The Benefits of EW in Academia

Prof. Warren P. du Plessis
wduplessis@ieee.org

Department of Electrical, Electronic and Computer Engineering
Faculty of Engineering, Built Environment and Information Technology
University of Pretoria

6 November 2019

Presentation Outline

1. Why not EW?
   - Secrecy
   - Military Connotations
2. Why EW?
   - Systems Thinking
   - Limited Information
   - Academic Dishonesty
3. Conclusion

Why not EW?

Secrecy

Academia is Open, EW is Secret

Academia is Open, EW is Secret

Academia is Open, EW is Secret

- Academic freedom
  - Essential for true research
- Sharing of information
  - Publish or perish!
- Military is secretive
  - Keep advantage
- Publications on EW
  - Many books
  - Classic problems
  - Dual-use

Military Connotations

Military Connotations

Military Connotations

Military Connotations

Military Connotations

Military Connotations

Military Connotations

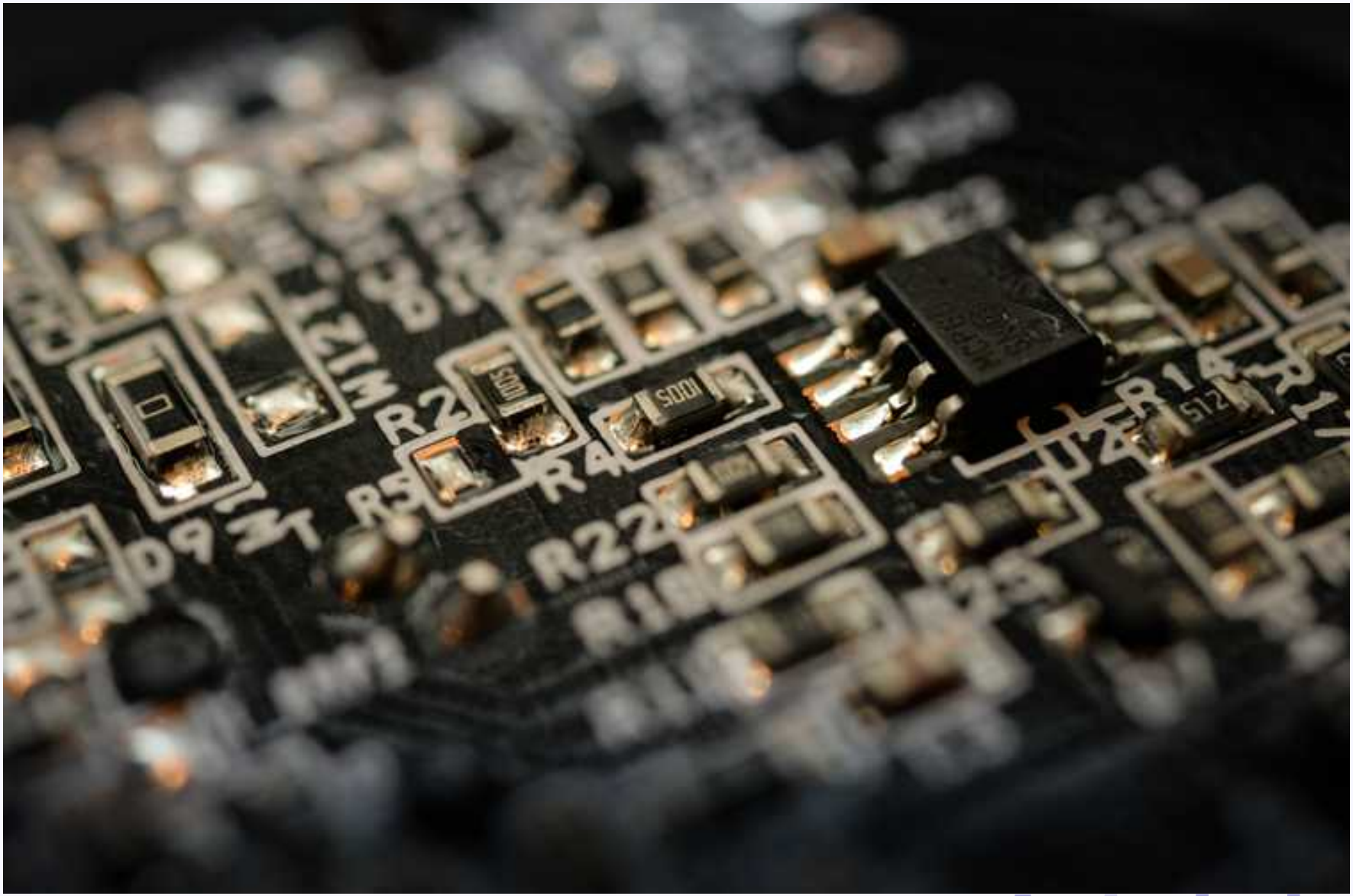
Military Connotations

- Peace is a requirement
  - Growth
  - Development
- No soldier actually wants to fight
  - Deterrence
  - Insurance
- Most violent continent
  - Coming our way
- EW
  - Protecting people
  - Maximising effectiveness

Why EW?

