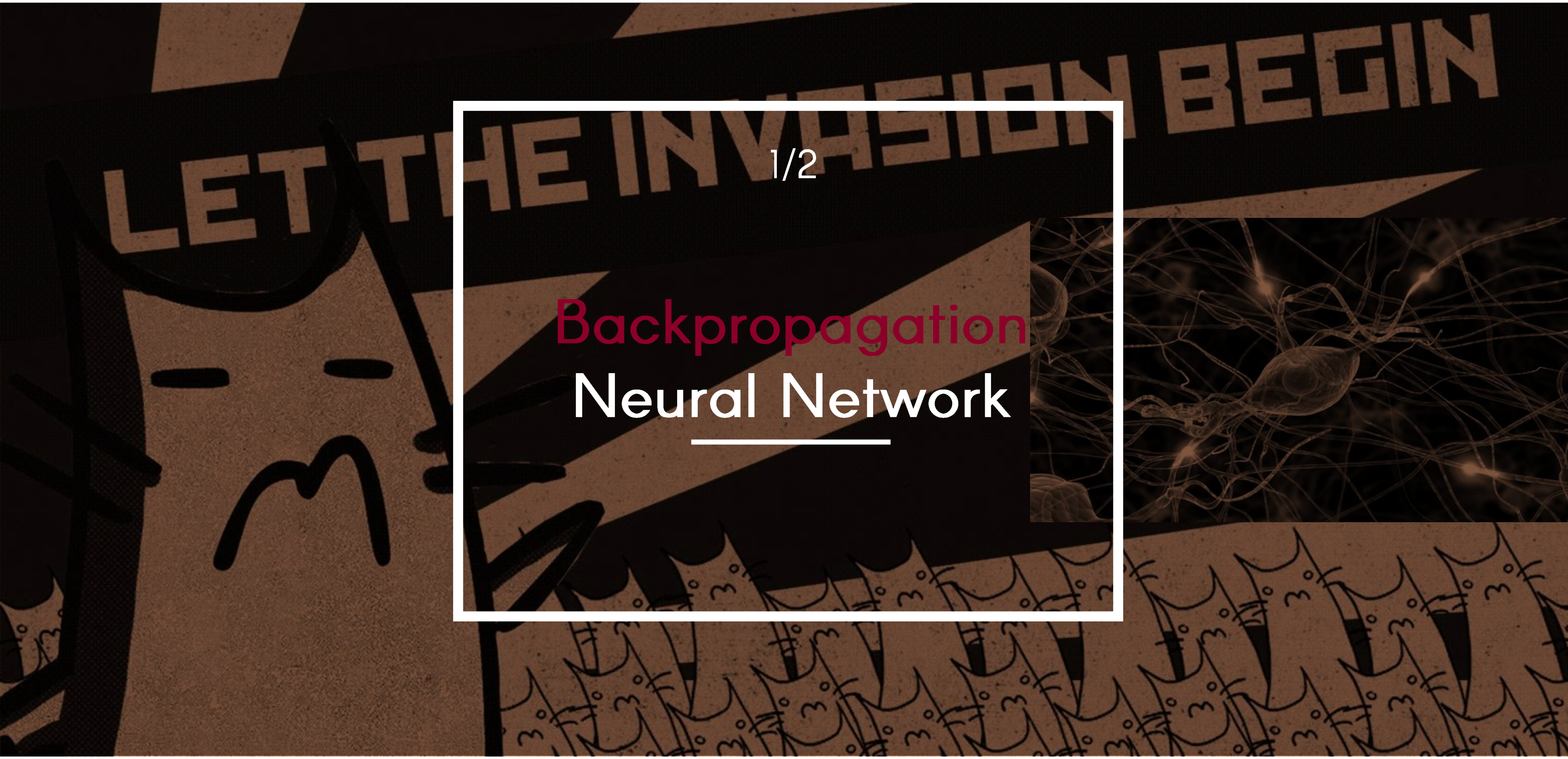
Image source: https://loadingdata.nl/allegiance-propaganda/

Backpropagation, Neural Network

Haedong Kim



1/2

Backpropagation
Neural Network

Motivation

source: http://cs231n.stanford.edu/syllabus.html

Def.

- Backpropagation: a way of computing gradients of functions (∇f)

Problem statement

  - By applying **chain rule** recursively
  - In a **computational way**

Purposes

1. Perform a **parameter update**
2. Other useful purposed
   - Visualization
   - Interpreting neural networks

Compounded function

source: http://cs231n.stanford.edu/syllabus.html

Kernel machines

- We deal with compounded functions in neural networks
  - Multiple hidden layers

1. Support Vector Machine (SVM)

(From the lecture note of Professor Seok, H., Introduction to Machine Learning, Korea Univ., 2016)

   - Convex optimization (single global optimum)

2. Neural Network (NN)
   - Fixing the number of basis functions in advance
   - Allow them to be adaptive in *parametric form*
   - Non-convex optimization -> local minimums

