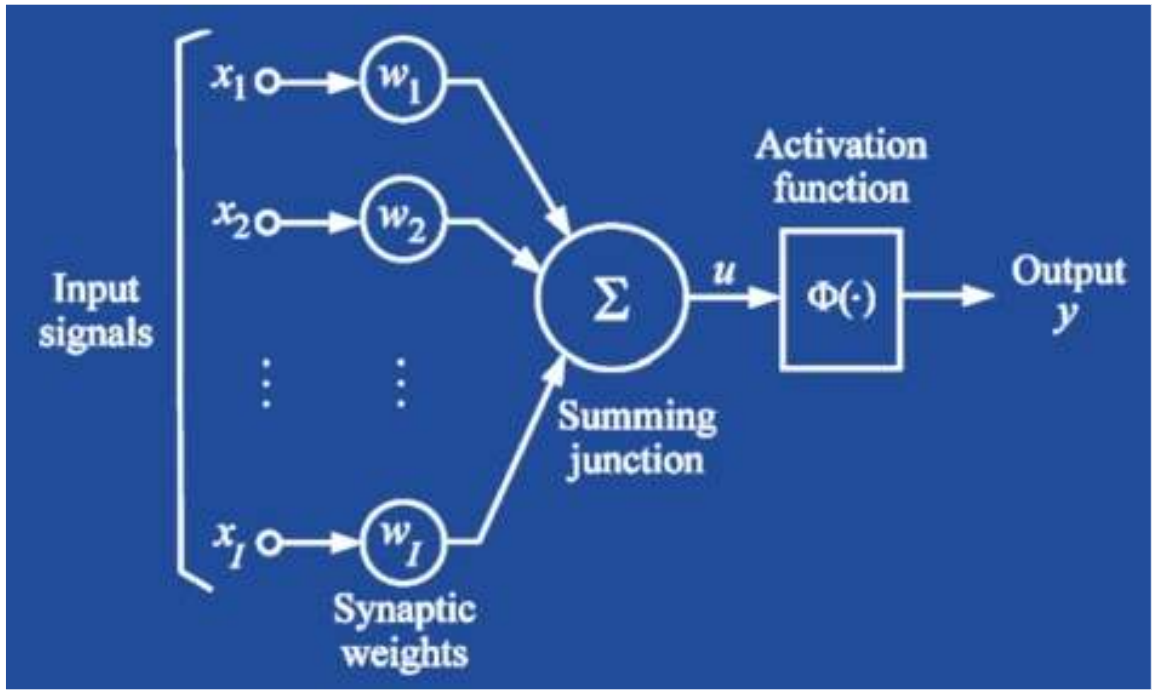
https://www.oreilly.com/ideas/self-driving-trucks-enter-the-fast-lane-using-deep-learn

