CS489/698: Intro to ML

Lecture 03: Multi-layer Perceptron

UNIVERSITY OF WATERLOO

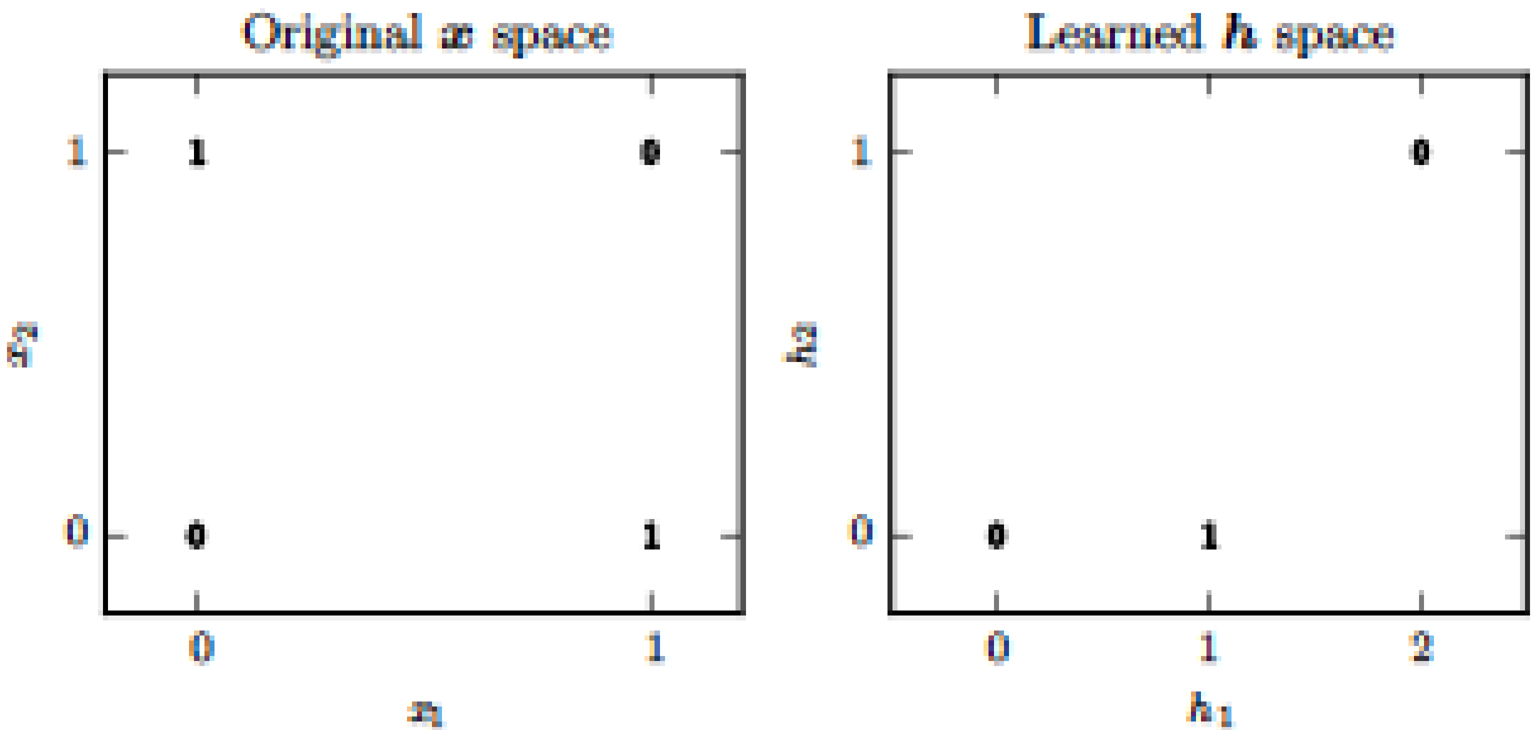
Outline

- Failure of Perceptron
- Neural Network
- Backpropagation
- Universal Approximator

Outline

- Failure of Perceptron
- Neural Network
- Backpropagation
- Universal Approximator

The XOR problem



- $X = \{ (0,0), (0,1), (1,0), (1,1) \}$, $y = \{-1, 1, 1, -1\}$
- $f^*(x) = \text{sign}[f_{xor} - ½]$, $f_{xor}(x) = (x_1 \& -x_2) \mid (-x_1 \& x_2)$

No separating hyperplane

$$y(\langle \mathbf{w}, \mathbf{x} \rangle + b) > 0$$

- $x_1 = (0,0)$, $y_1 = -1$ → $b < 0$
- $x_2 = (0,1)$, $y_2 = 1$ → $w_2 + b > 0$
- $x_3 = (1,0)$, $y_3 = 1$ → $w_1 + b > 0$

$$\left.\begin{array}{l} w_2 + b > 0 \\ w_1 + b > 0 \end{array}\right\} (w_1 + w_2 + b) + b > 0$$

- $x_4 = (1,1)$, $y_4 = -1$ → $w_1 + w_2 + b < 0$

Ex. what happens if we run perceptron/winnow on this example?

