Deep Learning with Java, part 2

PU Machine learning mit Java, Weka und UIMA

Nils Reiter,
nils.reiter@uni-koeln.de

February 2, 2021
Winter term 2020/21

Section 1

Looking Back

Exercise 11

`https://github.com/idh-cologne-machine-learning-mit-java/exercise-11`

Libraries

- Data representation: ND4J
- ETL library: Datavec
  - Stores n-dimensional arrays *outside* of the Java heap
  - "API mimics the semantics of Numpy, Matlab and scikit-learn"
  - libnd4j: C++ part
- Neural networks: Deeplearning4J (DL4J)
  - Can run on CUDA (GPU), partially written in C++
  - Complexity sources
    - Data sets that don't fit in memory (unlike Weka)
    - Efficiency for productive use

Libraries

- Data representation: ND4J
- ETL library: Datavec
  - Stores n-dimensional arrays *outside* of the Java heap
  - "API mimics the semantics of Numpy, Matlab and scikit-learn"
  - libnd4j: C++ part
- Neural networks: Deeplearning4J (DL4J)
  - Can run on CUDA (GPU), partially written in C++
  - Complexity sources
    - Data sets that don't fit in memory (unlike Weka)
    - Efficiency for productive use
- Similar to Python

Libraries

- Data representation: ND4J
- ETL library: Datavec
  - Stores n-dimensional arrays *outside* of the Java heap
  - "API mimics the semantics of Numpy, Matlab and scikit-learn"
  - libnd4j: C++ part
- Neural networks: Deeplearning4J (DL4J)
  - Can run on CUDA (GPU), partially written in C++
  - Complexity sources
    - Data sets that don't fit in memory (unlike Weka)
    - Efficiency for productive use
- Similar to Python
- Plan
  - ~~Today / Exercise 11: Data preparation~~
  - Next week / Exercise 12: Training an actual network

Section 2

Deep Learning with Java

DL4J

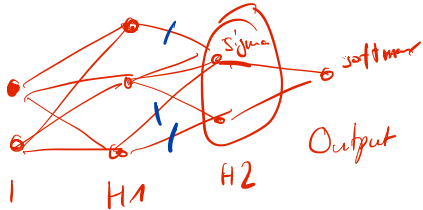
- Open source, integrated with Hadoop and Spark
  → Learning can be distributed over compute clusters

DL4J

- Open source, integrated with Hadoop and Spark
  - → Learning can be distributed over compute clusters
- Two steps
  - Define layout of artificial neural network and training parameters
  - Launch training, supervise progress

Designing Artificial Neural Networks

- Parameters
  - Activation function
  - Regularization
  - Weight updater
  - Weight initialization
  - Loss function
  - Seed values



Designing Artificial Neural Networks

- Parameters
  - Activation function
  - Regularization
  - Weight updater
  - Weight initialization
  - Loss function
  - Seed values
- Layout
  - Number of layers
  - Size of each layer
  - Type of layers

Designing Artificial Neural Networks

- Parameters
  - Activation function
  - Regularization
  - Weight updater
  - Weight initialization
  - Loss function
  - Seed values
- Layout
  - Number of layers
  - Size of each layer
  - Type of layers

```
1  MultiLayerConfiguration conf
2    = new NeuralNetConfiguration.Builder()
3    .seed(seed).list()
4    .layer(new DenseLayer.Builder().nIn(5)
5      .nOut(10).activation(Activation.TANH)
6      .weightInit(WeightInit.ZERO).build())
7    .layer(new DenseLayer.Builder().nIn(10).nOut(3)
8      .activation(Activation.SIGMOID)
9      .weightInit(WeightInit.UNIFORM).build())
10   .layer(new OutputLayer.Builder(
11     LossFunctions.LossFunction.HINGE)
12     .activation(Activation.SOFTMAX).nIn(3)
13     .nOut(outputNum).build())
14   .build();
```

Designing Artificial Neural Networks

- Parameters
  - Activation function
  - Regularization
  - Weight updater
  - Weight initialization
  - Loss function
  - Seed values
- Layout
  - Number of layers
  - Size of each layer
  - Type of layers

```
1  MultiLayerConfiguration conf
2    = new NeuralNetConfiguration.Builder()
3    .seed(seed).list()
4    .layer(new DenseLayer.Builder().nIn(5)
5      .nOut(10).activation(Activation.TANH)
6      .weightInit(WeightInit.ZERO).build())
7    .layer(new DenseLayer.Builder().nIn(10).nOut(3)
8      .activation(Activation.SIGMOID)
9      .weightInit(WeightInit.UNIFORM).build())
10   .layer(new OutputLayer.Builder(
11     LossFunctions.LossFunction.HINGE)
12     .activation(Activation.SOFTMAX).nIn(3)
13     .nOut(outputNum).build())
14   .build();
```

Builder design patterns: `https://en.wikipedia.org/wiki/Builder_pattern`

Training Artificial Neural Networks

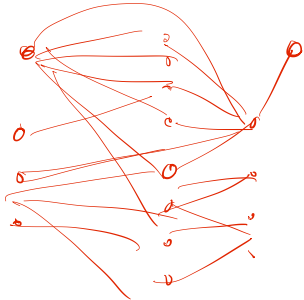
```
1  // Create a network from
2  // a configuration
3  MultiLayerNetwork network
4    = new MultiLayerNetwork(conf);
5
6  // Initialize it
7  network.init();
8
9  // Train one epoch
10 network.fit(dataset);
11
12 // Train n epochs
13 network.fit(dataset, n);
```

Training Artificial Neural Networks

```
1 // Create a network from
2 // a configuration
3 MultiLayerNetwork network
4   = new MultiLayerNetwork(conf);
5
6 // Initialize it
7 network.init();
8
9 // Train one epoch
10 network.fit(dataset);
11
12 // Train n epochs
13 network.fit(dataset, n);
```

- Epoch: One time training on the entire training data set
  - More epochs make the result better, but take more time
- Argument types to `fit(...)`
  - `fit(DataSet)` / `fit(DataSetIterator)`
  - `fit(INDArray, INDArray)`
  - `fit(INDArray, int[])`
  - `fit(MultiDataSet)` / `fit(MultiDataSetIterator iterator)`

Non-Linear Network Architectures



- `MultiLayerNetwork`: Stacked layers
- `ComputationGraph`: Arbitrary directed acyclic graphs
  - Multiple inputs and outputs
  - Complex structures
- Joint learning of multiple tasks

demo

The Problem of Teaching Deep Learning

- Deep learning is best taught using examples
- Examples need to be brief and easily graspable
  - $\rightarrow$ Standard data sets with pre-defined reading routines
- Reality: Your data is always a little different from standard data sets
  - $\Rightarrow$ What is taught is not reflective of reality

A Real Dataset

- Amazon reviews, used for sentiment classification
- It's large: Test data: 400 000 reviews, Training data: 3 600 000
- Preprocessing
  - Establish vocabulary (count words, preserve words with frequency > 15)
  - Lowercase everything
  - Replace punctuation and digits by space
  - Count words in vocabulary, replace string column by NDArray
  - Convert labels to category
  - Convert category to integer

Vectorizer

demo

Section 3

Next Exercise

Exercise 12 – the last one!

`https://github.com/idh-cologne-machine-learning-mit-java/exercise-12`