Deep Learning on Java

Breandan Considine

JFokus 2017
Who am I?

- Background in Computer Science, Machine Learning
- Worked for a small ad-tech startup out of university
- Spent two years as Developer Advocate @JetBrains
- Interested in machine learning and speech recognition
- Enjoy writing code, traveling to conferences, reading
- Say hello! @breandan | breandan.net | bre@ndan.co
What is “three”?
Size
Shape
Distance
Similarity
Separation
Orientation
What is “dog”?
Length

Neck

Chest

Girth

**DOG SIZE**

<table>
<thead>
<tr>
<th>Extra Small</th>
<th>Chihuahua</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Miniature Breed</td>
</tr>
<tr>
<td></td>
<td>Yorkie</td>
</tr>
</tbody>
</table>

| Small             | Cavalier Spaniel |
|                   | Miniature Poodle |
|                   | Dachshund       |
|                   | Jack Russell    |
|                   | Westie         |
|                   | Whippet        |

| Medium            | Beagle        |
|                   | Cocker Spaniel |
|                   | Springer Spaniel |
|                   | Staffordshire Bull Terrier |
|                   | Standard Poodle |

| Large              | Boxer         |
|                    | Doberman      |
|                    | German Shepherd |
|                    | Labrador      |
|                    | Retriever     |
|                    | Setter        |

| Extra Large        | Rottweiler    |
|                    | Wolfhound     |
|                    | Pyrenean      |
|                    | Bloodhound    |
|                    | Great Dane    |
|                    | St. Bernard   |
What is “Swedish”? 
Early Speech Recognition

- Requires lots of handmade feature engineering
- Poor results: >25% WER for HMM architectures
Automatic speech recognition in 2011

* Milestones in phone recognition accuracy using the TIMIT database
Year over year Top-5 Recognition Error
What happened?

• Bigger data
• Faster hardware
• Smarter algorithms
What is machine learning?

• Prediction
• Categorization
• Anomaly detection
• Personalization
• Adaptive control
• Playing games
Traditional education

- One-size-fits-all curriculum
- Teaching process is repetitive
- Students are not fully engaged
- Memorization over understanding
- Encouragement can be inconsistent
- Teaches to the test (not the real world)
How can we improve education?

• Personalized learning
• Teaching assistance
• Adaptive feedback
• Active engagement
• Spaced repetition
• Assistive technology
Google today is announcing the release of version 5.0 of its Google Translate service.
Had Walter Alvarez not asked his father, the Nobel Prize-winning physicist Luis Alvarez, how long the clay had taken to deposit, the younger Alvarez may not have thought to use iridium, an element rarely found on earth but more plentiful in meteorites, to answer this question. Iridium, in the form of microscopic grains of cosmic dust, is constantly raining down on the planet. The Alvarezes reasoned that if the clay layer had taken a significant amount of time to deposit, it would contain detectable levels of iridium.

---

**physicist**

expert in physics. Continue reading.

**iridium**

chemical element, atomic number 77, atomic weight 193.1, a very hard steel-grey metal with a high melting-point, having a number of important uses in industry, for example, in watch-making and pen making. Continue reading.
Lemma 0.1. Let $C$ be a set of the construction.

Let $C$ be a gerber covering. Let $F$ be a quasi-coherent sheaves of $O$-modules. We have to show that

$$O_{\Omega} = O_X(L)$$

Proof. This is an algebraic space with the composition of sheaves $F$ on $X_{\text{etale}}$ we have

$$O_X(F) = \{\text{morph}_{\text{etale}} \times_{O_X} (G, F)\}$$

where $G$ defines an isomorphism $F \to F$ of $O$-modules.

Lemma 0.2. This is an integer $Z$ is injective.

Proof. See Spaces, Lemma ??.

Lemma 0.3. Let $S$ be a scheme. Let $X$ be a scheme and $X$ is an affine open covering. Let $U \subseteq X$ be a canonical and locally of finite type. Let $X$ be a scheme. Let $X$ be a scheme which is equal to the formal complex.

