Getting involved in deep learning with DL4J
AI, Machine & Deep Learning

Artificial Intelligence

Machine Learning

Deep Learning
A paradigm shift

**Algorithmic approach**

**Goal**: apply the algorithm rules to the input

**Deep Learning**

**Goal**: make the input-output association “emerge” as a trained neural network (generalize from data)

In essence a Neural Network is an association machine
Deep Learning Origins

- First steps in the 1940s - McCulloch and Pitts
  "The early model of an artificial neuron is introduced by Warren McCulloch and Walter Pitts in 1943."

  ![Symbolic Illustration of Linear Threshold Gate](image)

- 1958 - Frank Rosenblatt - The Perceptron

- Youtube “History of Neural Networks”
  https://www.youtube.com/watch?v=6fXNiJXUheI
Inputs, Weights & Activation

How a node state depends from its inputs and their weights

Input#1: Active

Input#2: Active

Input#3: Inactive

excitatory connection

inhibitory connection

Activation Function

\[ \Sigma \]

Input States and Connections:

- **Input#1**: Active (1)
- **Input#2**: Active (1)
- **Input#3**: Inactive (0)

Activation Functions:

- **Sigmoid function**
- **RELU function**
Multi Layer Perceptron

Organizing nodes in layers ...
## Typical deep learning tasks

**Fraud detection**

<table>
<thead>
<tr>
<th>Data</th>
<th>Fraud?</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>true</td>
</tr>
</tbody>
</table>

**MNIST**

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Face recognition**

<table>
<thead>
<tr>
<th>Image</th>
<th>is person A?</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>true</td>
</tr>
</tbody>
</table>

**Face classification**

<table>
<thead>
<tr>
<th>Image</th>
<th>who is?</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>A</td>
</tr>
</tbody>
</table>

**Automated Driving Systems**

<table>
<thead>
<tr>
<th>Road Images &amp; sensor inputs</th>
<th>Steering wheel angle?</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0°</td>
</tr>
<tr>
<td>...</td>
<td>-10°</td>
</tr>
<tr>
<td>...</td>
<td>+23°</td>
</tr>
</tbody>
</table>

**Rain mm forecast**

<table>
<thead>
<tr>
<th>sensors</th>
<th>rain mm</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Classification tasks**

**Regression tasks**
A very minimal network & its training

<table>
<thead>
<tr>
<th>I1</th>
<th>I2</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.12</td>
<td>0.87</td>
<td>0.70</td>
</tr>
<tr>
<td>0.45</td>
<td>0.72</td>
<td>0.95</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Training Set

Network to be trained

\[ \text{Network to be trained} \]
Global error for a given weights set

- weights \((w_1, w_2)\) randomly initialized
- \(\forall\) item in the Training Set
  - calc the Actual output
  - calc the distance from the Target output (=error)
- calc the global error at \((w_1, w_2)\): \(e.g. \sum e_i^2\)
  [global=for the entire Training Set]
Error as a function in the weights domain

Error = Cost = Loss Function

error = f (w1, w2, ... )
The goal: finding the minimum cost

http://al-roomi.org/3DPlot/index.html

\[
\frac{(x^3+4*y^2)*\sqrt{\text{abs}(\sin(x^2+y^2)))}}{100}
\]
The goal: finding the minimum cost

<table>
<thead>
<tr>
<th>Ideal Goal</th>
<th>Practical Goal</th>
</tr>
</thead>
</table>
| • find **the global** minimum  
• if it were a mathematical function we could use the derivative  
• the domain is a continuous | • find **a good** minimum  
• we can just move around the weights domain and do samplings for the error  
• steps are discrete (may be too small or too large) |
Adjusting weights for the path to a good minimum

Error (Cost)

where to move from here ??

... and how much to move ??