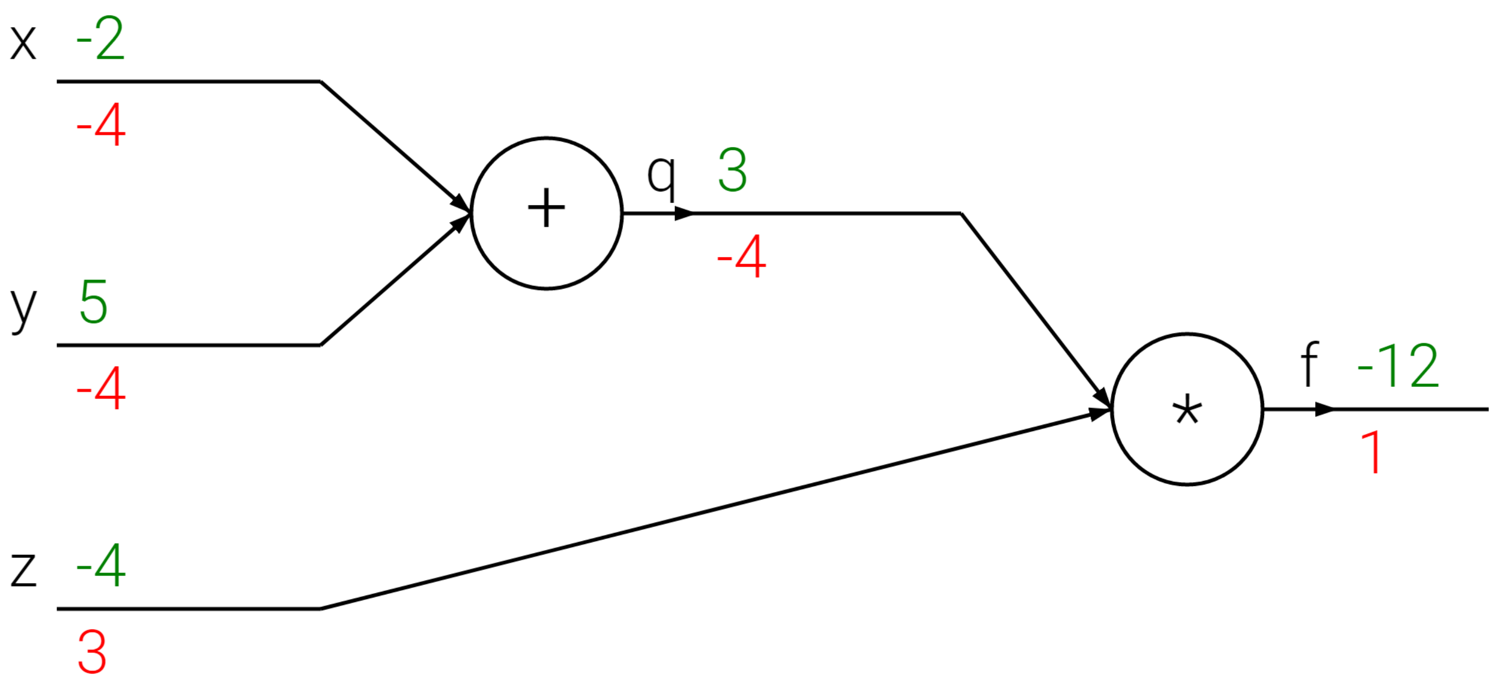
Computational graph

source: http://cs231n.stanford.edu/syllabus.html

e.g.

- $f(x, y, z) = (x + y)z$

Computational graph of the f



Goal

- $\dfrac{\partial f}{\partial z}, \dfrac{\partial f}{\partial x}, \dfrac{\partial f}{\partial y}$

Computational graph

source: http://cs231n.stanford.edu/syllabus.html

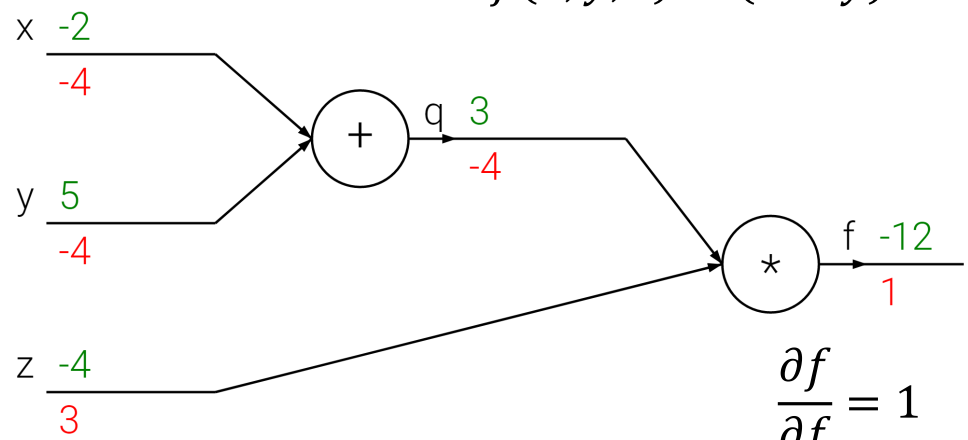
Multiplication

- $f(x,y) = xy$
  - $\dfrac{\partial f}{\partial x} = y, \ \dfrac{\partial f}{\partial y} = x$

