Neural Networks - Deep Learning

Artificial Intelligence © Allegheny College

Janyl Jumadinova

November 13-15, 2018

Credit: Google Workshop
Structure of a prototypical biological neuron

- cell core
- axon
- myelin sheath
- cell body (soma)
- terminal button
- synapse
- dendrites
Neural computing requires a number of **neurons**, to be connected together into a **neural network**.

Neurons are arranged in layers.

\[ a = f(p_1w_1 + p_2w_2 + p_3w_3 + b) = f\left(\sum p_iw_i + b\right) \]
Activation Functions

- The activation function is generally non-linear.
- Linear functions are limited because the output is simply proportional to the input.

McCulloch and Pitts: every Boolean function can be implemented
Activation Functions

Linear Transfer Function

Symmetric Hard Limit Trans. Funct.

Satlin Transfer Function

Tan-Sigmoid Transfer Function

Log-Sigmoid Transfer Function

Radial Basis Function
Network structures

Feed-forward networks:
- Single-layer perceptrons
- Multi-layer perceptrons
Feed-forward example

Feed-forward network = a parameterized family of nonlinear functions:

\[ a_5 = g(W_{3,5} \cdot a_3 + W_{4,5} \cdot a_4) \]
\[ = g(W_{3,5} \cdot g(W_{1,3} \cdot a_1 + W_{2,3} \cdot a_2) + W_{4,5} \cdot g(W_{1,4} \cdot a_1 + W_{2,4} \cdot a_2)) \]
Single-layer Perceptrons

Output units all operate separately – no shared weights.

Adjusting weights moves the location, orientation, and steepness of cliff.
Multi-layer Perceptrons

- Layers are usually fully connected.
- Numbers of hidden units typically chosen by hand.
Neural Networks

Input $\rightarrow$ ["this", "movie", "was", "great"]

Hidden $\rightarrow$

Output (score) $\rightarrow$ [.7]
Neural Networks

pixels()

Input  Hidden  Output (label)

["cat"]
A fully connected NN layer

\[
\begin{align*}
y_1 &= \sigma \left( W_{1,1} x_1 + W_{1,2} x_2 + W_{1,3} x_3 + b_1 \right) \\
y_2 &= \sigma \left( W_{2,1} x_1 + W_{2,2} x_2 + W_{2,3} x_3 + b_2 \right) \\
y_3 &= \sigma \left( W_{3,1} x_1 + W_{3,2} x_2 + W_{3,3} x_3 + b_3 \right)
\end{align*}
\]
Implementation as Matrix Multiplication

\[
\begin{align*}
y_1 &= \sigma \left( W_{1,1} x_1 + W_{1,2} x_2 + W_{1,3} x_3 \right) + b_1 \\
y_2 &= \sigma \left( W_{2,1} x_1 + W_{2,2} x_2 + W_{2,3} x_3 \right) + b_2 \\
y_3 &= \sigma \left( W_{3,1} x_1 + W_{3,2} x_2 + W_{3,3} x_3 \right) + b_3
\end{align*}
\]

\[
\begin{bmatrix}
y_1 \\
y_2 \\
y_3
\end{bmatrix} = \sigma \left( \begin{bmatrix}
W_{1,1} & W_{1,2} & W_{1,3} \\
W_{2,1} & W_{2,2} & W_{2,3} \\
W_{3,1} & W_{3,2} & W_{3,3}
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix} + \begin{bmatrix}
b_1 \\
b_2 \\
b_3
\end{bmatrix} \right)
\]
Non-Linear Data Distributions

The Problem

Linear Model

Neural Network
Is this a Cat or Dog?

Deep Neural Network

Input Layer

Activated Neurons

Output Layer
Most current machine learning works well because of human-designed representations and input features.

Machine learning becomes just optimizing weights to best make a final prediction.
Deep Learning

- Most current machine learning works well because of human-designed representations and input features.
- Machine learning becomes just optimizing weights to best make a final prediction.
- **Deep learning** algorithms attempt to learn multiple levels of representation of increasing complexity/abstraction.
The Need for Distributed Representations

Current NLP systems are incredibly fragile because of their atomic symbol representations.

Crazy sentential complement, such as for “likes [(being) crazy]”
Each neuron implements a relatively simple mathematical function.

\[ y = g(\bar{w} \cdot \bar{x} + b) \]
Each neuron implements a relatively simple mathematical function.

\[ y = g(\mathbf{w} \cdot \mathbf{x} + b) \]

The composition of $10^6 - 10^9$ such functions is powerful.
Deep Learning

Book: http://www.deeplearningbook.org/

Chapter 5

“A core idea in deep learning is that we assume that the data was generated by the composition of factors or features, potentially at multiple levels in a hierarchy.”
Results get better with:

- more data
- bigger models
- more computation
Results get better with:

- more data
- bigger models
- more computation

Better algorithms, new insights and improved methods help, too!
 TensorFlow

- Open source Machine Learning library
- Especially useful for **Deep Learning**
- For research and production
- Apache 2.0 license
- tensorflow.org
Adoption of Deep Learning Tools on GitHub

- TensorFlow
  - GitHub Stars: 25041
  - GitHub Forks: 9607
  - GitHub Launch: Nov. 2015

- Caffe
  - GitHub Stars: 10474
  - GitHub Forks: 6227
  - GitHub Launch: Sep. 2013

- Torch
  - GitHub Stars: 4671
  - GitHub Forks: 1288
  - GitHub Launch: Jan. 2012

- Theano
  - GitHub Stars: 3829
  - GitHub Forks: 1399
  - GitHub Launch: Jan. 2008
**Epoch**: a training iteration (one pass through the dataset).