LR = Learning Rate
Gradient Descent

The **Gradient Descent** algorithm allows finding **local** minima:

1. from a given (random) initial point

2. find the direction with the **steepest gradient descent**

3. make a little step ($\Delta w$) in this direction (Note: it's a N-dimensional space)

4. go to step 2

(like a skier in the fog ... )
Backpropagation algorithm


"Learning representations by back-propagating errors"
Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J.
(8 October 1986 - Nature. 323)

https://www.nature.com/articles/323533a0

https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd
https://www.linkedin.com/pulse/gradient-descent-backpropagation-ken-chen
Neural Networks numbers

A simple neural network for classifying images

- input: images 400x300 pixels
  => 120,000 input nodes

- hidden layer#1: 50,000 nodes
  => 120,000 x 50,000
  => 6,000,000,000 connections (weights)
  just between input and first hidden layer!
Neural Networks ages

• bright & dark ages in the Neural Network history (several “winter” seasons)

• training requires:
  ✓ lots of data
  ✓ great elaboration effort
  ✓ huge amount of time

• why so much success now?
  • availability of a great amount of data (Big Data)
  • high elaboration power in modern pc's
Why “Deep” learning?

- each hidden layer can learn different features
Why “Deep” learning?

• e.g. in image classification tasks

- image
- lines, corners, ...
- shapes
- ... complex figures
- output

+ abstraction
Many degrees of freedom

High number of parameters and topology choices

• how many hidden layers?

• which Activation Function?

• Learning Rate value?

• topology:
  • Standard Multilayer Neural Network
  • Convolutional Neural Network (CNN)
  • Recurrent Neural Network (RNN)
  • other ...
Some priority

Input & Output layer topology & mapping have to be designed first!
The MNIST dataset can be considered:
- a sort of “Hello World” in the Deep Learning domain
- a reference example against which to test learning paradigms or techniques
- 70,000 $28 \times 28$ pixels images of handwritten 0-9 digits
MNIST: slicing an image for input

784 input 0|1 nodes
MNIST: output layer

- 784 input nodes
- 10 output nodes (only one on=1)
MNIST, Training & Test Set

- 60,000 in the **Training Set** (used to train the network)
- 10,000 in the **Test Set** (used to measure how the network is really learning)

This set, which is excluded from the training, plays the role of a “third party evaluator”.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

784 input nodes 10 output nodes
MNIST Dataset setup

- download & unzip

http://github.com/myleott/mnist_png/raw/master/mnist_png.tar.gz
Deeplearning4j website

Open-source, distributed, deep learning library for the JVM

Deep Learning for Java
Eclipse New Project

• New Maven Project
DL4J Maven dependencies

```xml
  <modelVersion>4.0.0</modelVersion>
  <groupId>it.rcpvision.dl4j</groupId>
  <artifactId>it.rcpvision.dl4j.workbench</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
    <dependency>
      <groupId>org.nd4j</groupId>
      <artifactId>nd4j-native-platform</artifactId>
      <version>1.0.0-beta5</version>
    </dependency>
    <dependency>
      <groupId>org.deeplearning4j</groupId>
      <artifactId>deeplearning4j-core</artifactId>
      <version>1.0.0-beta5</version>
    </dependency>
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-jdk14</artifactId>
      <version>1.7.26</version>
    </dependency>
  </dependencies>
</project>
```
// The absolute path of the folder containing MNIST training and testing subfolders
private static final String MNIST_DATASET_ROOT_FOLDER = "/home/vincenzo/dl4j/mnist_png/";

// Height and width in pixel of each image
private static final int HEIGHT = 28;
private static final int WIDTH = 28;

// The total number of images into the training and testing set
private static final int N_SAMPLES_TRAINING = 60000;
private static final int N_SAMPLES_TESTING = 10000;

// The number of possible outcomes of the network for each input,
// correspondent to the 0..9 digit classification
private static final int N_OUTCOMES = 10;

private static DataSetIterator getDataSetIterator(String folderPath, int nSamples) throws IOException {
    }
}
private static DataSetIterator getDataSetIterator(String folderPath, int nSamples) throws IOException {
    File folder = new File(folderPath);
    File[] digitFolders = folder.listFiles();