Summation

- $f(x,y) = x + y$
  - $\dfrac{\partial f}{\partial x} = 1, \ \dfrac{\partial f}{\partial y} = 1$

$f(x,y,z) = (x+y)z$



$$\frac{\partial f}{\partial f} = 1$$

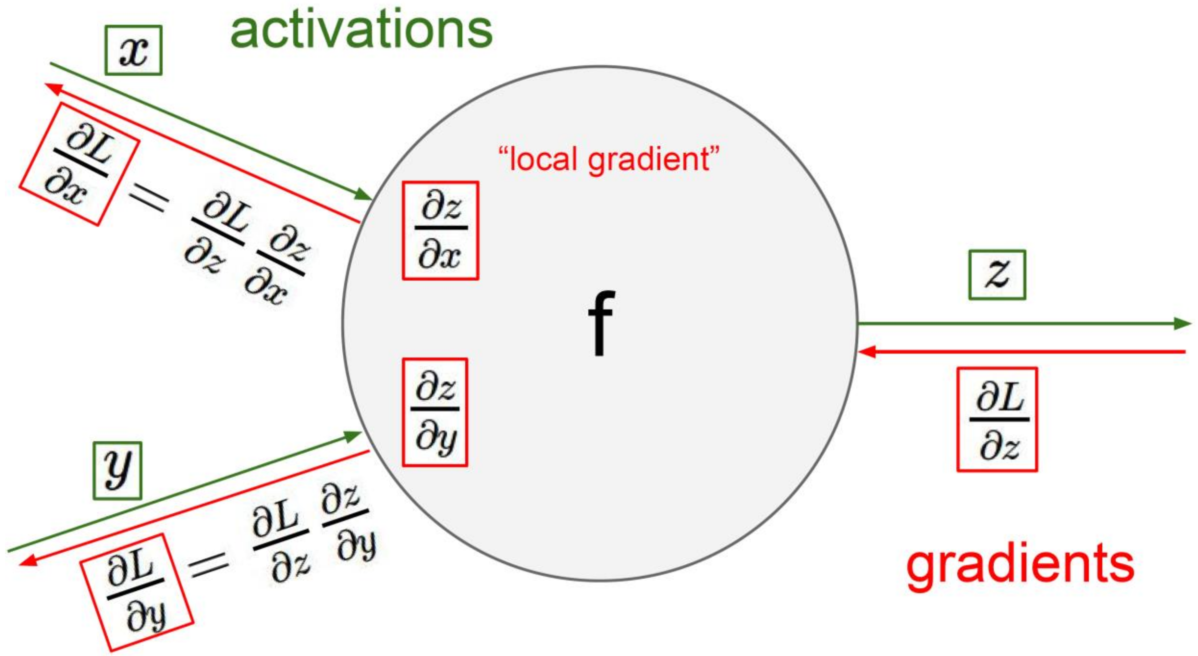
- Let $x + y = q$ and $f(x,y,z) = qz$
- $\dfrac{\partial f}{\partial x} = \dfrac{\partial f}{\partial q}\dfrac{\partial q}{\partial x}, \ \dfrac{\partial f}{\partial y} = \dfrac{\partial f}{\partial q}\dfrac{\partial q}{\partial y}$

Local property of backpropagation

source: http://cs231n.stanford.edu/syllabus.html

Picture for explaining local property

- Backpropagation is a local process!



- Backpropagation can be thought of as gates communicating to each other

高麗大學校 DS·BA Data Science & Business Analytics

Local property of backpropagation

source: http://cs231n.stanford.edu/syllabus.html

e. g. Sigmoid function

- $$f(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}}} = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