Systems Thinking

Systems Thinking



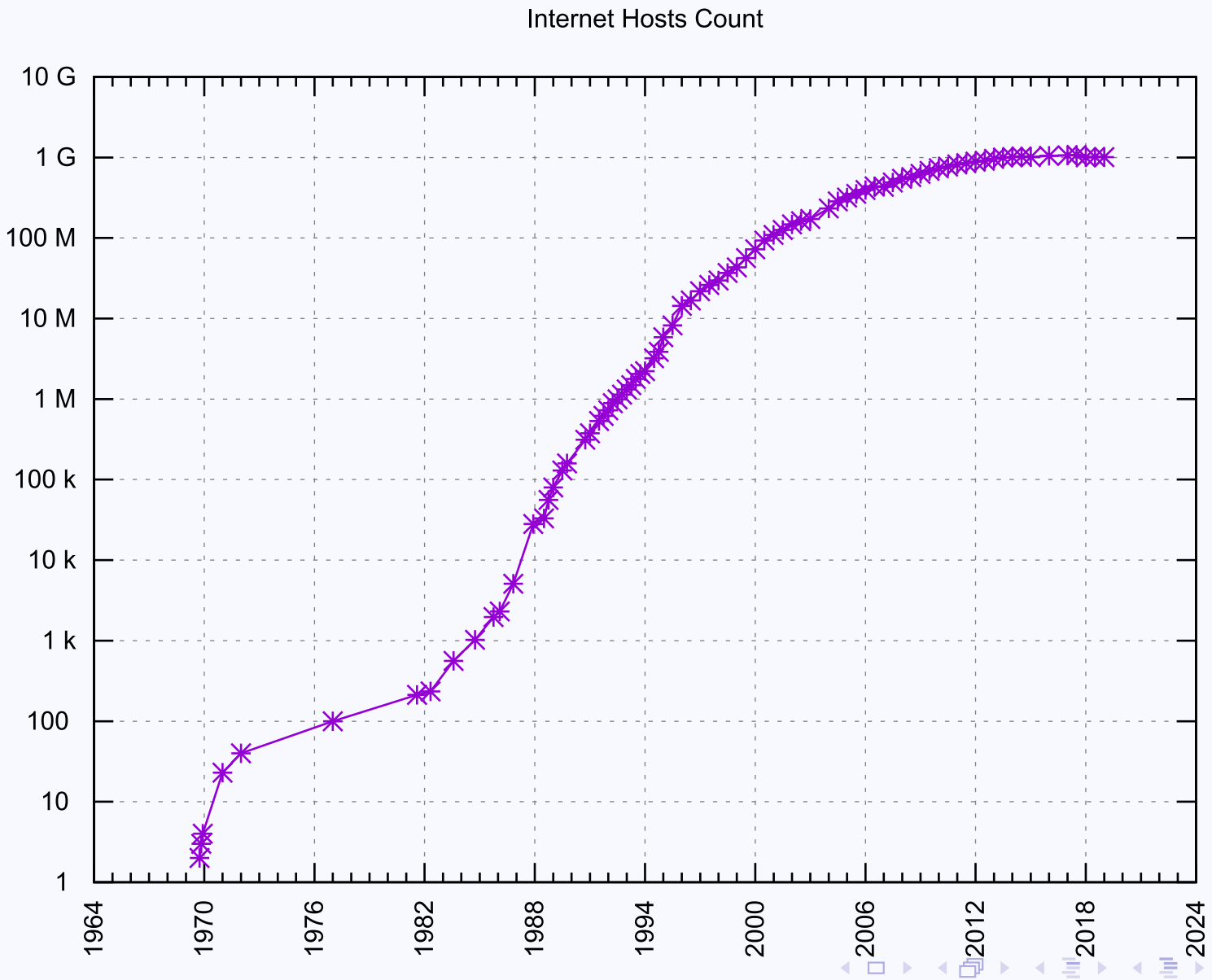
Systems Thinking

Systems Thinking

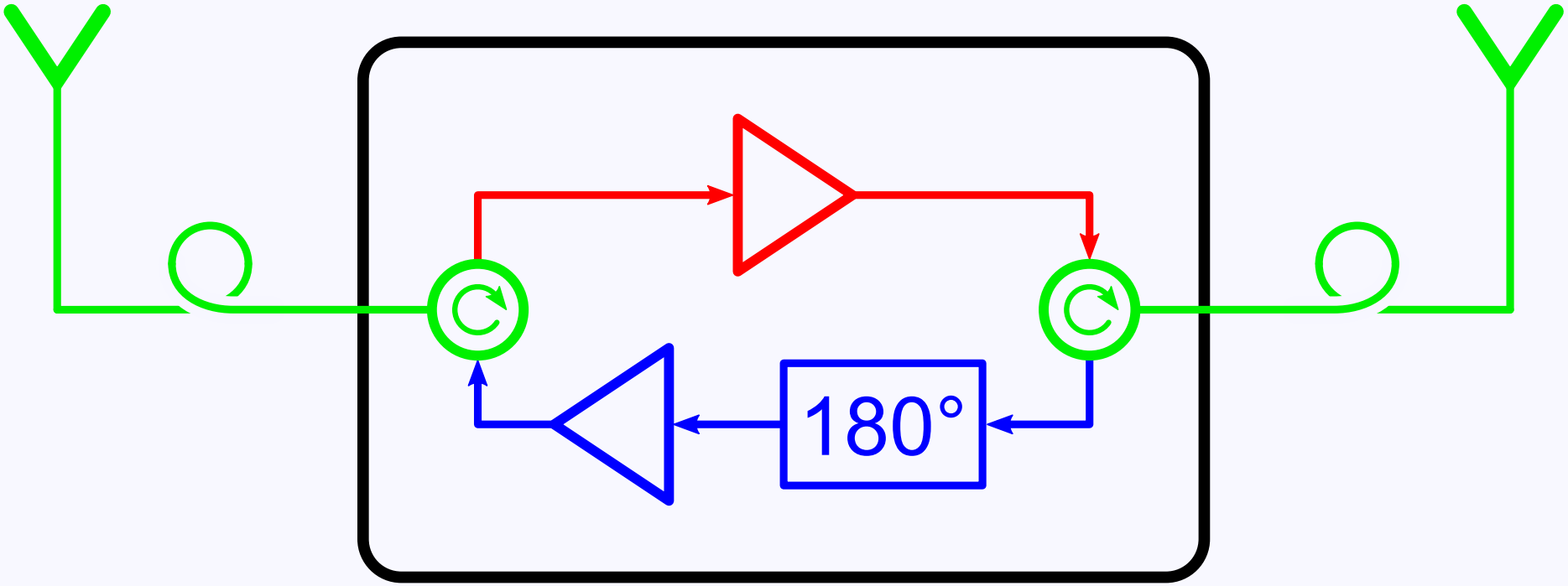
- Engineering education
  - Too much information
  - Many independent topics
  - Seldom connected
- EW is about systems of systems
  - Interaction of systems
  - Cannot consider sub-sub-subsystems in isolation