The following to the construction of the lemma follows.

Let $X$ be a scheme. Let $X$ be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$ be a morphism of algebraic spaces over $S$ and $Y$.

Proof. Let $X$ be a nonzero scheme of $X$. Let $X$ be an algebraic space. Let $F$ be a quasi-coherent sheaf of $O_X$-modules. The following are equivalent

1. $F$ is an algebraic space over $S$.
2. If $X$ is an affine open covering.

Consider a common structure on $X$ and $X$ the functor $O_X(U)$ which is locally of finite type.

This since $F \in F$ and $x \in G$ the diagram

$$\begin{array}{c}
S \\
\downarrow \\
\xi \\
\downarrow \\
O_X(Y) \\
\downarrow \\
\alpha
\end{array}$$

is a limit. Then $G$ is a finite type and assume $S$ is a flat and $F$ and $G$ is a finite type $S$. This is of finite type diagrams, and

- the composition of $G$ is a regular sequence,
- $O_X = \text{sheaf of rings}$.

Proof. We have see that $X = \text{Spec}(R)$ and $F$ is a finite type representable by algebraic space. The property $F$ is a finite morphism of algebraic stacks. Then the cohomology of $X$ is an open neighbourhood of $U$.

Proof. This is clear that $G$ is a finite presentation, see Lemmas ??.

A reduced above we conclude that $U$ is an open covering of $C$. The functor $F$ is a

$$\text{field} \quad O_{X/} \to F_P \to (O_{X_{\text{etale}}})_P \to O_{X/}(O_{X/})$$

is an isomorphism of covering of $O_X$. If $F$ is the unique element of $F$ such that $X$ is an isomorphism.

The property $F$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $O_X$-algebra with $F$ are opens of finite type over $S$.

If $F$ is a scheme theoretic image points.

If $F$ is a finite direct sum $O_{X/}$ is a closed immersion, see Lemma ??.

This is a sequence of $F$ is a similar morphism.
COLLABORATIVE FILTERING

Read by both users

Similar users

Read by her, recommended to him!

CONTENT-BASED FILTERING

Read by user

Similar articles

Recommended to user
Rasta Imposta Lightweight Penguin Costume

Price: $21.50 - $99.69 & FREE Returns. Details
Sale: Lower price available on select options

Fit: As expected (80%) ▼

Size:
Select ▼ Size Chart

Color: Black/White

- 100% Polyester
- Imported
- Hand Wash
- Rasta Imposta's lightweight penguin is both affordable and comfortable
- This costume comes with a tunic with an attached head and shoe covers

Customers who viewed this item also bought:

- Rasta Imposta: $9.00 - $49.98
- Forum Novelties: $13.67 - $52.49
- Rasta Imposta: $19.99 - $50.98
- Rasta Imposta: $54.99 - $116.99
Handwriting recognition

\[ f(z) = \frac{1}{2\pi} \int_0^{2\pi} u(e^{i\psi}) \frac{e^{i\psi} + z}{e^{i\psi} - z} \, d\psi, \quad |z| < 1 \]

\[ F(z) = \frac{1}{2\pi} \int_0^{2\pi} u(e^{i\psi}) \frac{e^{i\psi} + z}{e^{i\psi} - z} \, d\psi, \quad |z| < 1 \]
Handwriting recognition
Speech recognition

colorless green ideas sleep furiously
Speech Verification / Recitation
Machine learning, for humans

- Self-improvement
- Language learning
- Computer training
- Special education
- Reading comprehension
- Content generation
What’s a Tensor?

- A “tensor’ is just an n-dimensional array
- Useful for working with complex data
- We use (tiny) tensors every day!
What’s a Tensor?

• A “tensor’ is just an n-dimensional array
• Useful for working with complex data
• We use (tiny) tensors every day!
What’s a Tensor?

• A “tensor’ is just an n-dimensional array
• Useful for working with complex data
• We use (tiny) tensors every day!
What’s a Tensor?

