INTRODUCTION TO ARTIFICIAL NEURAL NETWORK AND TENSORFLOW

QML Journal Club
Friday, Aug 24

MOTIVATION

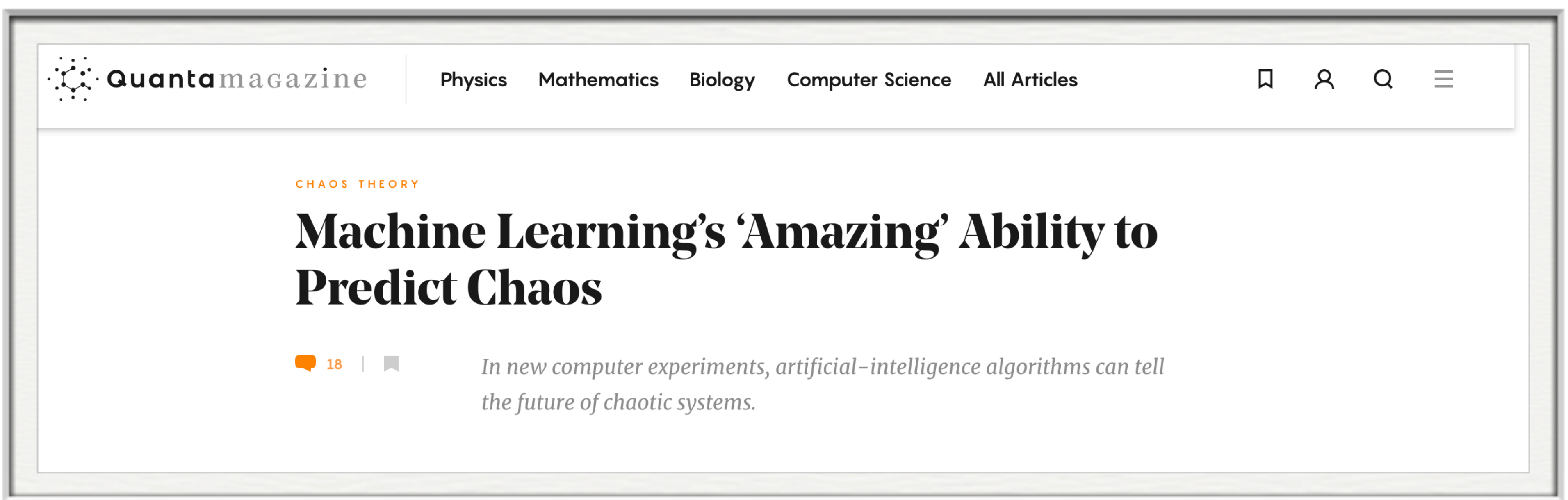
- Foundational:
  - Understanding the principles behind intelligence

Human vs a machine

MOTIVATION

- Practical:
  - “AI is the new electricity ”
    1. Healthcare
    2. Transportation
    3. Communication

  - Applications in Physics



MOTIVATION

SHARE

0

0

RESEARCH ARTICLES | PHYSICS

Solving the quantum many-body problem with artificial neural networks

Giuseppe Carleo[1,*], Matthias Troyer[1,2]

+ See all authors and affiliations

Science 10 Feb 2017:
Vol. 355, Issue 6325, pp. 602-606
DOI: 10.1126/science.aag2302

Article | Figures & Data | Info & Metrics | eLetters | PDF

Machine learning and quantum physics

Elucidating the behavior of quantum interacting systems of many particles remains one of the biggest challenges in physics. Traditional numerical methods often work well, but some of the most interesting problems leave them stumped. Carleo and Troyer harnessed the power of machine learning to develop a variational approach to the quantum many-body problem (see the Perspective by Hush). The method performed at least as well as state-of-the-art approaches, setting a benchmark for a prototypical two-dimensional problem. With further development, it may well prove a valuable piece in the quantum toolbox.

