Lecture 19: 3 April, 2025

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Data Mining and Machine Learning
January–April 2025

Support Vector Machines

$w \cdot x + b > \cancel{0}\ +1$

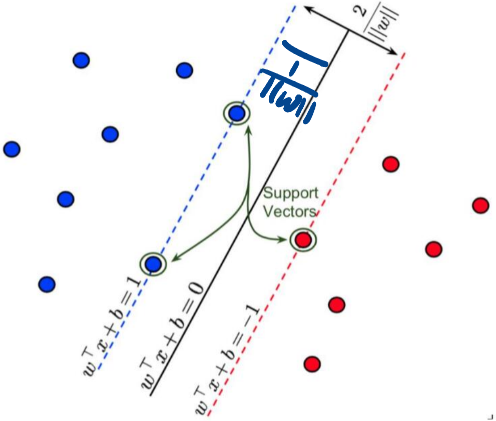
$\qquad\quad\ \ < \cancel{0}\ -1$

Minimize $\dfrac{|w|}{2}$

Subject to

$w_1x_1^i + w_2x_2^i + \cdots w_nx_n^i + b > 1$, if $y_i = 1$

$w_1x_1^i + w_2x_2^i + \cdots w_nx_n^i + b < -1$, if $y_i = -1$



- The constraints are linear
- The objective function is not linear
  $|w| = \sqrt{w_1^2 + w_2^2 + \cdots + w_n^2}$
- This is a quadratic optimization problem, not linear programming

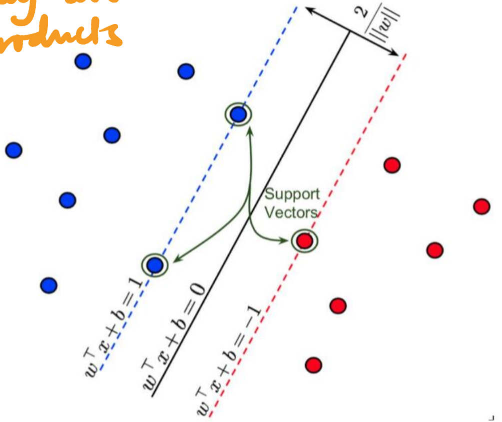
Support Vector Machines

- Convex optimization theory
- Can be solved using computational techniques
- Solution expressed in terms of Lagrange multipliers $\alpha_1, \alpha_2, \ldots, \alpha_N$
- α_i is non-zero iff x_i is a support vector
- Final classifier for new input z

$$\text{sign}\left[\sum_{i \in sv} y_i \alpha_i (x_i \cdot z) + b\right]$$

- sv is set of support vectors

Only dot products



Soft margin optimization

Minimize $\frac{|w|}{2} + \sum_{i=1}^{N} \xi_i^2$

Subject to

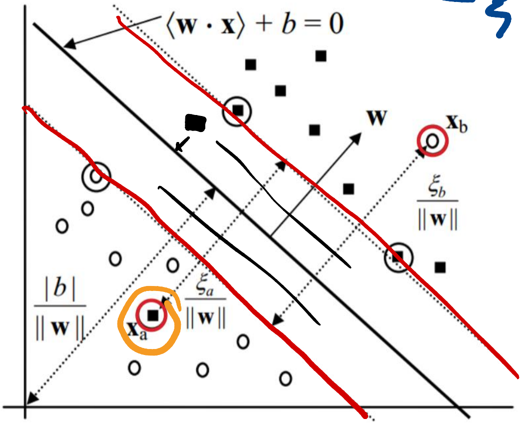
$\xi_i \geq 0$

$w \cdot x_i + b > 1 - \xi_i$, if $y_i = 1$

$w \cdot x_i + b < -1 + \xi_i$, if $y_i = -1$

- Constraints include requirement that error terms are non-negative
- Again the objective function is quadratic

$wx + b > 1 - \xi$



Kernels

- K is a kernel for transformation φ if $K(x,z) = \varphi(x)\cdot\varphi(z)$
- If we have a kernel, we don't need to explicitly compute transformed points
- All dot products can be computed implicitly using the kernel on original data points