Last June, tuSimple drove an autonomous truck

for 200 miles from Yuma, AZ to San Diego,

An introduction to Deep Learning

The neuron



$$\sum_{i=1}^{I} x_i * w_i = u$$

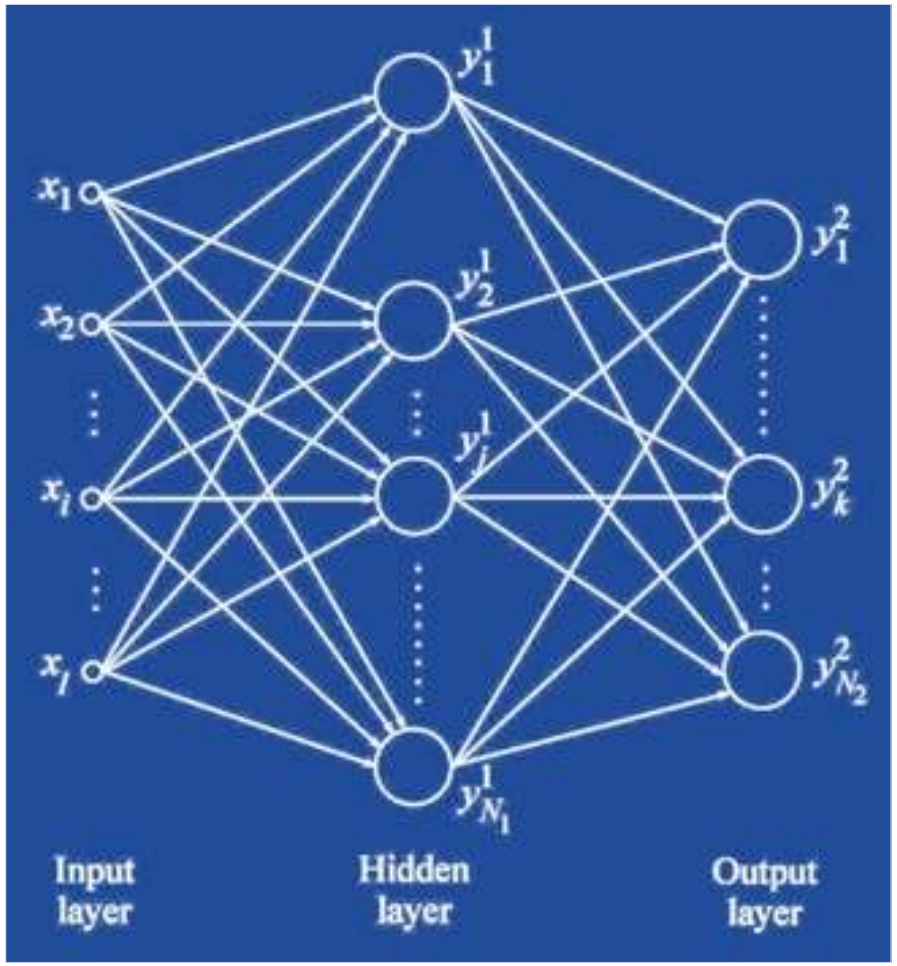
”Multiply and Accumulate”

Activation functions

Name	Plot	Equation
Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Logistic (a.k.a. Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$
ArcTan		$f(x) = \tan^{-1}(x)$
Softsign [7][8]		$f(x) = \frac{x}{1 + \lvert x \rvert}$
Rectified linear unit (ReLU)[9]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$

Source: Wikipedia

Neural networks



x =

$x_{11}, x_{12}, \ldots x_{1l}$
$x_{21}, x_{22}, \ldots x_{2l}$
… … …
$x_{m1}, x_{m2}, \ldots x_{ml}$