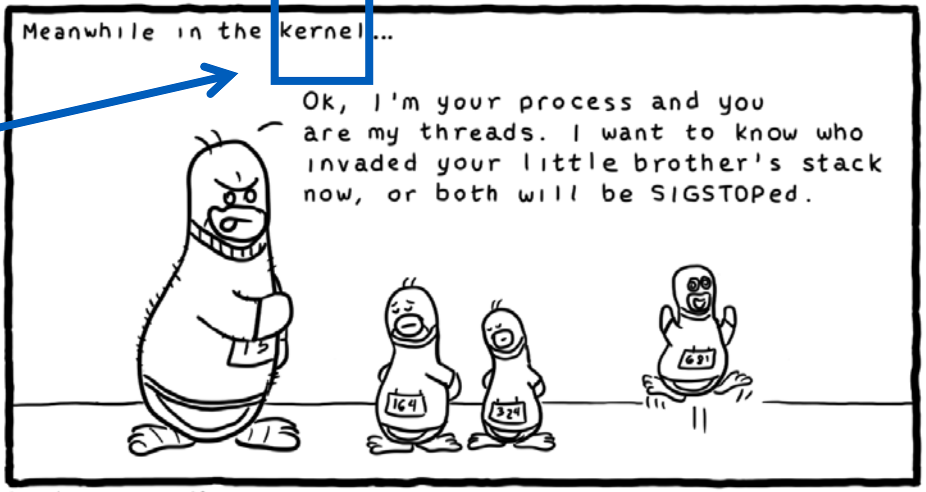
- Contradiction!
- [At least one of the blue or green inequalities has to be strict]

Fixing the problem

- Our model (hyperplanes) underfits the data (xor func)

- Fix representation, richer model

- Fix model, richer representation



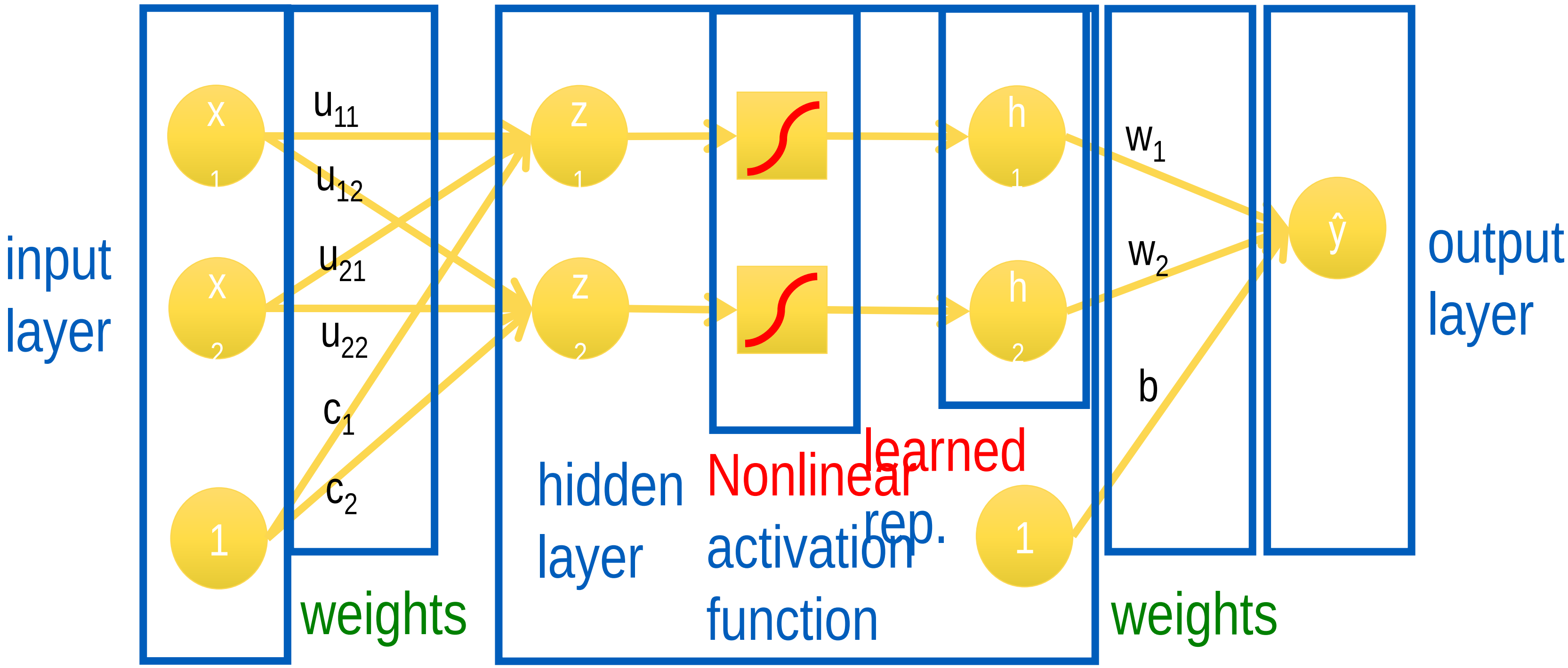
Daniel Stori {turnoff.us}

- NN: still use hyperplane, but learn representation simultaneously

Outline

- Failure of Perceptron
- Neural Network
- Backpropagation
- Universal Approximator