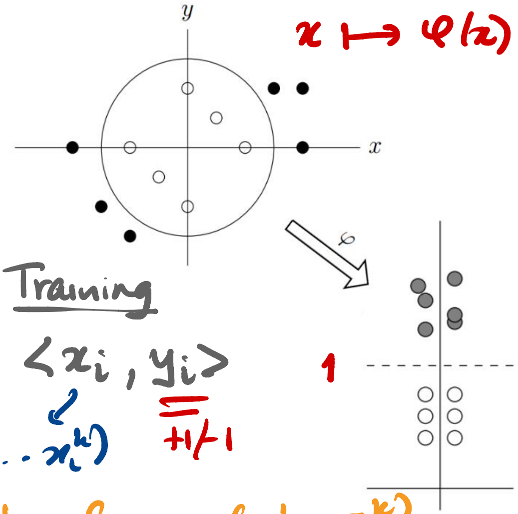
Dual

$$\sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j \, \varphi(x_i)\cdot\varphi(x_j)$$

Soln

$$\text{sign}\left[\sum_{i\in sv} y_i \alpha_i (\varphi(x_i)\cdot\varphi(z)) + b\right]$$

$x \mapsto \varphi(x)$



Training

$\langle x_i, y_i \rangle$

$(x_i^1 \ldots x_i^k)$

$+1/-1$

1

Classify $z = (z^1, \ldots z^k)$

Kernels

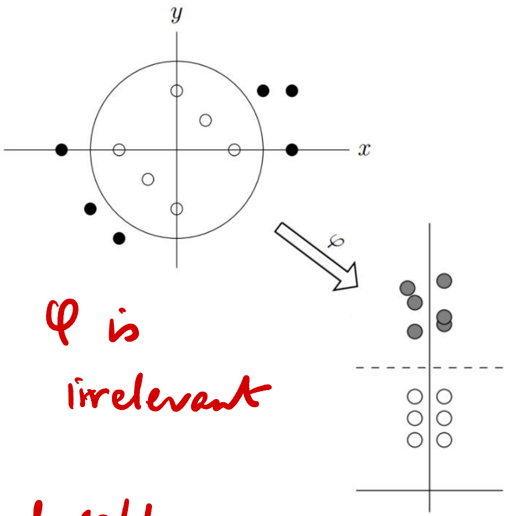
- K is a kernel for transformation φ if $K(x,z) = \varphi(x) \cdot \varphi(z)$
- If we have a kernel, we don't need to explicitly compute transformed points
- All dot products can be computed implicitly using the kernel on original data points

Dual optim

$$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j \, K(x_i, x_j)$$

$$\text{sign}\left[\sum_{i \in sv} y_i \alpha_i K(x_i, z) + b\right]$$

Dual solution



φ is irrelevant

Known kernels

- Fortunately, there are many known kernels

  real no.

- Polynomial kernels

  any

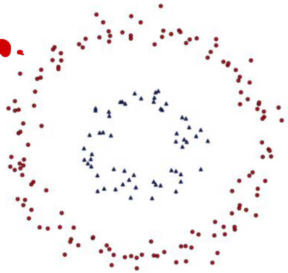
  $(r + x \cdot z)^k$

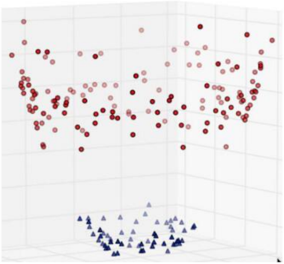
$$K(x, z) = (1 + x \cdot z)^k$$

- Any $K(x, z)$ representing a similarity measure

- Gaussian radial basis function — similarity based on inverse exponential distance