• A “tensor” is just an n-dimensional array
• Useful for working with complex data
• We use (tiny) tensors every day!
What’s a Tensor?

- A “tensor” is just an n-dimensional array
- Useful for working with complex data
- We use (tiny) tensors every day!
What’s a Tensor?

- A “tensor” is just an n-dimensional array
- Useful for working with complex data
- We use (tiny) tensors every day!
$N \times M$ image is a point in $\mathbb{R}^{NM}$
Male-Female

Verb tense

Country-Capital

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Example 1</th>
<th>Example 2</th>
<th>Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>France - Paris</td>
<td>Italy: Rome</td>
<td>Japan: Tokyo</td>
<td>Florida: Tallahassee</td>
</tr>
<tr>
<td>big - bigger</td>
<td>small: larger</td>
<td>cold: colder</td>
<td>quick: quicker</td>
</tr>
<tr>
<td>Miami - Florida</td>
<td>Baltimore: Maryland</td>
<td>Dallas: Texas</td>
<td>Kona: Hawaii</td>
</tr>
<tr>
<td>Einstein - scientist</td>
<td>Messi: midfielder</td>
<td>Mozart: violinist</td>
<td>Picasso: painter</td>
</tr>
<tr>
<td>Sarkozy - France</td>
<td>Berlusconi: Italy</td>
<td>Merkel: Germany</td>
<td>Koizumi: Japan</td>
</tr>
<tr>
<td>copper - Cu</td>
<td>zinc: Zn</td>
<td>gold: Au</td>
<td>uranium: plutonium</td>
</tr>
<tr>
<td>Berlusconi - Silvio</td>
<td>Sarkozy: Nicolas</td>
<td>Putin: Medvedev</td>
<td>Obama: Barack</td>
</tr>
<tr>
<td>Microsoft - Windows</td>
<td>Google: Android</td>
<td>IBM: Linux</td>
<td>Apple: iPhone</td>
</tr>
<tr>
<td>Microsoft - Ballmer</td>
<td>Google: Yahoo</td>
<td>IBM: McNealy</td>
<td>Apple: Jobs</td>
</tr>
<tr>
<td>Japan - sushi</td>
<td>Germany: bratwurst</td>
<td>France: tapas</td>
<td>USA: pizza</td>
</tr>
</tbody>
</table>

Types of machine learning

Machine Learning

- Supervised
- Unsupervised
- Reinforcement
Supervised Learning

Regression

Classification
Supervised Learning

Classification
\[ y = mx + b \]
$$z = mx + ny + b$$
Cool learning algorithm

def classify(datapoint, weights):
Cool learning algorithm

def classify(datapoint, weights):
    prediction = sum(x * y for x, y in zip([1] + datapoint, weights))
def classify(datapoint, weights):
    prediction = sum(x * y for x, y in zip([1] + datapoint, weights))
    if prediction < 0:
        return 0
    else:
        return 1
inputs \( x_1, x_2, x_3, \ldots, x_n \) with weights \( w_{1j}, w_{2j}, w_{3j}, \ldots, w_{nj} \) are summed to form the net input \( net_j \). This net input is then passed through an activation function \( \varphi \) to produce the output \( o_j \). The threshold \( \theta_j \) is also indicated.
Cool learning algorithm

def classify(datapoint, weights):
  prediction = sum(x * y for x, y in zip([1] + datapoint, weights))
  if prediction < 0:
    return 0
  else:
    return 1
Cool learning algorithm

def train(data_set):
Cool learning algorithm

def train(data_set):

class Datum:
    def init(self, features, label):
        self.features = [1] + features
        self.label = label
Cool learning algorithm

def train(data_set):
    weights = [0] * len(data_set[0].features)

