Intro to Deep Learning with Keras

Thu 20th September 2018 5pm - 6:30pm

PyData Bristol

TensorFlow

Install

pip install tensorflow

pip install keras

Then visit

coefficient.training

ARTIFICIAL NEURAL NETWORKS

1943: The Mechanical Tortoise

1956: The Dartmouth Conference

> **Every aspect** of learning or other feature of intelligence can in principle be so precisely described that **a machine can be made to simulate it**

Top Down?

- Make rules
- Wait for something to break
- Refine the rules
- "Expert systems"
- Brittle & expensive

Bottom Up?

- Define goals
- Systems have complex design
- Systems find patterns
- Lots of computing power
- More like nature

1970s: The AI Winter

> In **three to eight years** we will have a machine with the **general intelligence** of an **average human being**

– 1970, Marvin Minsky, Founder of MIT Artificial Intelligence Lab

> There has been **no machine translation** of general scientific text and **none is in immediate prospect**

– 1966 report by US government advisory committee

> **In no part of the field** have discoveries made so far produced the **major impact that was promised**

– 1973, Professor Sir James Lighthill, "Artificial Intelligence: A General Survey"

80s - 90s: Smaller Ambitions, Big Business

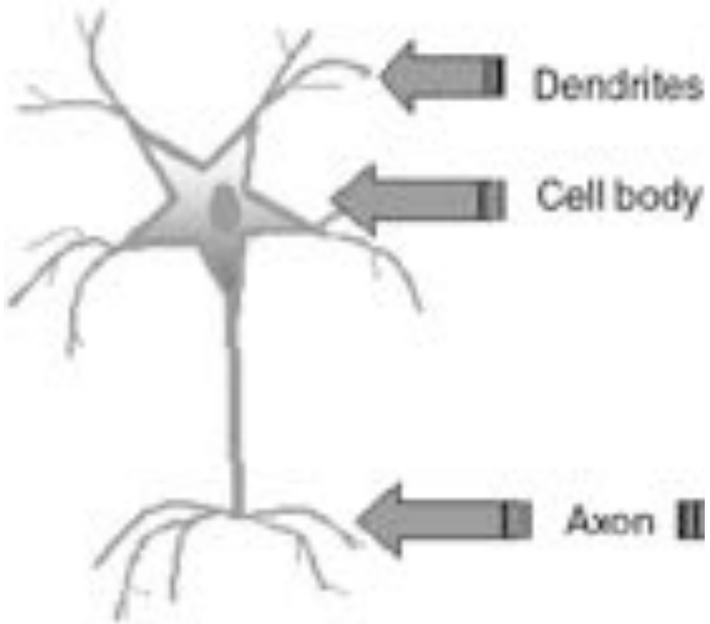
1981:
Expert systems at **DEC** configure orders for new computer systems, **saves company $40m per year**

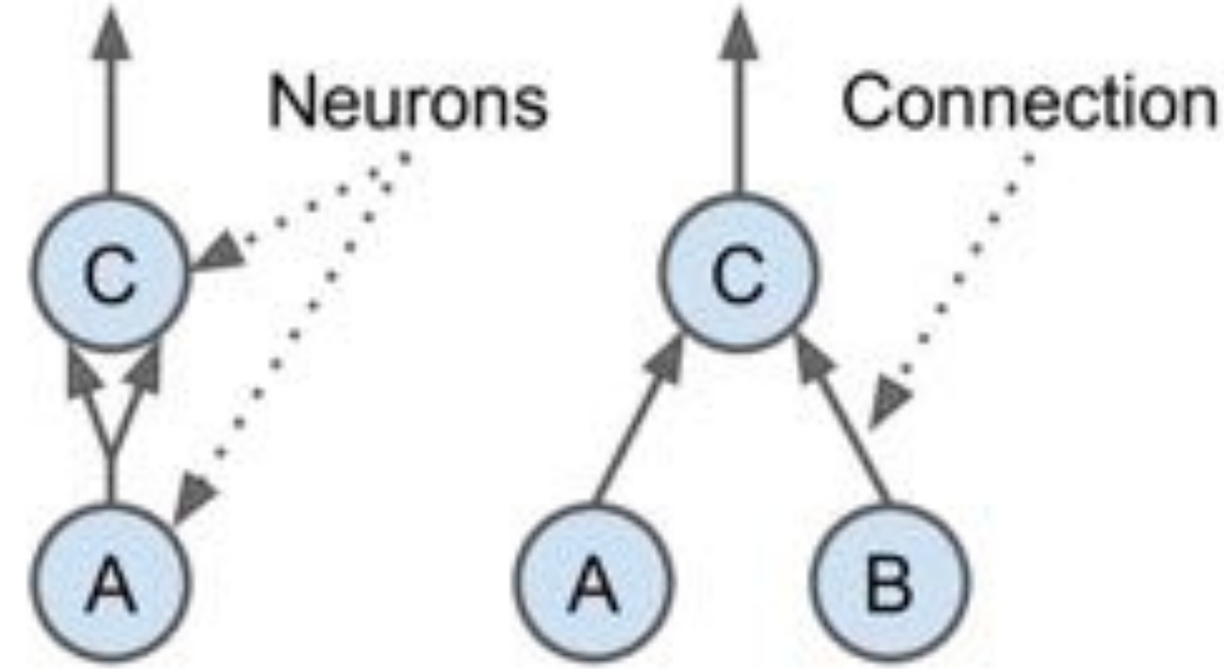
1997:
IBM's Deep Blue defeats world chess champion **Gary Kasparov**