**Batch**: Portion of the dataset (number of samples after dataset has been divided).

**Regularization**: a set of techniques that helps learning models to converge (http://www.godeep.ml/regularization-using-tensorflow/).
playground.tensorflow.org
TensorFlow

- Operates over **tensors**: n-dimensional arrays
TensorFlow

- Operates over **tensors**: n-dimensional arrays
- Using a **flow graph**: data flow computation framework
TensorFlow

- Operates over **tensors**: n-dimensional arrays
- Using a **flow graph**: data flow computation framework

A multidimensional array.

A graph of operations.
5.7 ← Scalar

- Number, Float, etc.
TensorFlow

\[(x_1, x_2, \ldots, x_n)\]

\[\{ (x_1, x_2, \ldots, x_n) \}

\[
\begin{array}{cccc}
X_{11} & X_{12} & \cdots & X_{1n} \\
X_{m1} & X_{m2} & \cdots & X_{mn} \\
\end{array}
\]
Tensors have a **Shape** that is described with a vector.
TensorFlow

- Tensors have a **Shape** that is described with a vector
  - \([1000, 256, 256, 3]\)
- 10000 Images
- Each Image has 256 Rows
- Each Row has 256 Pixels
- Each Pixel has 3 values (RGB)
Computation is a dataflow graph

Graph of $\textit{Nodes}$, also called $\textit{Operations}$ or $\textit{ops.}$.
Computation is a dataflow graph with tensors

Edges are N-dimensional arrays: Tensors
Computation is a dataflow graph with state

'Biases' is a variable

Some ops compute gradients

-= updates biases
Core TensorFlow data structures and concepts

- **Graph**: A TensorFlow computation, represented as a dataflow graph:
  - collection of ops that may be executed together as a group.
Core TensorFlow data structures and concepts

- **Graph**: A TensorFlow computation, represented as a dataflow graph:
  - collection of ops that may be executed together as a group.
- **Operation**: a graph node that performs computation on tensors
Core TensorFlow data structures and concepts

- **Graph**: A TensorFlow computation, represented as a dataflow graph:
  - collection of ops that may be executed together as a group.
- **Operation**: a graph node that performs computation on tensors
- **Tensor**: a handle to one of the outputs of an Operation:
  - provides a means of computing the value in a TensorFlow Session.
**TensorFlow**

- **Constants**

Placeholders: must be fed with data on execution.

Variables: a modifiable tensor that lives in TensorFlow's graph of interacting operations.

Session: encapsulates the environment in which Operation objects are executed, and Tensor objects are evaluated.
- **Constants**
- **Placeholders**: must be fed with data on execution.
- **Constants**
- **Placeholders**: must be fed with data on execution.
- **Variables**: a modifiable tensor that lives in TensorFlow’s graph of interacting operations.
TensorFlow

- **Constants**
- **placeholders**: must be fed with data on execution.
- **Variables**: a modifiable tensor that lives in TensorFlow’s graph of interacting operations.
- **Session**: encapsulates the environment in which Operation objects are executed, and Tensor objects are evaluated.
### TensorFlow

<table>
<thead>
<tr>
<th>Category</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element-wise math ops</td>
<td>Add, Sub, <strong>Mul</strong>, Div, Exp, Log, Greater, Less...</td>
</tr>
<tr>
<td>Matrix ops</td>
<td><strong>Concat</strong>, Slice, <strong>Split</strong>, Constant, Rank, <strong>Shape</strong>...</td>
</tr>
<tr>
<td>Matrix ops</td>
<td><strong>MatMul</strong>, MatrixInverse, MatrixDeterminant...</td>
</tr>
<tr>
<td>Stateful ops</td>
<td><strong>Variable</strong>, Assign, AssignAdd...</td>
</tr>
<tr>
<td>NN building blocks</td>
<td><strong>SoftMax</strong>, Sigmoid, <strong>ReLU</strong>, <strong>Convolution2D</strong>...</td>
</tr>
<tr>
<td>Checkpointing ops</td>
<td>Save, Restore</td>
</tr>
<tr>
<td>Queue &amp; synch ops</td>
<td>Enqueue, Dequeue, MutexAcquire...</td>
</tr>
<tr>
<td>Control flow ops</td>
<td>Merge, Switch, Enter, Leave...</td>
</tr>
</tbody>
</table>