DEEP LEARNING FOR ENTERPRISE

Ari Kamlani | ari@skymind.io

Bank of America
Merrill Lynch

skymind

DL4J NLP Session

March 2019

skymind | DAILY SCHEDULE

09:00am - 12:00pm - Section 1 & Labs

12:00pm - 01:00pm - Lunch

01:00pm - 04:00pm - Section 2 & Labs

04:00pm - 05:00pm - Rapid Fire Q&A / Vendor Use Cases

Session Agenda

- Approximately 1 Hour Session
- Post 15 Min Huddle for Discussions
- 15 min Break

Rate your Experience

This is just to give us a feel of your knowledge with Deep Learning and DL4J Suite

- Language Expertise: Python, Scala, Java (Scale 1-5 for Each Language)
- Machine Learning Knowledge (Scale 1-5)
- Deep Learning Knowledge (Scale 1-5)
- Deep Learning Framework Knowledge (Scale 1-5, Give Examples)
- Experimentation and Deployment Platforms (As applicable)
- NLP Use Cases useful to the Organization

skymind

Agenda

Skymind

Overview of DL4J Suite & Installation

Intro to NLP & Use Cases

NLP Processing: Data Preparation

DL4J Constructs

Feature Extraction for Text

Shallow Networks for Language Models

Deep Networks for Language Models & Forecasting

Network Evaluation & Interpretation

Scaling and Tuning for Context

Q&A / Wrap-Up

skymind

Labs

- Iterators for Unstructured Text
- Tokenization
- Word Vectors
- Shallow Networks: Word2Vec
- Custom Iterator
- Deeper Networks: LSTM Cells