l features

m samples

y =

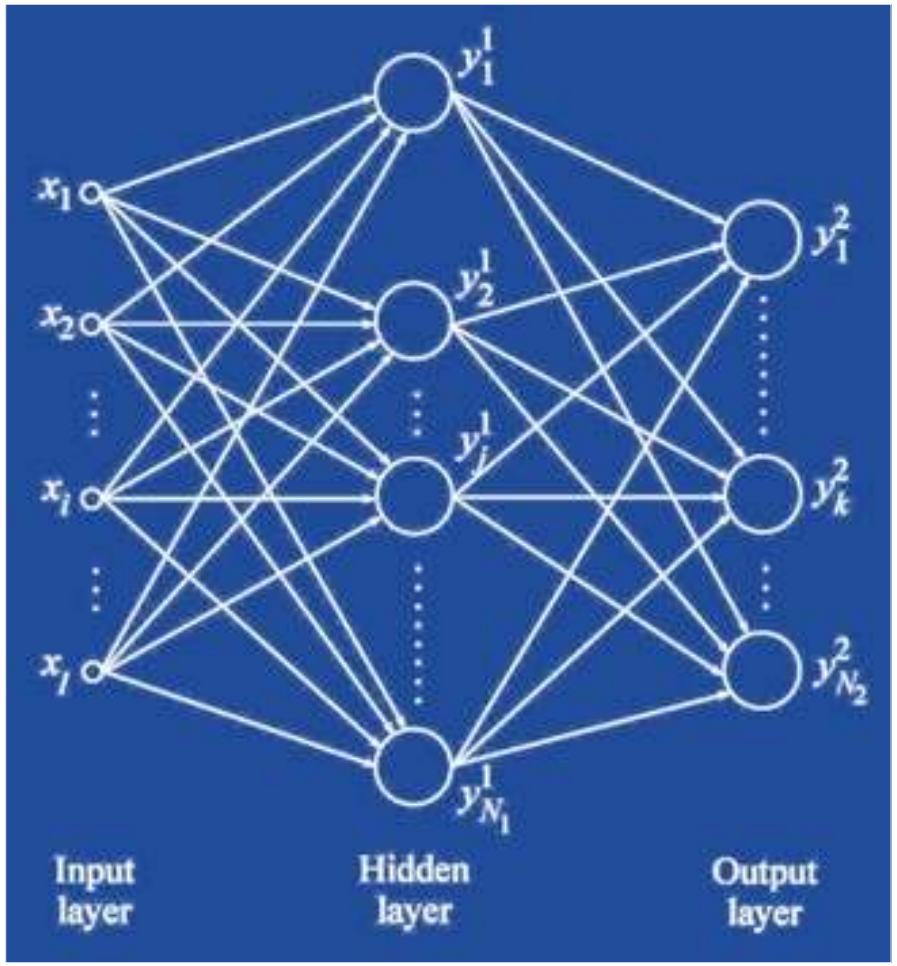
y_1
y_2
…
y_m

0,0,1,0,0,…,0
1,0,0,0,0,…,0
…
0,0,0,0,1,…,0

One-hot encoding

m labels,
N_2 outputs

Neural networks



l features

$$x = \begin{bmatrix} x_{11}, x_{12}, \ldots x_{1l} \\ x_{21}, \ldots x_{2l} \\ \ldots \\ x_{m1}, \ldots \end{bmatrix}$$

m samples

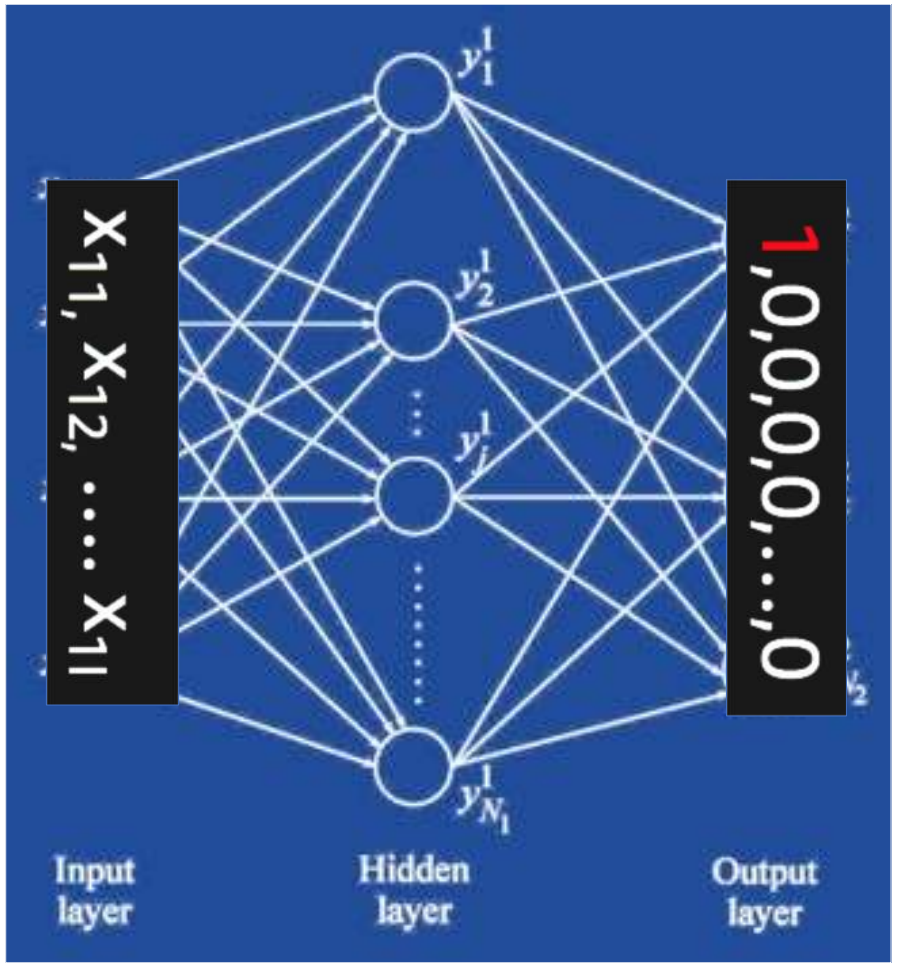
$$y = \begin{bmatrix} y_1 \\ y_2 \\ \ldots \\ y_m \end{bmatrix} \quad \begin{bmatrix} 0,0,1,\ldots,0 \\ 1,0,0,\ldots,0 \\ \ldots \\ 0,0,0,\ldots,0 \end{bmatrix}$$