Science, this issue p. 602; see also p. 580

APPLYING MACHINE LEARNING TO PHYSICS

- "Quantum Codes from Neural Networks", Johannes Bausch, Felix Leditzky, arXiv: 1806.08781, 6/2018
- "Analytic continuation via 'domain-knowledge free' machine learning", Hongkee Yoon, Jae-Hoon Sim, Myung Joc Han, arXiv: 1806.03841, 6/2018
- "Machine-learning Skyrmions", Vinit Kumar Singh, Jung Hoon Han, arXiv: 1806.03749, 6/2018
- "Machine learning of quantum phase transitions", Xiao-Yu Dong, Frank Pollmann, Xue-Feng Zhang, arXiv: 1806.00829, 6/2018
- "Adversarial quantum circuit learning for pure state approximation", Marcello Benedetti, Edward Grant, Leonard Wossnig, Simone Severini, arXiv: 1806.00463, 6/2018
- "Automated discovery of characteristic features of phase transitions in many-body localization", Patrick Huembeli, Alexandre Dauphin, Peter Wittek, Christian Gogolin, arXiv: 1806.00419, 6/2018
- "Machine learning many-body localization: Search for the elusive nonergodic metal", Yi-Ting Hsu, Xiao Li, Dong-Ling Deng, S. Das Sarma, arXiv: 1805.12138, 5/2018
- "Deep Learning Topological Invariants of Band Insulators", Ning Sun, Jinmin Yi, Pengfei Zhang, Huitao Shen, Hui Zhai, arXiv: 1805.10503, 5/2018
- "Identifying topological order via unsupervised machine learning", Joaquin F. Rodriguez-Nieva, Mathias S. Scheurer, arXiv: 1805.05961, 5/2018

PHYSICS-INSPIRED IDEAS APPLIED TO MACHINE LEARNING

- "Supervised learning with generalized tensor networks", Ivan Glasser, Nicola Pancotti, J. Ignacio Cirac, arXiv: 1806.05964, 6/2018
- "Universal Statistics of Fisher Information in Deep Neural Networks: Mean Field Approach", Ryo Karakida, Shotaro Akaho, Shun-ichi Amari, arXiv: 1806.01316, 6/2018
- "Entropy and mutual information in models of deep neural networks", Marylou Gabrié, Andre Manoel, Clément Luneau, Jean Barbier, Nicolas Macris, Florent Krzakala, Lenka Zdeborová, arXiv: 1805.09785, 5/2018
- "Physically optimizing inference", Audrey Huang, Benjamin Sheldan, David A. Sivak, Matt Thomson, arXiv: 1805.07512, 5/2018
- "Neural Networks as Interacting Particle Systems: Asymptotic Convexity of the Loss Landscape and Universal Scaling of the Approximation Error", Grant M. Rotskoff, Eric Vanden-Eijnden, arXiv: 1805.00915, 5/2018
- "Neural networks as Interacting Particle Systems: Asymptotic convexity of the Loss Landscape and Universal Scaling of the Approximation Error", Grant M. Rotskoff, Eric Vanden-Eijnden, arXiv: 1805.00915, 5/2018
- "Supervised machine learning algorithms based on generalized Gibbs ensembles", Tatjana Puskarov, Axel Cortes Cubero, arXiv: 1804.03546, 4/2018
- "The Loss Surface of XOR Artificial Neural Networks", Dhagash Mehta, Xiaojun Zhao, Edgar A. Bernal, David J. Wales, arXiv: 1804.02411, 4/2018
- "Matrix Product Operators for Sequence to Sequence Learning", Chu Guo, Zhanming Jie, Wei Lu, Dario Poletti, arXiv: 1803.10908, 3/2018
- "Protection against Cloning for Deep Learning", Richard Kenway, arXiv: 1803.10995, 3/2018