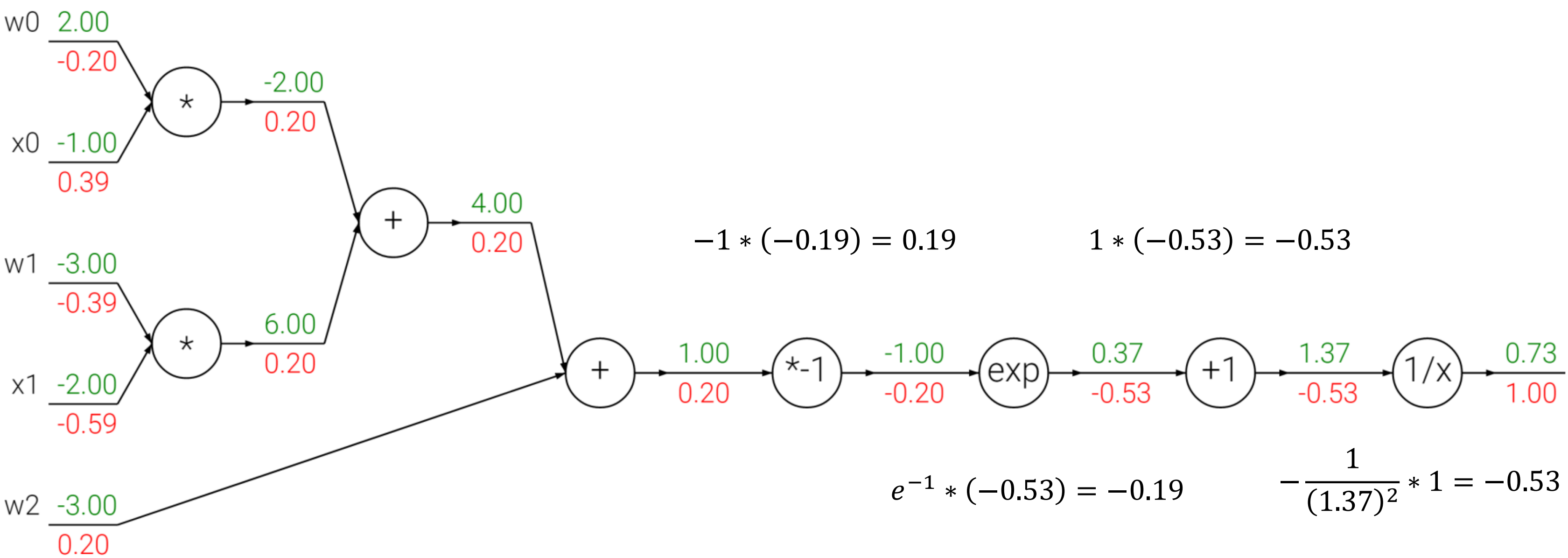
Local gradients of the sigmoid

- $f(x) = \frac{1}{x}$ -> $\frac{df}{dx} = -\frac{1}{x^2}$
- $f_c(x) = c + x$ -> $\frac{df}{dx} = 1$
- $f(x) = e^x$ -> $\frac{df}{dx} = e^x$
- $f_a(x) = ax$ -> $\frac{df}{dx} = a$

Local property of backpropagation

source: http://cs231n.stanford.edu/syllabus.html

Computational graph of the sigmoid



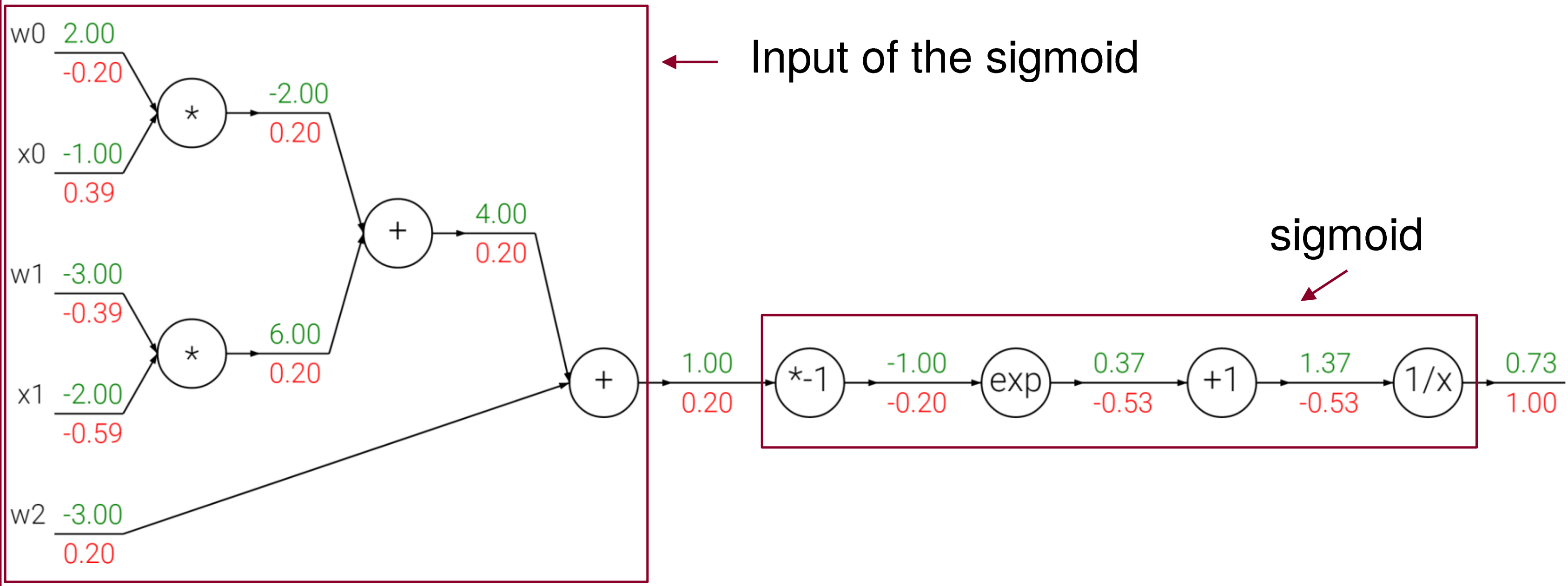
$$f(x) = \frac{1}{x} -> \frac{df}{dx} = -\frac{1}{x^2} \qquad f_c(x) = c + x -> \frac{df}{dx} = 1$$