[0, 0, 0]
def train(data_set):
    weights = [0] * len(data_set[0].features)
    total_error = threshold + 1
def train(data_set):
    weights = [0] * len(data_set[0].features)
    total_error = threshold + 1

    while total_error > threshold:
        total_error = 0
        for item in data_set:
            error = item.label - classify(item.features, weights)
            weights = [w + RATE * error * i
                        for w, i in zip(weights, item.features)]
            total_error += abs(error)
Cool learning algorithm

def train(data_set):
    weights = [0] * len(data_set[0].features)
    total_error = threshold + 1
    while total_error > threshold:
        total_error = 0
        for item in data_set:
            error = item.label - classify(item.features, weights)
            weights = [w + RATE * error * i for w, i in zip(weights, item.features)]
            total_error += abs(error)
Cool learning algorithm

def train(data_set):
    weights = [0] * len(data_set[0].features)
    total_error = threshold + 1
    while total_error > threshold:
        total_error = 0
        for item in data_set:
            error = item.label - classify(item.features, weights)
            weights = [w + RATE * error * i
                        for w, i in zip(weights, item.features)]
            total_error += abs(error)
Cool learning algorithm

weights = [w + RATE * error * i
for w, i in zip(weights, item.features)]
Cool learning algorithm

```python
def train(data_set):
    weights = [0] * len(data_set[0].features)
    total_error = threshold + 1
    while total_error > threshold:
        total_error = 0
        for item in data_set:
            error = item.label - classify(item.features, weights)
            weights = [w + RATE * error * i
                        for w, i in zip(weights, item.features)]
        total_error += abs(error)
```
def train(data_set):
    weights = [0] * len(data_set[0].features)
    total_error = threshold + 1
    while total_error > threshold:
        total_error = 0
        for item in data_set:
            error = item.label - classify(item.features, weights)
            weights = [w + RATE * error * i
                        for w, i in zip(weights, item.features)]
        total_error += abs(error)
<table>
<thead>
<tr>
<th>Structure</th>
<th>Regions</th>
<th>XOR</th>
<th>Meshed regions</th>
</tr>
</thead>
<tbody>
<tr>
<td>single layer</td>
<td>Half plane bounded by hyper-plane</td>
<td><img src="image" alt="XOR single layer" /></td>
<td><img src="image" alt="Meshed regions single layer" /></td>
</tr>
<tr>
<td>two layer</td>
<td>Convex open or closed regions</td>
<td><img src="image" alt="XOR two layer" /></td>
<td><img src="image" alt="Meshed regions two layer" /></td>
</tr>
<tr>
<td>three layer</td>
<td>Arbitrary (limited by # of nodes)</td>
<td><img src="image" alt="XOR three layer" /></td>
<td><img src="image" alt="Meshed regions three layer" /></td>
</tr>
</tbody>
</table>
Backpropogation

\[ \text{train(trainingSet)} : \]

initialize network weights randomly
until average error stops decreasing (or you get tired):
  for each \textbf{sample} in trainingSet:
    prediction = network.output(sample)
    compute error (prediction - sample.output)
    compute error of (hidden -> output) layer weights
    compute error of (input -> hidden) layer weights
    update weights across the network
  save the weights
Gradient Descent

http://cs231n.github.io/
“Deep” neural networks
Artificial Neural Networks: How can I estimate the number of neurons and layers?

Yoshua Bengio, Head of Montreal Institute for Learning Algorithms, Professor @ U. Montreal

14.2k Views • Upvoted by Zeeshan Zia, PhD in Computer Vision, CV/ML researcher in Silicon Valley and Hadayat Seddiqi, engineering @ biotech startup

Most Viewed Writer in Artificial Neural Networks (ANNs) with 30+ answers

Very simple. Just keep adding layers until the test error does not improve anymore.

As for number of units, it is a hyper-parameter to be optimized, as usual. See my paper on guidelines for setting up hyper-parameters in deep networks, *Practical recommendations for gradient-based training of deep architectures*. 

Written May 7, 2013 • View Upvotes

Upvoted 192 Downvote Comments 3+
ImageNet LSVR Competition
What is a kernel?