Limited Information

Lack of Detailed Information



Lack of Detailed Information



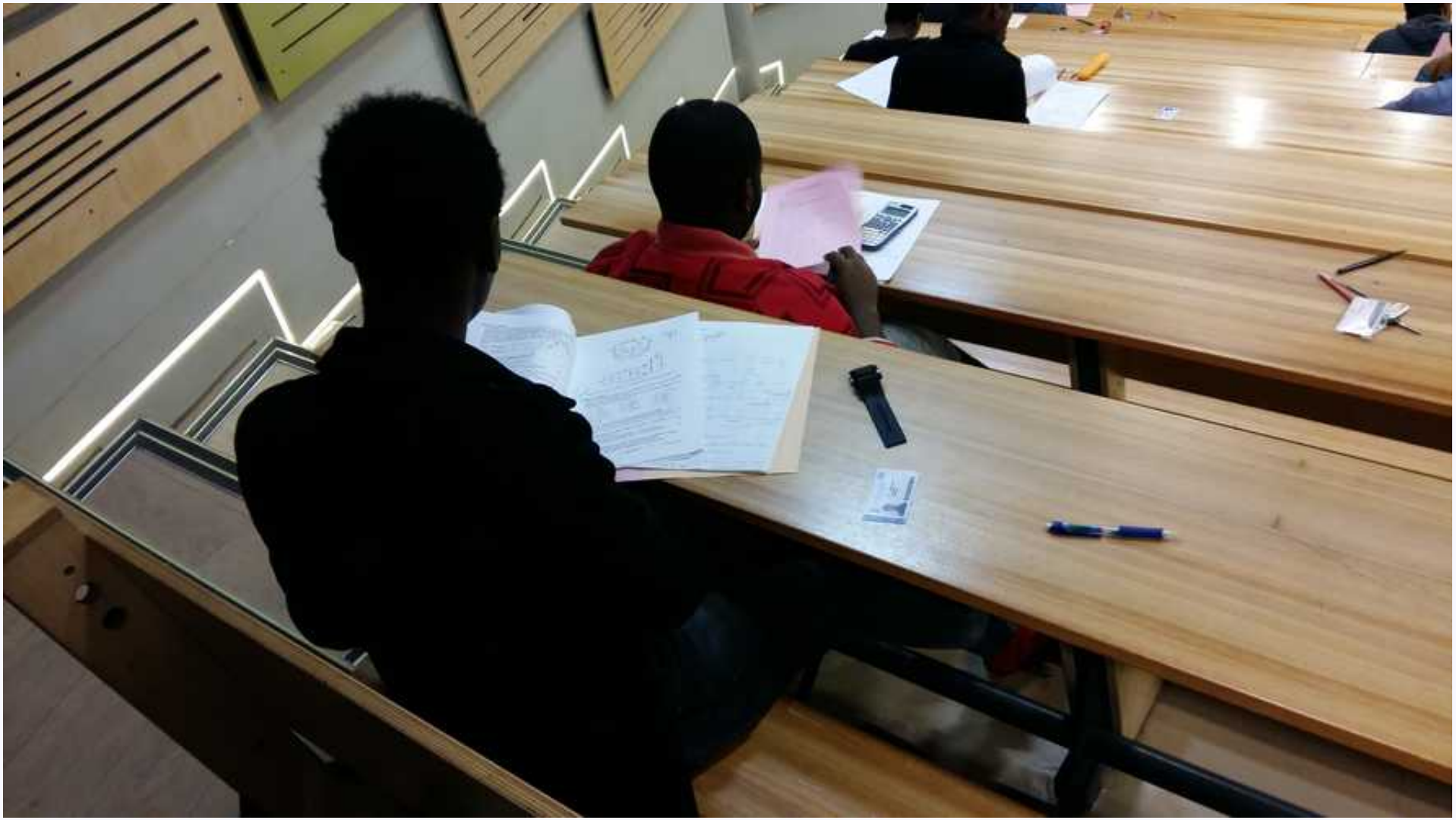
$$\theta_i = \theta_r + \theta_e \frac{1 - a^2}{1 + a^2 + 2a\cos(\phi)}$$