    NativeImageLoader nil = new NativeImageLoader(HEIGHT, WIDTH);
    ImagePreProcessingScaler scaler = new ImagePreProcessingScaler(0, 1);

    INDArray input = Nd4j.create(new int[]{nSamples, HEIGHT * WIDTH});
    INDArray output = Nd4j.create(new int[]{nSamples, N_OUTCOMES});

    int n = 0;
    for (File digitFolder : digitFolders) {
        int labelDigit = Integer.parseInt(digitFolder.getName());
        File[] imageFiles = digitFolder.listFiles();
        for (File imageFile : imageFiles) {
            INDArray img = nil.asRowVector(imageFile);
            scaler.transform(img);
            input.putRow(n, img);
            output.put(n, labelDigit, 1.0);
            n++;
        }
    }

    DataSet dataSet = new DataSet(input, output);
    List<DataSet> listDataSet = dataSet.asList();
    Collections.shuffle(listDataSet, new Random(System.currentTimeMillis()));
    DataSetIterator dsi = new ListDataSetIterator<DataSet>(listDataSet, 10); // batchSize=10
    return dsi;
}
Building the MLP Network

```java
public static void main(String[] args) throws IOException {
    DataSetIterator dsi = getDataSetIterator(MNIST_DATASET_ROOT_FOLDER + "training", N_SAMPLES_TRAINING);
    MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
        .seed(123) // include a random seed for reproducibility
        .updater(new Nesterovs(0.006, 0.9))
        .l2(1e-4)
        .list()
        .layer(new DenseLayer.Builder() // create the first, input layer with xavier initialization
                    .nIn(HEIGHT*WIDTH)
                    .nOut(1000)
                    .activation(Activation.RELU)
                    .weightInit(WeightInit.XAVIER)
                    .build())
        .layer(new OutputLayer.Builder(LossFunction.NEGATIVELOGLIKELIHOOD) // create hidden layer
                    .nIn(1000)
                    .nOut(N_OUTCOMES)
                    .activation(Activation.SOFTMAX)
                    .weightInit(WeightInit.XAVIER)
                    .build())
        .build();
```
Training & testing the Network

```java
MultilayerNetwork model = new MultilayerNetwork(conf);
model.init();
//print the score with every 500 iteration
model.setListeners(new ScoreIterationListener(500));
model.fit(dsi);
```

```java
DataSetIterator testDsi = getDataSetIterator( MNIST_DATASET_ROOT_FOLDER + "testing", N_SAMPLES_TESTING );
Evaluation eval = model.evaluate(testDsi); //org.nd4j.evaluation.classification.Evaluation;
System.out.println(eval);
```

```
==================================================================
# of classes: 10
Accuracy: 0.9689
Precision: 0.9688
Recall: 0.9688
F1 Score: 0.9687
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)
==================================================================
```
Layers are not fully connected:

- A node receives input only from a group of adjacent nodes in previous layer
- The same set of weights ("filter") is applied to all nodes between two consecutive layers
Convolutional Neural Networks

- specific for images

E.g. 3x3 (or 5x5) filter sliding over the image, from top-left to bottom-right
Convolutional Neural Networks