skymind

Today's Objectives

Skymind

Overview of DL4J Suite & Installation

Intro to NLP & Use Cases

NLP Processing: Data Preparation

DL4J Constructs

Feature Extraction for Text

Shallow Networks for Language Models

Deep Networks for Language Models & Forecasting

Extending Layers to Deep Networks

Combining Networks for Improved Performance

Scaling and Tuning for Context

Network Evaluation & Interpretation

Q&A / Wrap-Up

skymind

DL4J Constructs

Data Preparation

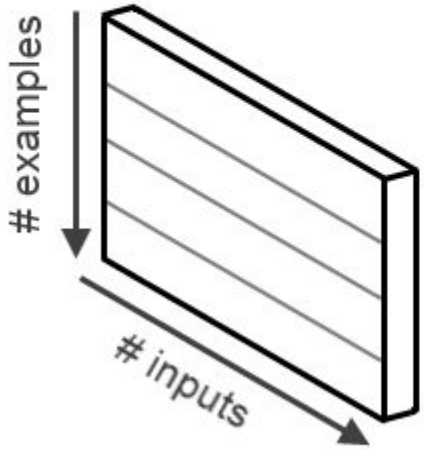
Layers

Sub-Networks

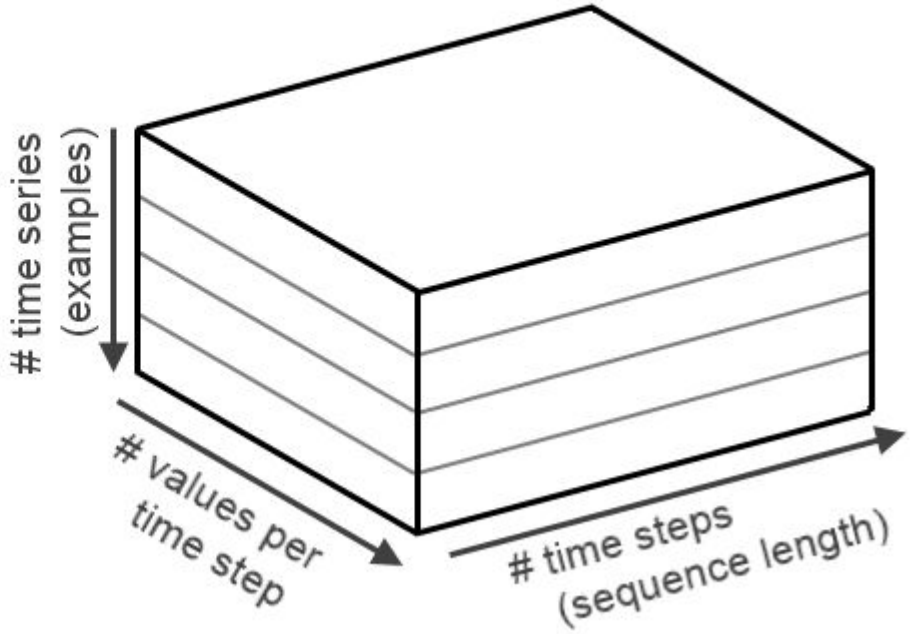
skymind | Network Shapes

- Feed Forward: [numExamples, inputSize], [numExamples, outputSize]
- RNNs/LSTMs: [numExamples, inputSize, sequenceLength], [numExamples, outputSize, sequenceLength]

Feed Forward Network Data



Recurrent Network Data



skymind | Input PreProcessors

- Input PreProcessor Layers are used to Convert between different Layer Types, operating on the input to the Layer
  - FeedForwardToRnnPreProcessor
  - FeedForwardToCnnPreProcessor, FeedForwardToCnn3DPreProcessor
  - CnnToRnnPreProcessor
  - CnnToFeedForwardPreProcessor, Cnn3DToFeedForwardPreProcessor
  - RnnToFeedForwardPreProcessor, RnnToCnnPreProcessor
- Instead of using the PreProcessors above, use .setInputTypes(InputType.<type>(args))
  - Automatically associated conversion between layer types
  - Automatically converts the association between number of Input Units (.nIn)
  - Can be used for Multiple Inputs of different types, e.g. CNN, Feed Forward

skymind | <type>DataSet, <type>DataSetIterator

- DataSet
  - **(Features, Labels) Pairing** of INDArray types per a given data partition
  - For use with Single Input, Single Output Array usage
  - MultiDataSet: For use with Multiple Inputs or Outputs, w/selectable indexable columns
- DataSetIterator
  - Iterates on a DataSet object
  - To be used in conjunction with DataVec RecordReaders (e.g. CSVRecordReader)
  - Sequences: CSVSequenceRecordReader, SequenceRecordReaderDataSetIterator
  - MIMO: RecordReaderMultiDataSetIterator, MultiDataSetIterator

Often in NLP, you will find yourself subclassing **DataSetIterator** and writing a custom Iterator

As the majority of the existing iterators are written for Structured Data not Unstructured

skymind | Document Text Iterators

DocumentIterator	Operates on a **stream (InputStream)**, based on underlying backed LineIterator
LabelAwareDocumentIterator	Given a File path of subdirectories, returns an InputStream for content to be read
FileDocumentIterator	Given a File path, returns an InputStream of file content to read

```
FileDocumentIterator iteratorDoc = new FileDocumentIterator(new File(path));
while(iteratorDoc.hasNext()) {
   InputStream inputStream = iteratorDoc.nextDocument();
   byte[] data        = new byte[1024];
   int    bytesRead = inputStream.read(data);
   log.info("Sub Document: {}, Bytes Read: {}", new String(data, "UTF-8").substring(0, bytesRead), bytesRead);
   inputStream.close();
}
```

skymind | Document Label Aware Text Iterators

Iterator<LabeledDocument>	Iterates over empassing text as a document (iterator.next())
LabelAwareIterator	Container for LabeledDocument and LabelsSource
FileLabelAwareIterator	Iterate over LabeledDocument based on **Document Categories**
FilenamesLabelAwareIterator	Iterate over LabeledDocument based on **Files (No Categories)**
SimpleLabelAwareIterator	Iterate over external generated data using an existing iterator
BasicLabelAwareIterator	Iterate over sentence/label pairs, e.g. as in Doc2Vec
DocumentIteratorConverter	Converts from DocumentIterator type to LabelAwareIterator type
AsyncLabelAwareIterator	Based on underlying AsyncIterator and existing LabelAwareIterator as input Fetched per buffer size, much like AsyncIterator

```
int bufferSize = 32;
LabelAwareIterator iterLabeled = new FileLabelAwareIterator.Builder()
                                        .addSourceFolder(new File(path))
                                        .build();
AsyncLabelAwareIterator iterAsyncDir = new AsyncLabelAwareIterator(iterLabeled, bufferSize);
LabelsSource srcDir    = iterAsyncDir.getLabelsSource();
List<String> labelsDir  = srcDir.getLabels();
log.info("Annotated --- Number Labels: {}, Labels: {}", srcDir.getNumberOfLabelsUsed(), labelsDir);
```

```
iterAsyncDir.reset();
while(iterAsyncDir.hasNext()) {
   LabelledDocument doc = iterAsyncDir.next();
   log.info("Annotated --- label:{}, content: {}",
     doc.getLabels(),
     doc.getContent().substring(0, 20)) ;
}
```