Artificial Neurons

- Proposed by *McCulloch & Pitts*
- Inspired by biological neurons, artificial neurons can replicate logical operations using **binary I/O** and **thresholds**.

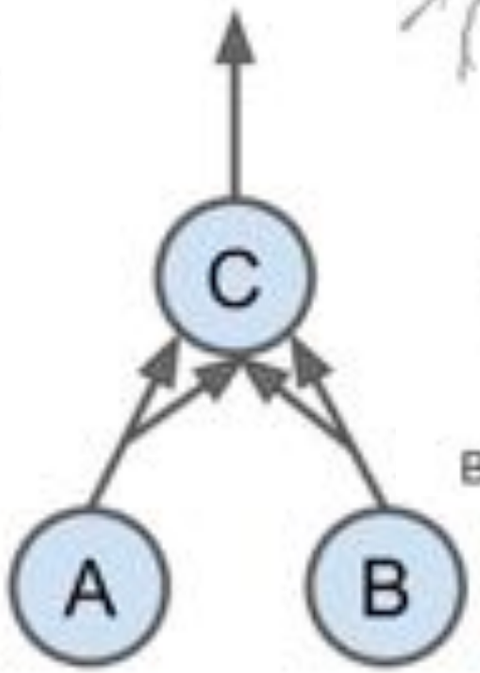
- e.g. neuron fires if **2+ inputs fire**.





