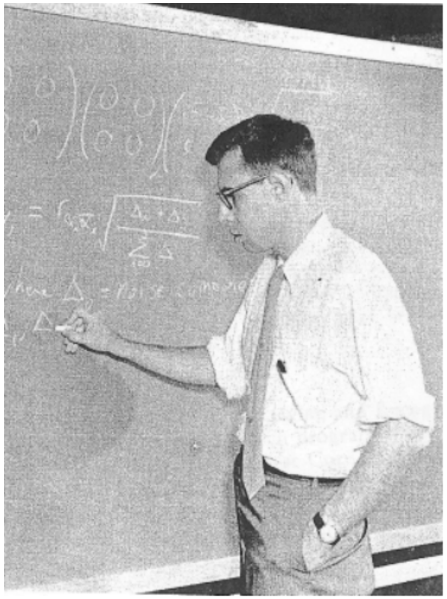
(Artificial) Neural Networks

6.036 Introduction to Machine Learning

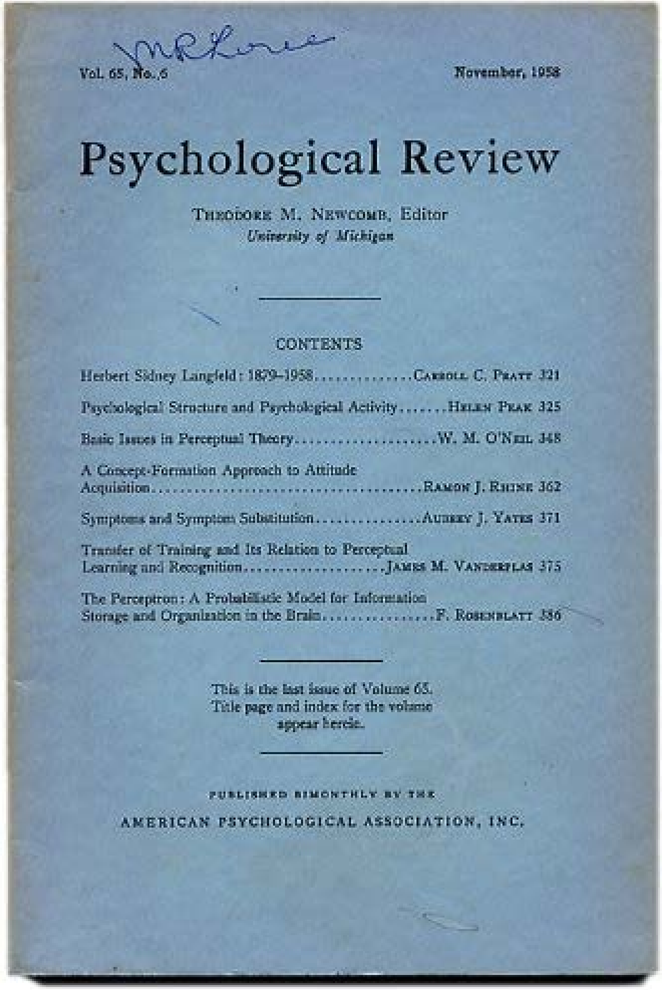
Review

- Linear models/linear regression
  - Can be fit reliably (convex optimization)
  - Model capacity is limited to linear functions
- Non-linear kernel
  - $\phi(x)$ – (nonlinear) feature map
  - Apply linear model to a transformed input $\phi(x)$
  - Yields nonlinear decision boundaries for classification, or nonlinear functions for regression
- Question: How to choose ϕ ?
  - Use generic ϕ
  - Manually engineer ϕ
  - Learn ϕ
    - We give up the convexity of the training
    - We gain an increased model capacity

Perceptrons, 1958



http://www.ecse.rpi.edu/homepages/nagy/PDF_chrono/2011_Nagy_Pace_FR.pdf. Photo by George Nagy



Vol. 65, No. 6 November, 1958

Psychological Review

Theodore M. Newcomb, Editor
University of Michigan

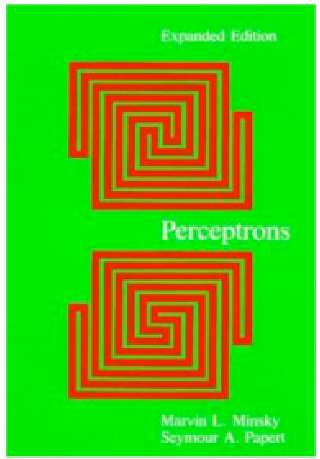
CONTENTS

This is the last issue of Volume 65.
Title page and index for the volume appear herein.

Published bimonthly by the
American Psychological Association, Inc.

http://www.manhattanrarebooks-science.com/rosenblatt.htm

Minsky and Papert, Perceptrons, 1972



Perceptrons, expanded edition

An Introduction to Computational Geometry

By Marvin Minsky and Seymour A. Papert

Overview

Perceptrons - the first systematic study of parallelism in computation - has remained a classical work on threshold automata networks for nearly two decades. It marked a historical turn in artificial intelligence, and it is required reading for anyone who wants to understand the connectionist counterrevolution that is going on today.

Artificial-intelligence research, which for a time concentrated on the programming of ton Neumann computers, is swinging back to the idea that intelligence might emerge from the activity of networks of neuronlike entities. Minsky and Papert's book was the first example of a mathematical analysis carried far enough to show the exact limitations of a class of computing machines that could seriously be considered as models of the brain. Now the new developments in mathematical tools, the recent interest of physicists in the theory of disordered matter, the new insights into and psychological models of how the brain works, and the evolution of fast computers that can simulate networks of automata have given *Perceptrons* new importance.

Witnessing the swing of the intellectual pendulum, Minsky and Papert have added a new chapter in which they discuss the current state of parallel computers, review developments since the appearance of the 1972 edition, and identify new research directions related to connectionism. They note a central theoretical challenge facing connectionism: the challenge to reach a deeper understanding of how "objects" or "agents" with individuality can emerge in a network. Progress in this area would link connectionism with what the authors have called "society theories of mind."

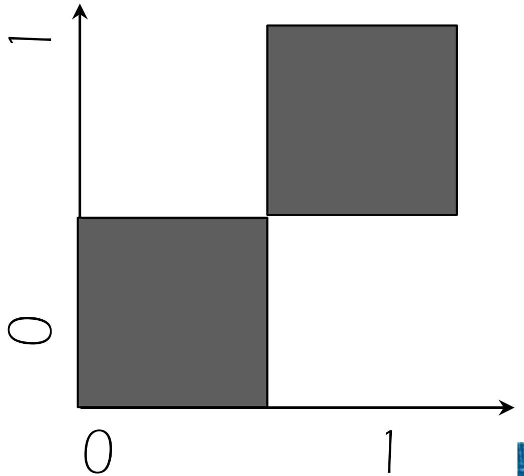
Paperback | $35.00 Short | £24.95 |
ISBN: 9780262631112 | 308 pp. | 6 x 8.9 in | December 1987

Perceptrons

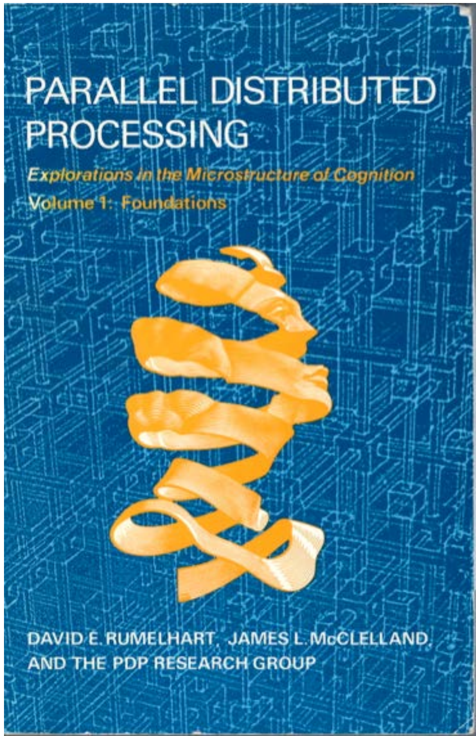
Minsky and Papert

Parallel Distributed Processing, 1986

Inputs		Output
0	0	0
1	0	1
0	1	1
1	1	0



PDP authors pointed to the backpropagation algorithm as a breakthrough, allowing multi-layer neural networks to be trained. Among the functions that a multi-layer network can represent but a single-layer network cannot: the XOR function.



LeCun conv nets, 1998

PROC. OF THE IEEE, NOVEMBER 1998 7

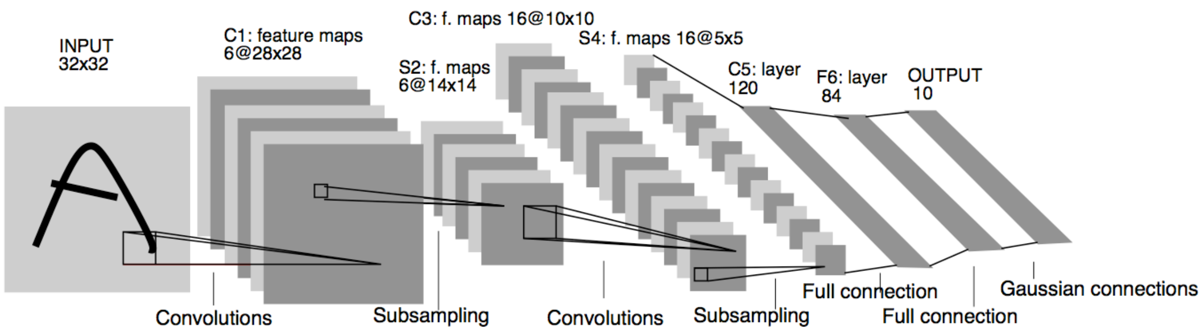


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

PROC. OF THE IEEE, NOVEMBER 1998 17

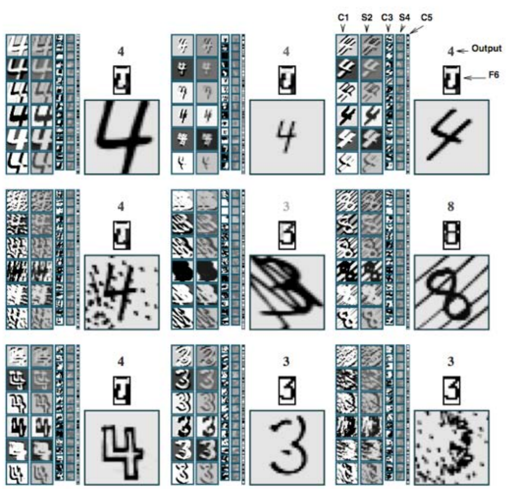


Fig. 13. Examples of unusual, distorted, and noisy characters correctly recognized by LeNet-5. The grey-level of the output label represents the penalty (lighter for higher penalties).

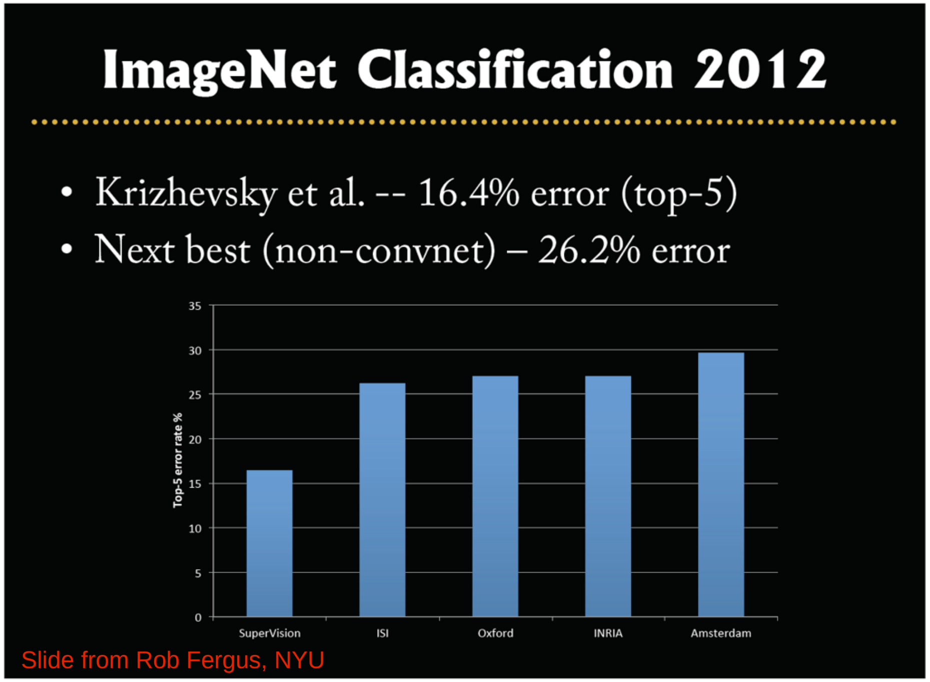
Neural networks to recognize handwritten digits? yes

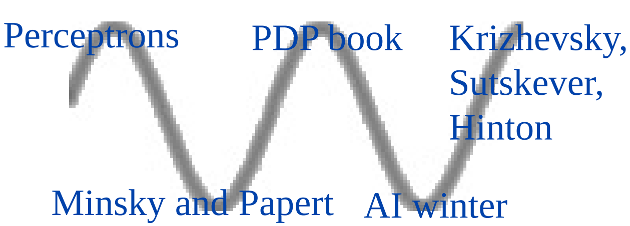
Neural networks for tougher problems? not really

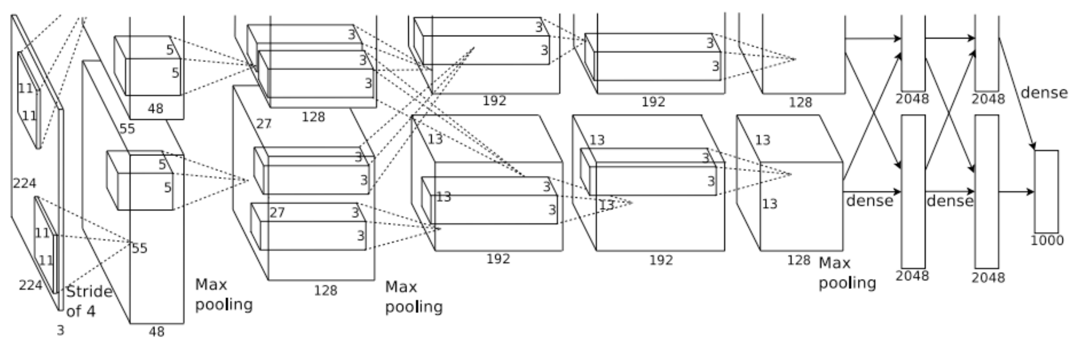
NIPS 2000

- NIPS, Neural Information Processing Systems, is the premier conference on machine learning. Evolved from an interdisciplinary conference to a machine learning conference.
- For the NIPS 2000 conference:
  - <u>title words predictive of paper acceptance</u>: “Belief Propagation” and “Gaussian”.
  - <u>title words predictive of paper rejection</u>: “Neural” and “Network”.

