Today's Agenda

Introduction to Deep Learning [Tom Hanlon, 15 min]

Leveraging Cloudera DS Workbench [Vartika Singh, 30 min]

Deep Learning Healthcare Application [Dave Kale, 30 min]

Break 3:00 PM

One Hour Hands on Lab

Productionizing Deep Learning Models [Josh Patterson, 15 min]

Q&A
Introduction to Deep Learning
Table of Contents

What is DL? DL4J?

Spark, Parallelism, and DL4J

Use Cases
What is Deep Learning

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.
A Diagram

Field of Artificial Intelligence

Field of Machine Learning

Deep Learning

skymind
A Neural Network
Biological Neurons

- Biological Neuron: An electrically excitable cell that processes and transmits information through electrical and chemical signals
- Biological Neural Network: An interconnected group of neurons
Features of Artificial Neural Networks Inspired by Biological Neurons

- Connections
  - Each Neuron is connected to other Neurons
- Learning
  - Each Connection has a weight and bias parameters assigned
  - As network Learns parameters are adjusted
- Activation
  - Output of Neuron is input moderated by nonlinear activation function
Role of Artificial Neural Network

Learns or Trains to perform tasks that traditional programming methods find rather challenging.

- Speech recognition
- Object recognition
- Computer vision
- Pattern recognition.
Deep Learning Considerations

- Inspired by the brain
  - Very basic implementation
- Brain has huge number of Neurons
- Brain has non-linear connections
- Creates similar distributed units of functionality
- Knowledge comes through connections
The Rise and Fall of Neural Networks

- 40 year old theory and practice
- Hype cycle followed by delusion
- Repeat cycle
- Incremental improvements over time
Why Neural Networks Now?

- 2012 Neural Network dominates image recognition
- Leads to current boom
Simple Neural Network

- Very Simple Neural Network to demonstrate internals
Simple Network

The Goal: Input=0.5 Output=0.8
The Design:

- Apply Random weights, plus activation function per neuron => output

\[
\text{Input} \times \text{Weight} + \text{Bias} \Rightarrow \text{Activation Function}
\]

\[
(0.5 \times 2.01 - 0.00) \times -0.34 + 0.01 = -0.3317
\]
Calculate error

- backpropagate to adjust weights
- repeat

![Diagram showing neural network with input, single node hidden layer, and output layer with Backpropagation highlighted.]
Intro to DeepLearning4J

DeepLearning Framework built in Java

Supported by Skymind.io

Enterprise focus
Goals of the DeepLearning4J project

- Provide a Toolkit for using DeepLearning on the JVM
  - Enterprise users
  - Security
  - Flexibility
DeepLearning4J sub-projects

- DataVec
  - Tools for ETL
- ND4J
  - NUmeric Arrays
  - NumPY for the JVM
- libnd4j
  - Native Libraries for GPUs/CPUs
- DeepLearning4J
  - Tools to train Neural Networks
DataVec

- ETL Libraries purpose built for Neural Networks
  - Neural networks process numeric arrays
  - Datavec helps you get from your_data => numeric array
Data Sources Supported by DataVec

- Log files
- Text documents
- Tabular data
- Images and video
- and more !!
DataVec Features

- Transformation
- Scaling
- Shuffling
- Joining
- Splitting
Commonly Used Features

- **RecordReaders**
  - Read files or input, convert to List of Writables

- **Normalizers**
  - Standardize, scale or normalize the data

- **Transform Process**
  - Join datasets, replace strings with numerics, extract labels
Diagram of available ETL paths
Applying Labels

- DataVec provides the following tools for generating Labels
  - ParentPathLabelGenerator
  - PathLabelGenerator
Available ND4J Pre-Processors

- **ImagePreProcessingScaler**
  - min max scaling default 0 + - 1

- **NormalizerMinMaxScaler**
  - Scale values observed min -> 0, observed max -> 1

- **NormalizerStandardize**
  - moving column wise variance and mean
  - no need to pre-process
Image Transforms

• Included Libraries
  ◦ JavaCV
  ◦ OpenCV
  ◦ ffmpeg

skymind
Frequently Used DataVec classes

- CSVRecordReader
  - CSV text data
- ImageRecordReader
  - Convert image to numeric array representing pixel values
- JacksonRecordReader
  - Parses JSON records
- ParentPathLabelGenerator
  - Builds labels based on directory path
- Transform, Transform Process Builder, TransformProcess
  - Conversion tools
ND4J

- Provides scientific computing libraries
- Main features
  - Versatile n-dimensional array object
  - Multiplatform functionality including GPUs
  - Linear algebra and signal processing functions
ND4J and DeepLearning

- Classes frequently Used
  - DataSet
    - Container for INDArrays of Features/Labels
  - DataSetIterator
    - Build DataSet from RecordReader
libND4J

- The C++ engine that powers ND4J
  - Speed
  - CPU and GPU support
Distributed Training in Spark

- Documentation
  - https://deeplearning4j.org/spark
- Examples
  - https://github.com/deeplearning4j/dl4j-examples/tree/master/dl4j-spark-examples
Non-Distributed Training overview

- Process minibatch
  - Calculate Loss Function
  - Calculate direction of less error
  - Update weights in that direction
Non-Distributed Training Challenges

- Number of parameters may be large
- CPU or preferably GPU intensive
- Update large multi-dimensional matrix of numeric values
Distributed Training Implementation

- Workers process minibatch
- Calculate Loss Function
- Calculate Gradient update
- Submit to Parameter server
- Parameter Server averages weights from workers
- Ships averaged weights to workers
New Feature Parameter Sharing

- Peer to Peer Sharing of Parameters
- Remove dependency on master process
- More efficient
DL4J Spark Examples

- [https://github.com/deeplearning4j/dl4j-examples/tree/master/dl4j-spark-examples](https://github.com/deeplearning4j/dl4j-examples/tree/master/dl4j-spark-examples)
JavaSparkContent sc = ...;
JavaRDD<DataSet> trainingData = ...;
MultiLayerConfiguration networkConfig = ...;

//Create the TrainingMaster instance
int examplesPerDataSetObject = 1;
TrainingMaster trainingMaster = new ParameterAveragingTrainingMaster.Builder(exampl
  .(other configuration options)
  .build();

//Create the SparkDl4jMultiLayer instance
SparkDl4jMultiLayer sparkNetwork = new SparkDl4jMultiLayer(sc, networkConfig, train

//Fit the network using the training data:
sparkNetwork.fit(trainingData);
How to Participate and Contribute

- Chat with us on Gitter
  - https://gitter.im/deeplearning4j/deeplearning4j
- Contribute
  - https://github.com/deeplearning4j/
Paths to Production
With DL4J

1. Build Model from Scratch
2. Literature Search
3. Load PreTrained Model
4. Train in DL4J
5. Deploy For Inference
From Keras
Execution

- Your Model
  - CPU
  - Parallel Wrapper
  - SparkDL4jMultiLayer
    - GPU/s
    - Spark

Skymind