$$K(x, z) = e^{-c|x-z|^2}$$

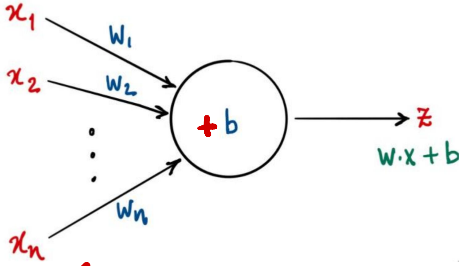


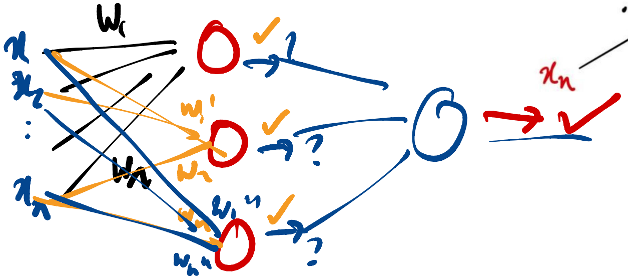


Linear separators and Perceptrons

$x = \langle x_1, x_2, \ldots, x_n \rangle$

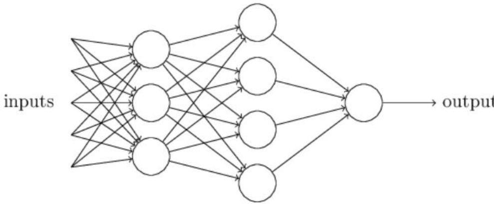
- Perceptrons define linear separators $w \cdot x + b$
  - $w \cdot x + b > 0$, classify Yes $(+1)$
  - $w \cdot x + b < 0$, classify No (-1)





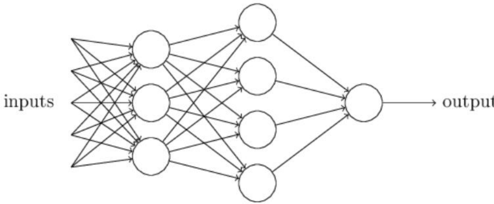
Linear separators and Perceptrons

- Perceptrons define linear separators $w \cdot x + b$
  - $w \cdot x + b > 0$, classify Yes $(+1)$
  - $w \cdot x + b < 0$, classify No (-1)
- What if we cascade perceptrons?



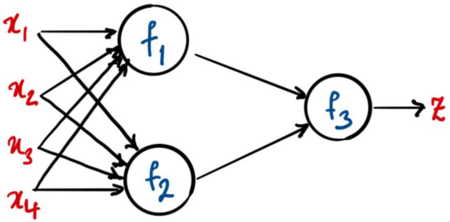
Linear separators and Perceptrons

- Perceptrons define linear separators $w \cdot x + b$
  - $w \cdot x + b > 0$, classify Yes $(+1)$
  - $w \cdot x + b < 0$, classify No (-1)
- What if we cascade perceptrons?
- Result is still a linear separator



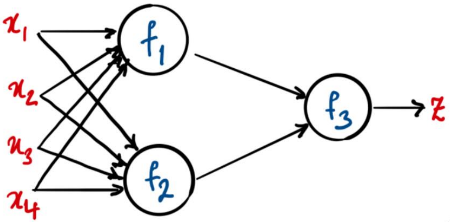
Linear separators and Perceptrons

- Perceptrons define linear separators $w \cdot x + b$
  - $w \cdot x + b > 0$, classify Yes $(+1)$
  - $w \cdot x + b < 0$, classify No (-1)
- What if we cascade perceptrons?
- Result is still a linear separator
  - $f_1 = w_1 \cdot x + b_1, f_2 = w_2 \cdot x + b_2$



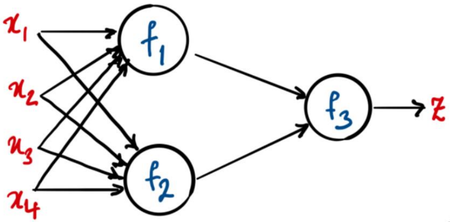
Linear separators and Perceptrons

- Perceptrons define linear separators $w \cdot x + b$
  - $w \cdot x + b > 0$, classify Yes $(+1)$
  - $w \cdot x + b < 0$, classify No (-1)
- What if we cascade perceptrons?
- Result is still a linear separator
  - $f_1 = w_1 \cdot x + b_1, f_2 = w_2 \cdot x + b_2$
  - $f_3 = w_3 \cdot \langle f_1, f_2 \rangle + b_3$