Deep Learning







Krizhevsky, Sutskever, and Hinton, NIPS 2012

Why do we care?

- Self-driving cars

e.g.

(Mobileye)

(Neural Networks)

Why do we care?

- Self-driving cars

e.g. 

(Mobileye)

(Neural Networks)

- Dialogue systems

e.g. 

(Neural Networks)

Why do we care?

- Self-driving cars

  e.g. 

  (Mobileye)

  (Neural Networks)

- Dialogue systems

  e.g. 

  (Neural Networks)

- Image understanding

  e.g., $\boldsymbol{h}\Bigg($  $\Bigg) =$ A group of people shopping at an outdoor market

  (Neural Networks)

Why now?

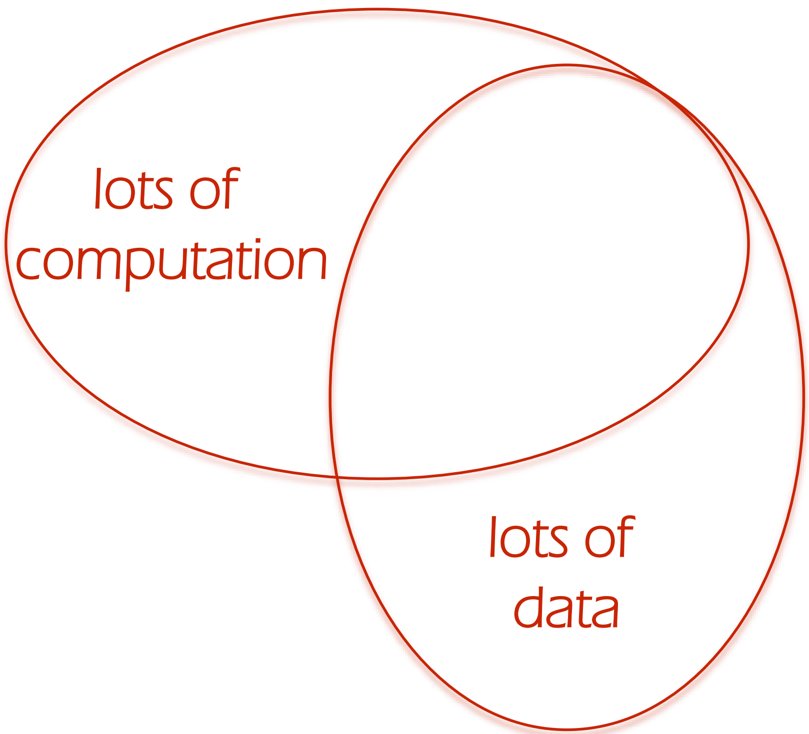
- Building blocks are similar to those already introduced decades ago… what is different now?

Why now?

- Building blocks are similar to those already introduced decades ago… what is different now?

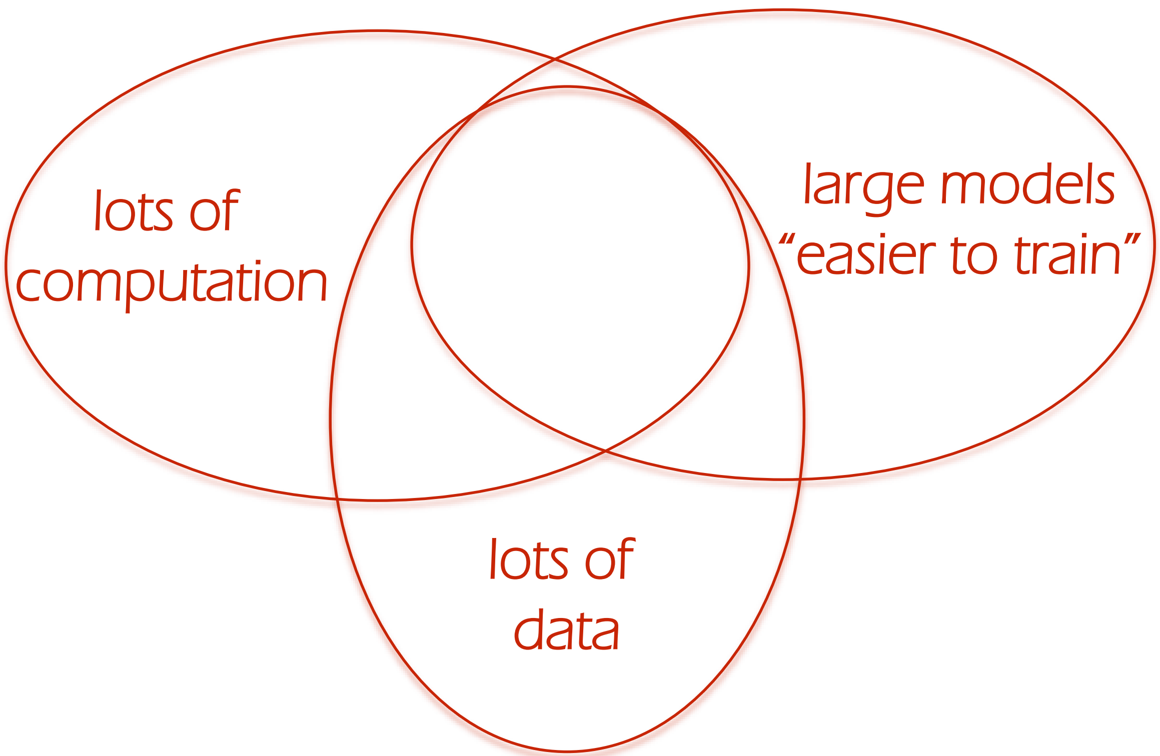
Why now?

- Building blocks are similar to those already introduced decades ago… what is different now?

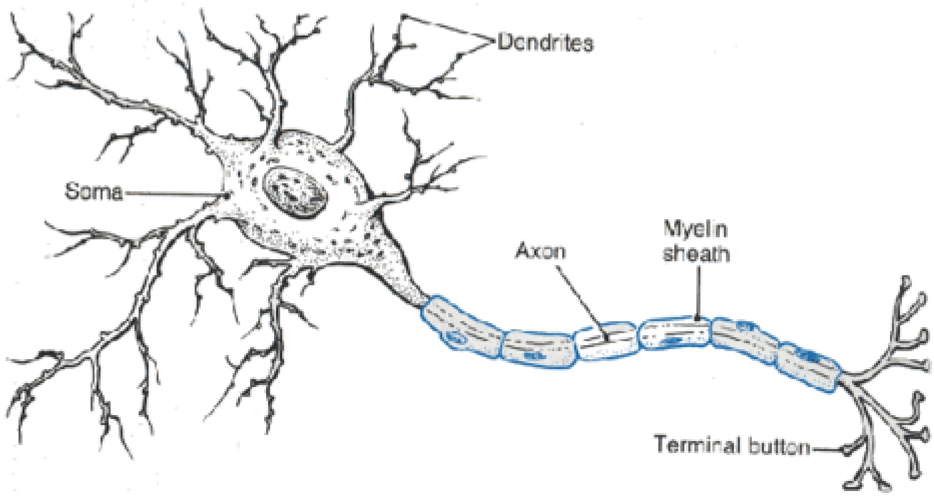


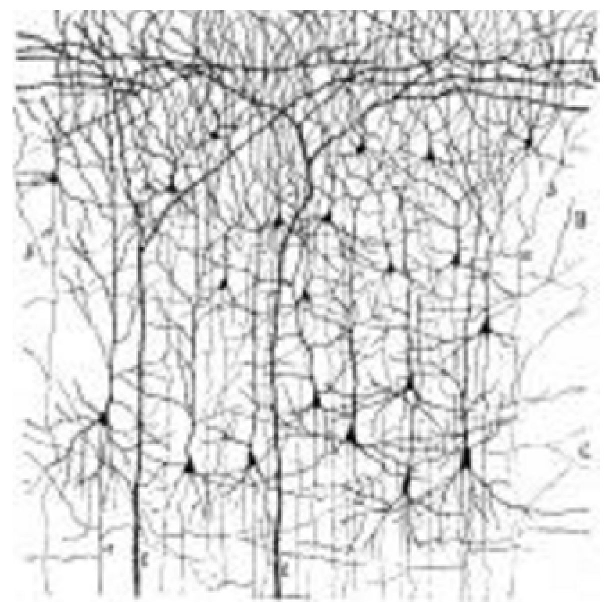
Why now?

- Building blocks are similar to those already introduced decades ago… what is different now?

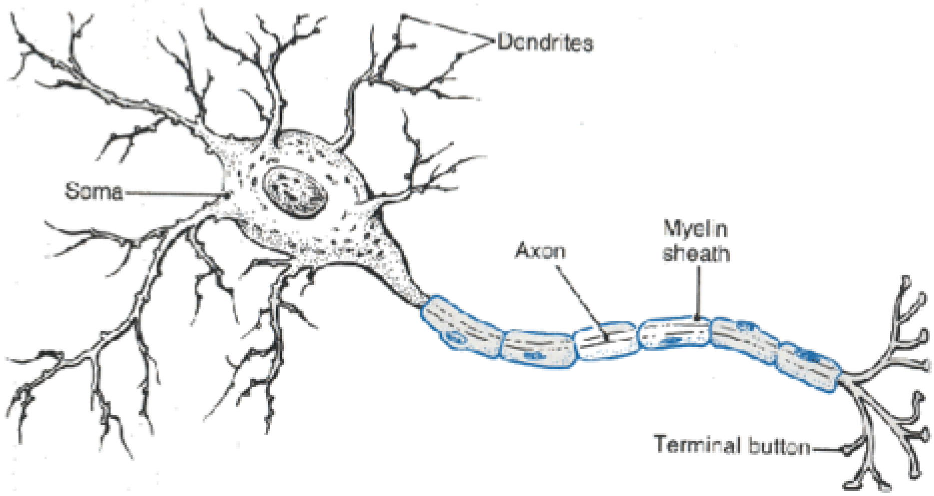


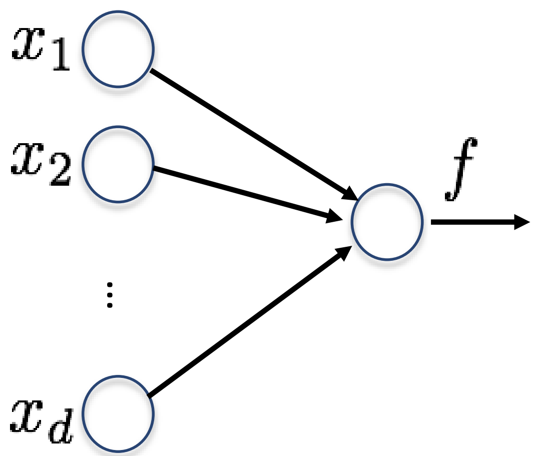
Neural Networks





(Artificial) Neural Networks

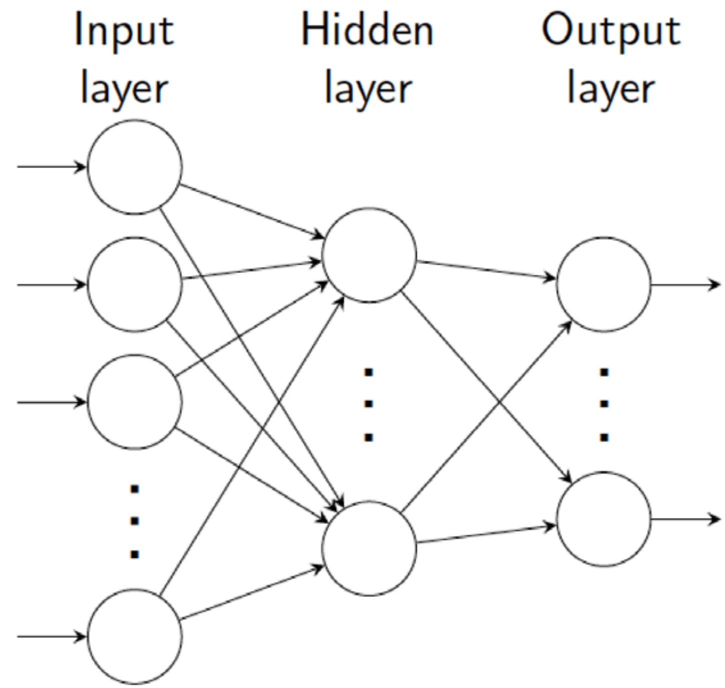




(e.g., a linear classifier)