skymind | Mapping Providers

LabeledSentenceProvider	Iterates over sentences/docs
FileLabeledSentenceProvider	Hashmap Interface (labels, files)
CollectionLabeledSentenceProvider	List Interface (labels, content)
LabelAwareConverter	Converters from a LabeledAwareIterator

```
iterAsyncDir.reset();
LabeledSentenceProvider providerConv = new LabelAwareConverter(iterAsyncDir, labelsDir);
log.info("Annotated --- Num Classes: {}, Labels: {}", providerConv.numLabelClasses(),
providerConv.allLabels());
while (providerConv.hasNext()) {
   Pair<String, String> docPair = providerConv.nextSentence();
   log.info("Annotated --- Label: {}, Text: {}", docPair.getValue(), docPair.getKey());
}
```

Alternatively, instead of using an existing iterator, provide a hashmap or list interface with the other above classes...

skymind | Sentence Text Iterators

SentenceIterator	Iterates over Sentence Strings
BasicLineIterator	Single Line Iterator
LineSentenceIterator	Single Sentence Iterator, where each line is a sentence
CollectionSentenceIterator	Utility Operations to iterate over a sequence of strings as a collection
PrefetchingSentenceIterator	Asynchronous Data Latency Hiding via provided SentenceIterator
StreamLineIterator	Iterate over InputStream or DocumentIterator as lines of text
FileSentenceIterator	Iterate over a File or Directory to iterate over sequence of lines as sentences
UimaSentenceIterator	Iterate based on Analysis Engine
SentencePreProcessor	PreProcess a Sentence of Text

skymind | PreProcessors

DataSetPreProcessor

*PreProcess (.preProcess) type **DataSet***

Typical examples include NormalizerMinMaxScaler, NormalizerStandardizer

For Text, we want to process this differently, e.g. Lemmatization, Parts of Speech (POS)

SentencePreProcessor

*PreProcess (.preProcess) a type **String** Sentence*

*Typically use on subclasses of **SentenceIterator** to preprocess text per batch prior to Tokenization*

TokenPreProcess

*PreProcess (.preProcess) a type **String** Token*

```
SentenceIterator iter = new LineSentenceIterator(file);
iter.setPreProcessor(new SentencePreProcessor() {
  @Override
  public String preProcess(String sentence) {
     //execute a series of preProcessing steps
     return sentence.toLowerCase();
  }
});
```

skymind | Tokenization

Tokenization is categorized into the following categories:
*As of Release 1.0.0-Beta3, Bert functionality is not included, but available on the **master** branch*

- Tokenizer Factory: <type>TokenizerFactory
  - Type = {Default, NGram, BertWordPiece, PosUima}
- Tokenization: <type>Tokenizer
  - Type = {Default, DefaultStream, NGram, BertWordPiece, BertWordPieceStream}
- Token PreProcessing: <type>PreProcessor
  - Type = {Common, LowCase, Ending} via <tokenization>.setTokenPreProcessor
    - Common: removes common punctuation and lower cases
    - Ending: removes ‘ed’, ‘ly’, ‘ing’, ‘s’, ‘ss’, ‘.’
    - LowCase: lower cases a string

```
TokenizerFactory tokenizerFactory = new DefaultTokenizerFactory();
tokenizerFactory.setTokenPreProcessor(new CommonPreprocessor());
Tokenizer t = tokenizerFactory.create(sentence);
log.info("Number Tokens: {}", t.countTokens());
//Extract Tokens: moving cursor to end
log.info("Tokens: {}", t.getTokens());
//Alternatively iterate over tokens on a different tokenizer (.hasMoreTokens, .nextToken)
```

skymind | Record Readers

Record Readers per **ingesting and vectorizing Data** (just a few noted below)