Two-layer perceptron



$$\mathbf{z} = U\mathbf{x} + \mathbf{c}$$ 1st linear layer

makes all the difference! $$\mathbf{h} = f(\mathbf{z})$$ nonlinear transform

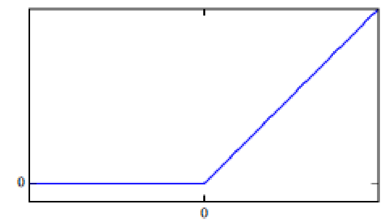
2nd linear layer $$\hat{y} = \langle \mathbf{h}, \mathbf{w} \rangle + b$$

Does it work?

$$U = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} 2 \\ -4 \end{bmatrix} \quad b = -1$$

Rectified Linear Unit (ReLU)

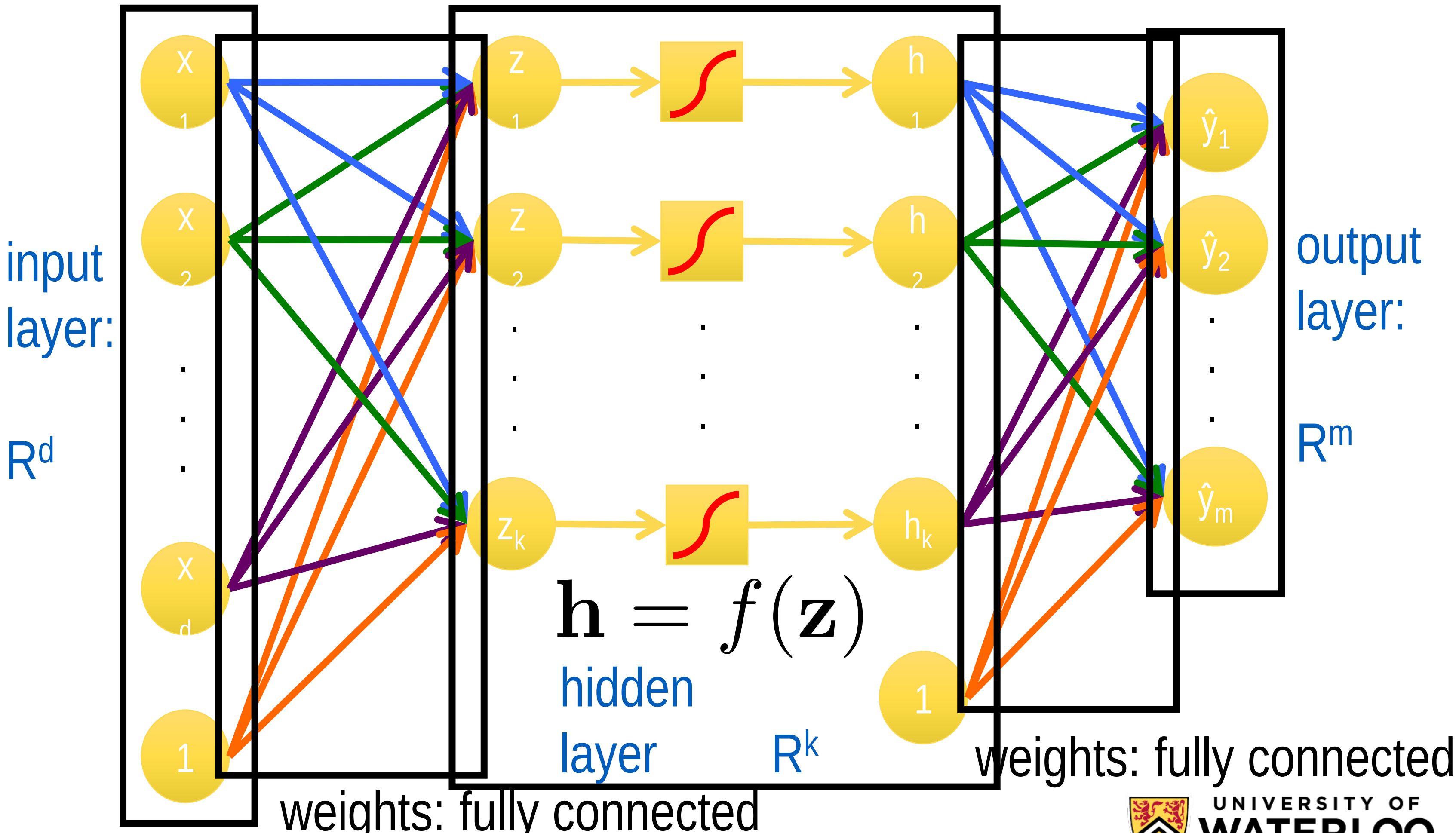