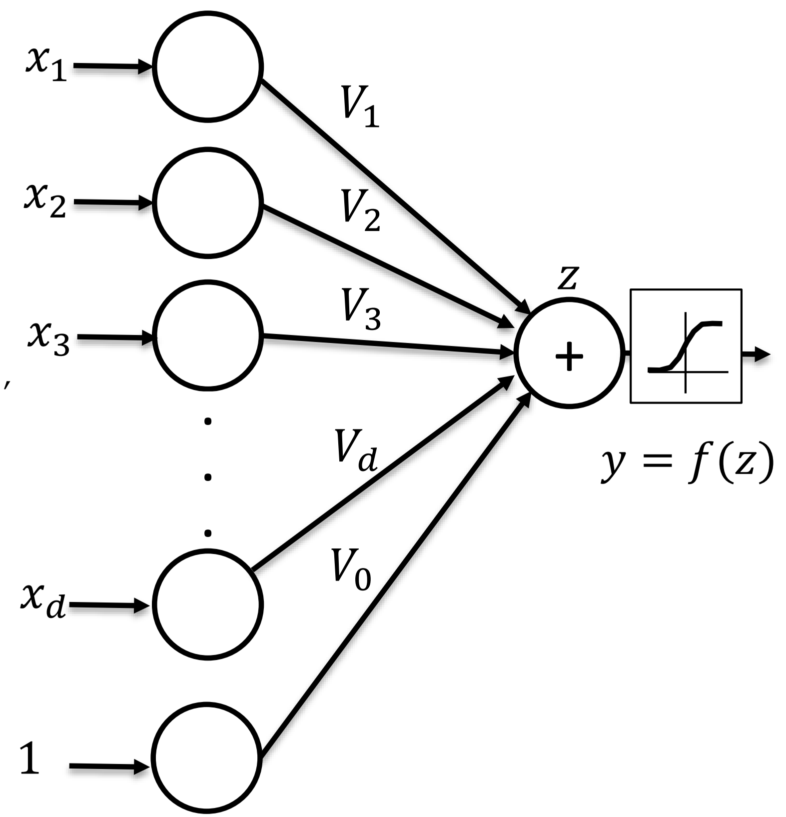
Neural Networks

- Composed of a sequence of layers
- Each layer contains artificial neurons
- Each layer computes some function of the previous layer
- Inputs mapped in a feed-forward fashion to output
- For now, feed-forward model (no cycles)



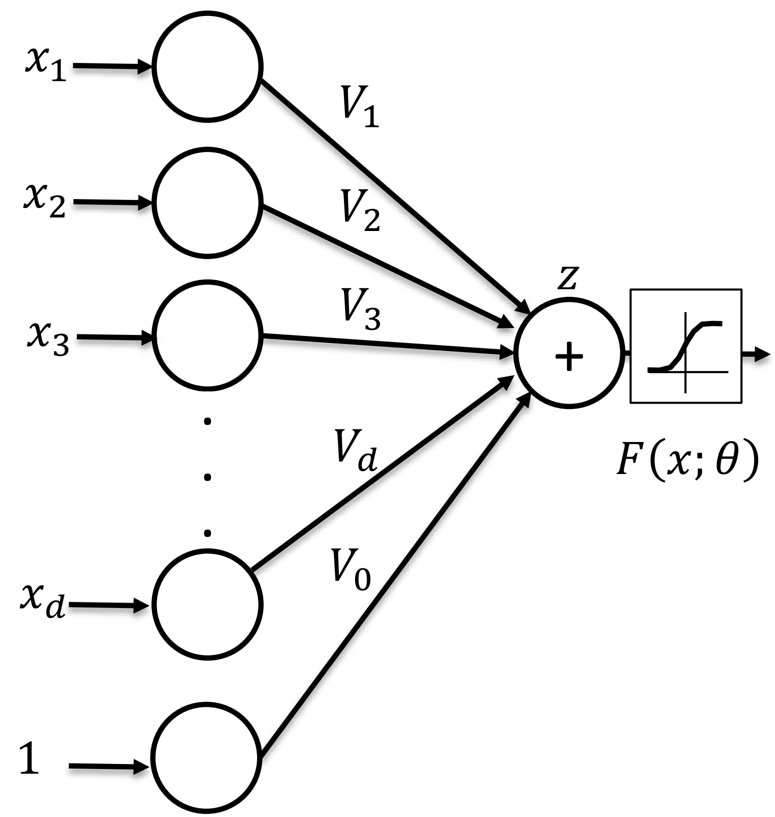
An Individual Neuron

- Input: vector $\boldsymbol{x}$ (size d×1)
- Unit parameters: $\boldsymbol{\theta} = \{V, V_0\}$
  - weights $\boldsymbol{V_i}$(size d×1)
  - bias $\boldsymbol{V_0}$
- Unit activation: $z = \sum_{i=1}^{d} x_i V_i + V_0$
  - You can think of a bias $\boldsymbol{V_0}$ as weight $\boldsymbol{V_0}$, connected to a constant input 1
- Activation function: $f(z)$
  - e.g., $f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
- Output:$y = f(z)$



Simplest Neural Network

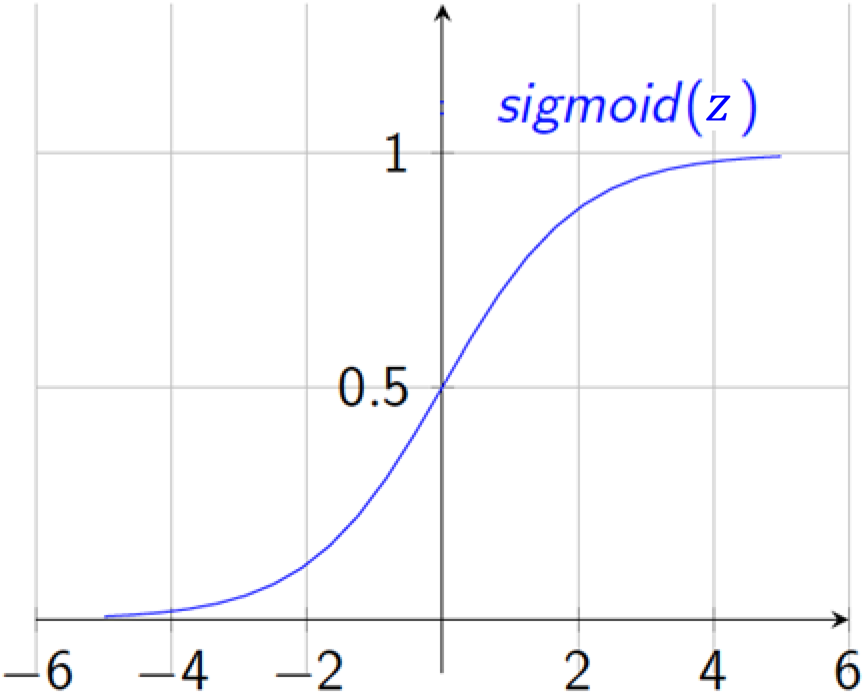
- A linear classifier
- Input: vector $\boldsymbol{x}$ (size d× 1)
- Layer parameters: $\boldsymbol{\theta} = \{\boldsymbol{V}, \boldsymbol{V_0}\}$
  - weights $\boldsymbol{V_i}$(size d× 1)
  - bias $\boldsymbol{V_0}$
- $z = \sum_{i=1}^{d} x_i V_i + V_0 = x \cdot V + V_0$
- Activation function: $f(z) = z$
- Output: $F(x; \theta) = f(z) = z$



Non-linearities: sigmoid

$$f(z) = sigmoid(z) = \frac{1}{1+e^{-z}}$$

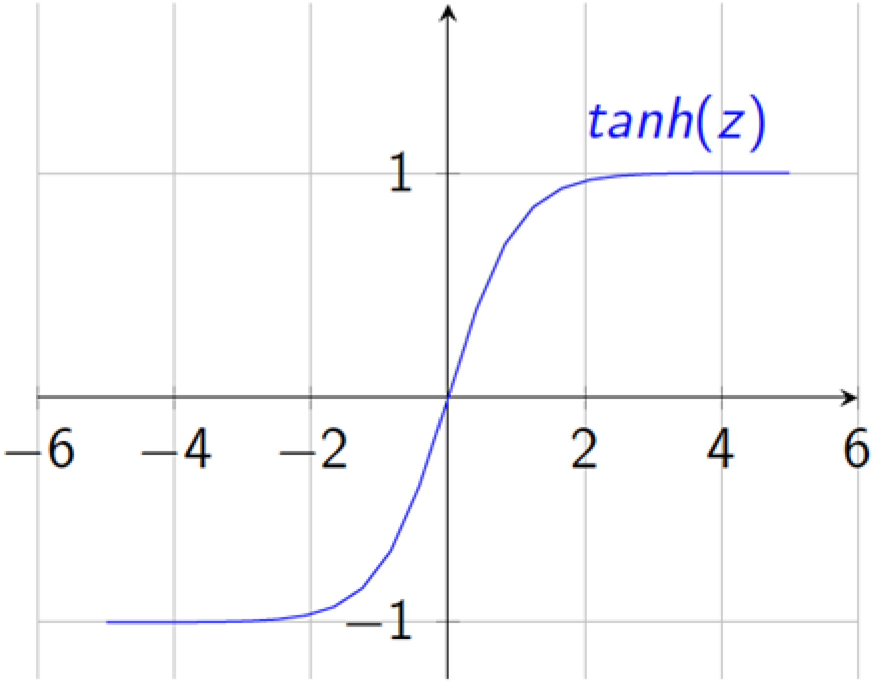
- Interpretation as a ring rate of neurons
- Bounded between [0,1]
- Saturation for large positive and negative inputs
- Gradients go to zero
- Outputs centered at 0.5
- Not used in practice



Non-linearities: tanh

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

- Bounded between [-1,+1]
- Saturation for large positive and negative inputs
- Gradients go to zero
- Outputs centered at 0
- Preferable to sigmoid
  $\tanh(z) = 2\mathrm{sigmoid}(2z) - 1$



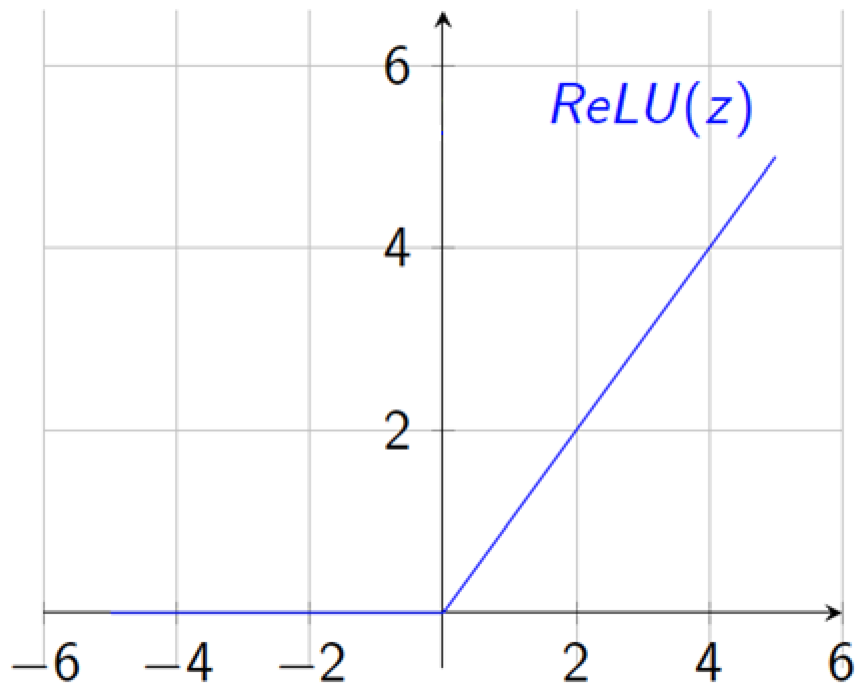
Non-linearities: Rectified Linear (ReLU)

- Unbounded output (on positive side)
- Efficient to implement:

$$f'(z) = \frac{df}{dz} = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}$$