$$f(x) = e^x -> \frac{df}{dx} = e^x \qquad f_a(x) = ax -> \frac{df}{dx} = a$$

Modularity

source: http://cs231n.stanford.edu/syllabus.html

Merge the graph



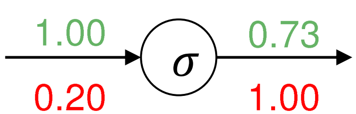
Sigmoid fn. $\sigma(x)$

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

Derivative of the sigmoid

- $\frac{d\sigma(x)}{dx} = \big(1 - \sigma(x)\big)\sigma(x)$

Sigmoid gate



$(1 - 0.73) * 0.73 * 1 = 0.2$

高麗大學校 DS·BA Data Science & Business Analytics

Patterns in backward flow

source: http://cs231n.stanford.edu/syllabus.html

Multiplication gate

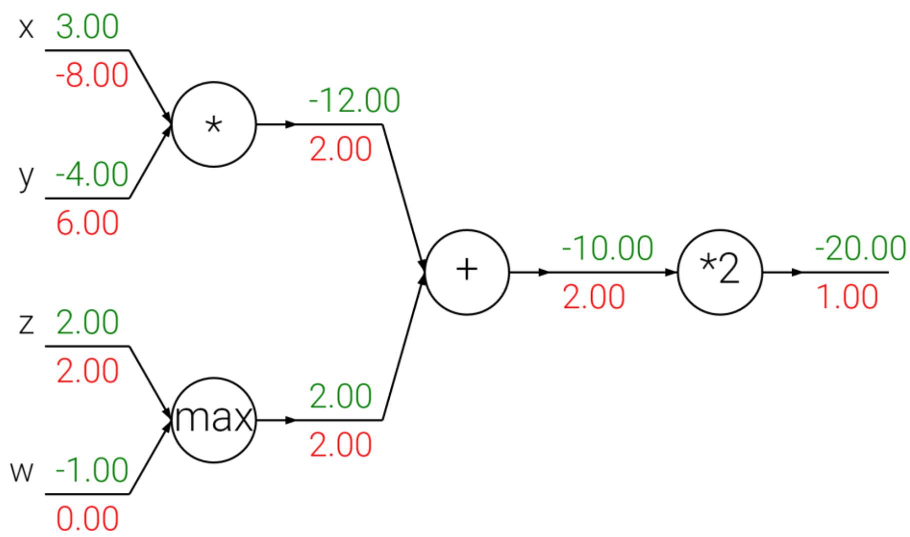
- Gradient swapper
  - $f(x, y) = xy$
    - $\dfrac{\partial f}{\partial x} = y, \ \dfrac{\partial f}{\partial y} = x$

Add gate

- Gradient distributor
  - $f(x, y) = x + y$
    - $\dfrac{\partial f}{\partial x} = 1, \ \dfrac{\partial f}{\partial y} = 1$

Max gate

- Gradient router
  - $f(x, y) = \max(x, y)$
    - $\dfrac{\partial f}{\partial x} = 1\ (x \geq y), 0\ o/w$
    - $\dfrac{\partial f}{\partial x} = 1\ (y \geq x), 0\ o/w$