Useful for reading complex and distributed records

Can furthermore had multiple Record Readers on a single iterator (.addReader)

- CSVRecordReader, JacksonRecordReader, FileRecordReader, InMemoryRecordReader
- CSVSequenceRecordReader, InMemorySequenceRecordReader
- CSVMultiSequenceRecordReader
- FileRecordReader, TfidfRecordReader (subclassed FileRecordReader)
- TransformProcessRecordReader, TransformProcessSequenceRecordReader

Per Unstructured Text:

The existing record readers would probably not fit many NLP tasks out of the box, so custom versions would need to be built, e.g.:

- Custom DataSetIterator
- Custom RecordReader

skymind | RecordReader Configuration

InputSplit

List of loadable locations used with a RecordReader, common ones used below as part of Initialization:

<type>InputSplit	Description	Usage
NumberedFileInputSplit	Per Sequences of Numbered Files	Transform the naming convention of your files
FileSplit	Splits up a root directory or single file	Commonly used with BalancedPathFilter
CollectionInputSplit	Per collection of URIs	Similar to LoadFromMetaData
InputStreamInputSplit	For Streaming raw data into records	Possibly w/External Connectors

*In more **Production based environments**, URIs will be provided from an external database, referencing the appropriate object store repository locations to load from and vectorize.*

Configuration

Key:Value settings used with a RecordReader underlying implementation

skymind

DL4J Constructs

Data Preparation

Layers

Sub-Networks

skymind | DL4J Layer Types

Commonly used NN Layers in DL4J Suite

- Feed-Forward (Dense) Layers, Embedding Layers (Dense Layer)
- Output Layers
- Convolutional Layers
- Recurrent Layers
- Unsupervised Layers (<type>AutoEncoder)
- Custom Layers
- Graph Vertices
- InputPreProcessors

Core Building Blocks of a NN

- Dense (FC) Layers: Associate Full Connectivity (leading to large number of Parameters)
- Convolutional: Capture Spatial Local Properties and Translational Invariances
- RNN/LSTM Cells: Capture Temporal Patterns in Sequences
- Custom Layers via existing layers and configuration via .instantiate:
  - Configuration (org.deeplearning4j.nn.conf.layers.Layer)
  - Implementation (org.deeplearning4j.nn.api.Layer)

*In Practice - For more **efficient computation***
the latter FC layers have been can replaced with fully convolutional layers

Embedding Layers

- Extends the FeedForward Layer
  - Mathematically Embedding Layer ~ Dense Layer w/OHE
  - Computationally more efficient, avoiding costly matrix multiplication and redundancy in dimensions
- Embedding Layer (2D) - Non-Sequence
  - Encodes each token element to a Dense Vector, e.g Data Dictionary (indices to vectors)
- Embedding Sequence Layer (3D) - Sequences
  - Embedding Layer for Sequences expecting fixed-length input

The Embedding Layers can be used only as the first Layer in the NN Configuration

skymind | Embedding Layers

Terminology

Term	Description
Look-Up Table	Weight Matrix (Embedding Matrix)
Embedding Layer	Encapsulates Hidden Layer, encapsulates Embedding Matrix
Number of Embeddings (.nIn)	Size of Vocab Dictionary = vocab.size()
Embedding Dimension (.nOut)	Embedding Width = Dimension of Dense Embedding (OHE Vectors -> Dense Vectors)
Input Length (.inputLength)	for 3D Embedding, Length of Input Sequence (e.g. fixed set number of words)

Shapes

Layer	Input	Output
Embedding Layer (2D)	[numExamples, 1]	[numExamples, numClasses]
Embedding Sequence Layer (3D)	[numExamples, inputLength]	[numExamples, nOut, inputLength]

skymind | Embedding Layers

Embedding Layer (2D)

```
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
        .activation(Activation.TANH)
        .weightInit(WeightInit.XAVIER)
        .list()
        .layer(0, new EmbeddingLayer.Builder()
                        .hasBias(true)
                        .nIn(vocabSize)
                        .nOut(embeddingDim)
                        .build())
        .layer(1, new OutputLayer.Builder()
                        .nIn(embeddingDim)
                        .nOut(numClasses)
                        .lossFunction(LossFunctions.LossFunction.MCXENT)
                        .activation(Activation.SOFTMAX)
                        .build())
        .build();
```