C = A

C = A ∧ B



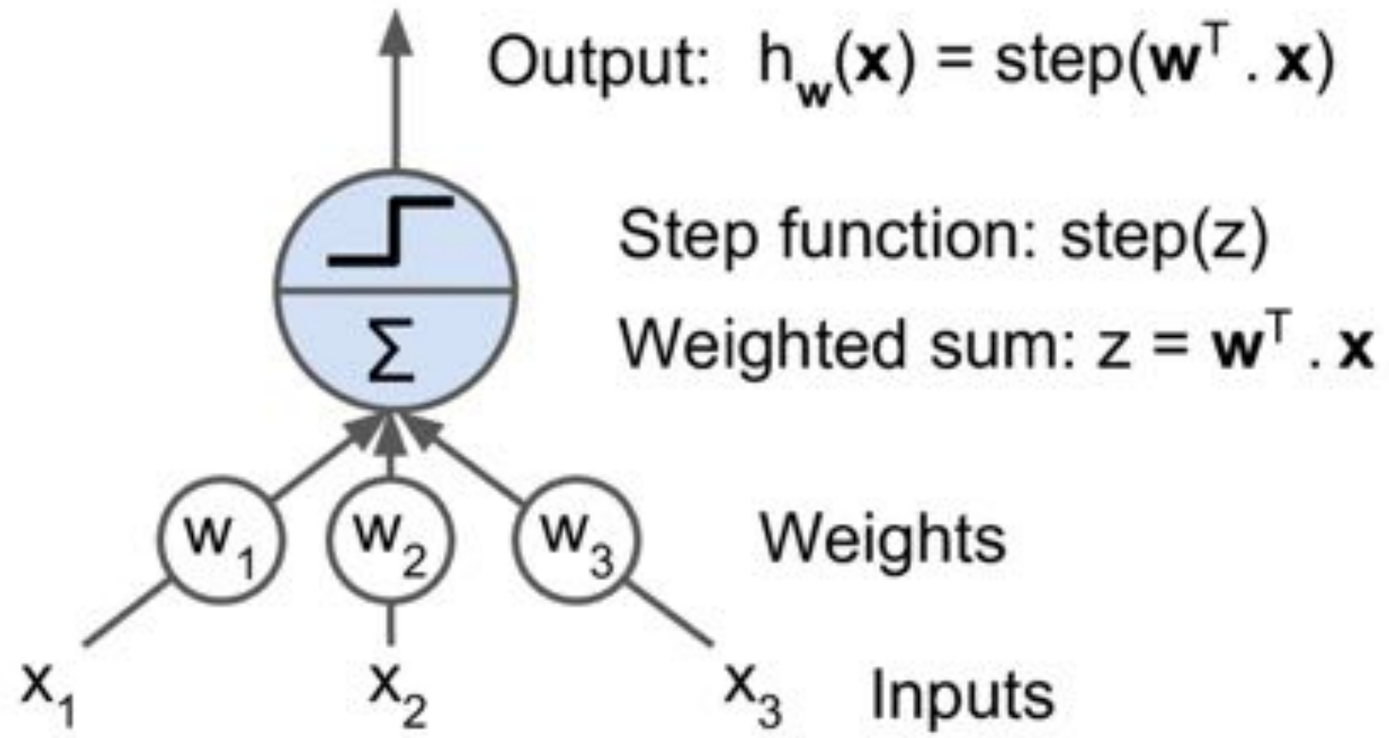
C = A ∨ B

The Linear Threshold Unit

- Frank Rosenblatt, 1957
- Replaces neuron with **linear threshold unit (LTU)**
  - Binary I/O –> **continuous I/O**
  - 1. Compute **weighted sum of inputs** (linear, i.e. dot product)
  - 2. Apply **step function** (e.g. *heaviside* or *sign)*
- Can be used as a simple binary classifier (add a bias + sigmoid step function and you have logistic regression!)

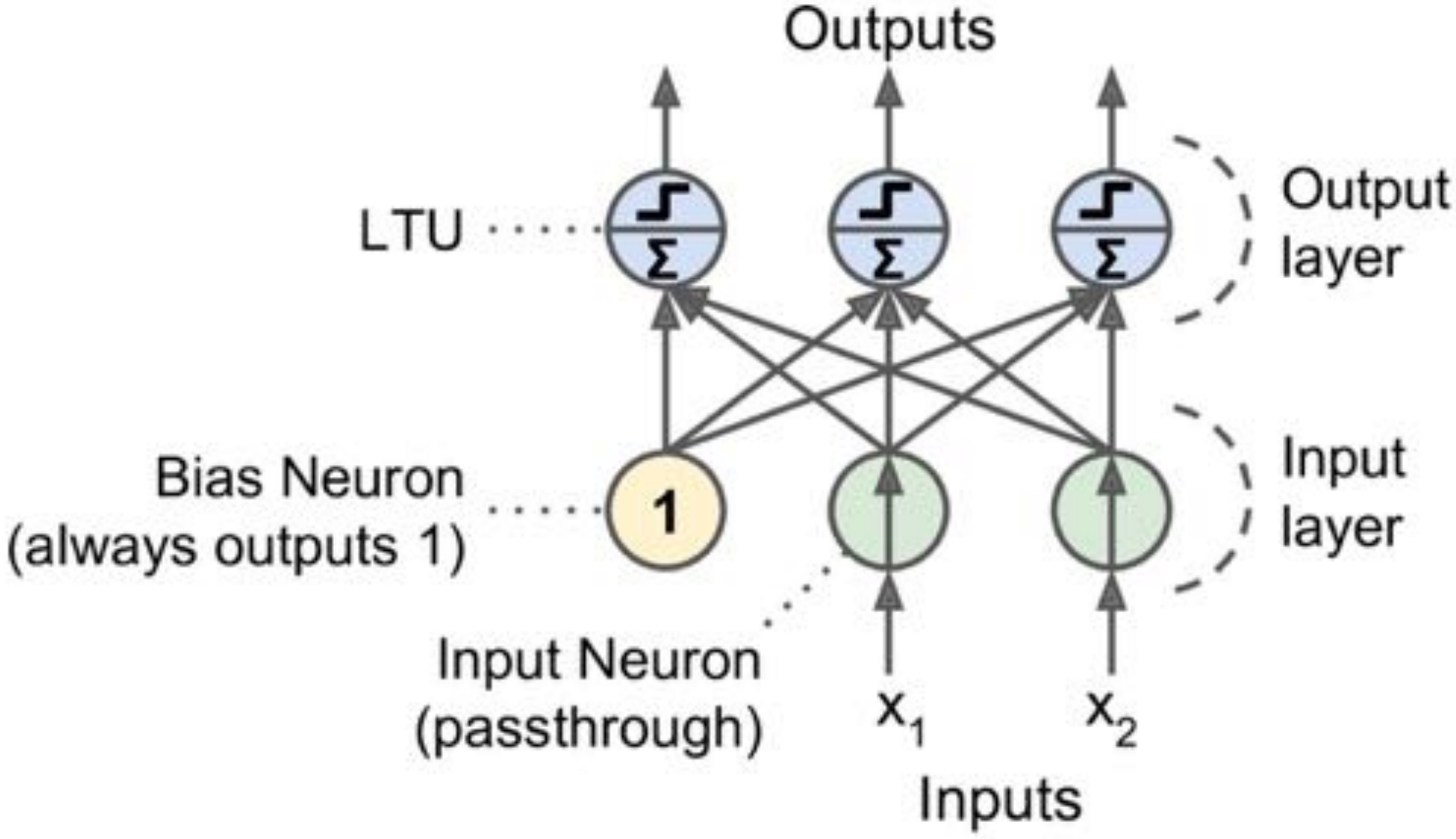
$$\text{heaviside}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