- Also seems to help convergence
- Drawback: if strongly in negative region, unit is dead forever (no gradient).
- Default choice: widely used in current models

$$f(z) = \max(z, 0)$$



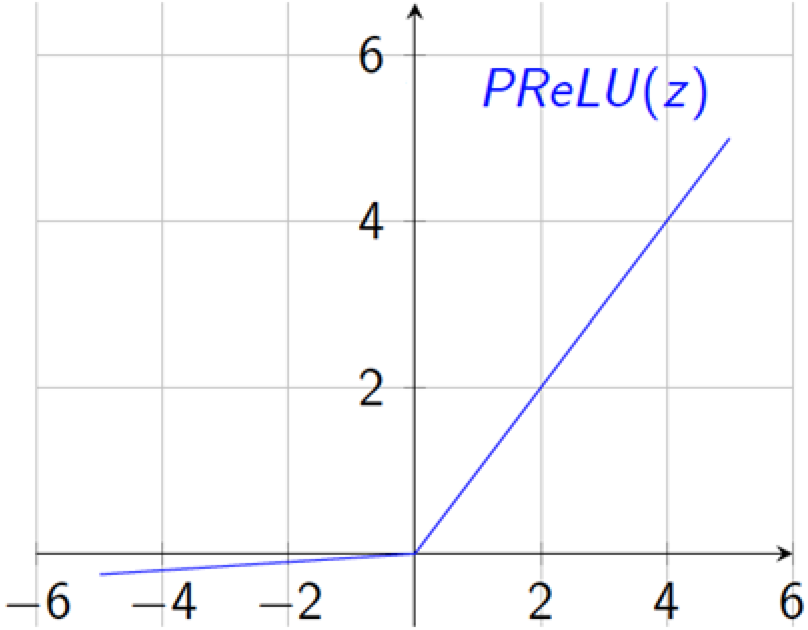
Non-linearities: Leaky ReLU

$$f(z) = \begin{cases} \max(0, z) & z > 0 \\ \alpha \min(0, z) & z < 0 \end{cases}$$

- α is small (e.g. 0.02)
- Efficient to implement:

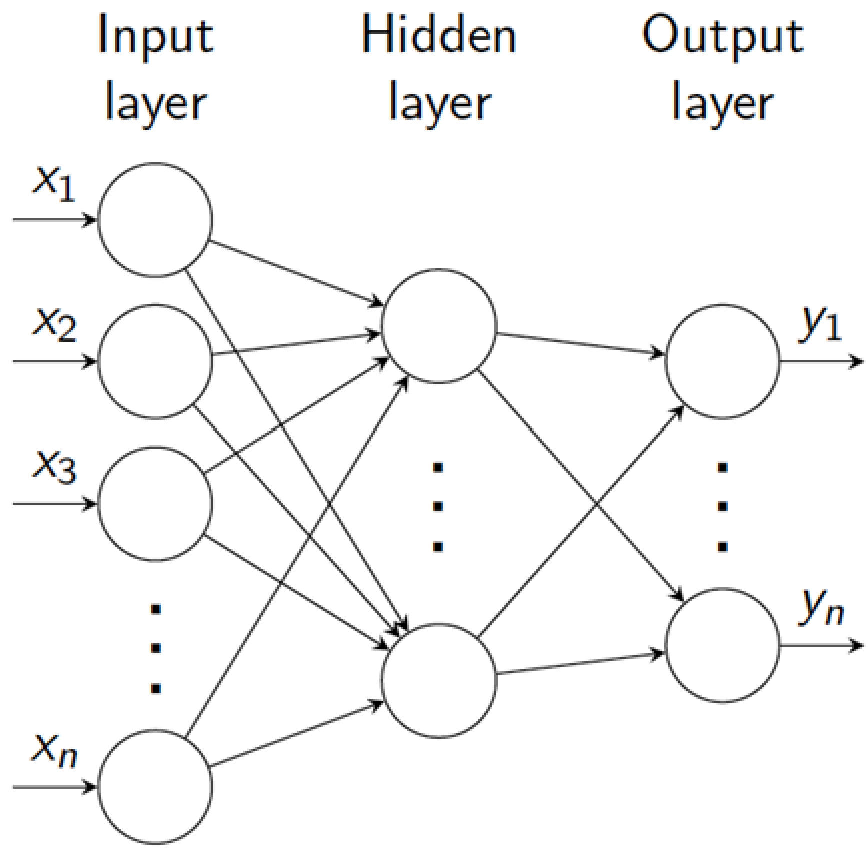
$$f'(z) = \frac{df}{dz} = \begin{cases} -\alpha & z < 0 \\ 1 & z > 0 \end{cases}$$

- Also known as parametric ReLU (PReLU)
- Has non-zero gradients everywhere (unlike ReLU)



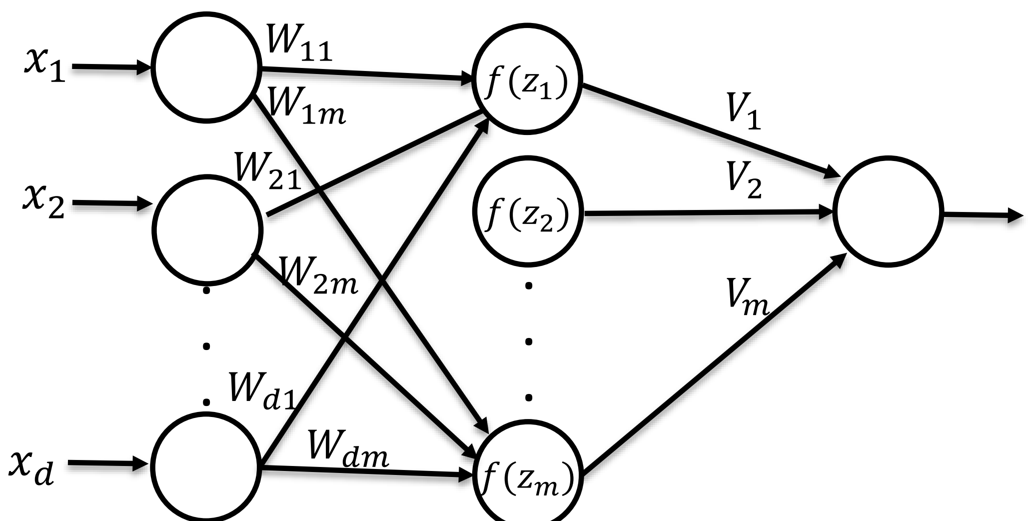
Multiple Layers

- Neural networks are composed of multiple layers of neurons.
- Acyclic structure. Basic model assumes full connections between layers.
- Layers between input and output are called hidden.
- Various names used:
  - Artificial Neural Net (ANN)
  - Multi-layer Perceptron (MLP)
  - Fully-connected network



Example: 2-Layer Neural Network

- By convention, # of layers is: # of hidden layers + output,
  - e.g., 2-layer model has 1 hidden layers.
- Parameters:
  - $\theta = \{W_{ij}, W_{0j}\}$ & $\{V_j, V_0\}$

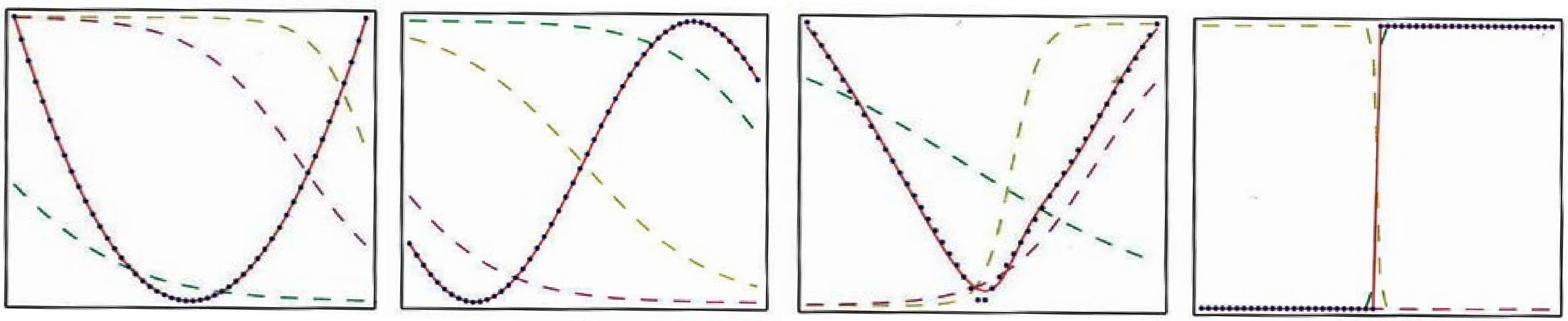


$$F(x;\theta) = z$$

$$z = \sum_{j=1}^{m} f(z_j) V_j + V_0$$

$$z_j = \sum_{i=1}^{d} x_i W_{ij} + W_{0j} \qquad f(z_j) = \max\{0, z_j\}$$

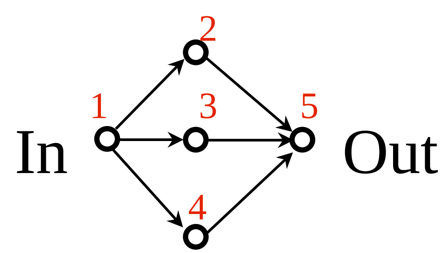
Representational Power of 2-layer Networks



$f(x) = x^2$ $\quad$ $f(x) = \sin(x)$ $\quad$ $f(x) = |x|$ $\quad$ $f(x) = H(x)$

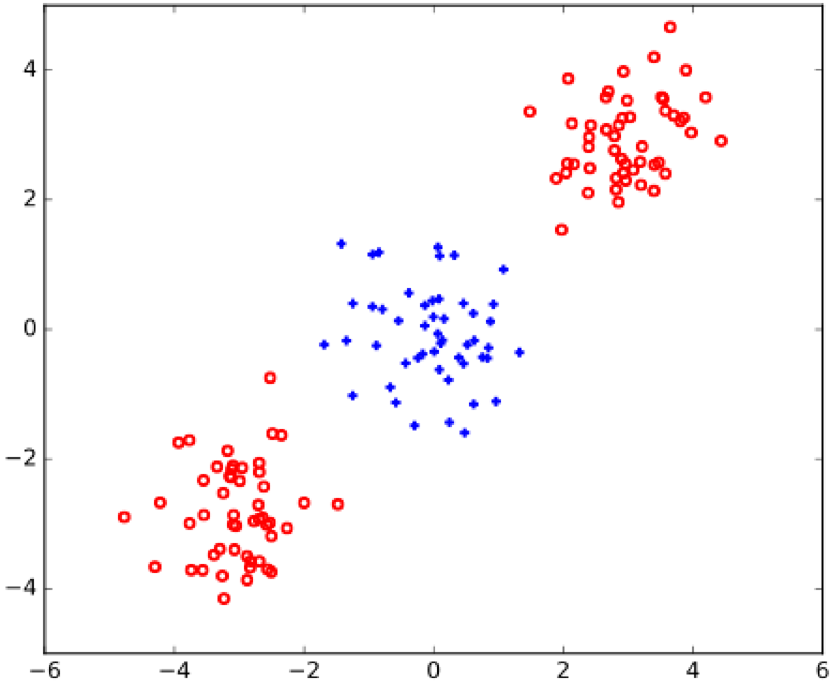
H $- step\ function$

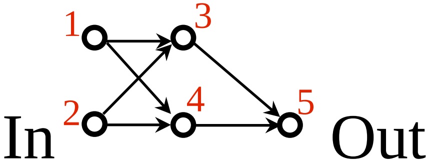
- Two-layer network
  - 1 input, 3 hidden units, 1output
- 50 training points (sampled uniformly)
- Result
  - Red curve (predicted value)
  - Dashed curves (hidden unit outputs)



$$z_5 = \sum_{i=2}^{i=4} w_{i5} \tanh(w_{1i} z_1 + \overset{\substack{\text{bias} \\ \downarrow}}{w_{0i}})$$