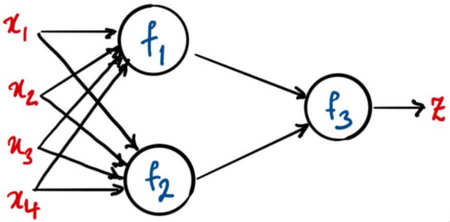
Linear separators and Perceptrons

- Perceptrons define linear separators $w \cdot x + b$
  - $w \cdot x + b > 0$, classify Yes $(+1)$
  - $w \cdot x + b < 0$, classify No (-1)
- What if we cascade perceptrons?
- Result is still a linear separator
  - $f_1 = w_1 \cdot x + b_1, f_2 = w_2 \cdot x + b_2$
  - $f_3 = w_3 \cdot \langle f_1, f_2 \rangle + b_3$
  - $f_3 = w_3 \cdot \langle w_1 \cdot x + b_1, w_2 \cdot x + b_2 \rangle + b_3$

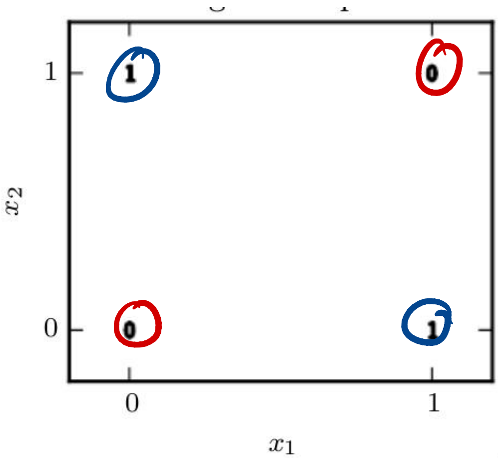


Linear separators and Perceptrons

- Perceptrons define linear separators $w \cdot x + b$
  - $w \cdot x + b > 0$, classify Yes $(+1)$
  - $w \cdot x + b < 0$, classify No (-1)
- What if we cascade perceptrons?
- Result is still a linear separator
  - $f_1 = w_1 \cdot x + b_1, f_2 = w_2 \cdot x + b_2$
  - $f_3 = w_3 \cdot \langle f_1, f_2 \rangle + b_3$
  - $f_3 = w_3 \cdot \langle w_1 \cdot x + b_1, w_2 \cdot x + b_2 \rangle + b_3$
  - $f_3 = \sum_{i=1}^{4} (w_{3_1} w_{1_i} + w_{3_2} w_{2_i}) \cdot x_i$
    $+(w_{3_1} b_1 + w_{3_2} b_2 + b_3)$

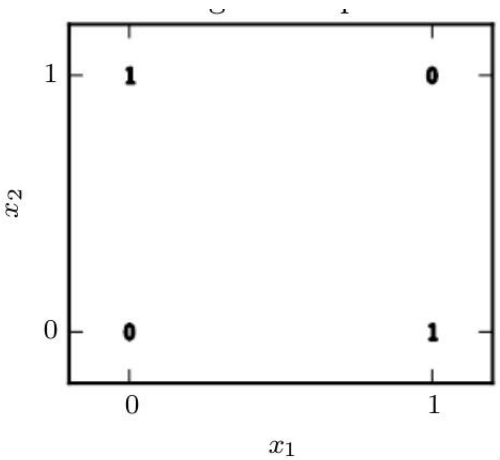


- Cannot compute *exclusive-or* (XOR)
- $XOR(x_1, x_2)$ is true if exactly one of x_1, x_2 is true (not both)