$$\text{sgn}(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ +1 & \text{if } z > 0 \end{cases}$$



The Perceptron

- A **Perceptron** is simply a **layer of LTUs** plus a **bias** term.
- Every input feature flows into every LTU.
- The Perceptron shown below can model three-class binary classification problems.

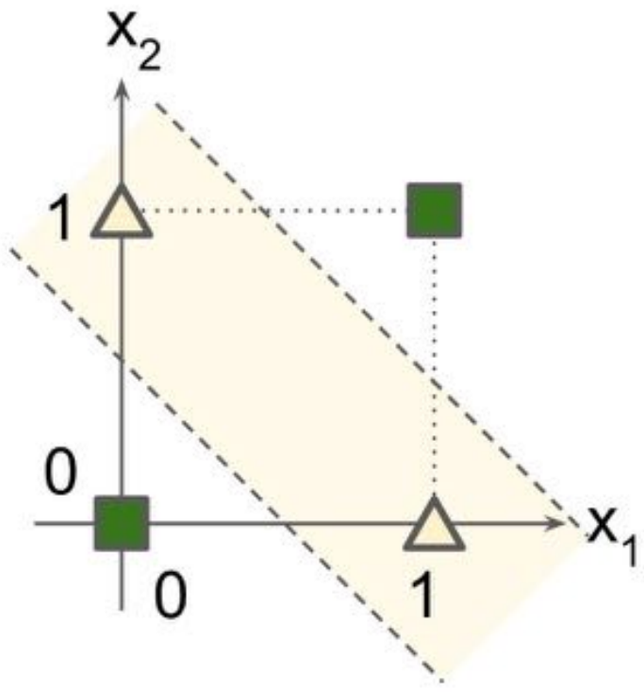


The Perceptron

- **Training Perceptrons**
  - "*Cells that fire together, wire together"* – Siegrid Löwel
  - Input single training instances one at a time. For each incorrect answer, reinforce the weights that would've helped get it right.