MOTIVATION

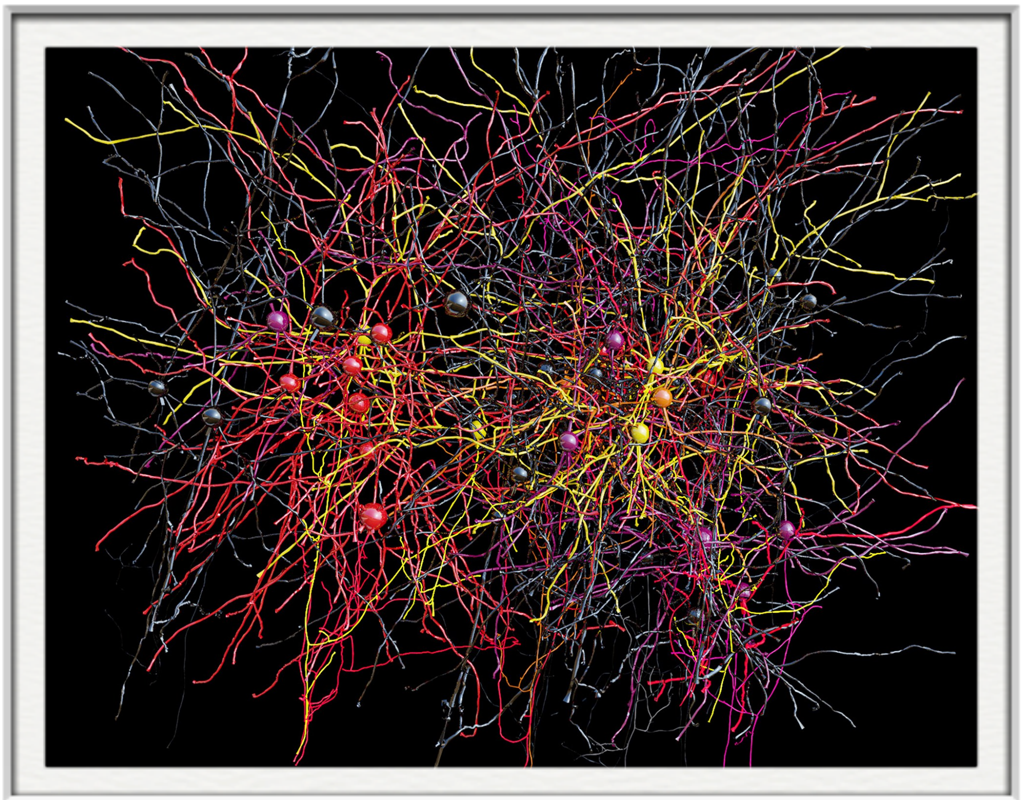
QUANTUM COMPUTATION AND QUANTUM ALGORITHMS FOR MACHINE LEARNING

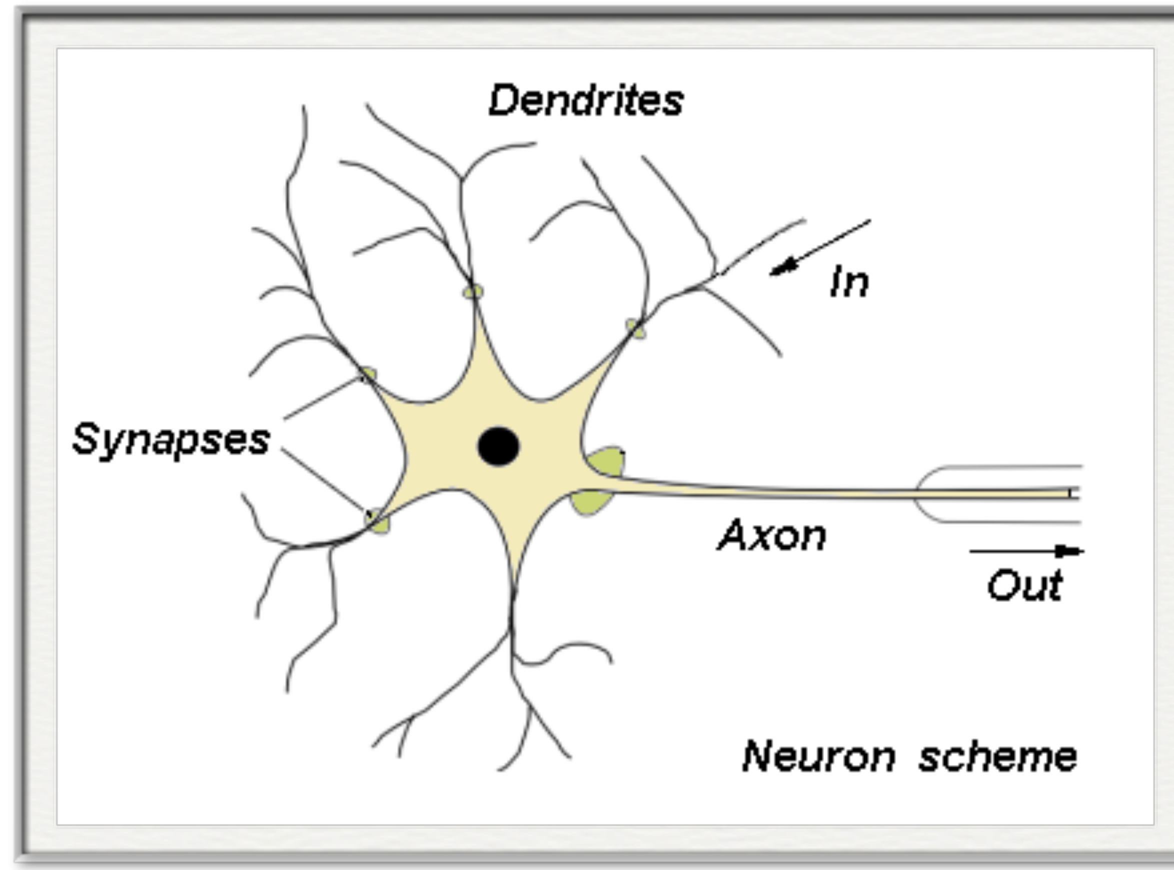
- "A Universal Training Algorithm for Quantum Deep Learning", Guillaume Verdon, Jason Pye, Michael Broughton, arXiv: 1806.09729, 6/2018
- "Artificial Quantum Neural Network: quantum neurons, logical elements and tests of convolutional nets", V. I. Dorozhinsky, O. V. Pavlovsky, arXiv: 1806.09664, 6/2018
- "Quantum Kitchen Sinks: An algorithm for machine learning on near-term quantum computers", C. M. Wilson, J. S. Otterbach, N. Tezak, R. S. Smith, G. E. Crooks, M. P. da Silva, arXiv: 1806.08321, 6/2018
- "Continuous-variable quantum neural networks", Nathan Killoran, Thomas R. Bromley, Juan Miguel Arrazola, Maria Schuld, Nicolás Quesada, Seth Lloyd, arXiv: 1806.06871, 6/2018
- "Decoherence in a quantum neural network", Deniz Türkpençe, Tahir Çetin Akıncı, Serhat Şeker, arXiv: 1806.07251, 6/2018
- "Bayesian Quantum Circuit", Yuxuan Du, Tongliang Liu, Dacheng Tao, arXiv: 1805.11089, 5/2018
- "Quantum classification of the MNIST dataset via Slow Feature Analysis", Iordanis Kerenidis, Alessandro Luongo, arXiv: 1805.08837, 5/2018
- "Universal discriminative quantum neural networks", Hongxiang Chen, Leonard Wossnig, Simone Severini, Hartmut Neven, Masoud Mohseni, arXiv: 1805.08654, 5/2018

SOURCES

- *Deep Learning Specialisation (Coursera's 5 course series)*
- *"TensorFlow for dummies" by Mathew Scarpino*
- *"Deep Learning" by Ian Goodfellow, Yoshua Bengio and Aaron Courville*
- *"Neural Networks and Deep Learning" by Michael Nielsen*