Embedding Sequence Layer (3D)

```
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
        .activation(Activation.TANH)
        .weightInit(WeightInit.XAVIER)
        .list()
        .layer(0, new EmbeddingSequenceLayer.Builder()
                .hasBias(true)
                .nIn(vocabSize)
                .nOut(embeddingDim)
                .inputLength(sequenceLength)
                .build())
        .layer(1, new RnnOutputLayer.Builder()
                .nIn(embeddingDim)
                .nOut(numClasses)
                .lossFunction(LossFunctions.LossFunction.MCXENT)
                .activation(Activation.SOFTMAX)
                .build())
        .build();
```

skymind | Custom Layers

Custom Layers via core DL4J

Via subclassing Configuration and Implementation Layers

*Transform Configuration to Implementation via .instantiate of *.conf.Layers.Layer*

- Layer Configuration Settings: org.deeplearning4j.nn.conf.layers.Layer

  import org.deeplearning4j.nn.conf.layers.Layer.FeedForwardLayer

  public class CustomLayerConf extends FeedForwardLayer

- Layer Implementation: org.deeplearning4j.nn.layers

  import org.deeplearning4j.nn.layers.BaseLayer

  public class CustomLayerImp extends BaseLayer<CustomLayerConf>

Custom Layers via SameDiff

Via org.deeplearning4j.nn.conf.layers.samediff.{SameDiffLayer, …} as Intermediate Layer

- Simplicity in defining and configuration the Network Layer
- Requiring only the gradients from the forward pass

skymind | Custom Loss Functions

Subclass functionality from **org.nd4j.linalg.lossfunctions.ILossFunction**

Interesting Loss Layers: LossKLD, LossHinge, LossMultiLabel

As the Loss Layer is a function of the Output Layer, provide it to the instantiation

- .computeScore
- .computeScoreArray
- .computeGradient
- .computeGradientAndScore

Or likewise, extend a SameDiffLayer, in this case **SameDiffOutputLayer**

- .createLayer returning an SDVariable type

skymind

DL4J Constructs

Data Preparation

Layers

Sub-Networks

skymind | DL4J Networks

- **MultiLayerNetwork**
  - Per basic linear sequence of layers
- **ComputationGraph** (functional api)
  - Introducing Different Types of Graph Vertices (acts a a Node) instead of Layers
  - Per Multiple Inputs or Outputs (MIMO), e.g. multi-task to the network
  - Per introducing multiple intermediate connections or skip connections

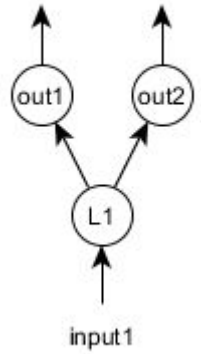
```
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
//set options ...
      .list()
      .layer(<indice>, new DenseLayer.Builder()
                        .nIn(neuronsIn)
                        .nOut(neuronsOut)
//set additional layer options
                        .activation(Activation.RELU)
                        .build())
//create additional layers
//create Output Layer
      .build();
```

```
ComputationGraphConfiguration conf = new NeuralNetConfiguration.Builder()
//set network options ...
   .graphBuilder()
   .addInput(<"input name">)  //.addInput or .addInputs
   .addLayer(<"layer name", new DenseLayer.Builder()
                                 .nIn(neuronsIn)
                                 .nOut(neuronsOut)
//set additional layer options
                                 .activation(Activation.RELU)
                                 .build(), <"input name to layer">)
//create additional layers
//create output Layer (e.g. "out")
   .setOutputs("out")
   .build();
```

skymind | Multiple Modalities (Multi-Task Learning=MTL)