- Same as the SGDClassifier in sklearn with **loss='perception'**, **learning_rate='constant'**, **eta0=1**, **penalty=None.**

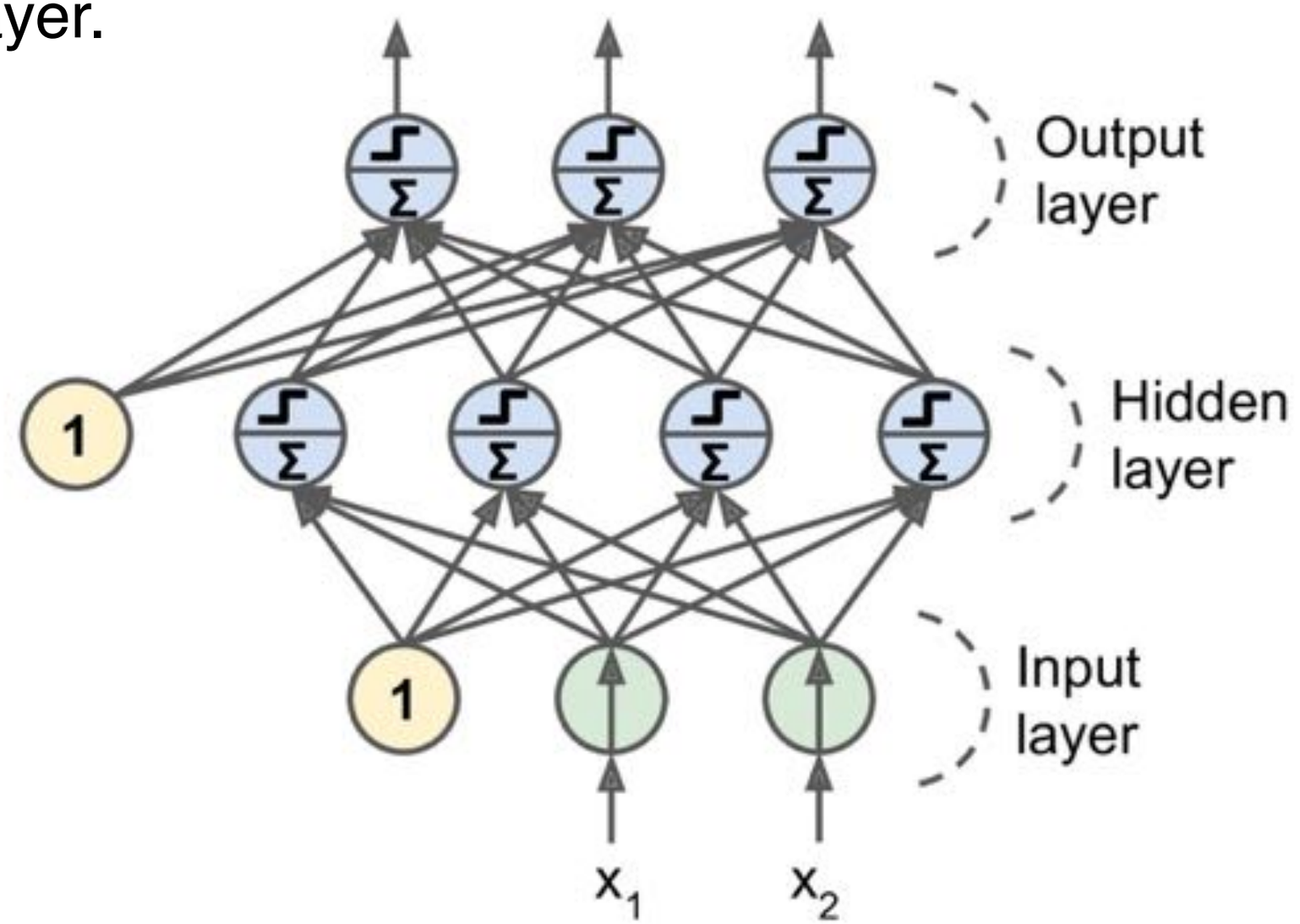
- Similar to Multi-class Logistic Regression, except that logistic regression is able to generate probabilities (via sigmoid function).



- **Issue:** Cannot solve XOR ("exclusive OR") —> (*Minsky & Papert, 1969; cue disappoint & AI Winter)*

The Multi-Layer Perceptron (MLP)

- A **Multi-Layer Perceptron** is a stack of multiple Perceptrons:
  - One input ("*passthrough*") layer.
  - At least one layer of "hidden" LTUs. If there are two or more hidden layers this is called a "*deep neural network*" (DNN).
  - One final output layer.