$$f(t) = t_+ := \max(t, 0)$$



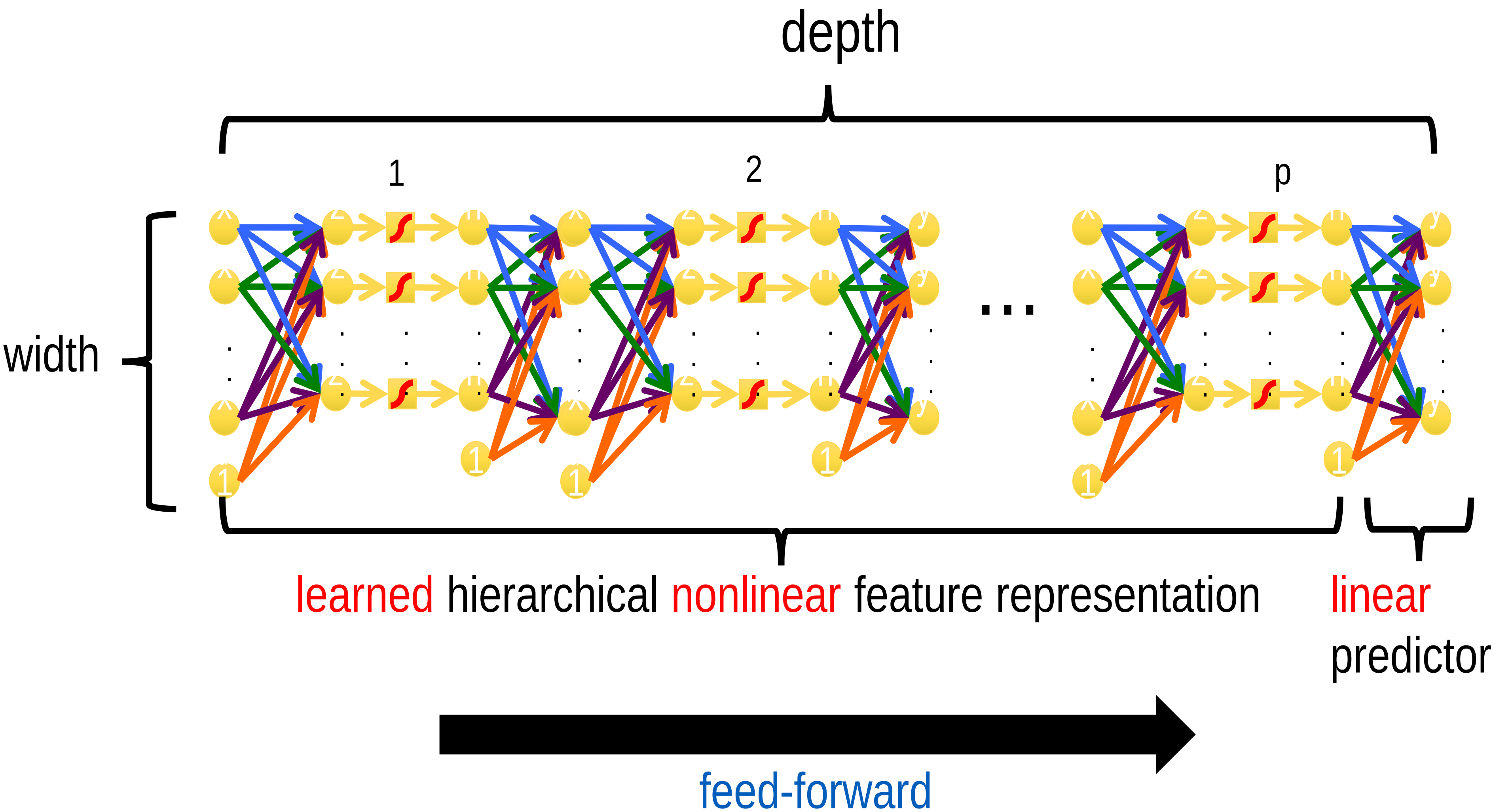
- $x_1 = (0,0)$, $y_1 = -1$ $\rightarrow$ $z_1 = (0,-1)$, $h_1 = (0,0)$ $\rightarrow$ $\hat{y}_1 = -1$ ✓
- $x_2 = (0,1)$, $y_2 = 1$ $\rightarrow$ $z_2 = (1,0)$, $h_2 = (1,0)$ $\rightarrow$ $\hat{y}_2 = 1$ ✓
- $x_3 = (1,0)$, $y_3 = 1$ $\rightarrow$ $z_3 = (1,0)$, $h_3 = (1,0)$ $\rightarrow$ $\hat{y}_3 = 1$ ✓
- $x_4 = (1,1)$, $y_4 = -1$ $\rightarrow$ $z_4 = (2,1)$, $h_4 = (2,1)$ $\rightarrow$ $\hat{y}_4 = -1$ ✓

