Deep Learning: a gentle introduction

Jamal Atif
jamal.atif@dauphine.fr

PSL, Université Paris-Dauphine, LAMSADE

February 8, 2016
Scientists See Promise in Deep-Learning Programs

By JOHN MARKOFF  NOV. 23, 2012

A voice recognition program translated a speech given by Richard F. Rashid, Microsoft's top scientist, into Mandarin Chinese. Hao Zhang/The New York Times
Comment le « deep learning » révolutionne l'intelligence artificielle

Cette technologie d'apprentissage, basée sur des réseaux de neurones artificiels, a complètement bouleversé le domaine de l'intelligence artificielle en moins de cinq ans.

« J'ai n'ai jamais vu une révolution aussi rapide. On est passé d'un système un peu obscur à un système utilisé par des millions de personnes en seulement deux ans. » Yann LeCun, un des pionniers du « deep learning », n'en revient toujours pas. Après une longue traversée du désert,
Why a talk about deep learning?

Convolutional Networks (Yann Le Cun): challenge ILSVRC : 1000 categories and 1,461,406 images.
Neuron’s basic anatomy

Figure: A neuron’s basic anatomy consists of four parts: a **soma** (cell body), **dendrites**, an **axon**, and **nerve terminals**. Information is received by dendrites, gets collected in the cell body, and flows down the axon.
Artificial neuron

\[ \sum_{i=1}^{n} w_i x_i + b = 1 \]

Pre-activation

Activation

Dendrites

Cell body

Axon

Jamal Atif (Université Paris-Dauphine)
Perceptron

Rosenblatt 1957
Perceptron learning

Input: a sample $S = \{(x^{(1)}, y^{1}), \cdots, (x^{(n)}, y^{n})\}$

- Initialize a parameter $t$ to 0
- Initialize the weights $w_i$ with random values.
- Repeat
  - Pick an example $x^{(k)} = [x_1^{(k)}, \cdots, x_d^{(k)}]^T$ from $S$
  - Let $y^{(k)}$ be the target value and $\tilde{y}^{(k)}$ the computed value using the current perceptron
  - If $\tilde{y}^{(k)} \neq y^{(k)}$ then
    - Update the weights: $w_i(t + 1) = w_i(t) + \Delta w_i(t)$ where $\Delta w_i(t) = (y^{(k)} - \tilde{y}^{(k)})x_i^{(k)}$
  - End If
  - $t = t + 1$
- Until all the examples in $S$ were visited and no change occurs in the weights

Output: a perceptron for linear discrimination of $S$
Perceptron learning
Example: learning the OR function

Initialization: \( w_1(0) = w_2(0) = 1, w_3(0) = -1 \)

<table>
<thead>
<tr>
<th>( t )</th>
<th>( w_1(t) )</th>
<th>( w_2(t) )</th>
<th>( w_3(t) )</th>
<th>( x^{(k)} )</th>
<th>( \sum w_i x_i^k )</th>
<th>( \tilde{y}^{(k)} )</th>
<th>( y^{(k)} )</th>
<th>( \Delta w_1(t) )</th>
<th>( \Delta w_2(t) )</th>
<th>( \Delta w_3(t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>001</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>011</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>101</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>111</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>011</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
\( x^{(1)} = [0, 0]^T \quad t = 0 \)

\[ x = \sum_i w_i x_i \]

\[ \tilde{y} = 0 \quad y = 0 \]

\[ w_i = w_i + (y - \tilde{y}) x_i \]

\[ w_1 = 1 + 0 \times 0 \]
\[ w_2 = 1 + 0 \times 0 \]
\[ w_3 = -1 + 0 \times -1 \]
Perceptron illustration

\[ x^{(2)} = [0, 1]^T \]

\[ w_1 = 1 \]
\[ w_2 = -1 \]
\[ w_3 = 1 \]

\[ a = \sum_i w_i x_i \]

1 if \( a > 0 \)
0 elsewhere

\[ \tilde{y} = 0 \]
\[ y = 1 \]

\[ w_i = w_i + (y - \tilde{y}) x_i \]

\[ w_1 = 1 + 1 * 0 = 1 \]
\[ w_2 = 1 + 1 * 1 = 2 \]
\[ w_3 = -1 + 1 * 1 = 0 \]
\[ x^{(3)} = [1, 0]^T \quad t = 2 \]

\[ w_1 = 1 \]
\[ w_2 = 2 \]
\[ w_3 = 0 \]

\[ a = \sum_i w_ix_i \]