m labels,
N_2 categories

One-hot encoding

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Neural networks



Initially, the network will not predict correctly

$f(X_1) = Y'_1$

A loss function measures the difference between the real label Y_1 and the predicted label Y'_1

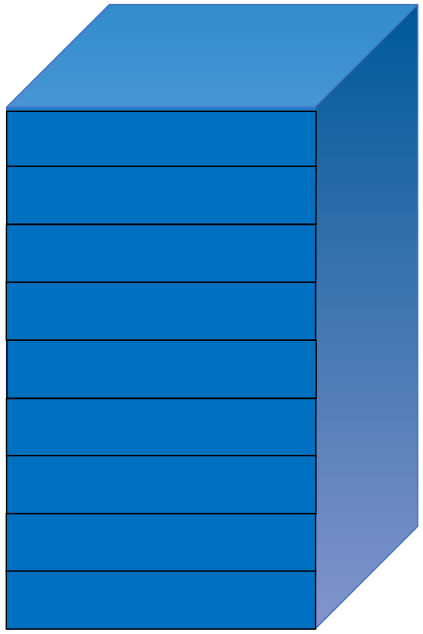
$error = loss(Y_1, Y'_1)$

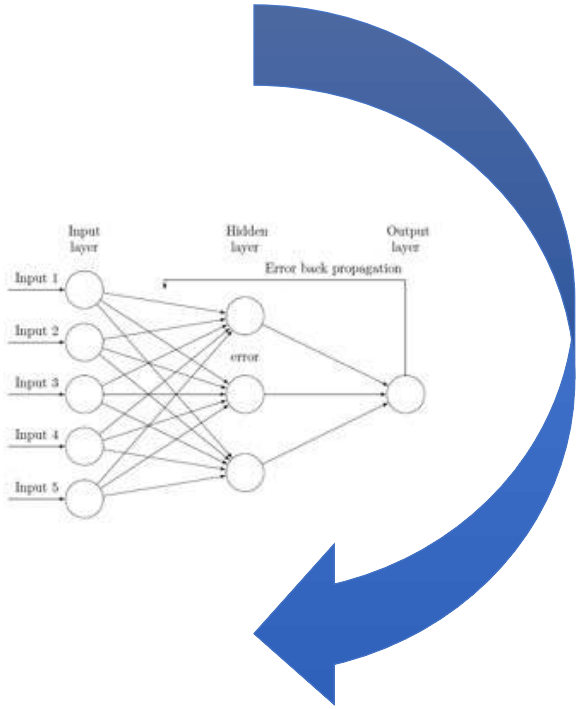
For a batch of samples:

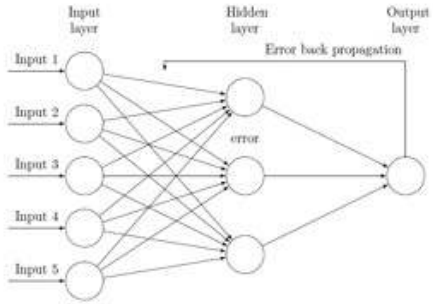
$$\sum_{i=1}^{batch\ size} loss(Y_i, Y'_i) = \text{batch error}$$

The purpose of the training process is to minimize error by gradually adjusting weights