• A kernel is just a matrix
• Used for edge detection, blurs, filters
Image

Convolved Feature

```
  1  1  1  1  0  0
  0  1  1  1  1  0
  0  0  1  1  1  1
  0  0  1  1  1  0
  0  0  1  1  0  0
```

4
Pooling (Downsampling)

Single depth slice

```
<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
```

max pool with $2 \times 2$ filters and stride 2

```
<table>
<thead>
<tr>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
```
Low level features
(a) Standard Neural Net

(b) After applying dropout.
Convolutional neural network
Google Inception Model
Google Inception Model
Google Inception Model
Open-source, distributed, deep-learning library for the JVM

What is Deeplearning4j?

Deeplearning4j is the first commercial-grade, open-source, distributed deep-learning library written for Java and Scala. Integrated with Hadoop and Spark, DL4J is designed to be used in business environments on distributed GPUs and CPUs. Skymind is its commercial support arm.
DEEPLEARNING4J: Open-source distributed DL for the JVM

ND4J: Scientific computing for Java
(Our linear algebra engine)

Hadoop
Spark
Mesos

DATA
CANOVA: “Rosetta Stone” of vectorization

GPUs
Swappable
& Parallel
Native

Predictions & classifications
Network Configuration

MultiLayerConfiguration mlc =
    new NeuralNetConfiguration.Builder()
    .seed(12345)
    .optimizationAlgo(STOCHASTIC_GRADIENT_DESCENT)
    .iterations(1)
    .learningRate(0.006)
    .updater(NESTEROVS)
    .momentum(0.9)
    .regularization(true)
    .l2(1e-4)
    .list()
Network Configuration

...layer(0, new DenseLayer.Builder()
  .nIn(28 * 28) // Number of input datapoints.
  .nOut(1000) // Number of output datapoints.
  .activation(Activation.RELU).weightInit(XAVIER)
  .build())
.layer(1, new OutputLayer.Builder(NEGATIVELOGLIKELIHOOD)
  .nIn(1000).nOut(10)
  .activation(SOFTMAX).weightInit(XAVIER).build())
.pretrain(false)
.backprop(true)
.build();
Model Initialization

MultiLayerNetwork mlpNet = new MultiLayerNetwork(conf);
mlpNet.init();
Training the model

DataSetIterator dataSetIterator = ...

for(int i=0; i < numEpochs; i++) {
    model.fit(dataSetIterator);
}
Evaluation

evaluator = new Evaluation(outputNum);
while (testSetIterator.hasNext()) {
    DataSet next = dataSetIterator.next();
    INDArray guesses =
        model.output(next.getFeatureMatrix(), false);
    INDArray realOutcomes = next.getLabels();
    evalaluator.eval(eval, output);
}

log.info(eval.stats());
A mostly complete chart of Neural Networks

©2016 Fjdor van 'Veen - asimovinstitute.org

Data Science/Engineering

- Data selection
- Data processing
  - Formatting & Cleaning
  - Sampling
- Data transformation
  - Feature scaling & Normalization
  - Decomposition & Aggregation
  - Dimensionality reduction
Original dataset

Training set

Validation set

Test set

Train the models

Select the best model

Test the model

New Data

Model in production
Fig. 1 Graphical illustration of bias and variance.
Common Mistakes

• Training set – 70%/30% split
• Test set – Do not show this to your model!
• Sensitivity vs. specificity
• Overfitting
Training your own model

• Requirements
  • Clean, labeled data set
  • Clear decision problem
  • Patience and/or GPUs

• Before you start
Preparing data for ML

• Generating Labels
• Dimensionality reduction
• Determining salient features
• Visualizing the shape of your data
• Correcting statistical bias
• Getting data in the right format
Further resources

• [CS231 Course Notes](#)
• [Deeplearning4j Examples](#)
• [Visualizing MNIST](#)
• [Neural Networks and Deep Learning](#)
• [Andrew Ng’s Machine Learning class](#)
• [Awesome Public Datasets](#)
Thank You!

Mary, Mark, Margaret, Hanneli