Activation Functions

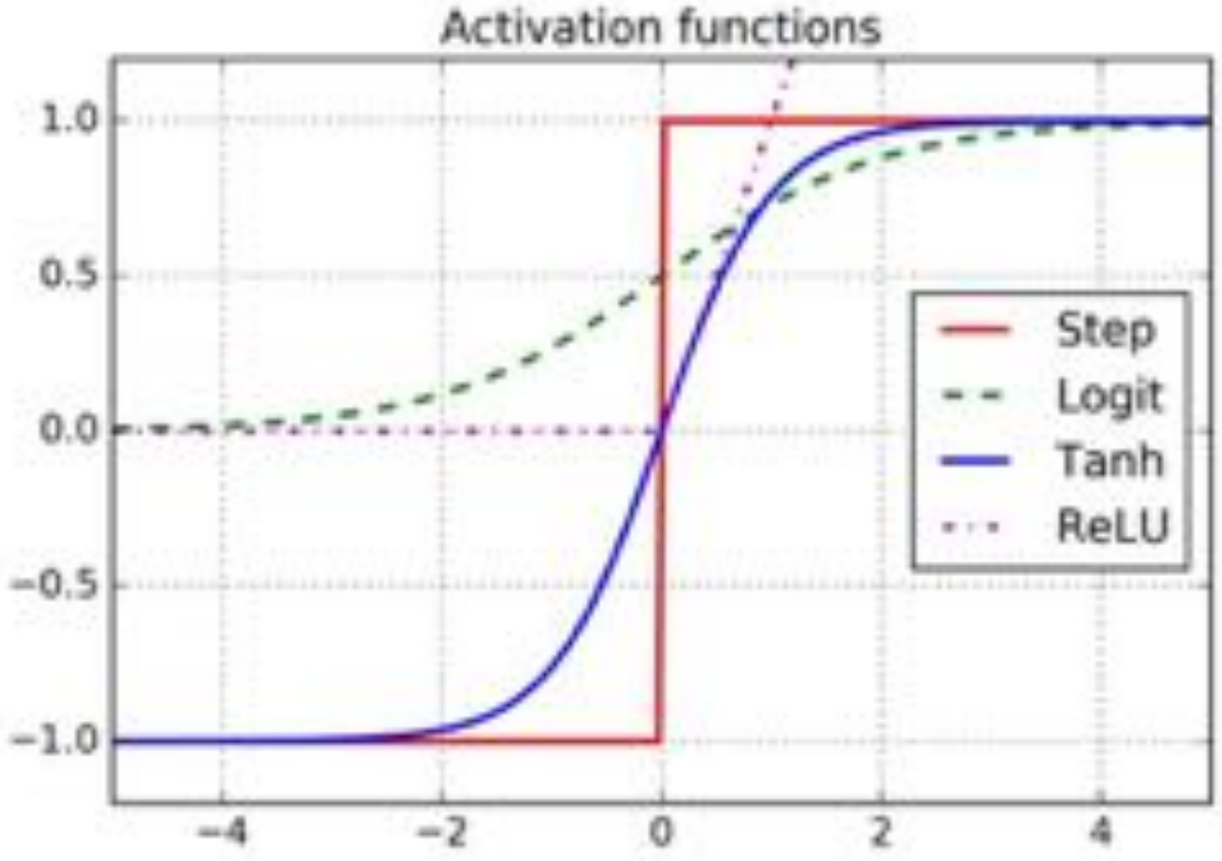
In order to calculate gradients, we need a smooth differentiable function (i.e. not a step function!).