Training







Trained neural network

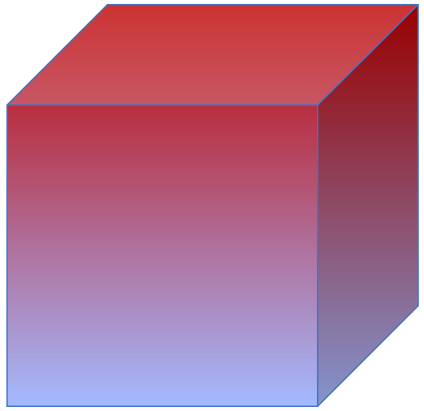
Training data set

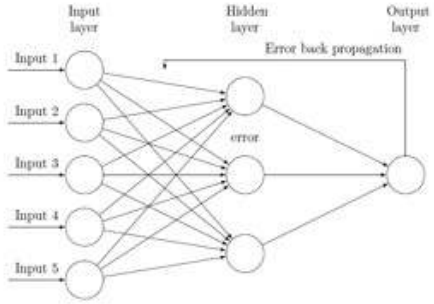
Backpropagation

Batch size
Learning rate
Number of epochs

Hyper parameters

Validation





Validation accuracy

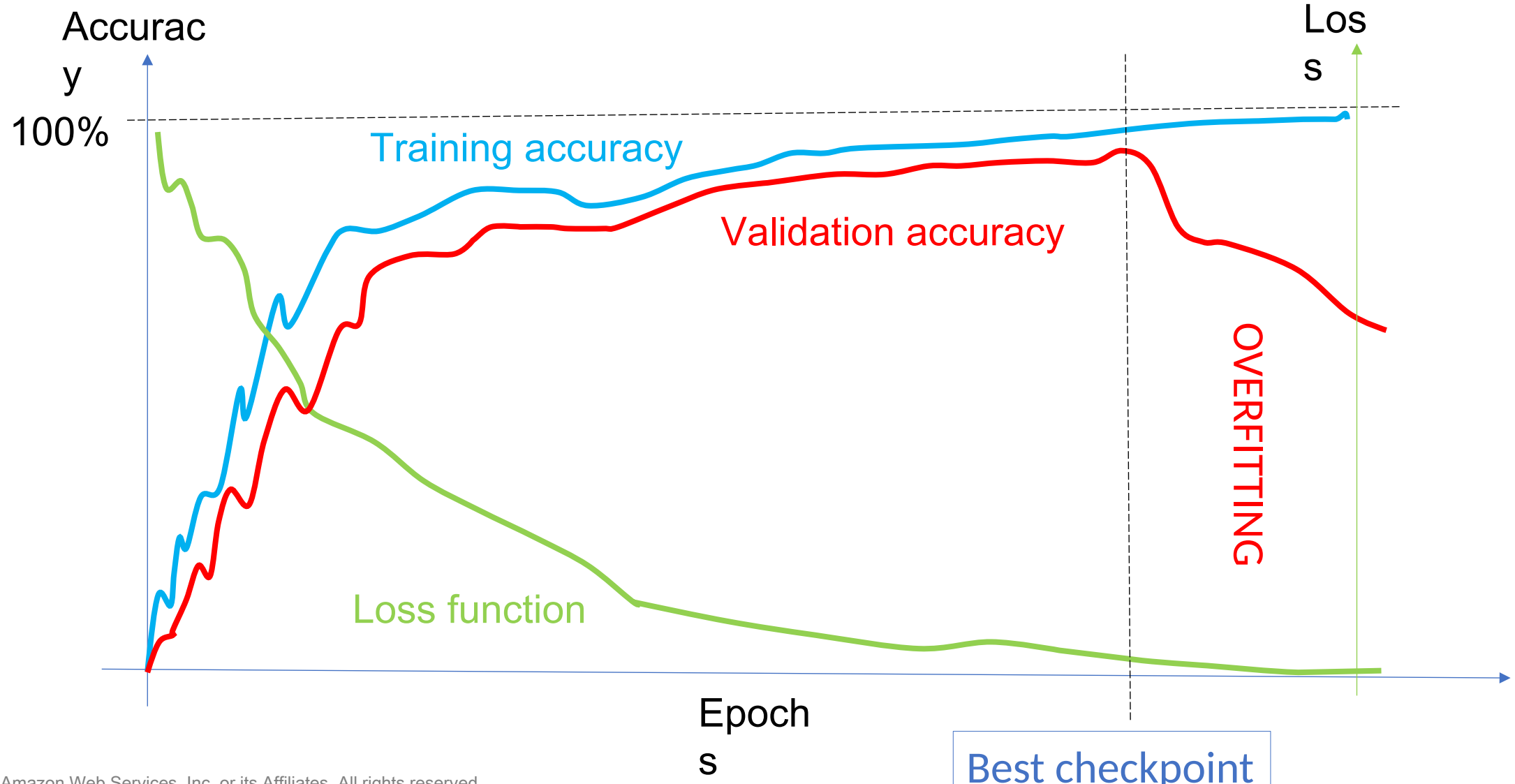
Validation data set

Trained neural network

Prediction at the end of each epoch

Save the model at the end of each epoch

Early stopping

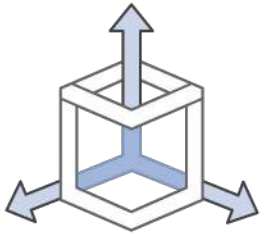


Apache MXNet

Apache MXNet: Open Source library for Deep Learning

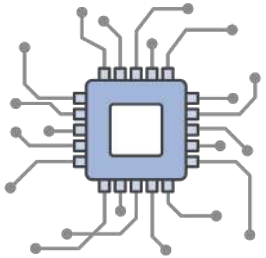
Programmable

Simple syntax, multiple languages



Portable

Highly efficient models for mobile and IoT

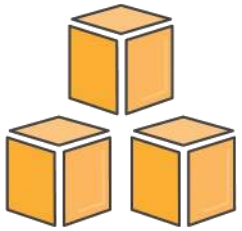


High Performance

Near linear scaling across hundreds of GPUs

Most Open

Accepted into the Apache Incubator



Best On AWS

Optimized for Deep Learning on AWS

MXNet 1.0 released on December 4th

The Apache MXNet API

- Storing and accessing data in multi-dimensional arrays
  → *NDArray* API
- Building models (layers, weights, activation functions)
  → *Symbol* API
- Serving data during training and validation
  → Iterators