NEURONS

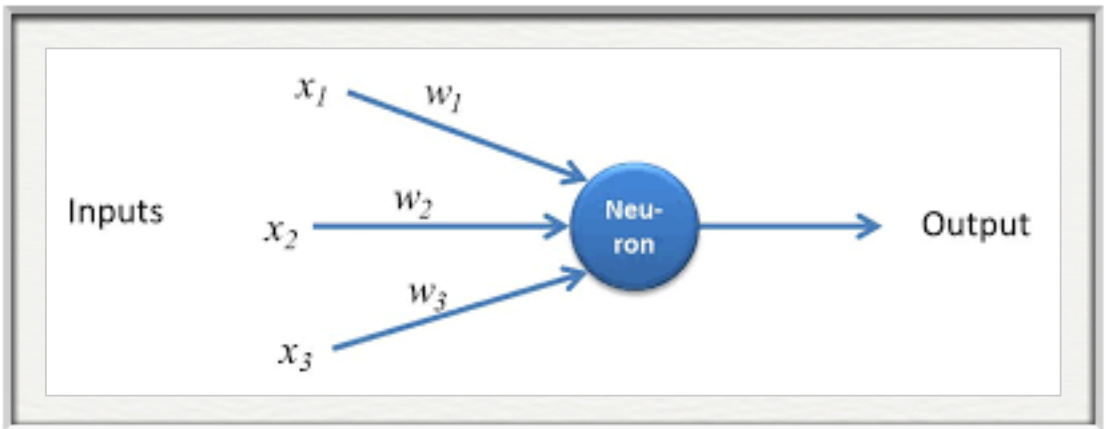


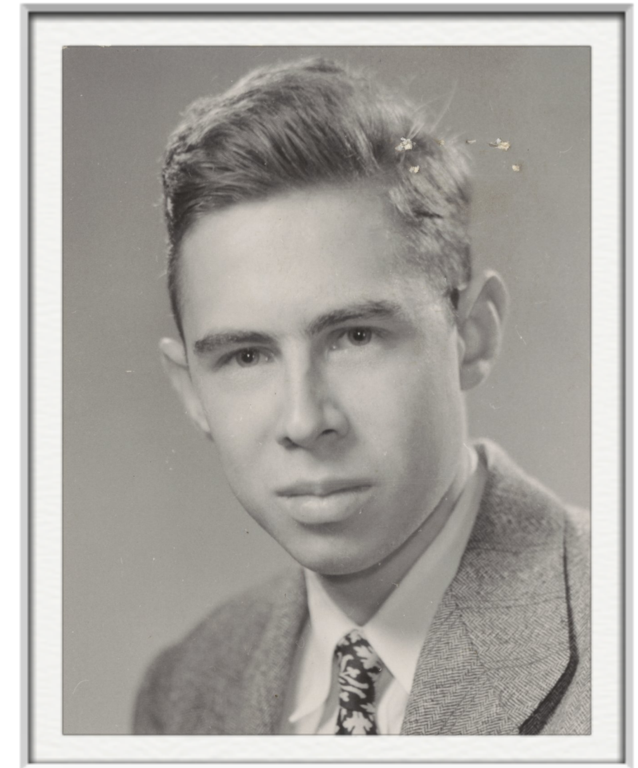


- *Every neuron receives one or more incoming signals and produces one outgoing signal*
- *Output of one neuron serves as input to another*
- *A neuron does not fire until its electricity passes some threshold value.*

PERCEPTRON

- *A type of artificial neuron*
- *Developed by Frank Rosenblatt in 1950s and 60s*