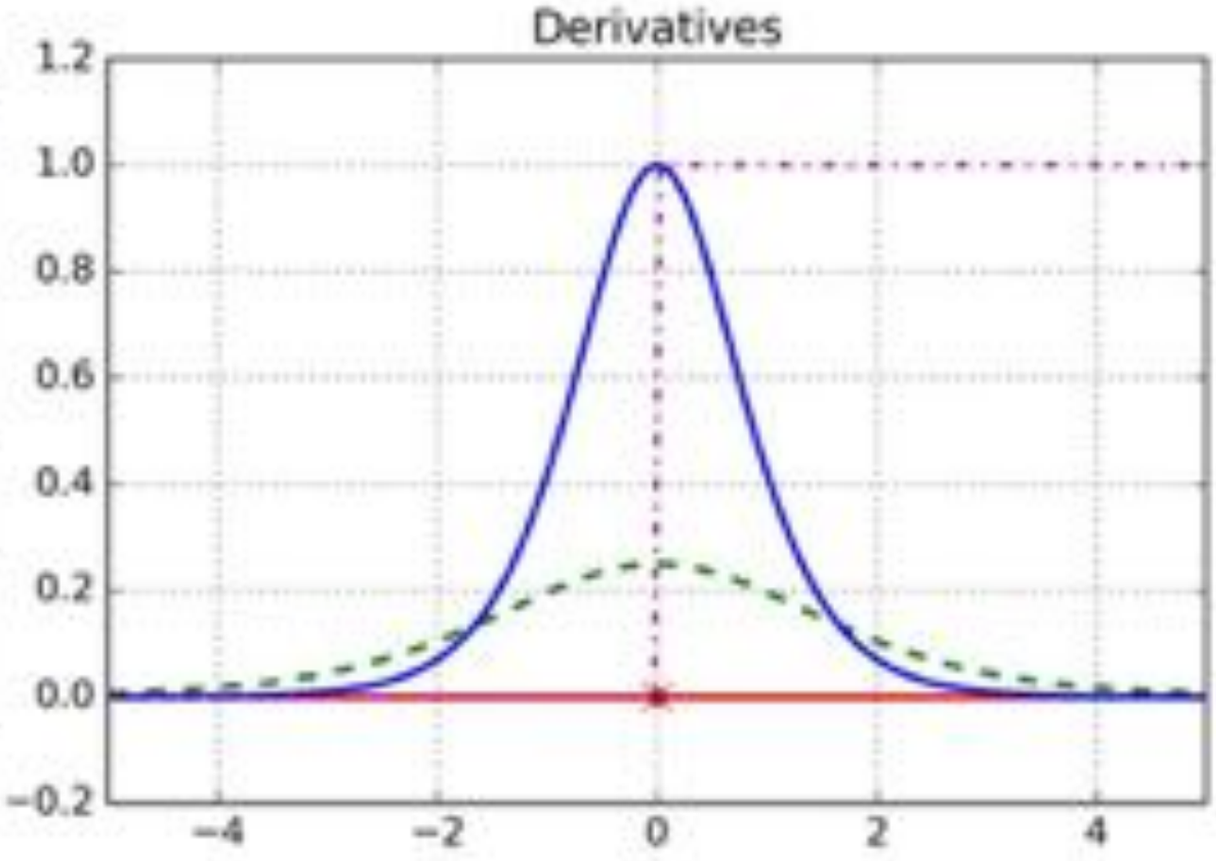
$$sigmoid: \quad \sigma(x) = \frac{1}{1+e^{-x}}$$

$$tanh: \quad \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

$$RELU: \quad relu(x) = \max(0, x)$$

Sigmoid was common early on as nature uses this in biological neurons.





LAB:

