CIS 192: Python Programming 🐍

Data Science III: Deep Learning

Arun Kirubaran
(kiruba@seas)
@kirubaran
kirubaran.com
Game Plan

1. Machine learning recap
2. What are neural networks
3. How the perceptron fits in
4. Loss functions and back-prop
5. Gradient Descent
A brief history of deep learning

1. **1940**: electronic brain with adjustable weights
2. **1957**: perceptron model with learnable
3. **1969**: perceptron model criticism (Minsky)
4. **AI WINTER** (little progress other than theories)
5. **2006**: first neural network trained (Hinton)
6. **2012**: AlexNet wins ImageNet challenge (Hinton)
7. **2019**: you take this class!
The Perceptron:

Machine learning model (linear classifier) and basic unit of most neural networks.
Parts of Perceptrons

1. Inputs (i.e. things to classify)
2. Weights (i.e. how to emphasize inputs)
3. Biases (i.e. defining confidence thresholds)
Thought Experiment

You want to decide whether or not to darty, and you only darty when it’s not raining, you don’t have much work or (more importantly) when you have enough friends darting.

How can we represent this model as a perceptron (i.e. inputs, weights, biases)?
Amount of rain

Amount of work

Amount of friends

Yes or No

What are some suitable weights?

Perceptron Model (Minsky-Papert in 1969)
Activation Function

Function that takes a neuron output and says whether it should fire.

\[ \phi(z) = \frac{1}{1 + e^{-z}} \]

Sigmoid activation function powers sigmoid neurons
What is a neural network?

**Answer:** a model inspired by how brains work that has a bunch of neurons wired together.
Why do we care?

Answer: can learn complex (non-linear) relationships with little human intervention (i.e. no manual features)
Universal Approximation Theorem

A neural network with a finite number of neurons can approximate compact subsets on continuous functions.
Areas neural networks shine:

1. Images
2. Speech
3. Text
4. Game playing
How do we build neural networks?

**Answer:** we can do this using perceptrons!
Layers get more and more abstract.
How do we classify new instances?

**Answer:** by inputting our instance to the neural network and letting the information “flow” until the output layer.
How do we find the optimal network parameters?

**Answer:** by adjusting our weights and biases until our model stops making mistakes.
Loss Function

A measure of how incorrect our model is. Can be very complicated, but is typically developed to be easy to differentiate (continuous) and minimize (ideally convex).

Why can’t we just count the number of failed classifications?
Backpropagation

Computing the gradient efficiently for a very big loss function (i.e. thousands of variables) by using the chain rule.
Gradient Descent

Start at the top of the hill, walk down the hill in the direction where it is steepest. Algorithm used to approximate optimal values (weights and biases) to minimize a function (loss function).
Deep Learning Training Process

1. Initialize random parameters
2. Use Backpropagation to compute gradients
3. Use Gradient Descent to **optimally** update parameters
4. Repeat steps 2 - 3
5. Model converges
Problems with implementation

1. Implementing backprop, gradient descent etc.
2. Doing complicated calculus in high-dimension
3. Utilizing GPU and/or CPU optimizations
4. Exporting into meaningful formats
Deep Learning Frameworks

1. Implementations of models and algorithms
2. Computational graphs
3. GPU parallelization and CPU optimizations
4. Standardized model formats
Computational Graphs

\[ e = c \times d \]
\[ e = 6 \]

\[ c = a + b \]
\[ c = 3 \]

\[ d = b + 1 \]
\[ d = 2 \]

\[ a = 2 \]
\[ b = 1 \]

http://colah.github.io/posts/2015-08-Backprop/
Derivatives on Computational Graphs

\[ e = c \times d \]
\[ e = 6 \]

\[ \frac{\partial e}{\partial c} = 2 \]
\[ \frac{\partial e}{\partial d} = 3 \]

\[ c = a + b \]
\[ c = 3 \]

\[ \frac{\partial c}{\partial a} = 1 \]
\[ \frac{\partial c}{\partial b} = 1 \]

\[ a = 2 \]

\[ d = b + 1 \]
\[ d = 2 \]

\[ \frac{\partial d}{\partial b} = 1 \]

\[ b = 1 \]

http://colah.github.io/posts/2015-08-Backprop/
Introducing Keras:

- Created by Francois Chollet (maintained by Google)
- High-Level API for deep learning
- Interface for TensorFlow’s computation
- Utilizes C to speed up processes on CPU
- Can parallelize training batches onto GPU
- Has a computational graph implementation