1 if \( a > 0 \)
0 elsewhere

\[ \tilde{y} = 1 \]
\[ y = 1 \]

\[ w_i = w_i + (y - \tilde{y})x_i \]

\[ w_1 = 1 + 0 \times 0 = 1 \]
\[ w_2 = 2 + 0 \times 1 = 2 \]
\[ w_3 = 0 + 0 \times 1 = 0 \]
Perceptron illustration

\[ x^{(4)} = [1, 1]^T \quad t = 3 \]

\[ a = \sum_i w_i x_i \]

\[ \tilde{y} = 1 \quad y = 1 \]

\[ w_i = w_i + (y - \tilde{y})x_i \]

\[ w_1 = 1 + 0 \times 0 = 1 \]
\[ w_2 = 2 + 0 \times 1 = 2 \]
\[ w_3 = 0 + 0 \times -1 = 0 \]
Perceptron illustration

\[ x^{(1)} = [0, 0]^T \]

\[ t = 4 \]

\[ a = \sum_i w_i x_i \]

\[ \tilde{y} = 0 \]

\[ y = 0 \]

\[ w_i = w_i + (y - \tilde{y}) x_i \]

\[ w_1 = 1 \]
\[ w_2 = 2 \]
\[ w_3 = 0 \]
Perceptron illustration

\[ x^{(2)} = [0, 1]^T \]

\[ t = 5 \]

\[ w_1 = 1 \]

\[ w_2 = 2 \]

\[ w_3 = 0 \]

\[ a = \sum_i w_i x_i \]

\[ \tilde{y} = 1 \]

\[ y = 1 \]

\[ w_i = w_i + (y - \tilde{y}) x_i \]

\[ w_1 = 1 + 0 \times 0 = 1 \]

\[ w_2 = 2 + 0 \times 1 = 2 \]

\[ w_3 = 0 + 0 \times 1 = 0 \]
Perceptron capacity

OR($x_1, x_2$) AND($x_1, x_2$)
Perceptron autumn

\[
\text{XOR}(x_1, x_2)
\]
Link with Logistic regression

\[ \sum_{i=1}^{d} w_i x_i + b \]

\[ h(x) = \frac{1}{1 + e^{-a(x)}} \]

Stochastic gradient update rule:

\[ w_j = w_j + \lambda (y(i) - h(x(i)))x^{(i)}_j \]

The same as the perceptron update rule!
But!

\[ \text{XOR}(x_1, x_2) \]

\[ \text{AND}(\overline{x}_1, x_2) \]

\[ \text{AND}(x_1, \overline{x}_2) \]
Multilayer Perceptron

Paul Werbos, 84. David Rumelhart, 86
1 Layer Perceptron

\[ f(x) = o(b^2 + \sum_{i=1}^{d} w_i^2 h_i(x)) \]

Hidden Layer
\[ h_j(x) = g(b + \sum_{i=1}^{n} w_{ij} x_i) \]

Output Layer
\[ f(x) = o(b^2 + \sum_{i=1}^{d} w_i^2 h_i(x)) \]
MLP training: backpropagation

XOR example

\[ h(x) = \frac{1}{1 + e^{-a(x)}} \]

\[ a(x) = \sum_k w_k h_k + b^2 \]

\[ \tilde{y} = \frac{1}{1 + e^{-a(x)}} \]
MLP training: backpropagation

XOR example: initialisation

\[ a_1 = \sum_i w^1_{i1} x_i + b^1_1 \]

\[ h(x) = \frac{1}{1 + e^{-a_1}} h_1 \]

\[ a_2 = \sum_i w^1_{i2} x_i + b^1_2 \]

\[ h(x) = \frac{1}{1 + e^{-a_2}} h_2 \]

\[ a_o = \sum_k w^2_k h_k + b^2 \]

\[ h_o(x) = \frac{1}{1 + e^{-a_o}} \]

\[ \tilde{y} \]
MLP training: backpropagation

XOR example: Feed-forward

\[ a_1 = \sum_i w^1_{1i} x_i + b^1_1 \]

\[ h_1(x) = \frac{1}{1+e^{-a_1(x)}} \]

\[ a^2 = 1 \]

\[ y = 0 \]

\[ a_0 = \sum_k w^2_k h_k + b^2 \]

\[ h_0(x) = \frac{1}{1+e^{-a_0(x)}} \]

\[ y = 0 \]

\[ a_1 = \sum_i w^1_{1i} x_i + b^1_1 \]

\[ h_1(x) = \frac{1}{1+e^{-a_1(x)}} \]

\[ a^2 = 1 \]

\[ y = 0 \]

\[ a_0 = \sum_k w^2_k h_k + b^2 \]

\[ h_0(x) = \frac{1}{1+e^{-a_0(x)}} \]

\[ y = 0 \]
MLP training: backpropagation

XOR example: Backpropagation

\[ w_{11} = 1 \]
\[ w_{121} = 1 \]
\[ w_{22} = 1 \]
\[ b_1 = -1 \]
\[ b_2 = -1 \]

\[ a_1 = \sum_i w_{1i} x_i + b_1 \]
\[ h_1(x) = \frac{1}{1 + e^{-a_1(x)}} \]
\[ a_2 = \sum_i w_{2i} x_i + b_2 \]
\[ h_2(x) = \frac{1}{1 + e^{-a_2(x)}} \]

\[ \delta_k = (y - \tilde{y})(1 - \tilde{y}) \tilde{y} h_k \]
\[ a_1 = -0.46 \]
\[ a_2 = -0.46 \]
\[ \tilde{y} = 0.39 \]

\[ y = 0 \]

\[ \delta_k = \frac{\partial E}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial a_o} \frac{\partial a_o}{\partial w^2_k} \]