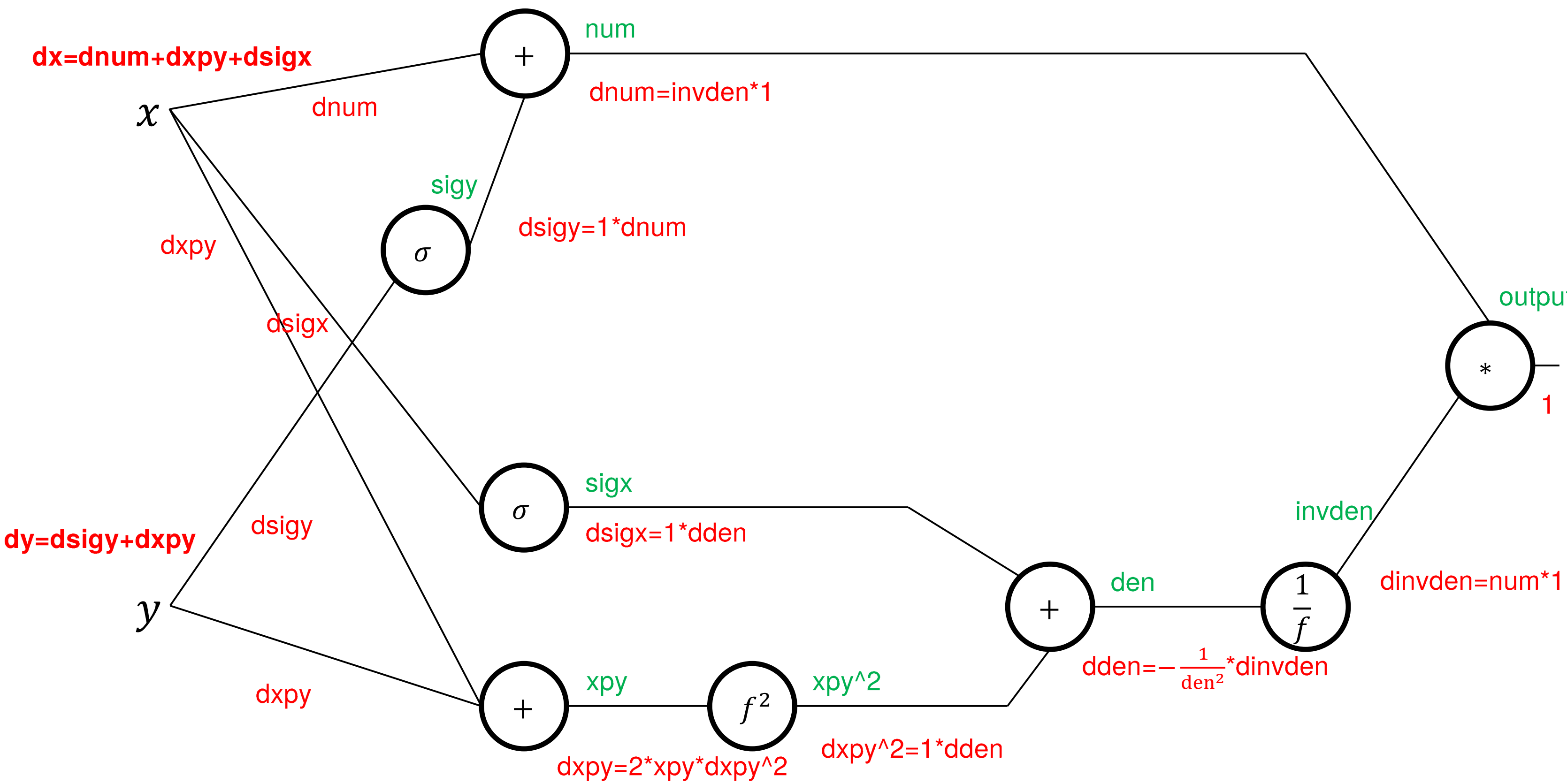
Another example

source: http://cs231n.stanford.edu/syllabus.html

Any function

- $f(x,y) = \dfrac{x+\sigma(y)}{\sigma(x)+(x+y)^2}$ (It's useless in practice)