Multi-layer perceptron

$$\mathbf{z} = U\mathbf{x} + \mathbf{c} \qquad \hat{\mathbf{y}} = W\mathbf{h} + \mathbf{b}$$



Multi-layer perceptron (stacked)

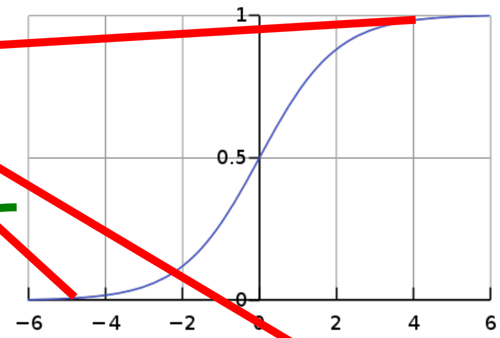


Activation function

饱和 saturate

- Sigmoid

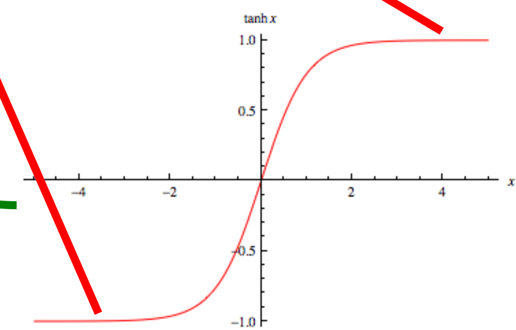
$$f(t) = \sigma(t) = \frac{1}{1+e^{-t}} = \frac{e^t}{1+e^t}$$



smooth

- Tanh

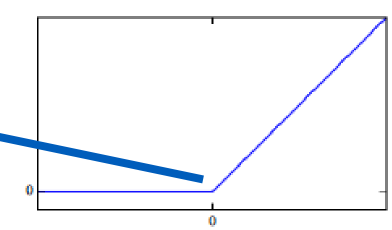
$$f(t) = \tanh(t) = \frac{e^t - e^{-t}}{e^t + e^{-t}}$$



- Rectified Linear

$$f(t) = t_+ := \max(t, 0)$$

nonsmooth

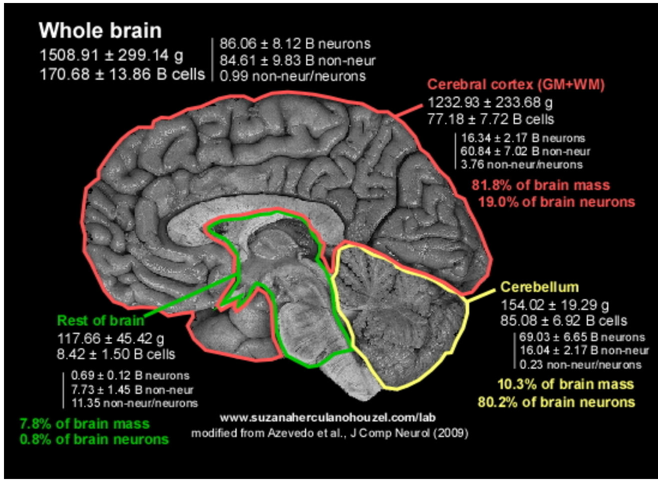


UNIVERSITY OF WATERLOO

Underfitting vs. Overfitting

- Linear predictor (perceptron / winnow / linear regression) underfits

- NNs learn *hierarchical nonlinear* feature jointly with linear predictor
  - may overfit
  - tons of heuristics (some later)

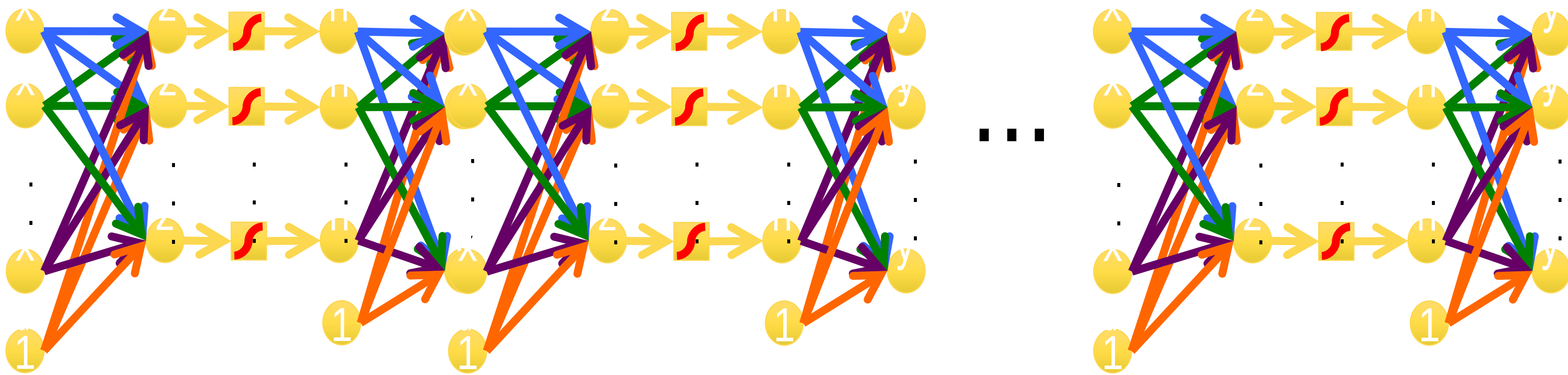


- Size of NN: # of weights/connections
  - ~ 1 billion; human brain 10^6 billions (estimated)