Limited Information

- Information (overload) age
  - Just look the solution up
  - What skills are truly valuable?
- Opportunities
  - Possibilities to develop new ideas
    - Even if only new in open literature...

Academic Dishonesty

Academic Dishonesty



Academic Dishonesty

Navigation

Machine Learning Mastery

Click to Take the FREE Crash-Course

Search...

How to Code a Neural Network with Backpropagation In Python

by **Jason Brownlee** on November 7, 2016 in **Code Algorithms From Scratch**

Tweet Share Share

Last Updated on September 26, 2019

The backpropagation algorithm is used in the classical feed-forward artificial neural network.

It is the technique still used to train large deep learning networks.

In this tutorial, you will discover how to implement the backpropagation algorithm for a neural network from scratch with Python.

After completing this tutorial, you will know:

- How to forward-propagate an input to calculate an output.
- How to back-propagate error and train a network.
- How to apply the backpropagation algorithm to a real-world predictive modeling problem.

Discover how to code ML algorithms from scratch including kNN, decision trees, neural nets, ensembles and much more in my new book, with full Python code and no fancy libraries.

Let's get started.

- **Update Nov/2016**: Fixed a bug in the activate() function. Thanks Alex!
- **Update Jan/2017**: Fixes issues with Python 3.
- **Update Jan/2017**: Updated small bug in update_weights(). Thanks Tomasz!
- **Update Apr/2018**: Added direct link to CSV dataset.
- **Update Aug/2018**: Tested and updated to work with Python 3.6.
- **Update Sep/2019**: Updated wheat-seeds.csv to fix formatting issues.

neurons in the hidden layer can use in the subsequent iteration. I chose the name 'delta' to reflect the change the error implies on the neuron (e.g. the weight delta).

You can see that the error signal for neurons in the hidden layer is accumulated from neurons in the output layer where the hidden neuron number **j** is also the index of the neuron's weight in the output layer **neuron['weights'][j]**.

```
# Backpropagate error and store in neurons
def backward_propagate_error(network, expected):
    for i in reversed(range(len(network))):
        layer = network[i]
        errors = list()
        if i != len(network)-1:
            for j in range(len(layer)):
                error = 0.0
                for neuron in network[i + 1]:
                    error += (neuron['weights'][j] * neuron['delta'])
                errors.append(error)
        else:
            for j in range(len(layer)):
                neuron = layer[j]
                errors.append(expected[j] - neuron['output'])
        for j in range(len(layer)):
            neuron = layer[j]
            neuron['delta'] = errors[j] * transfer_derivative(neuron['output'])
```

Let's put all of the pieces together and see how it works.

We define a fixed neural network with output values and backpropagate an expected output pattern. The complete example is listed below.

```
# Calculate the derivative of an neuron output
def transfer_derivative(output):
    return output * (1.0 - output)

# Backpropagate error and store in neurons
def backward_propagate_error(network, expected):
    for i in reversed(range(len(network))):
        layer = network[i]
        errors = list()
        if i != len(network)-1:
            for j in range(len(layer)):
                error = 0.0
                for neuron in network[i + 1]:
                    error += (neuron['weights'][j] * neuron['delta'])
                errors.append(error)
        else:
            for j in range(len(layer)):
                neuron = layer[j]
                errors.append(expected[j] - neuron['output'])
        for j in range(len(layer)):
            neuron = layer[j]
            neuron['delta'] = errors[j] * transfer_derivative(neuron['output'])

# test backpropagation of error
network = [{'output': 0.7105668883115941, 'weights': [0.13436424411240122, 0.84743373693723
           {'output': 0.6213859615555266, 'weights': [0.2550690257394217, 0.4954350870919409]
expected = [0, 1]
backward_propagate_error(network, expected)
for layer in network:
    print(layer)
```

Running the example prints the network after the backpropagation of error is complete. You can see that error values are calculated and stored in the neurons for the output layer and the hidden layer.