\[ w^2_k = w^2_k + \eta \delta_k \]
MLP training: backpropagation

XOR example: Backpropagation

\[
\begin{align*}
\delta_{ij} &= -(y - \tilde{y})(1 - \tilde{y})\tilde{y}w^2_h h_j (1 - h_j)x_i \\
\delta_k &= \frac{\partial E}{\partial w_k}, E_w = \frac{1}{2}(y - \tilde{y})^2 \\
&= \frac{\partial E_w}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial a_o} \frac{\partial a_o}{\partial h_k} \\
\delta_k &= -(y - \tilde{y})(1 - \tilde{y})\tilde{y}h_k
\end{align*}
\]

\[
\begin{align*}
w^1_{ij} &= w^1_{ij} - \eta \delta_{ij} \\
\delta_{ij} &= \frac{\partial E_w}{\partial w_{ij}} = \frac{\partial E_w}{\partial h_h} \frac{\partial h_h}{\partial a_o} \frac{\partial a_o}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial h_h} \frac{\partial h_h}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial w_{ij}}
\end{align*}
\]
MLP training: backpropagation

XOR example: Feed-forward

\[
\begin{align*}
  w_{11}^1 &= 1 \\
  w_{12}^1 &= 1 \\
  w_{21}^1 &= 1 \\
  w_{22}^1 &= 1 \\
  w_{11}^2 &= 1 \\
  w_{12}^2 &= 1 \\
  w_{21}^2 &= 1 \\
  w_{22}^2 &= 1 \\
  b_1^1 &= -0.98 \\
  b_1^2 &= -1.07 \\
  b_2^1 &= -0.98 \\
  b_2^2 &= -0.09 \\
  a_1 &= \sum_i w_{1i}^1 x_i + b_1^1 \\
  h_1(x) &= \frac{1}{1 + e^{-a_1(x)}} \\
  a_2 &= \sum_i w_{2i}^2 x_i + b_2^1 \\
  h_2(x) &= \frac{1}{1 + e^{-a_2(x)}} \\
  a_o &= \sum_k w_{k}^2 h_k + b_2^2 \\
  h_0(x) &= \frac{1}{1 + e^{-a_o(x)}} \\
  \tilde{y} &\approx 0.5 \\
  y &\approx 1 \\
\end{align*}
\]
Multilayer Perceptron as a deep NN

MLP with more than 2/3 layers is a deep network
Training issues

Setting the hyperparameters

- Initialization
- Number of iterations
- Learning rate $\eta$
- Activation function
- Early stopping criterion

Overfitting/Underfitting

- Number of hidden layers
- Number of neurons

Optimization

Vanishing gradient problem
Overfitting/Underfitting

Source: Bishop, PRML

Source: Bishop, PRML
What’s new!

Before 2006, training deep architectures was unsuccessful!

Underfitting
New optimization techniques

Overfitting
Bengio, Hinton, LeCun
  - Unsupervised pre-training
  - Stochastic “dropout" training
Unsupervised pre-training

Main idea
Initialize the network in an unsupervised way.

Consequences
- Network layers encode successive invariant features (latent structure of the data)
- Better optimization and hence better generalization
Unsupervised pre-training

How to?
Proceed greedily, layer by layer.

Unsupervised NN learning techniques
- Restricted Boltzmann Machines (RBM)
- Auto-encoders
- and many many variants since 2006
Unsupervised pre-training

Autoencoder

Encoder

\[ h(x) = g(a(x)) = \text{sigmoid}(b + Wx) \]

Decoder

\[ \hat{x} = o(\hat{a}(x)) = \text{sigmoid}(c + W^*h(x)) \] for binary inputs

Credit: Hugo Larochelle
Unsupervised pre-training
Sparse Autoencoders

Credit: Y. Bengio
Fine tuning

How to?

▶ Add the output layer
▶ Initialize its weights with random values
▶ initialize the hidden layers with the values from the pre-training
▶ Update the layers by Backpropagation
Drop out

Intuition
Regularize the network by **dropping out** stochastically some hidden units.

Procedure
Assign to each hidden unit a value 0 with probability \( p \) (common choice: .5)
Some applications
Computer vision
Convolutional Neural Networks

State of the art in digit recognition
Modern CNN


- 7 hidden layers, 650,000 units, 60,000,000 parameters
- Drop out
- $10^6$ images
- GPU implementation
- Activation function: $f(x) = max(0, x)$ (ReLu)
Computer vision
Convolutional Neural Networks

LeCun, Hinton, Bengio, Nature 2015
What is deep in deep learning!
Old vs New ML paradigms

- Hand-crafted features
- Learned features
- Raw data
- Classifier
- Decision
Softwares

▶ TensorFlow (google), https://www.tensorflow.org/
▶ Senna: NLP
  ▶ State-of-the-art performance on many tasks
  ▶ 3500 lines of C, extremely fast and using very little memory.
▶ Torch ML Library (C+Lua) : http://www.torch.ch/
▶ Recurrent Neural Network Language Model http://www.fit.vutbr.cz/imikolov/rnnlm
▶ Recursive Neural Net and RAE models for paraphrase detection, sentiment analysis, relation classification www.socher.org