UNIVERSITY OF WATERLOO

Weights training

x in $\mathbb{R}^d$



ŷ in $\mathbb{R}^m$

$$\boxed{\hat{\mathbf{y}} = q(x; \mathbf{\Theta})}$$

- Need a loss ℓ to measure diff. between pred $\hat{\mathbf{y}}$ and truth y
  - E.g., $(\hat{\mathbf{y}}-y)^2$; more later

- Need a training set $\{(x_1,y_1), \ldots, (x_n,y_n)\}$ to train weights Θ

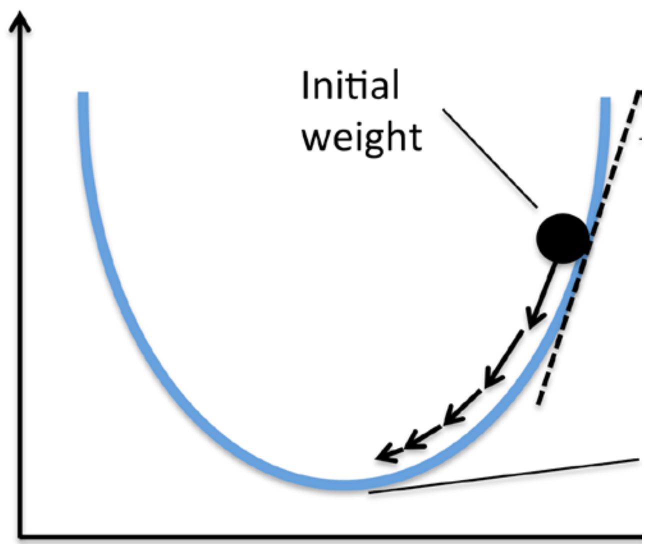
Gradient Descent

$$\boxed{\min_{\Theta}} L(\Theta) := \frac{1}{n}\sum_{i=1}^{n} \ell\left[\mathbf{q}(\mathbf{x}_i;\Theta), \mathbf{y}_i\right]$$

(Generalized) gradient

O(n) !

$$\Theta_{t+1} \leftarrow \Theta_t \boxed{-} \eta_t \nabla L(\Theta_t)$$



Step size (learning rate)

- const., if L is smooth
- diminishing, otherwise

Stochastic Gradient Descent (SGD)

$$\Theta_{t+1} = \Theta_t - \eta_t \cdot \frac{1}{n}\sum_{i=1}^{n} \nabla \ell\Big[\mathbf{q}(\mathbf{x}_i; \Theta_t), \mathbf{y}_i\Big]$$

average over *n* samples

a random sample suffices

$$\boxed{\Theta_{t+1} = \Theta_t - \eta_t \nabla \ell\Big[\mathbf{q}(\mathbf{x}_{i_t}; \Theta_t), \mathbf{y}_{i_t}\Big]}$$

- diminishing step size, e.g., 1/sqrt{t} or 1/t
- averaging, momentum, variance-reduction, etc.
- sample w/o replacement; cycle; permute in each pass

A little history on optimization

- Gradient descent mentioned first in (Cauchy, 1847)

ANALYSE MATHÉMATIQUE. — *Méthode générale pour la résolution des systèmes d'équations simultanées*; *par* M. AUGUSTIN CAUCHY.

- First rigorous convergence proof (Curry, 1944)