- Performs multiple independent predictions
- Includes two different domains: e.g. space (convolutional) and time (temporal dynamics)
- Hybrid Architecture that includes both convolutional and recurrent components
- Preprocessors are available to deal with reshaping the data between layer types

```
ComputationGraphConfiguration conf = new NeuralNetConfiguration.Builder()
   .updater(new Sgd(0.001))
   .graphBuilder()
   .addInputs("input")
   .addLayer("L1", new DenseLayer.Builder().nIn(3).nOut(4).build(), "input")
   // classification
   .addLayer("out1", new OutputLayer.Builder()
         .lossFunction(LossFunctions.LossFunction.NEGATIVELOGLIKELIHOOD)
         .nIn(4).nOut(3).build(), "L1")
   // regression
   .addLayer("out2", new OutputLayer.Builder()
         .lossFunction(LossFunctions.LossFunction.MSE)
         .nIn(4).nOut(2).build(), "L1")
   .setOutputs("out1","out2")
   .build();
```

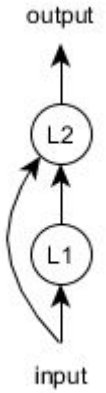


skymind | Skip (Residual) Connections

Method for adding the input matrix to the output of a Convolution

- Enables deep networks (many layers) ~ e.g. ResNet for Conv Networks
- Allowing for more complex architectures

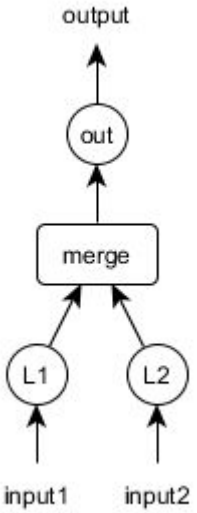
```
ComputationGraphConfiguration conf = new NeuralNetConfiguration.Builder()
   .updater(new Sgd(1e-3))
   .graphBuilder()
   .addInputs("input")
   .addLayer("L1", new LSTM.Builder().nIn(5).nOut(5).build(), "input")
   .addLayer("L2", new RnnOutputLayer.Builder().nIn(5+5).nOut(5).build(), "input", "L1")
   .setOutputs("L2")
   .build();
```



skymind | Merge Vertex

- Concatenates Output Layer Activations from each layer
- Useful for Architectures requires multiple auxiliary inputs
- Allowing for more complex architectures

```
ComputationGraphConfiguration conf = new NeuralNetConfiguration.Builder()
 .updater(new Sgd(1e-3))
 .graphBuilder()
 .addInputs("input1", "input2")
 .addLayer("L1",      new DenseLayer.Builder().nIn(3).nOut(4).build(), "input1")
 .addLayer("L2",      new DenseLayer.Builder().nIn(3).nOut(4).build(), "input2")
 //merge vertex should have 8 outputs (L1:4, L2:4)
 .addVertex("merge", new MergeVertex(), "L1", "L2")
 .addLayer("out",       new OutputLayer.Builder().nIn(4+4).nOut(3).build(), "merge")
 .setOutputs("out")
 .build();
```



DEEP LEARNING
FOR ENTERPRISE

help@skymind.ai

skymind

https://github.com/deeplearning4j/deeplearning4j

deeplearning4j / deeplearning4j

Watch 805 | Unstar 8,965 | Fork 4,267

Code | Issues 499 | Pull requests 9 | Projects 1 | Wiki | Insights

Deeplearning4j, ND4J, DataVec and more - deep learning & linear algebra for Java/Scala with GPUs + Spark - From Skymind http://deeplearning4j.org

java | gpu | deeplearning | neural-nets | deeplearning4j | dl4j | hadoop | spark | intellij | artificial-intelligence

22,525 commits | 105 branches | 49 releases | 219 contributors | Apache-2.0

skymind | Feedback

- From a scale of 1 to 10 how was your understanding of the topics covered?

  1 2 3 4 5 6 7 8 9 10

- How difficult was to understand?

  Too Easy Easy Just Right Hard Too Hard

- Were the topics covered what you expected?

- Any details you want to see explained tomorrow?

skymind

Material

Resources

Appendix

skymind | Increasing Memory

JVM Heap

Large Network Models need to have their heap space increased

Via IntelliJ

IntelliJ Preferences > Compiler > Command Line Options

-Xms1024m

-Xmx10g

-XX:MaxPermSize=2g

skymind | DOCUMENTATION

Notable Documentation

Skymind

https://skymind.ai/

https://blog.skymind.ai/

https://gitter.im/

OSS

https://deeplearning4j.org/

https://nd4j.org/

Notable Repositories

Skymind

https://github.com/deeplearning4j/deeplearning4j

https://github.com/deeplearning4j/dl4j-examples