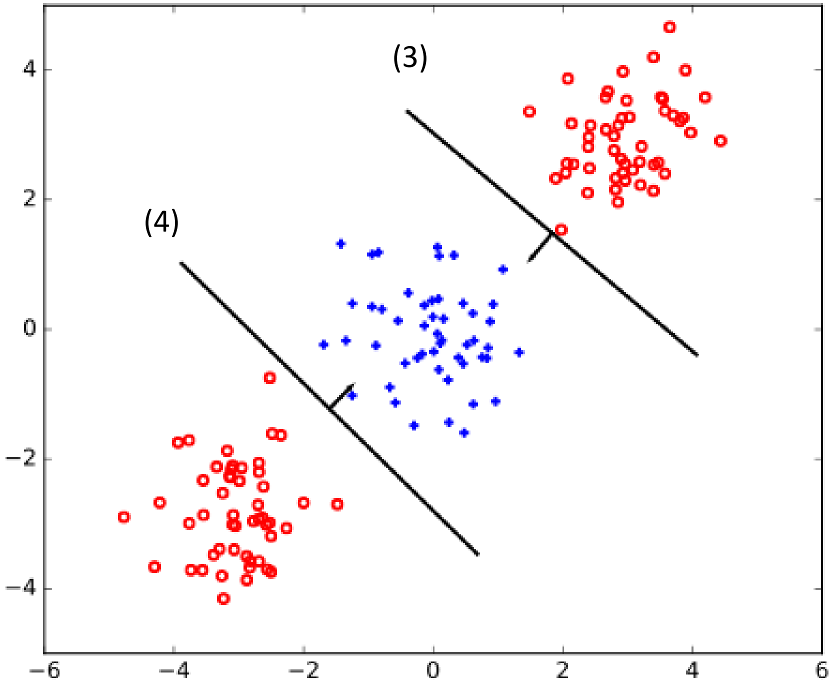
Example Problem

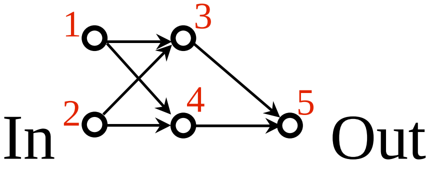




Hidden Layer Representation

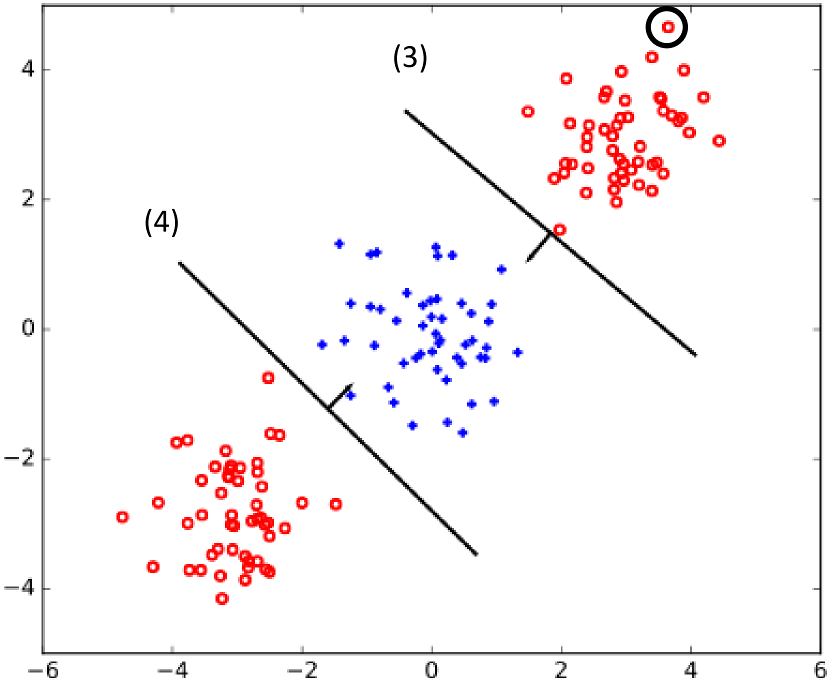
Hidden layer units



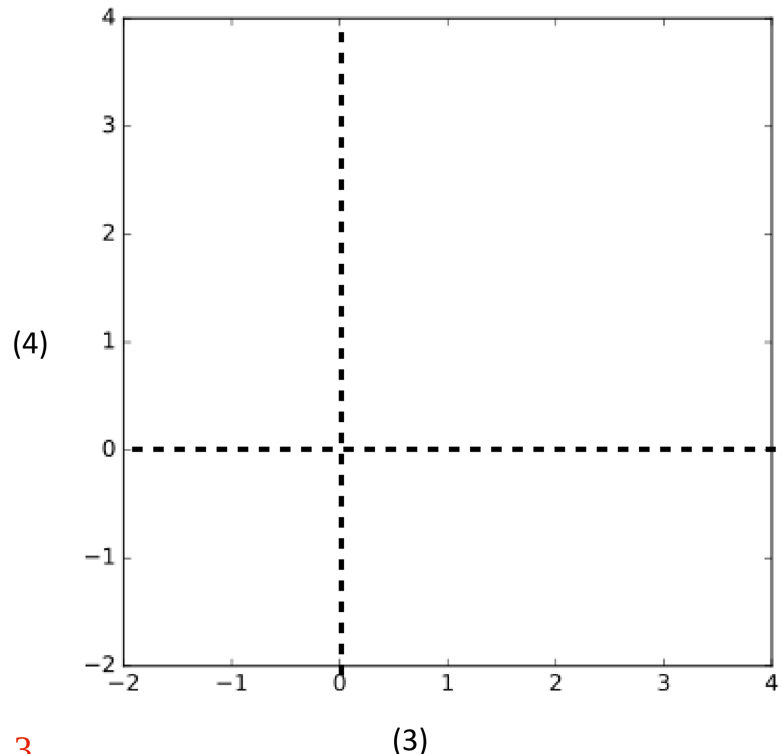


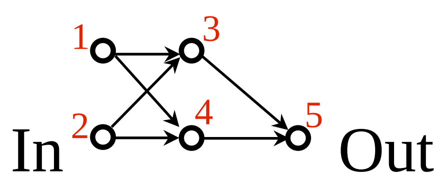
Hidden Layer Representation

Hidden layer units



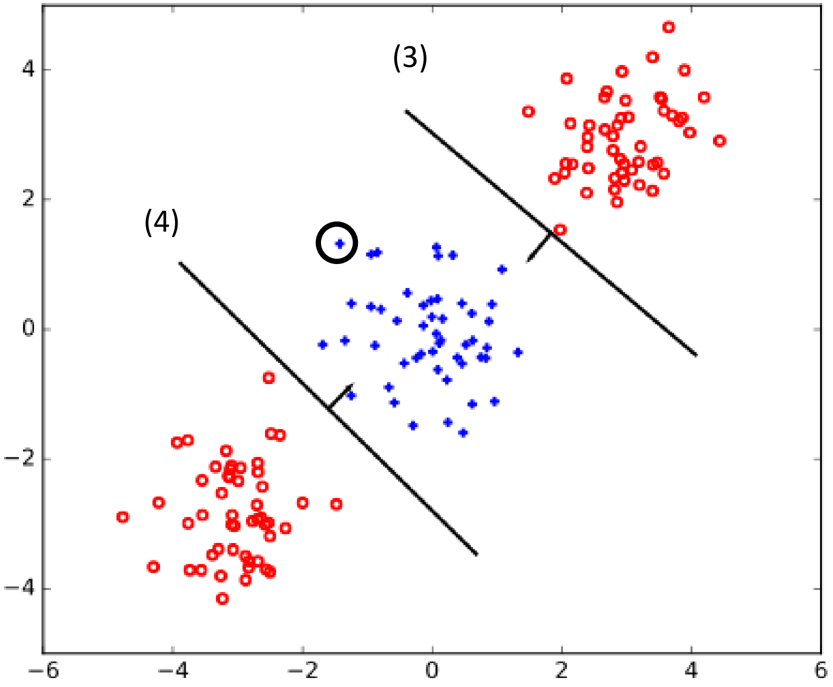
Linear activation



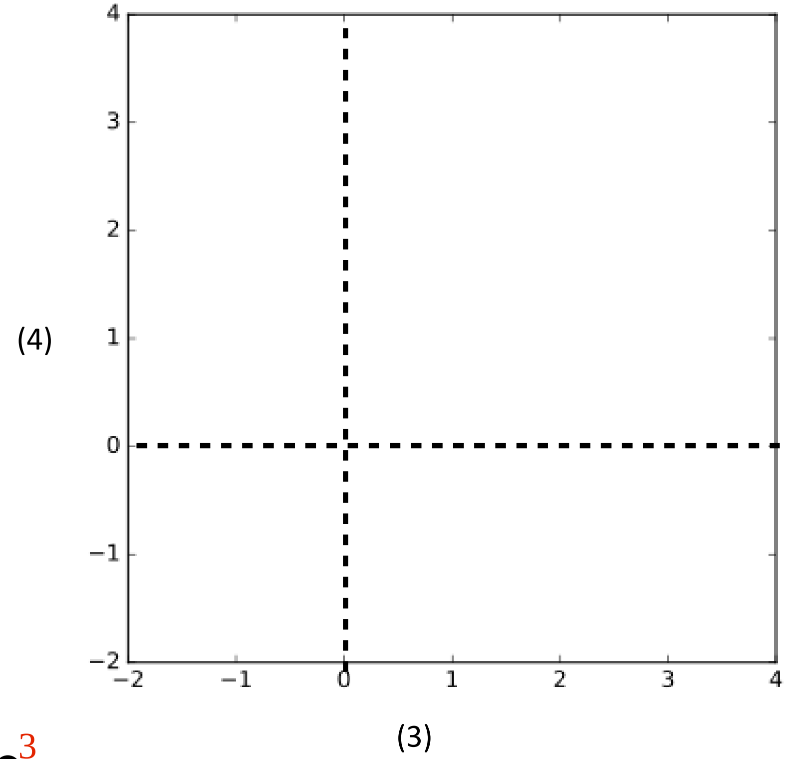


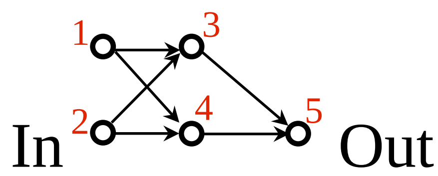
Hidden Layer Representation

Hidden layer units



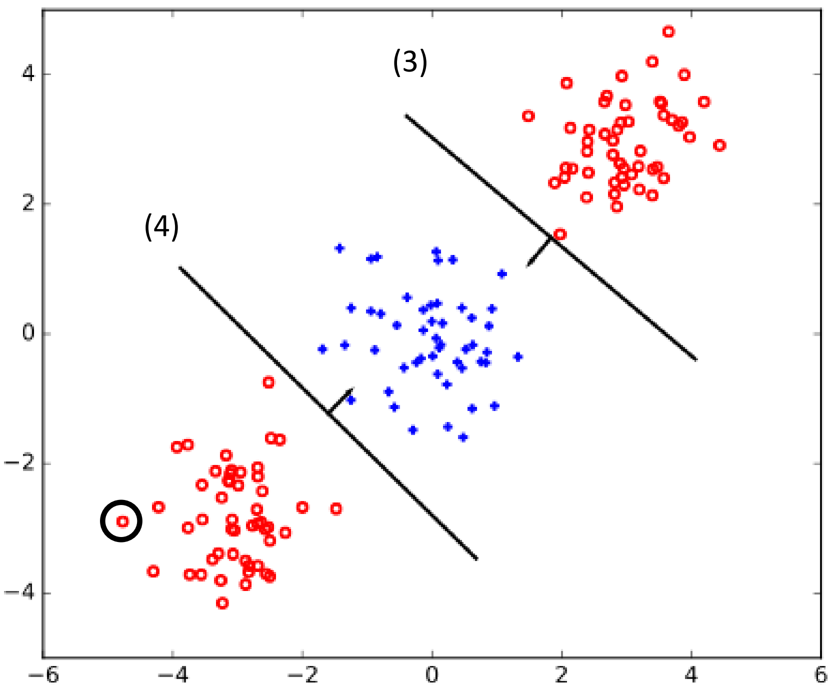
Linear activation



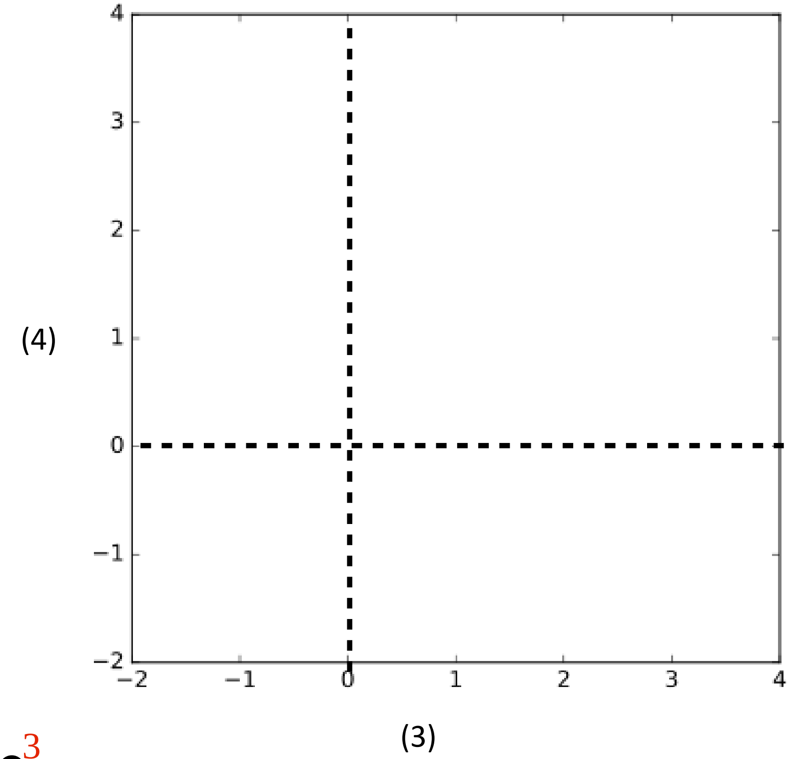


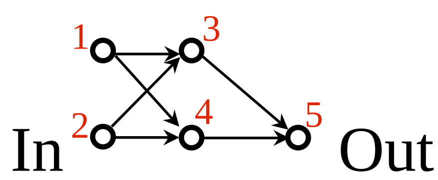
Hidden Layer Representation

Hidden layer units



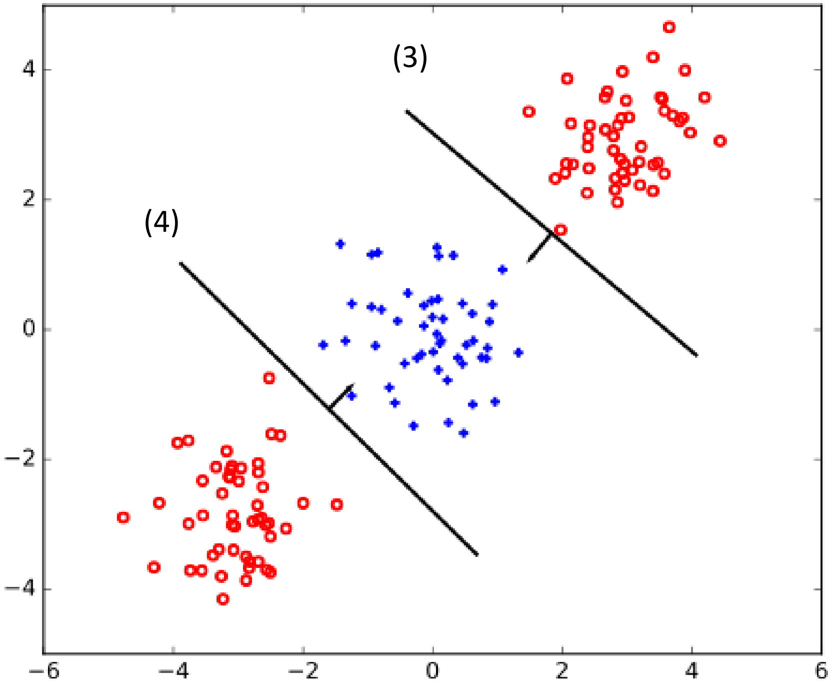