THE METHOD OF STEEPEST DESCENT FOR NON-LINEAR MINIMIZATION PROBLEMS*

BY HASKELL B. CURRY $\boxed{\textit{(Frankford Arsenal)}}$

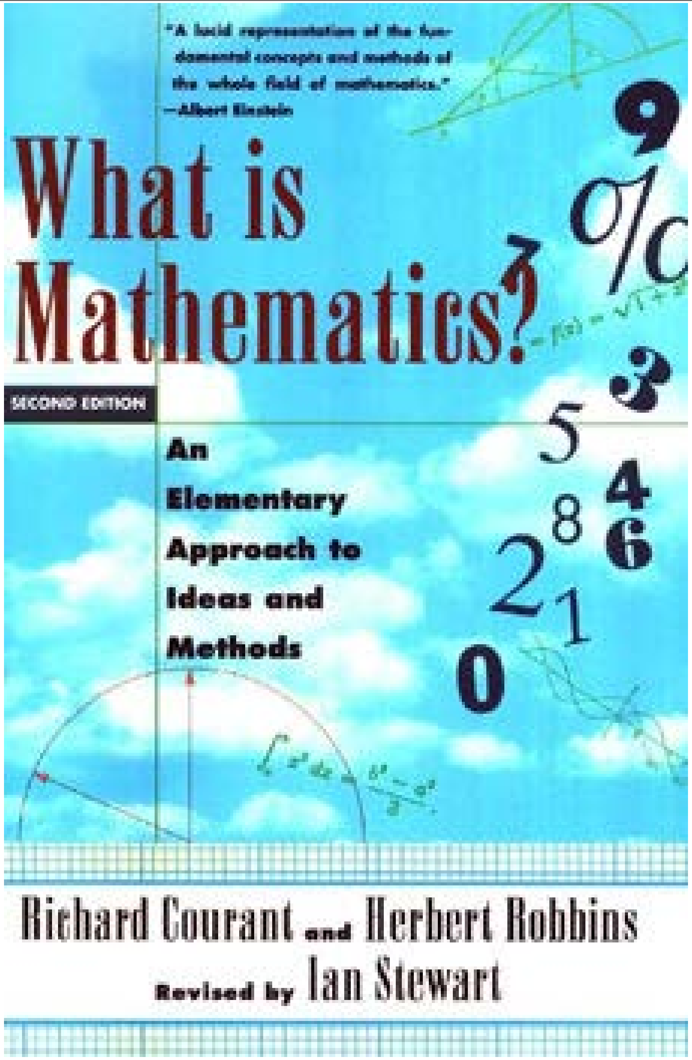
- SGD proposed and analyzed (Robbins & Monro, 1951)

A STOCHASTIC APPROXIMATION METHOD[1]

BY HERBERT ROBBINS AND SUTTON MONRO

University of North Carolina

Herbert Robbins (1915 – 2001)



UNIVERSITY OF WATERLOO

Outline

- Failure of Perceptron
- Neural Network
- Backpropagation
- Universal Approximator

Backpropogation

- A fancy name for the chain rule for derivatives:

$$f(x) = g[\,h(x)\,] \;\rightarrow\; f'(x) = g'[\,h(x)\,] * h'(x)$$

- Efficiently computes the derivative in NN

- Two passes; complexity = O(size(NN))
  - forward pass: compute function value sequentially
  - backward pass: compute derivative sequentially

Algorithm

Require: Network depth, l
Require: $\boldsymbol{W}^{(i)}, i \in \{1, \ldots, l\}$, the weight matrices
Require: $\boldsymbol{b}^{(i)}, i \in \{1, \ldots, l\}$, the bias parameters
Require: $\boldsymbol{x}$, the input to process
Require: $\boldsymbol{y}$, the target output

$\boldsymbol{h}^{(0)} = \boldsymbol{x}$
for $k = 1, \ldots, l$ **do**
  $\boldsymbol{a}^{(k)} = \boldsymbol{b}^{(k)} + \boldsymbol{W}^{(k)} \boldsymbol{h}^{(k-1)}$
  $\boldsymbol{h}^{(k)} = f(\boldsymbol{a}^{(k)})$
end for
$\hat{\boldsymbol{y}} = \boldsymbol{h}^{(l)}$
$J = L(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \lambda\Omega(\theta)$

Forward pass

f: activation function

$J = L + \lambda\Omega$: training obj.

After the forward computation, compute the gradient on the output layer:
$\boldsymbol{g} \leftarrow \nabla_{\hat{\boldsymbol{y}}} J = \nabla_{\hat{\boldsymbol{y}}} L(\hat{\boldsymbol{y}}, \boldsymbol{y})$
for $k = l, l-1, \ldots, 1$ **do**
  Convert the gradient on the layer's output into a gradient into the pre-nonlinearity activation (element-wise multiplication if f is element-wise):
  $\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{a}^{(k)}} J = \boldsymbol{g} \odot f'(\boldsymbol{a}^{(k)})$
  Compute gradients on weights and biases (including the regularization term, where needed):
  $\nabla_{\boldsymbol{b}^{(k)}} J = \boldsymbol{g} + \lambda \nabla_{\boldsymbol{b}^{(k)}} \Omega(\theta)$
  $\nabla_{\boldsymbol{W}^{(k)}} J = \boldsymbol{g}\, \boldsymbol{h}^{(k-1)\top} + \lambda \nabla_{\boldsymbol{W}^{(k)}} \Omega(\theta)$
  Propagate the gradients w.r.t. the next lower-level hidden layer's activations:
  $\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{h}^{(k-1)}} J = \boldsymbol{W}^{(k)\top} \boldsymbol{g}$
end for

Backward pass

History (perfect for a course project!)