- more filters
- each one detects different feature details
- after each Convolutional layer we put a Pooling (subsampling) layer in order to reduce subsequent dimensions
Building the Convolutional Network

```java
int channels = 1;
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
    .seed(123)
    .l2(0.0005)  // ridge regression value
    .updater(new Nesterovs(0.006, 0.9))
    .weightInit(WeightInit.XAVIER)
    .list()
    .layer(new ConvolutionLayer.Builder(5, 5)
        .nIn(channels )
        .stride(1, 1)
        .nOut(20)
        .activation(Activation.IDENTITY)
        .build())
    .layer(new SubsamplingLayer.Builder(SubsamplingLayer.PoolingType.MAX)
        .kernelSize(2, 2)
        .stride(2, 2)
        .build())
    .layer(new ConvolutionLayer.Builder(5, 5)
        .stride(1, 1) // nIn need not specified in later layers
        .nOut(50)
        .activation(Activation.IDENTITY)
        .build())
    .layer(new SubsamplingLayer.Builder(SubsamplingLayer.PoolingType.MAX)
        .kernelSize(2, 2)
        .stride(2, 2)
        .build())
    .layer(new ConvolutionLayer.Builder(5, 5)
        .nIn(channels )
        .stride(1, 1)
        .nOut(20)
        .activation(Activation.IDENTITY)
        .build())
    .layer(new SubsamplingLayer.Builder(SubsamplingLayer.PoolingType.MAX)
        .kernelSize(2, 2)
        .stride(2, 2)
        .build())
    .list();
```
Building the Convolutional Network

```
.layer(new DenseLayer.Builder().activation(Activation.RELU)
  .nOut(500)
  .build())
.layer(new OutputLayer.Builder(LossFunctions.LossFunction.NEGATIVELOGLIKELIHOOD)
  .nOut(N_OUTCOMES)
  .activation(Activation.SOFTMAX)
  .build())
.setInputType(InputType.convolutionalFlat(HEIGHT, WIDTH, channels)) // InputType.convolutional for normal image
  .build();
```

```
// InputType.convolutional for normal image
Layer#5 - Fully connected layer with 500 nodes
Layer#6 - Output
```

```
// InputType.convolutional for normal image
```

```
# of classes: 10

Accuracy: 0.9839
Precision: 0.9840
Recall: 0.9836
F1 Score: 0.9838
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)

Confusion matrix format: Actual (rowClass) predicted as (columnClass) N times
```

```
# of classes: 10

Accuracy: 0.9839
Precision: 0.9840
Recall: 0.9836
F1 Score: 0.9838
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)

Confusion matrix format: Actual (rowClass) predicted as (columnClass) N times
```
A real example taken from Kaggle

Malaria Cell Images Dataset
Cell Images for Detecting Malaria

Arunava · updated 10 months ago (Version 1)

Data Kernels (199) Discussion (7) Activity Metadata

Download (335 MB) New Notebook

Usability 7.5 Tags healthcare, online image galleries, infectious diseases

Description

Content
The dataset contains 2 folders - Infected - Uninfected
And a total of 27,558 images.

Acknowledgements
This Dataset is taken from the official NIH Website: https://ceb.nlm.nih.gov/repositories/malaria-datasets/ And uploaded here, so
```java
int channels = 3;
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
    .seed(123)
    .l2(0.0005) // ridge regression value
    .updater(new Nesterovs(0.005, 0.9))
    .weightInit(WeightInit.XAVIER)
    .list()
    .layer(new ConvolutionLayer.Builder(5, 5)
        .nIn(channels)
        .stride(1, 1)
        .nOut(40)
        .activation(Activation.RELU)
        .build())
    .layer(new SubsamplingLayer.Builder(SubsamplingLayer.PoolingType.MAX)
        .kernelSize(5, 5)
        .stride(2, 2)
        .build())
    .layer(new ConvolutionLayer.Builder(5, 5)
        .stride(1, 1) // nIn need not specified in later layers
        .nOut(50)
        .activation(Activation.RELU)
        .build())
    .layer(new SubsamplingLayer.Builder(SubsamplingLayer.PoolingType.MAX)
        .kernelSize(5, 5)
        .stride(2, 2)
        .build());

================================================================================
# of classes: 2
Accuracy: 0.9457
```
Resources

**Code in this talk**
https://github.com/vincenzocaselli/dl4j-ece2019

**Deeplearning4j Website**
deeplearning4j.org

**Deeplearning4j GitHub**
https://github.com/eclipse/deeplearning4j

**Deep Learning - A Practitioner's Approach**
By Adam Gibson, Josh Patterson
EVALUATE THE SESSIONS

Sign in and vote using the conference app or eclipsecon.org

-1 0 +1