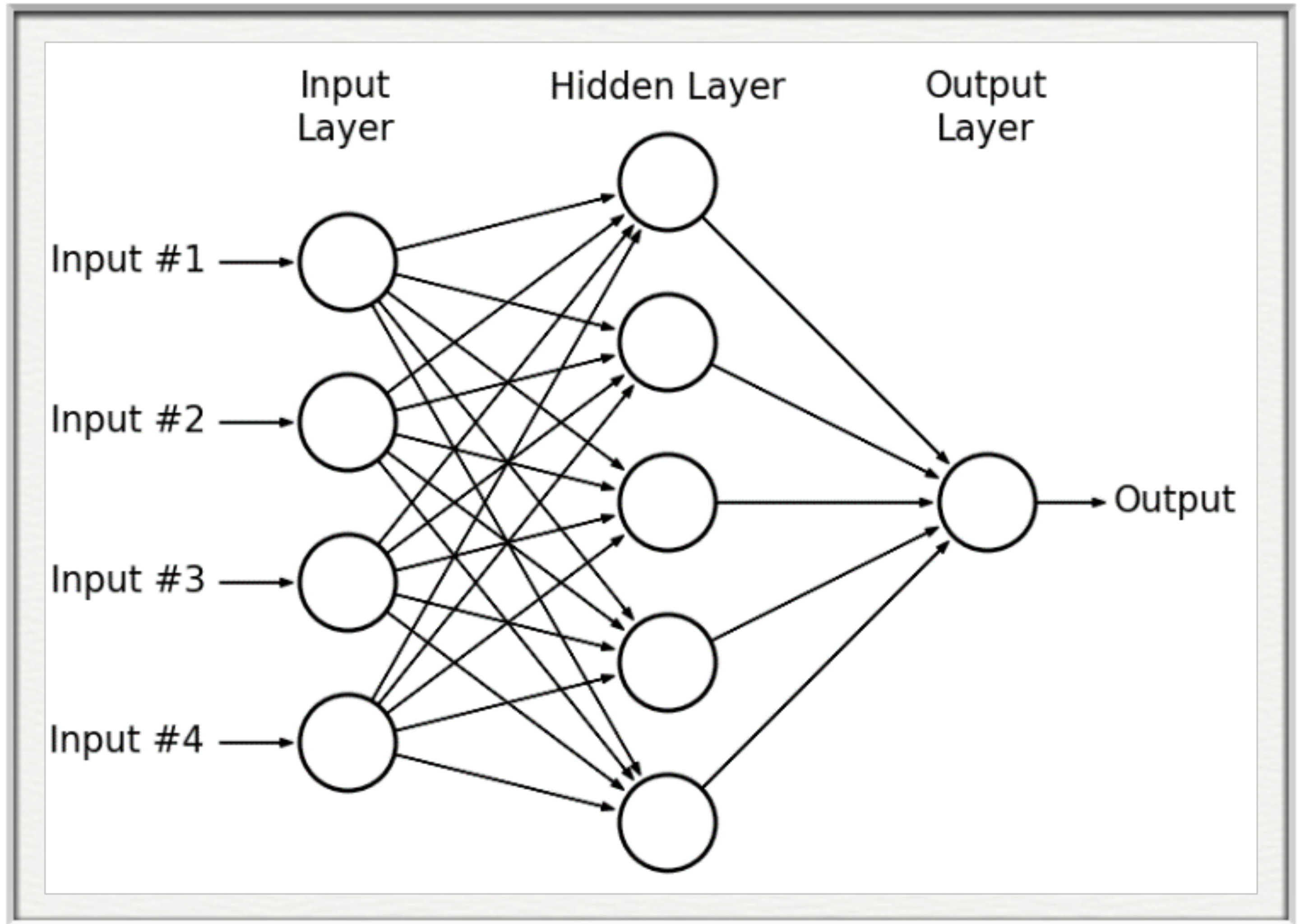


- *Takes several binary inputs*
- *Inputs are assigned weights*
- *Produces a single binary output*

$$\text{Output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{Threshold;} \\ 1 & \text{otherwise.} \end{cases}$$