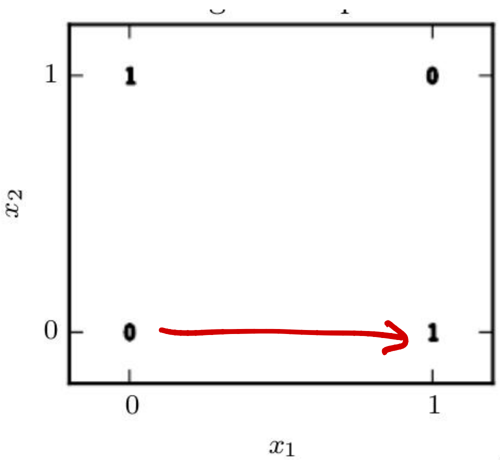
Limits of linearity

- Cannot compute *exclusive-or* (XOR)
- $XOR(x_1, x_2)$ is true if exactly one of x_1, x_2 is true (not both)
- Suppose $XOR(x_1, x_2) = ux_1 + vx_2 + b$



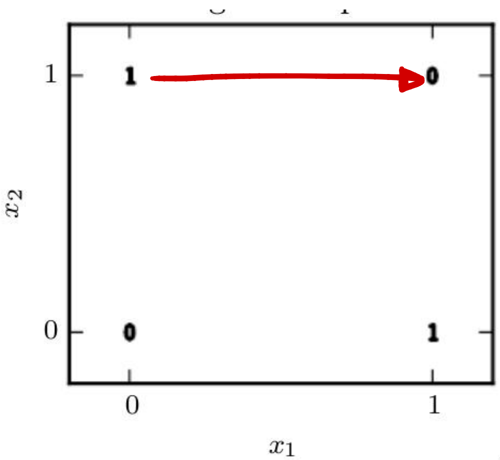
Limits of linearity

- Cannot compute *exclusive-or* (XOR)
- $XOR(x_1, x_2)$ is true if exactly one of x_1, x_2 is true (not both)
- Suppose $XOR(x_1, x_2) = ux_1 + vx_2 + b$
- $x_2 = 0$: As x_1 goes from 0 to 1, output goes from 0 to 1, so $u > 0$



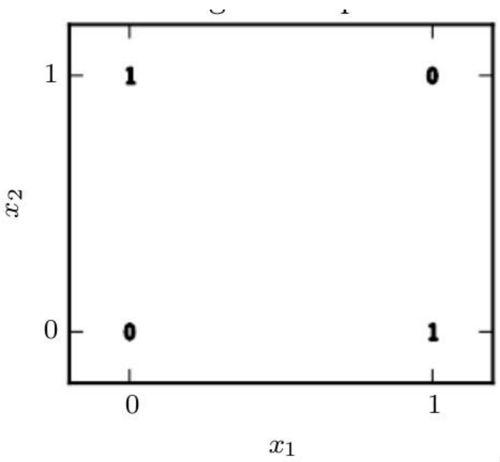
Limits of linearity

- Cannot compute *exclusive-or* (XOR)
- $XOR(x_1, x_2)$ is true if exactly one of x_1, x_2 is true (not both)
- Suppose $XOR(x_1, x_2) = ux_1 + vx_2 + b$
- $x_2 = 0$: As x_1 goes from 0 to 1, output goes from 0 to 1, so $u > 0$
- $x_2 = 1$: As x_1 goes from 0 to 1, output goes from 1 to 0, so $u < 0$



Limits of linearity

- Cannot compute *exclusive-or* (XOR)
- $XOR(x_1, x_2)$ is true if exactly one of x_1, x_2 is true (not both)
- Suppose $XOR(x_1, x_2) = ux_1 + vx_2 + b$
- $x_2 = 0$: As x_1 goes from 0 to 1, output goes from 0 to 1, so $u > 0$
- $x_2 = 1$: As x_1 goes from 0 to 1, output goes from 1 to 0, so $u < 0$
- Observed by Minsky and Papert, 1969, first "AI Winter"