UNIVERSITY OF WATERLOO

Outline

- Failure of Perceptron
- Neural Network
- Backpropagation
- Universal Approximator

Rationals are dense in R

- Any real number can be approximated by some rational number arbitrarily well

- Or in fancy mathematical language

$$\forall r \in \mathbf{R}, \forall \epsilon > 0, \exists s \in \mathbf{Q}, \text{ such that } |r - s| < \epsilon$$

domain of interest

subset in pocket

metric for approx.

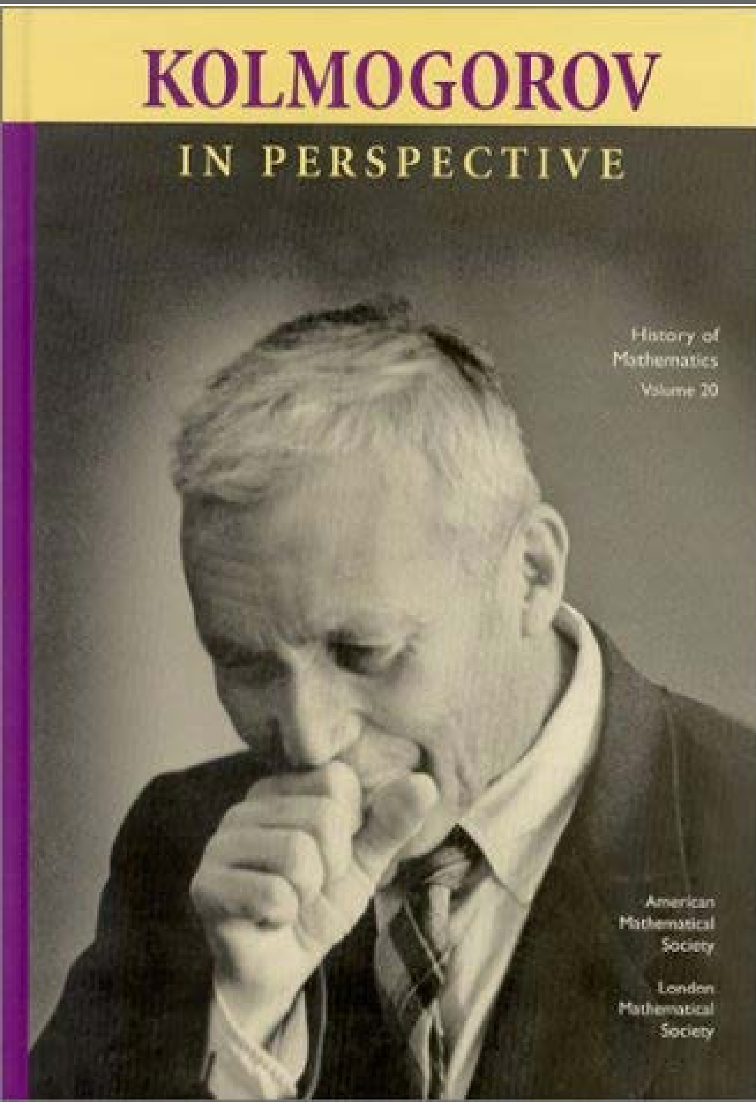
Kolmogorov-Arnold Theorem

Theorem (Kolmogorov, Arnold, Lorentz, Sprecher, ...).
Any continuous function *g*: $[0,1]^d \rightarrow R$ can be written (exactly!) as:

$$g(\mathbf{x}) = \sum_{j=1}^{2d+1} \varphi\Big(\sum_{i=1}^{d} \lambda_i \psi(x_i + \eta j) + j\Big)$$

- Binary addition is the "only" multivariate function!
- Solves Hilbert's 13th problem!
- φ, ψ can be constructed from *g*, which is unknown...

Andrey Kolmogorov (1903 - 1987)



ERGEBNISSE DER MATHEMATIK
UND IHRER GRENZGEBIETE
HERAUSGEGEBEN VON DER SCHRIFTLEITUNG
DES
„ZENTRALBLATT FÜR MATHEMATIK"
ZWEITER BAND
3

GRUNDBEGRIFFE DER
WAHRSCHEINLICHKEITS-
RECHNUNG

VON

A. KOLMOGOROFF

BERLIN
VERLAG VON JULIUS SPRINGER
1933

Foundations of the theory of probability

"Every mathematician believes that he is ahead of the others. The reason none state this belief in public is because they are intelligent people."

UNIVERSITY OF WATERLOO

Universal Approximator

Theorem (Cybenko, Hornik et al., Leshno et al., …).
Any continuous function g: $[0,1]^d \to R$ can be uniformly approximated in arbitrary precision by a two-layer NN with an activation function f that

1. is locally bounded
2. has "negligible" closure of discontinuous points
3. is not a polynomial

- conditions are necessary in some sense
- includes (almost) all activation functions in practice

Caveat and Remedy

- NNs were praised for being "universal"
  - but shall see later that many kernels are universal as well
  - desirable but perhaps not THE explanation

- May need exponentially many hidden units…

- Increase depth may reduce network size, exponentially!
  - can be a course project, ask for references

Questions?

UNIVERSITY OF WATERLOO