A device which decides by weighing available evidences

MULTI LAYER PERCEPTRONS



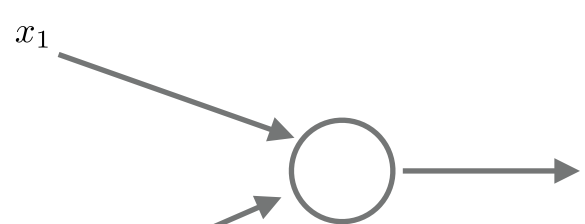
- ***Hidden Layers:*** *Training data does not include desired output for any of these layers.*

- *By varying weights and biases, one can obtain different decision making models.*

PERCEPTRON

- *A perceptron can simulate a NAND gate.*

$$w_1 = -2$$
$$w_2 = -2$$
$$\text{Threshold} = 3$$



$$\text{Output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq 0; \\ 1 & \text{otherwise.} \end{cases}$$

0	0	1
0	1	1
1	0	1
1	1	0

A new kind of NAND gate!

- *Devising learning algorithms to tune the weights and threshold*

TRAINING THE MODEL

- ***Forward Propagation:*** *Computing the output of the network, given network's weights and input*

- ***Loss Function:*** *Captures the difference between model and observed data. The model is improved by reducing the loss.*

Mean Square Error

$$\mathcal{L}(w,b) = \frac{1}{2n}\sum_x |y(x) - a(x)|^2$$

Cross entropy loss function

$$\mathcal{L}(w,b) = \frac{1}{n}\sum_x \left(y(x)\log\left(a(x)\right) + (1-y(x))\log\left(1-a(x)\right)\right)$$

- *Use an optimization algorithm to tune the weights and biases (thresholds)*

ACTIVATION FUNCTIONS

$$\text{Output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{Threshold;} \\ 1 & \text{otherwise.} \end{cases}$$

In general, one need not have step function.

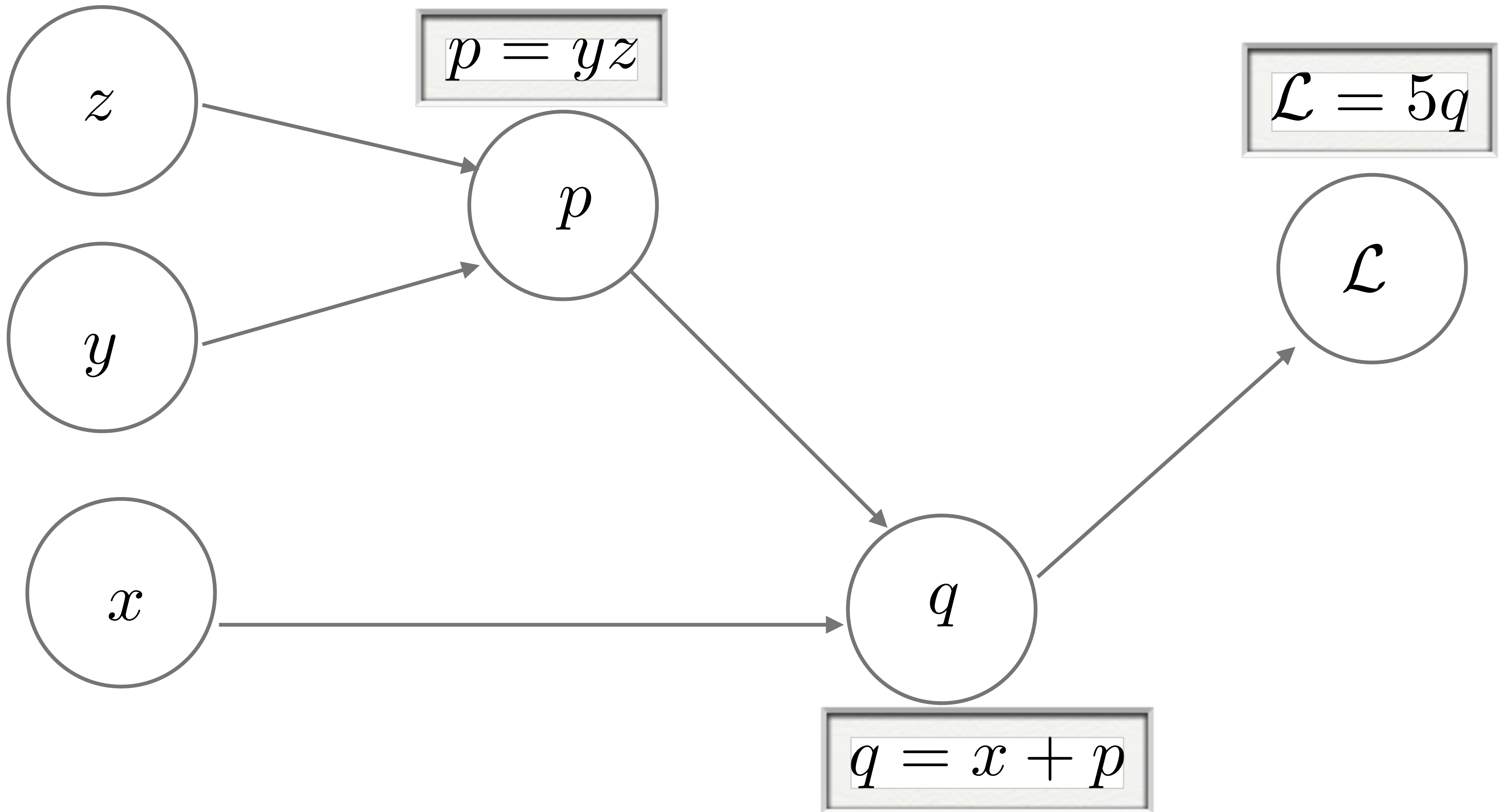
$$\text{Output} = u\left(\sum_i w_i x_i\right)$$