Computational graph



dx, dy

- add up gradients at forks

Summary

source: http://cs231n.stanford.edu/syllabus.html

Problem

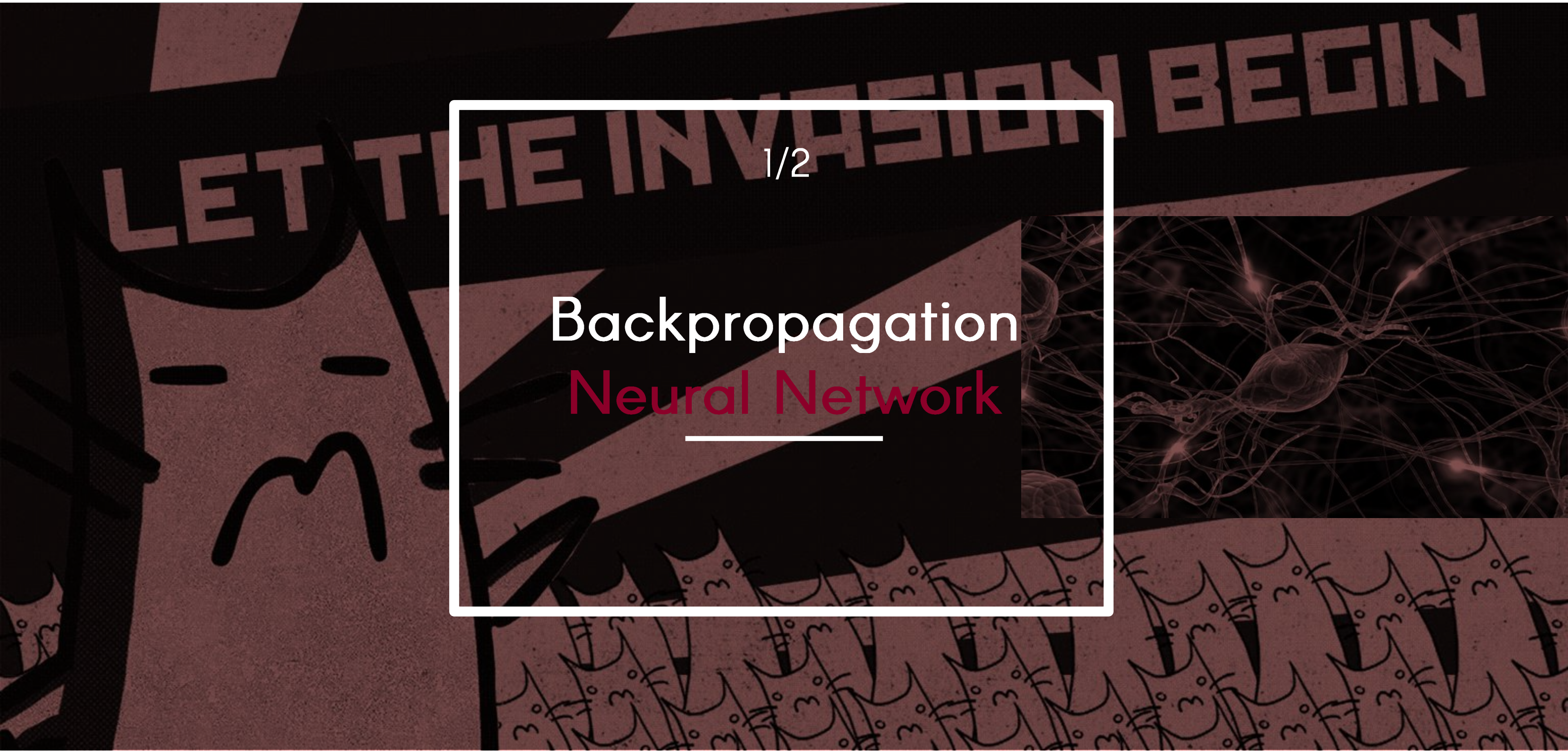
- Neural nets will be very large
  - Its gradient also will be very complex

Modularity

- Recursive application of the chain rule will save us!
  - Local property

Computational way

- It turns out that we don't need explicit expressions of gradients
  - Intermediate values
  - Staged computation



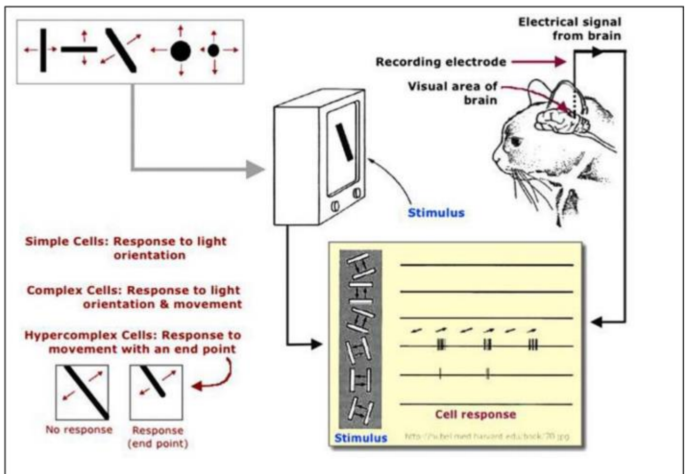
1/2

Backpropagation

Neural Network

- “The area of Neural Networks has originally been primarily inspired by the goal of modeling biological neural systems, but has since diverged and become a matter of engineering…” (*Andrej Karpathy*)

