Neural Networks - Deep Learning

Artificial Intelligence © Allegheny College

Janyl Jumadinova

March 4-6, 2020

Credit: Google Workshop
Structure of a prototypical biological neuron

- cell core
- axon
- myelin sheath
- cell body (soma)
- terminal button
- synapsis
- dendrites
Neural Networks

Neural computing requires a number of neurons, to be connected together into a neural network.

Neurons are arranged in layers.

\[ a = f(p_1 w_1 + p_2 w_2 + p_3 w_3 + b) = f(\sum p_i w_i + b) \]
Neural Networks

Neural computing requires a number of **neurons**, to be connected together into a **neural network**.

Neurons are arranged in layers.

\[
a = f(p_1 w_1 + p_2 w_2 + p_3 w_3 + b) = f(\sum p_i w_i + b)
\]

Two main **hyperparameters** that control the architecture or topology of the network: 1) the number of layers, and 2) the number of nodes in each hidden layer.
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: \( a = \text{purelin}(n) \)
- Symmetric Hard Limit Trans. Funct.: \( a = \text{hardlims}(n) \)
- Satlin Transfer Function: \( a = \text{satlin}(n) \)
- Tan-Sigmoid Transfer Function: \( a = \text{tansig}(n) \)
- Log-Sigmoid Transfer Function: \( a = \text{logsig}(n) \)
- Radial Basis Function: \( a = \text{radbas}(n) \)
Network structures

Two phases in each iteration:

1. Calculating the predicted output $y$, known as feed-forward
2. Updating the weights and biases, known as backpropagation
Feed-forward example

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

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

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

["this", "movie", "was", "great"]

Input →

Hidden →

Output (score) →

[0.7]
Neural Networks

pixels()

Input
Hidden
Output (label)

["cat"]
A fully connected NN layer

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

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

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

The Problem

Linear Model

Neural Network
Is this a CAT or DOG?

CAT   DOG

OUTPUT LAYER

ACTIVATED NEURONS

INPUT LAYER

DEEP NEURAL NETWORK
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.
Each neuron implements a relatively simple mathematical function.

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

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

The composition of \( 10^6 - 10^9 \) such functions is powerful.
“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 (to a degree) with:

- more data
- bigger models
- more computation
Results get better (to a degree) 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](http://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/).
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

$(x_1, x_2, x_3, ..., x_n) \quad \leftrightarrow \quad \text{Vector! (List, Tuple)}$
TensorFlow
Tensors have a **Shape** that is described with a vector.
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 Nodes, also called Operations or 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

bias

Add

Mul

learning rate

...
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.
**Constants**
- **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.
- **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.
<table>
<thead>
<tr>
<th><strong>Category</strong></th>
<th><strong>Examples</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Element-wise math ops</td>
<td>Add, Sub, Mul, Div, Exp, Log, Greater, Less...</td>
</tr>
<tr>
<td>Matrix ops</td>
<td>Concat, Slice, Split, Constant, Rank, Shape...</td>
</tr>
<tr>
<td>Matrix ops</td>
<td>MatMul, MatrixInverse, MatrixDeterminant...</td>
</tr>
<tr>
<td>Stateful ops</td>
<td>Variable, Assign, AssignAdd...</td>
</tr>
<tr>
<td>NN building blocks</td>
<td>SoftMax, Sigmoid, ReLU, Convolution2D...</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>
https://playground.tensorflow.org