Linear activation



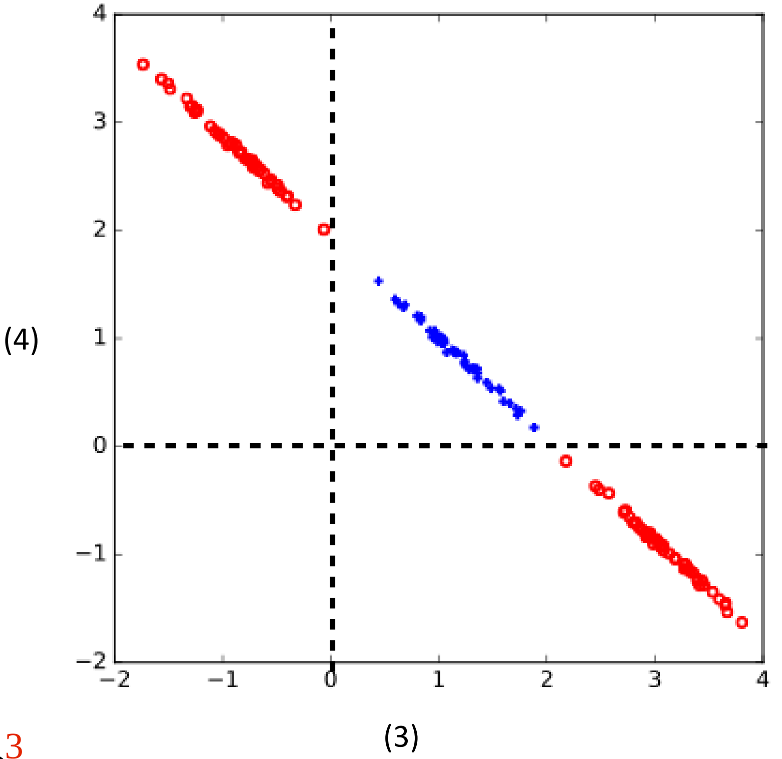


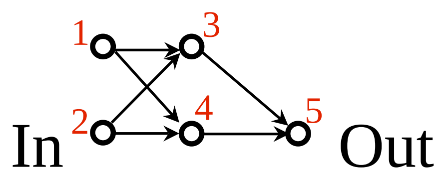
Hidden Layer Representation

Hidden layer units



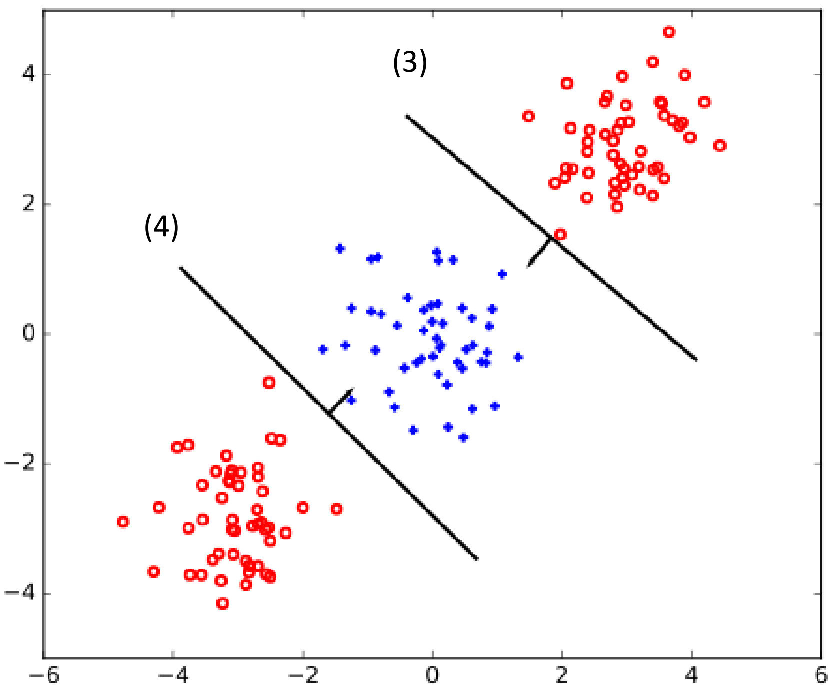
Linear activation



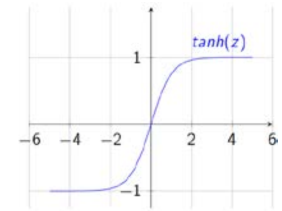


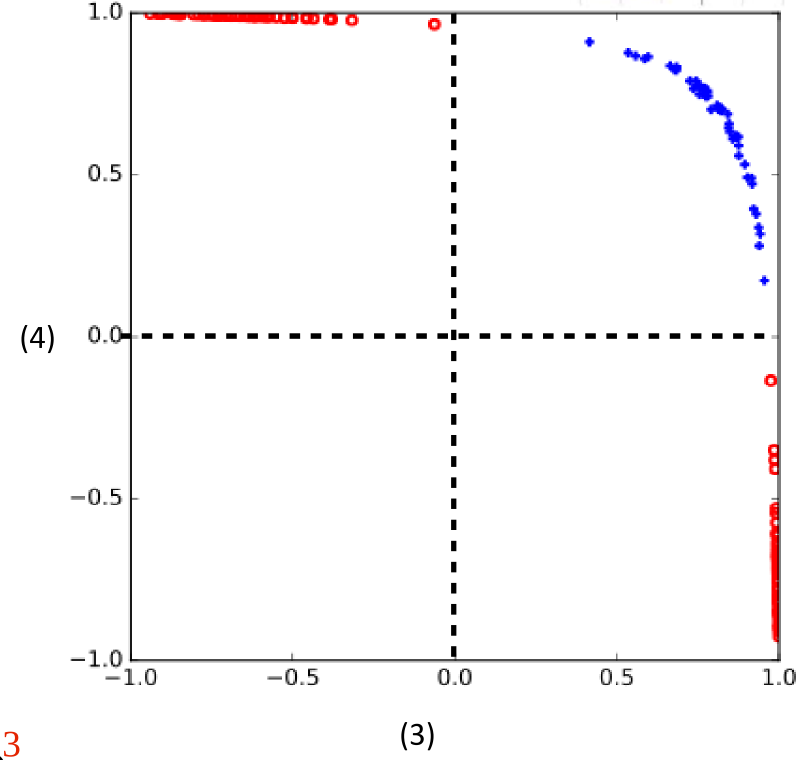
Hidden Layer Representation

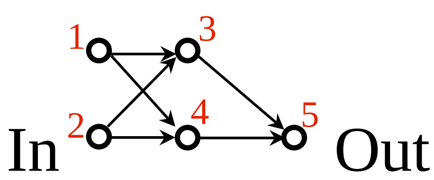
Hidden layer units



tanh activation

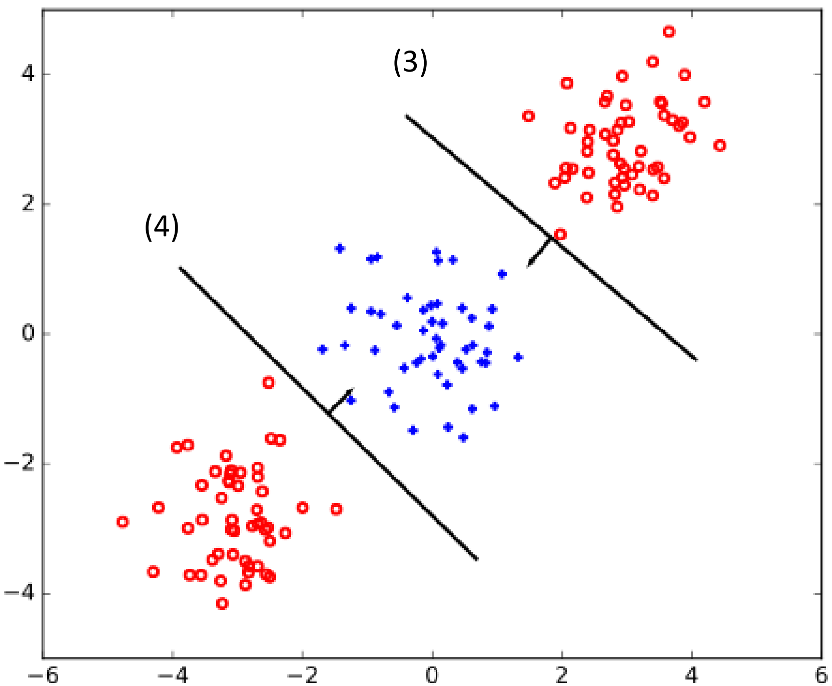




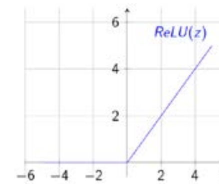


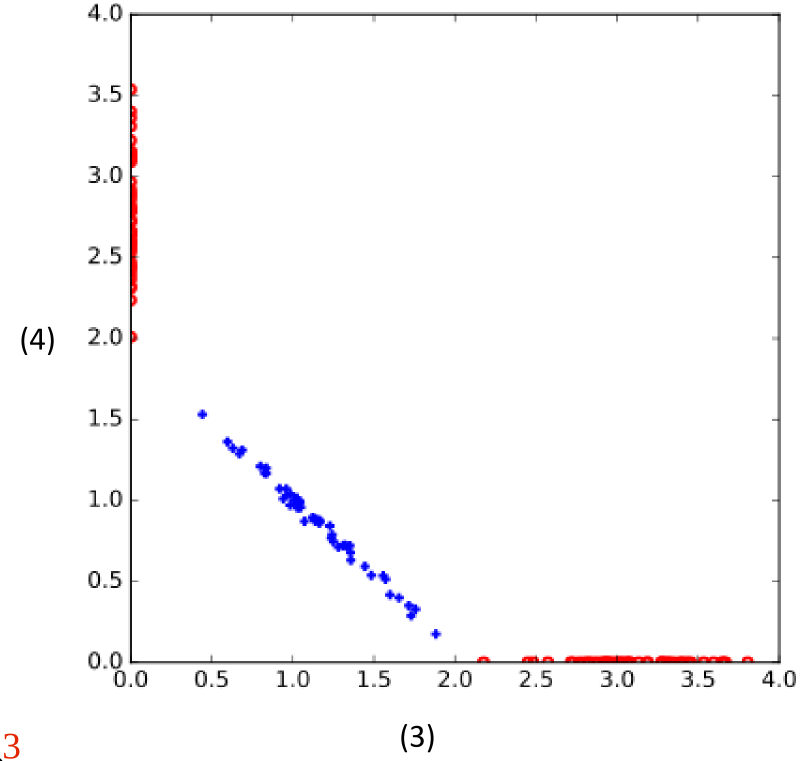
Hidden Layer Representation

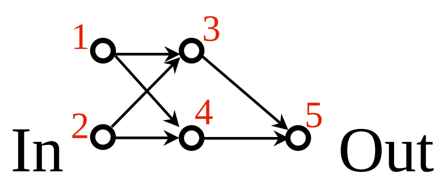
Hidden layer units



ReLU activation

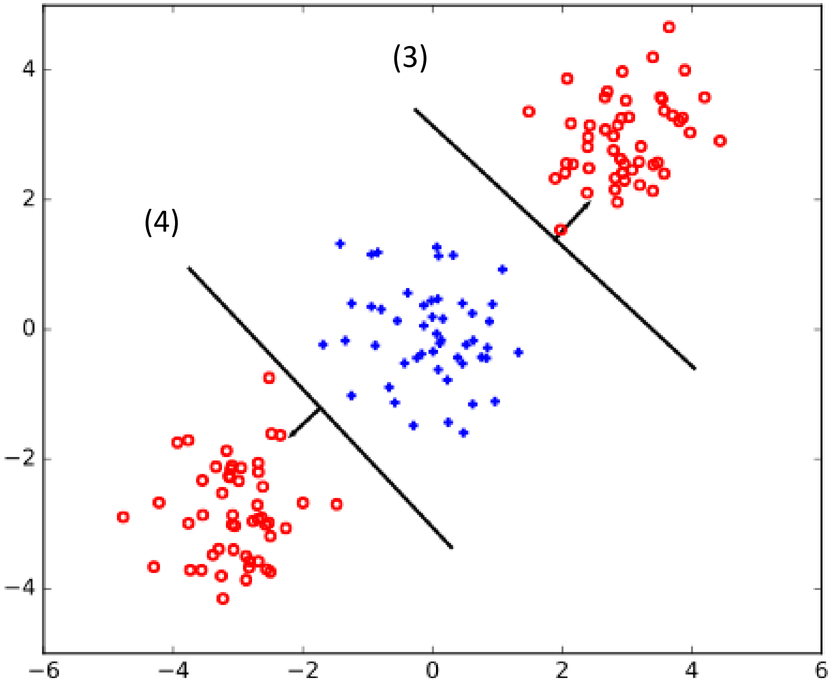


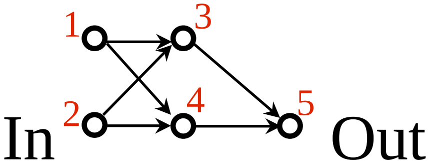




Does orientation matter?