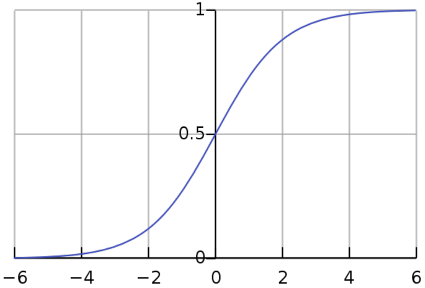


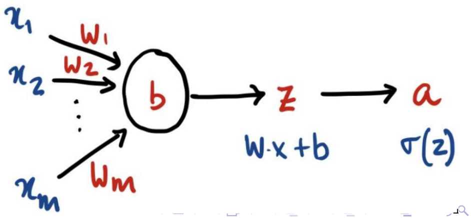
Non-linear activation

- Transform linear output z through a non-linear activation function
- Sigmoid function $\dfrac{1}{1+e^{-z}}$

a = activation

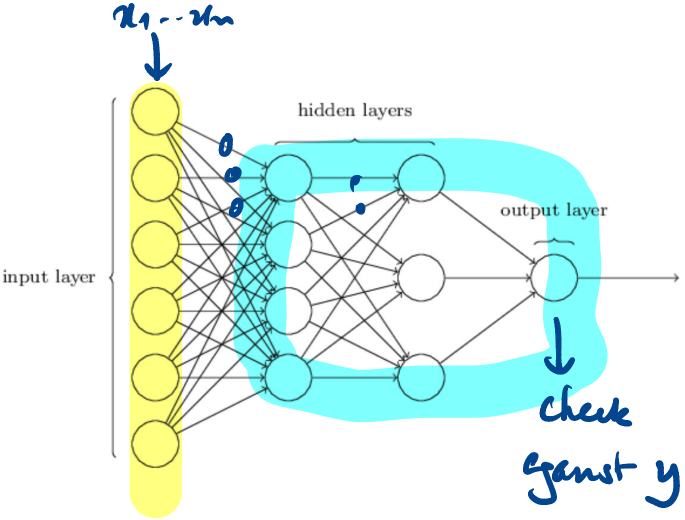
z = linear output





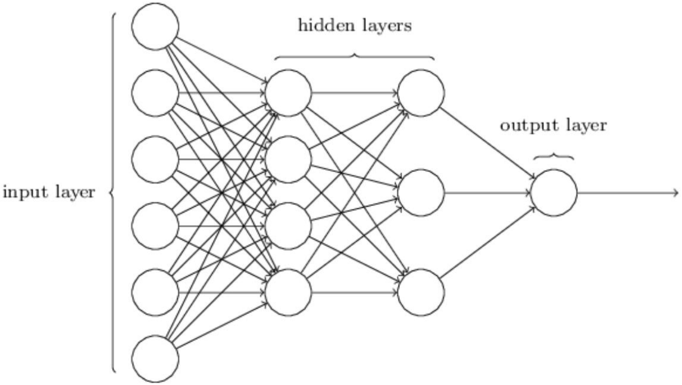
Structure of a neural network

- Acyclic
- Input layer, hidden layers, output layer



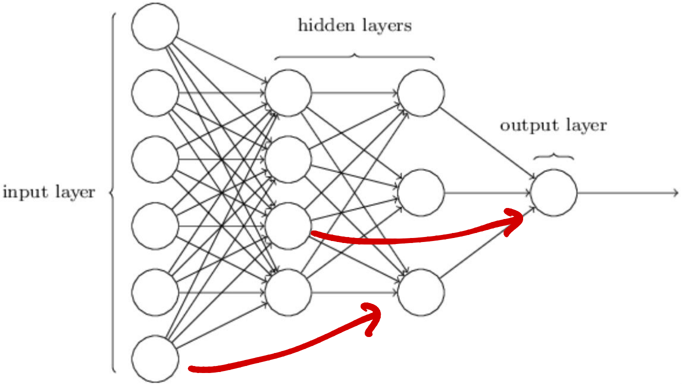
Structure of a neural network

- Acyclic
- Input layer, hidden layers, output layer
- Assumptions



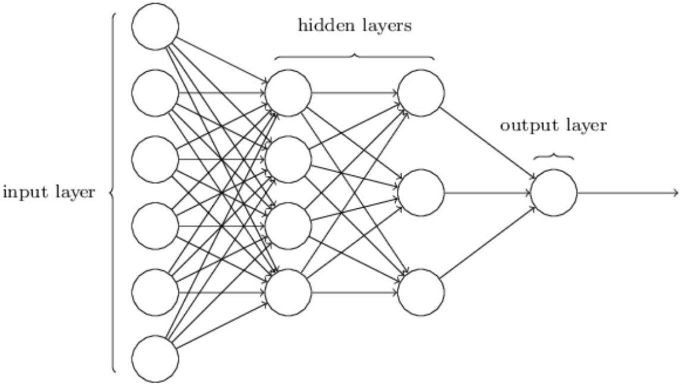
Structure of a neural network

- Acyclic
- Input layer, hidden layers, output layer
- Assumptions
  - Hidden neurons are arranged in layers



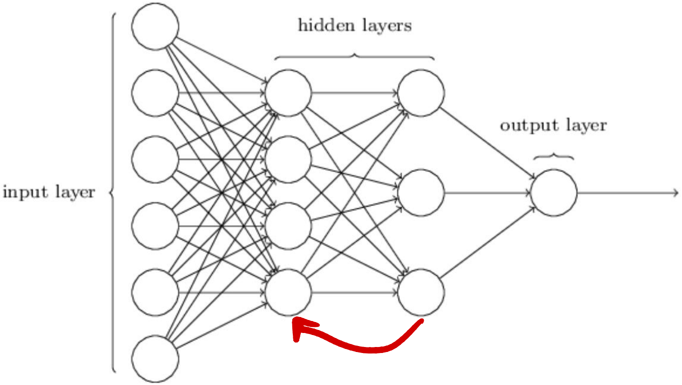