MULTI-LAYER PERCEPTRON

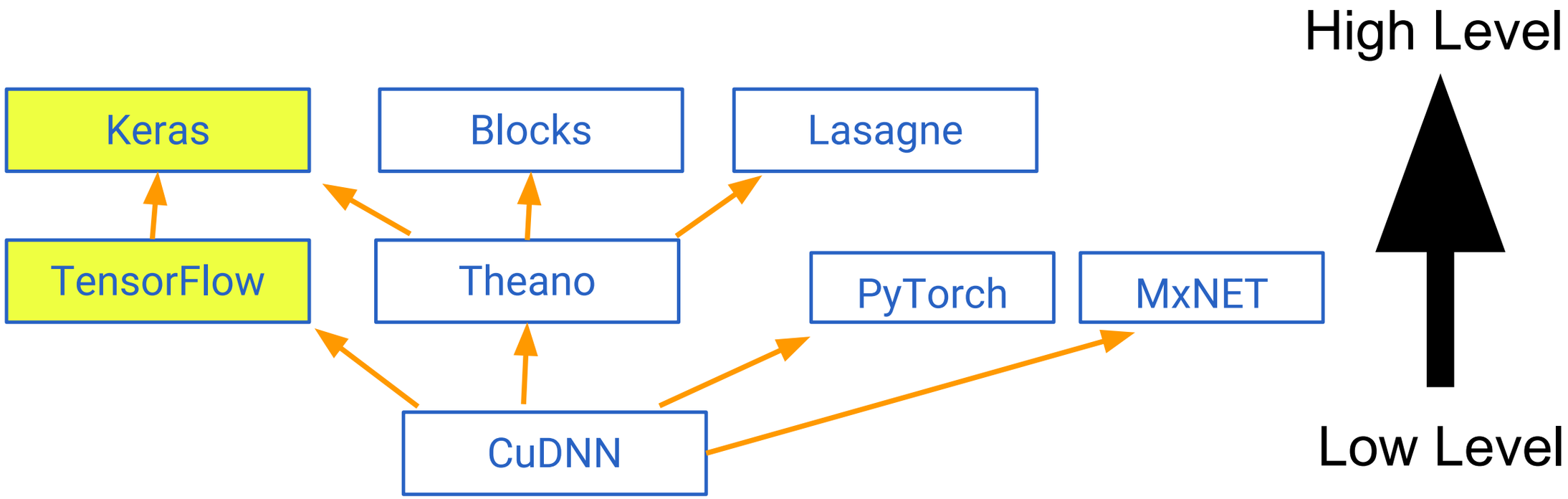

K
Keras

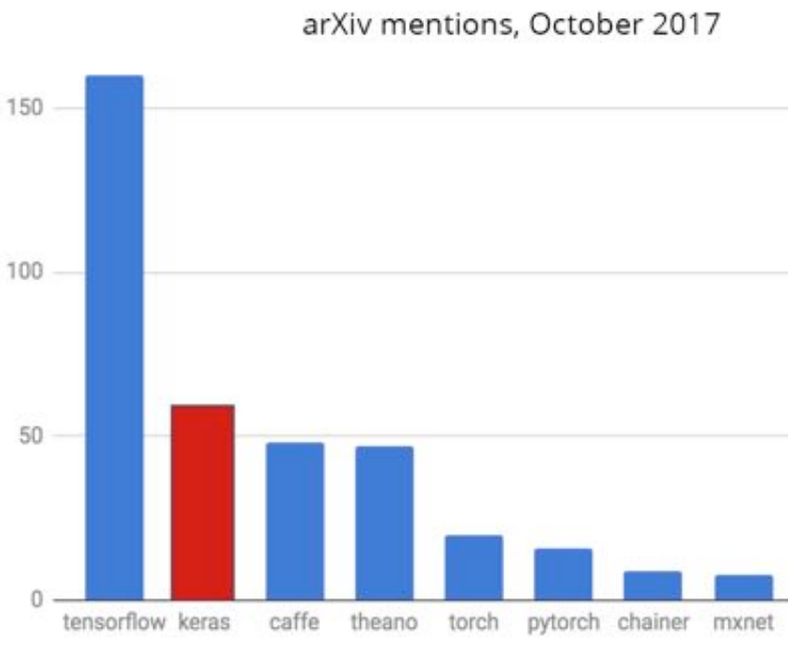
"Keras was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research."

- **Features**
  - High-level neural network API written in Python.
  - Just an API! Doesn't lock you into a framework, can currently run on top of TensorFlow (deeply integrated), CNTK (Microsoft) or Theano, plus MXNet (Apache/Amazon) coming...
  - Supports Feed-Forward Neural Networks, Convolutional NNs, Recurrent NNs, and complex combinations of all the above.
  - Runs seamlessly on both CPU and GPU.

- **Workflow:** **Define** –> **Compile** –> **Fit** –> **Predict**

The Deep Learning Landscape





<– 200k users (Nov'17) & #2 for scientific mentions

Keras runs everywhere!

- **Run Keras on...**
  - **iOS** via CoreML (official Keras support provided by Apple)
  - **Android** via TensorFlow Android runtime.
  - **Browsers** via GPU-accelerated JS runtimes (keras.js, WebDNN)
  - **Google Cloud** via TensorFlow-Serving
  - **JVM** via DL4J model import functionality

- **Supported hardware platforms**
  - **NVIDIA GPUs** via cuDNN
  - **Google TPUs** ("Tensor Processing Units", each providing up to 180 teraflops of performance)
  - **OpenCL-enabled GPUs** (including macOS!) via PlaidML's Keras backend
  - Built-in **multi-GPU support** plus **distributed training** via **Horovod (**Uber), **Dist-Keras** (Spark/CERN), and **Elephas**.

LAB:

BUILD A MLP IN KERAS

LAB: CONVOLUTIONAL NEURAL NETWORKS IN KERAS