- Training and using models
  → *Module* API

Model Server for Apache MXNet

ubuntu/python-2.7	ubuntu/python-3.5
AWS CodeBuild Passing	AWS CodeBuild Passing

Model Server for Apache MXNet (MMS) is a flexible and easy to use tool for serving Deep Learning models.

Use MMS Server CLI, or the pre-configured Docker images, to start a service that sets up HTTP endpoints to handle model inference requests.

https://github.com/awslabs/mxnet-model-server/

https://aws.amazon.com/blogs/ai/announcing-onnx-support-for-apache-mxnet/

Resources

https://aws.amazon.com/machine-learning

https://aws.amazon.com/blogs/ai

https://mxnet.incubator.apache.org

https://github.com/apache/incubator-mxnet

https://github.com/gluon-api

https://github.com/awslabs/sockeye

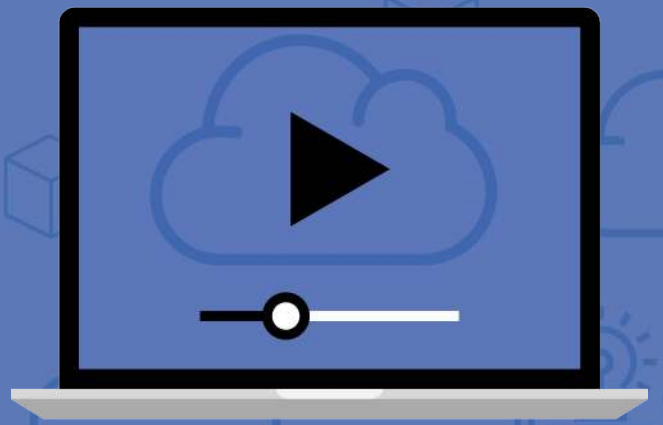
https://medium.com/@julsimon

Demos

https://github.com/juliensimon/dlnotebooks

1) Synthetic data set
2) Classify images with pre-trained models
3) Learn MNIST with a Multi-Layer Perceptron
4) Learn MNIST with the LeNet CNN
5) Predict handmade MNIST samples
6) Train and host a MNIST model with Amazon SageMaker

Thank you!



Julien Simon, AI Evangelist, EMEA
@julsimon