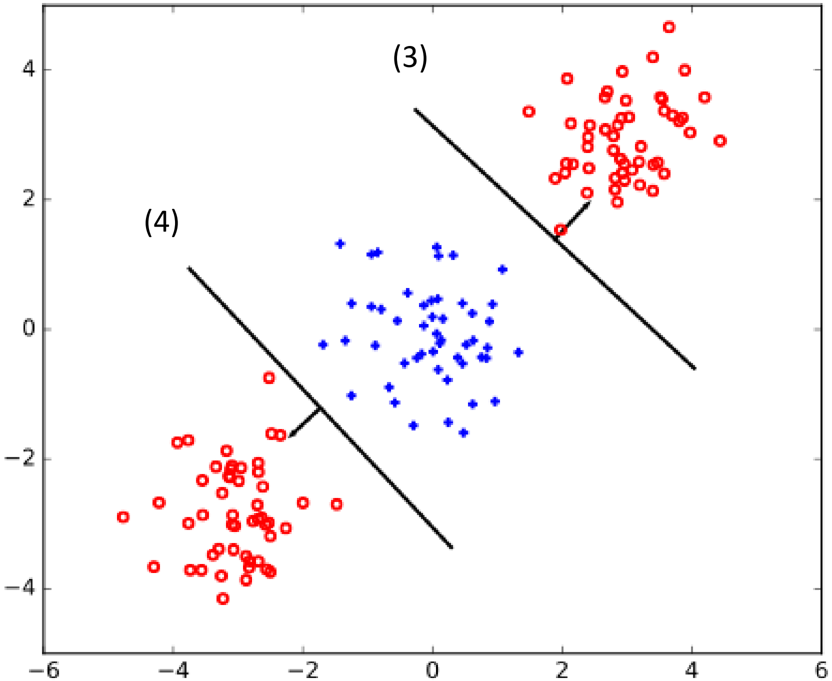
Hidden layer units



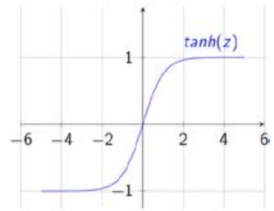


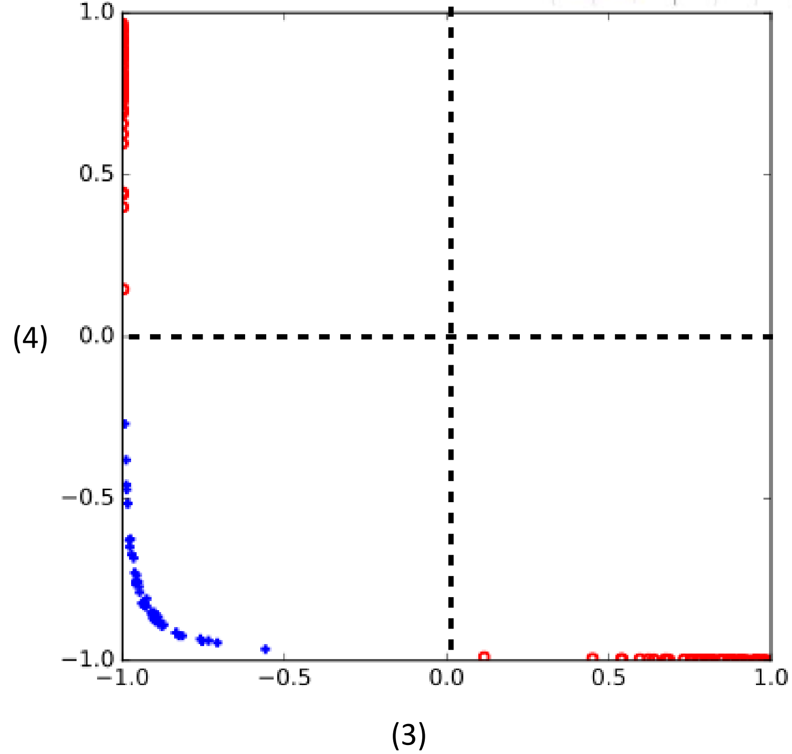
Does orientation matter?

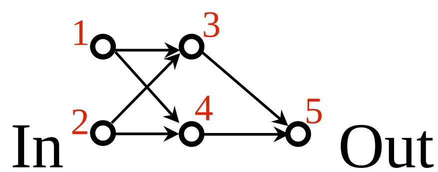
Hidden layer units



tanh activation

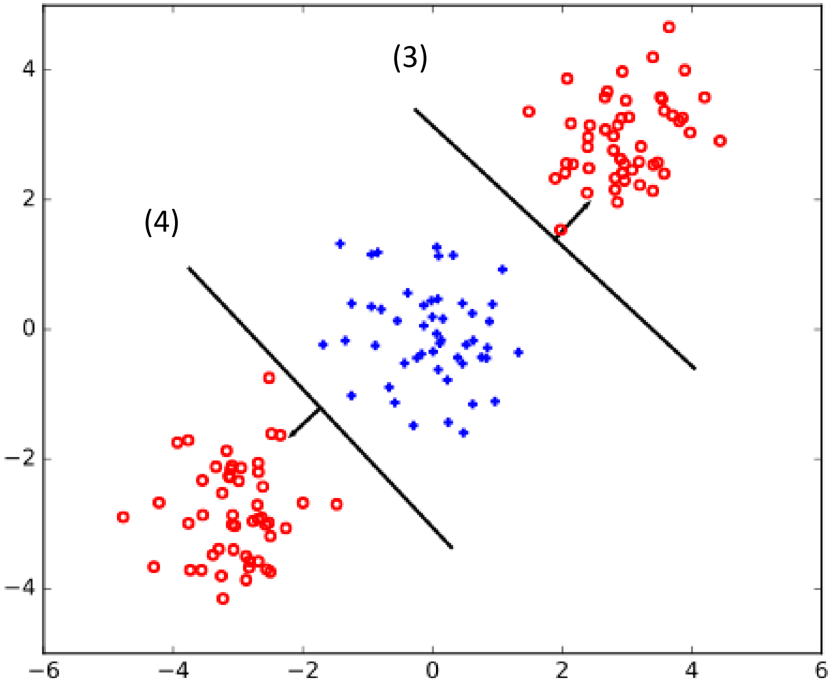




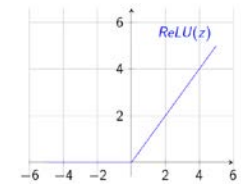


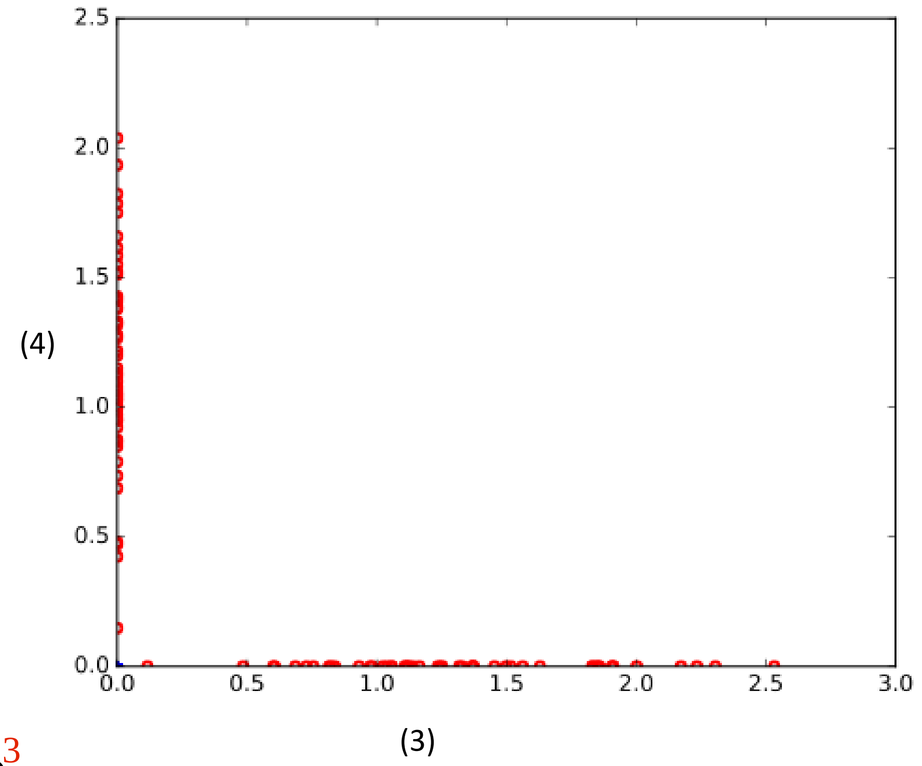
Does orientation matter?