Academic Dishonesty

Transform Grading into Learning. Meet Gradescope.

Meet Gradescope

Menu

Education with Integrity

Your culture of academic integrity begins with Turnitin.

What's New

Contact Sales

Prevent Plagiarism

Identify unoriginal content with the world's most effective plagiarism detection solution. Manage potential academic misconduct by highlighting similarities to the world's largest collection of internet, academic, and student paper content.

Learn More

Authorship

Address contract cheating with confidence. Data driven insights help determine whether students are doing their own work, enabling you to uphold your institution's commitment to educational excellence.

Learn More

Contact Sales

Academic Dishonesty

The Comprehensive GNU Radio Archive Network

The Comprehensive GNU Radio Archive Network (CGRAN) is a free open source repository for 3rd party GNU Radio applications (a.k.a Out Of Tree Modules) that are not officially supported by the GNU Radio project.

GNU Radio Packaging Legend:

OS	GR Version
Ubuntu-16.04	3.7.9.1
Ubuntu-18.04	3.7.11
Ubuntu-19.04	3.7.13.4

Name (?sort=name)	Most Recent Commit (?sort=last_commit)	Description	Categories (?sort=tags)
gr-aep (/2642/)	Sept. 10, 2019	A GNU Radio Module for performing beamforming using Adaptive Event Processing	beamforming, multiantenna, array processing, signal detection, blind
gr-correctiq (/2678/)	Sept. 9, 2019	Short description of gr-correctiq	sdr
gr-isdbt (/2699/)	Sept. 4, 2019	A complete receiver for the digital TV standard ISDB-T.	DTV, ISDB-T, OFDM, Viterbi
gr-microtelecom (/2692/)	Aug. 31, 2019	Microtelecom's Perseus SDR source module	hardware, source
gr-ieee802-11 (/2621/)	Aug. 26, 2019	IEEE 802.11 a/g/p Transceiver	IEEE 802.11, WiFi, OFDM
gr-rstt (/2661/)	Aug. 26, 2019	Receiver for Vaisala Weather Balloons	Vaisala, telemetry, weather balloon
gr-keyfob (/2668/)	Aug. 26, 2019	A transceiver for some Hella key fobs	Key Fob, Car, Hella
gr-fcdproplus (/2599/)	Aug. 17, 2019	A GNU Radio funcube dongle and funcube dongle pro+ source	funcube
gr-gsm (/2663/)	July 25, 2019	A GSM receiver	GSM
gr-lilacsat (/2619/)	July 11, 2019	None	None
gr-doa (/2629/)	July 8, 2019	Direction Finding with the USRP X-Series and TwinRX	doa, twinrx, sdr, array
gr-mixalot (/2657/)	June 1, 2019	Blocks/utilities to encode pager messages	pager, POCSAG, Motorola, Bravo, Advisor, ATI Wireless, CCP-6000
gr-hpsdr (/2665/)	May 17, 2019	modules for OpenHPSDR Hermes / Metis and Red Pitaya	Hermes, Metis, Red Pitaya, OpenHpsdr
gr-ccsds (/2628/)	April 17, 2019	Short description of gr-ccsds	ccsds, fec
gr-framers (/2636/)	April 7, 2019	None	None
gr-bluetooth (/2605/)	March 21, 2019	None	None

Academic Dishonesty

- Major problem
  - Easy to share information
  - Information (overload) age
- Detailed EW is classified
  - Cannot obtain even if try
  - Students' work is their own

Conclusion

CSIR Radar and EW Staff 1981

Thank you!