The function which determines neuron's output is called activation function.

- *Sigmoid*
- *Tanh*
- *Relu*

COMPUTATION GRAPH

$$\mathcal{L}(x, y, z) = 5\,(x + yz)$$



- *Move forward to calculate target function*
- *Move backward to calculate the derivatives*

TRAINING THE MODEL

1. *Initialise the weights and biases*
2. *Compute the output of the network (Forward propagation)*
3. *Calculate the loss*
4. *Compute the gradient of the loss by propagating backwards*
5. *Tune the weights and biases*
6. *Iterate 2-6 until loss goes below a tolerance level.*

TUNING THE WEIGHTS AND BIASES

- *Gradient Descent*

$$\boxed{\theta_t = \theta_{t-1} - \eta \nabla \mathcal{L}(\theta)}$$

Learning rate η

Parameters of the loss function at iteration t θ_t

- *Other methods:*
  - *Stochastic gradient descent*
  - *Momentum*
  - *Adam*

HYPERPARAMETERS

- *Learning rate*
- *Number of layers*
- *Number of nodes in a layer*

Drawbacks of adding hidden layers

- *Each extra hidden layer adds to extra time in training the network*
- *Increases the possibility of overfitting*

REGULARIZATION

- ***Dropout***
  - *It randomly removes the nodes from the network*
  - *Once a node is removed, the corresponding connections and weights are also removed.*
- ***L2 Regularization***
  - *Reduces overfitting by decreasing network's weights*
  - *A regulariser is added to the loss function*

TENSORFLOW

- *A machine learning framework by Google*
- *Python interface with backend C++*
- *The central data type is Tensor. TensorFlow means "Flow of tensors across computation graph".*
- *Steps:*
  - *Create tensors*
  - *Create operations*
  - *Execute the computation garph*

TENSORFLOW

- *Four simple codes*
  - *Hello World*
  - *Basic Mathematics*
  - *Optimisation*
  - *Creating deep neural networks*


KEEP
CALM
AND
ASK
QUESTIONS