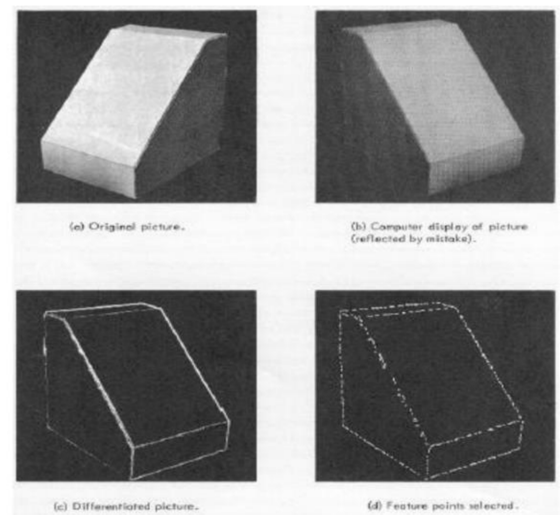
Revisit



Hubel & Wiesel, 1959

Block world

Larry Roberts, 1963



Parallel processing

- **Parallelism** is the main difference of the human brain from a computer

Parallel processing

SIMD

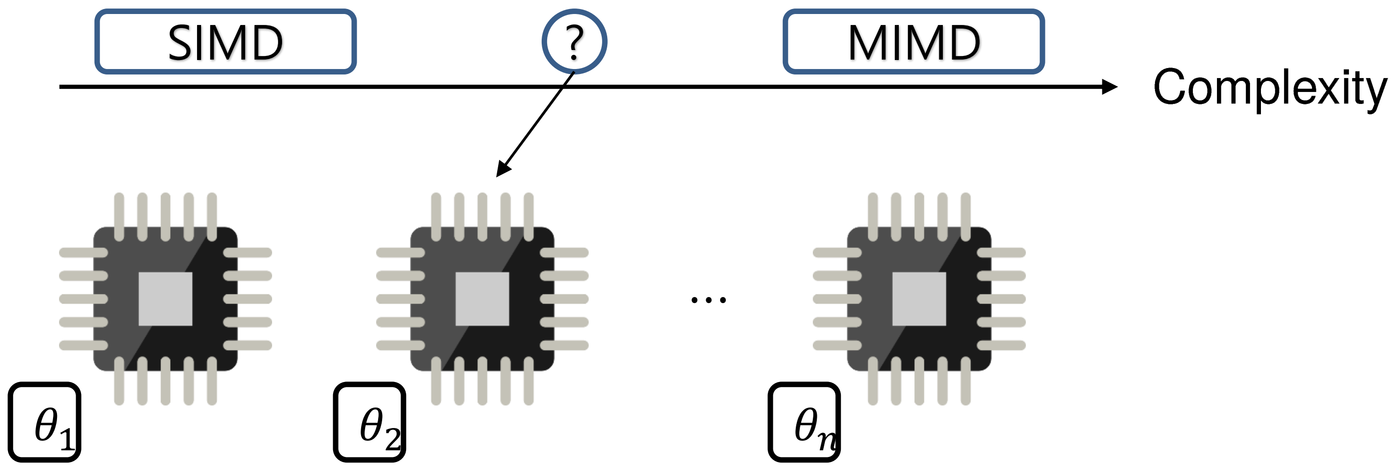
MIMD

- Two paradigms for *parallel processing*
- Single Instruction Multiple Data (SIMD)
  - All processors execute the same instruction but on different pieces of data
- Multiple Instruction Multiple Data (MIMD)
  - Different processors may execute different instructions on different data

Parallel processing

Alpaydin, E. (2014). Introduction to machine learning: MIT press.

Intermediate model



- Each processor is a fixed function and execute same instruction as SIMD

- By loading different parameters into a local memory, they can be doing different thing

Distributed representation

- The whole operation can be **distributed** over such processors

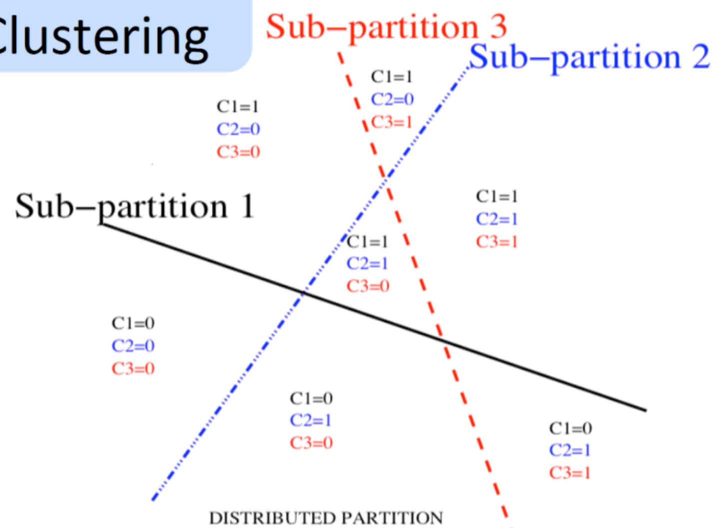
Bengio, B. (2015). Deep learning: Theoretical Motivations. Retrieved from http://videolectures.net/deeplearning2015_bengio_theoretical_motivations/

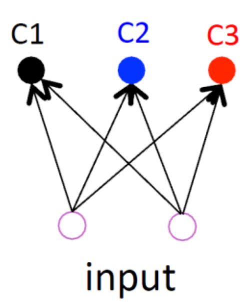
Curse of dim.

The need for dist. rep.

- Distributed representation allows us bypass the curse of dimensionality

Multi-Clustering





Non-mutually exclusive features/ attributes create a combinatorially large set of distinguiable configurations

# of distinguishable regions is linear in # of parameters

# of distinguishable regions grows almost exponentially with # of parameters

Generalize non-locally to never seen regions

Architecture design

Bengio, Y., Goodfellow, I. J., & Courville, A. (2015). Deep learning.

Architecture

- Depth and width

Universal approximation thm.

- A feedforward network with a single layer can approximate any Borel measurable function from one finite-dim. space to another, provided that the network is given enough hidden units

(Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, *2*(5), 359-366.)

Problems

1. Doesn't know appropriate parameters (Number of hidden units)

2. Overfitting

Practical solution

- Sufficient capability & Regularization

Q&A