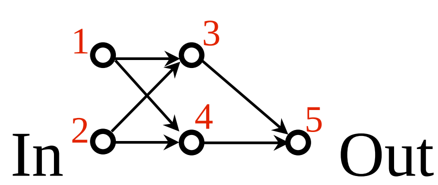
Hidden layer units



ReLU activation

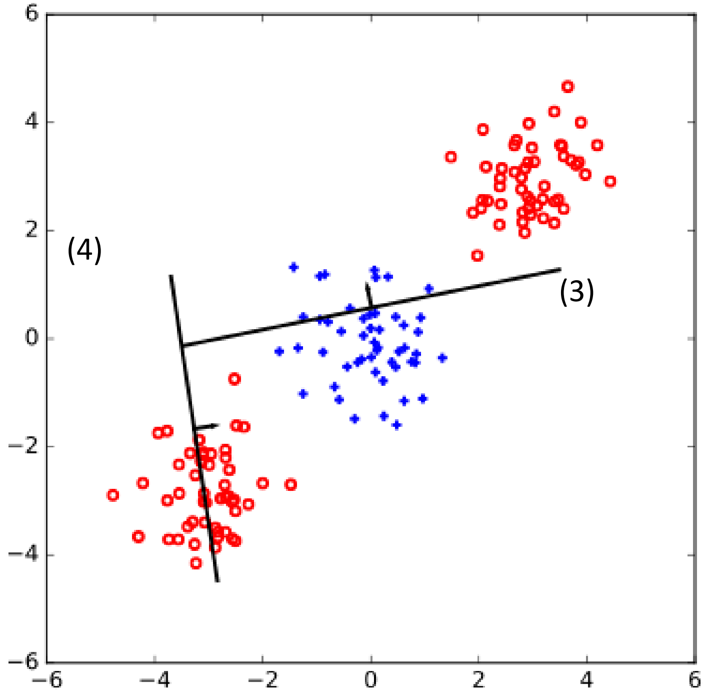


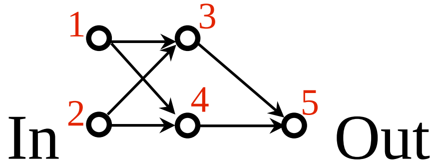




Random Hidden Units

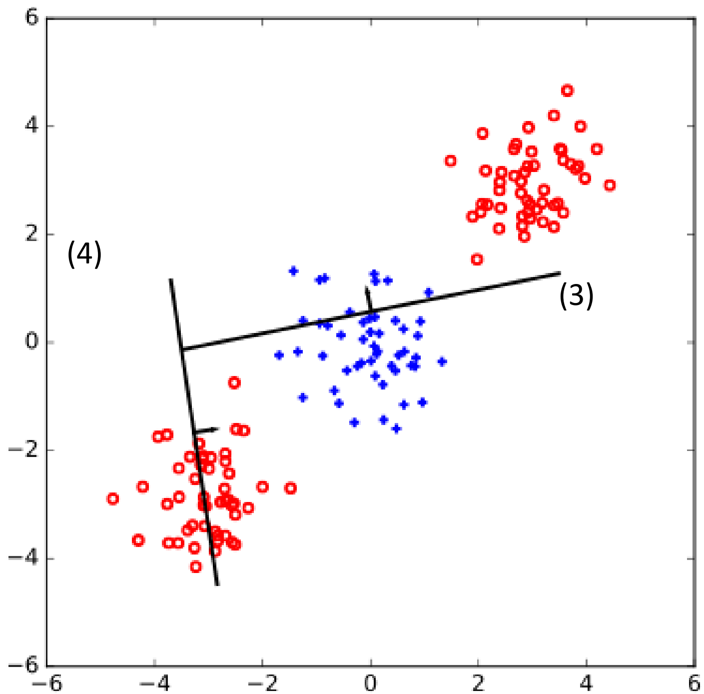
Hidden layer units



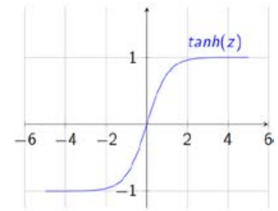


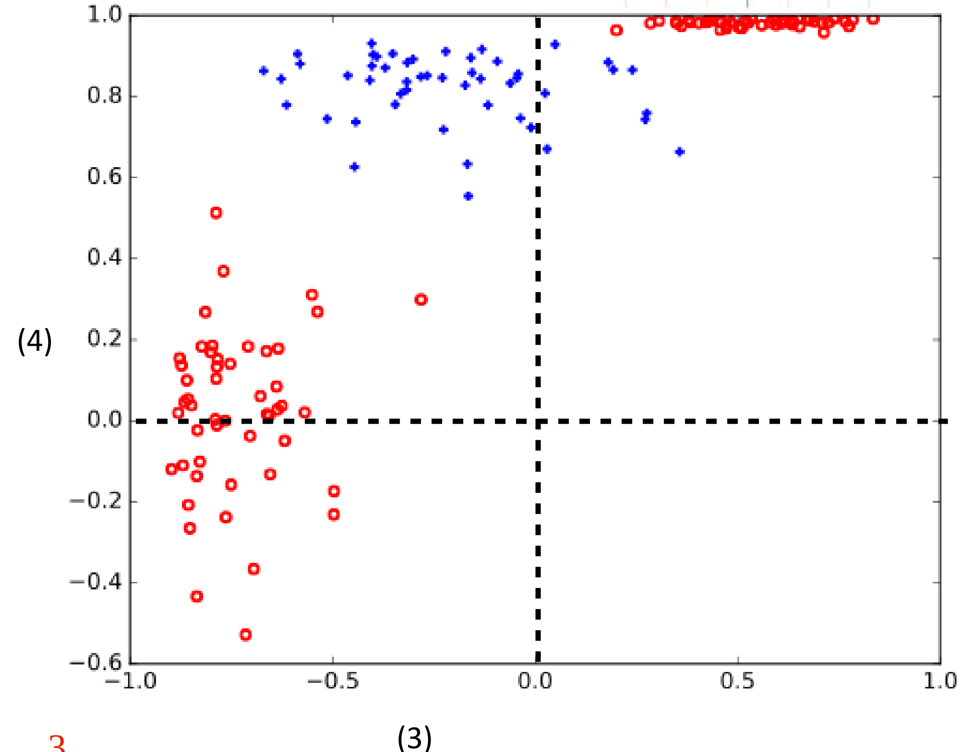
Random Hidden Units

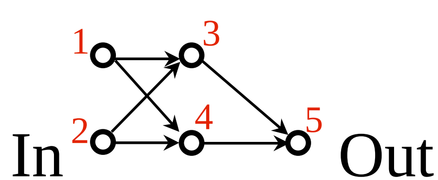
Hidden layer units



tanh activation

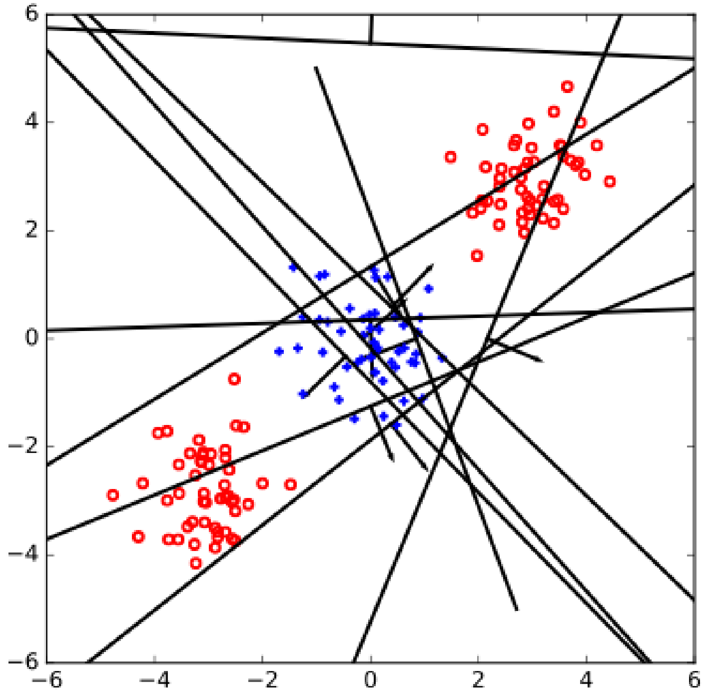




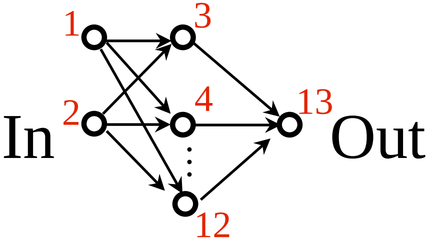


Random Hidden Units

Hidden layer units

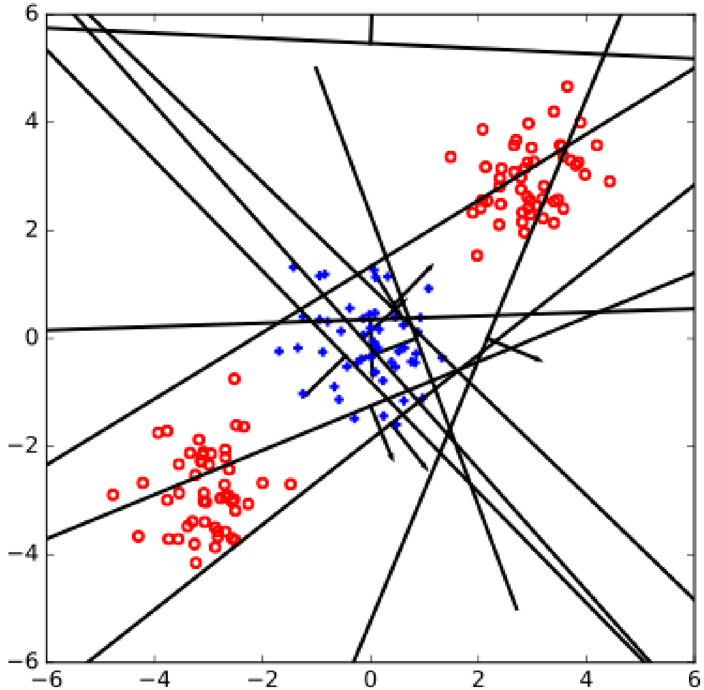


(10 randomly chosen units)



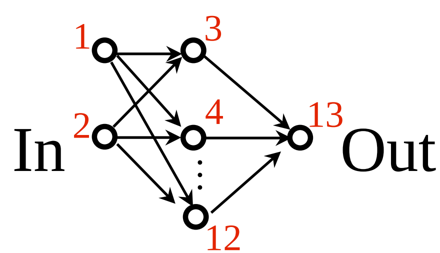
Random Hidden Units

Hidden layer units



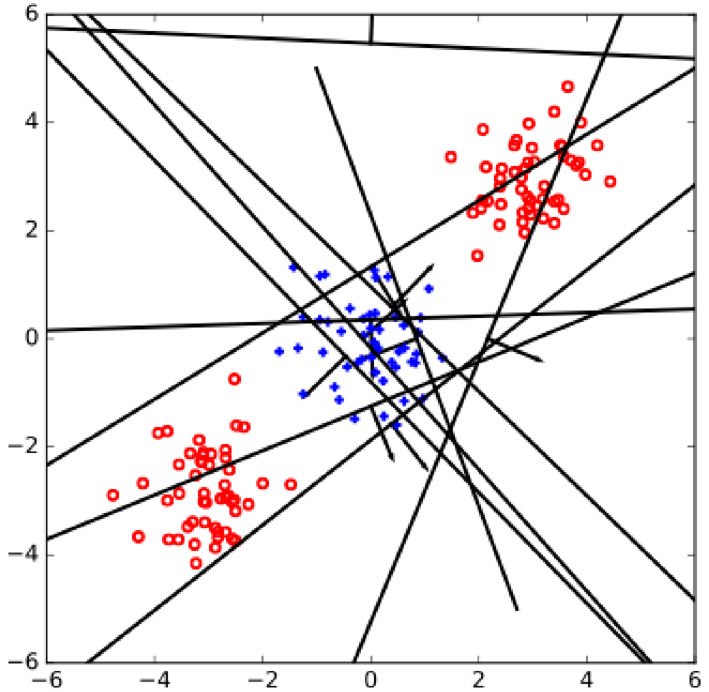
(10 randomly chosen units)

Are the points linearly separable in the resulting 10 dimensional space?



Random Hidden Units

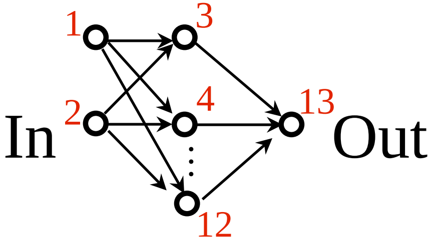
Hidden layer units



(10 randomly chosen units)

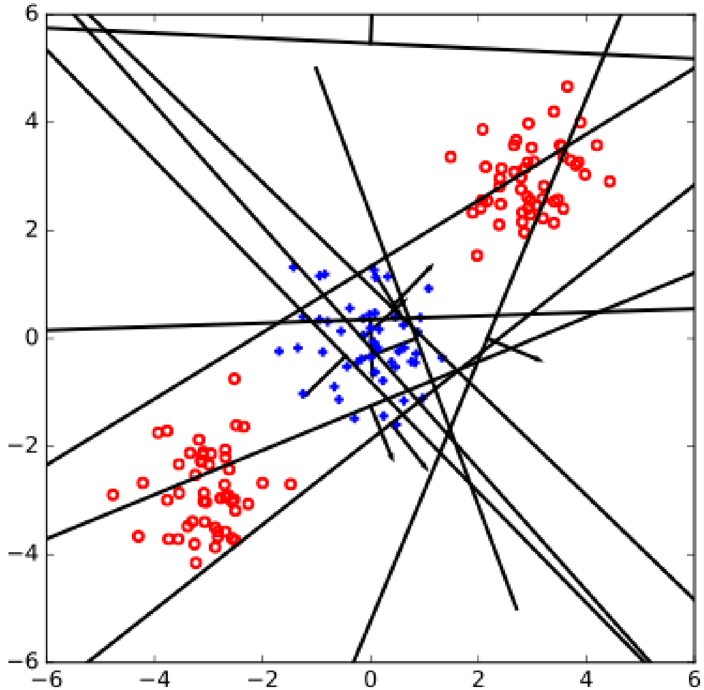
Are the points linearly separable in the resulting 10 dimensional space?

YES!

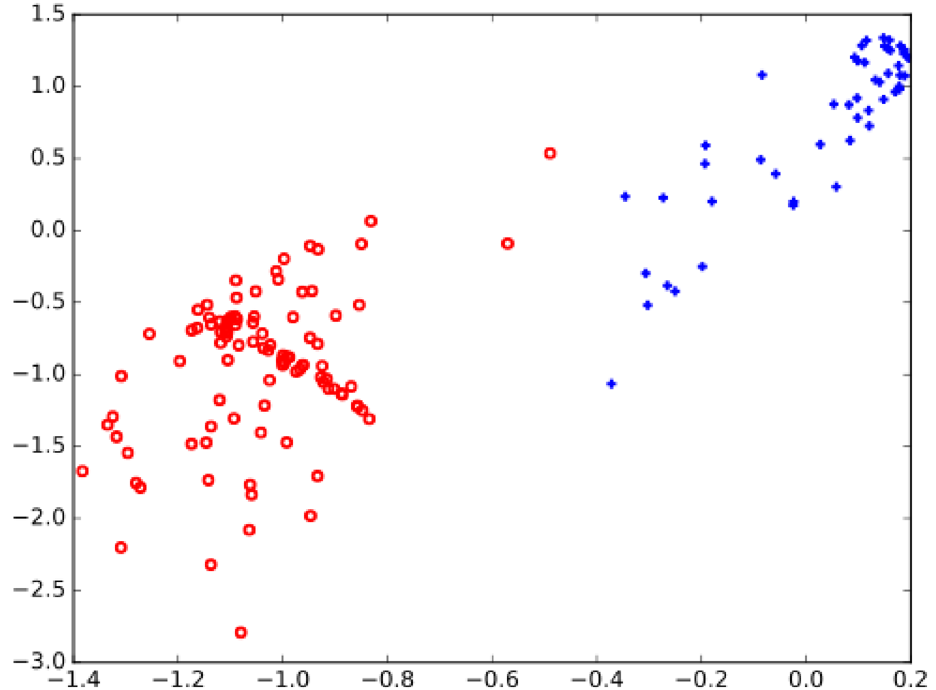


Random Hidden Units

Hidden layer units



(10 randomly chosen units)



what are the coordinates??

Summary

- Representation for feedforward neural networks
  - Input, hidden layers, output
  - Parameters/weights
  - Activation functions
- How to use a neural network for a classification problem
- Next lecture:
  - Training neural networks
  - Convolutional neural networks