Structure of a neural network

- Acyclic
- Input layer, hidden layers, output layer
- Assumptions
  - Hidden neurons are arranged in layers
  - Each layer is fully connected to the next



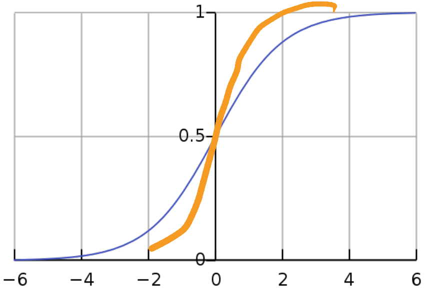
Structure of a neural network

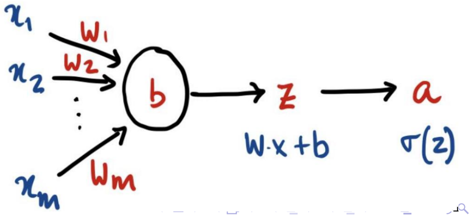
- Acyclic
- Input layer, hidden layers, output layer
- Assumptions
  - Hidden neurons are arranged in layers
  - Each layer is fully connected to the next
  - Set weight to zero to remove an edge



Non-linear activation

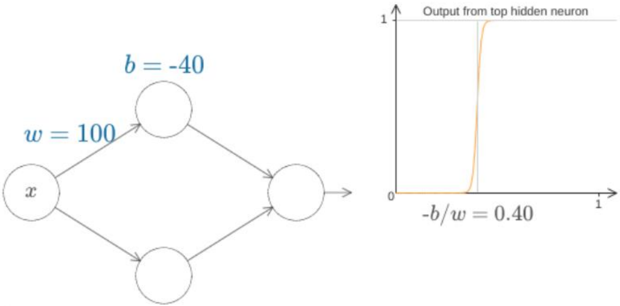
- Transform linear output z through a non-linear activation function
- Sigmoid function $\dfrac{1}{1+e^{-z}}$
- Step is at $z = 0$
  - $z = wx + b$, so step is at $x = -b/w$
  - Increasing w makes step steeper
  - Shift step by adjusting b





Universality

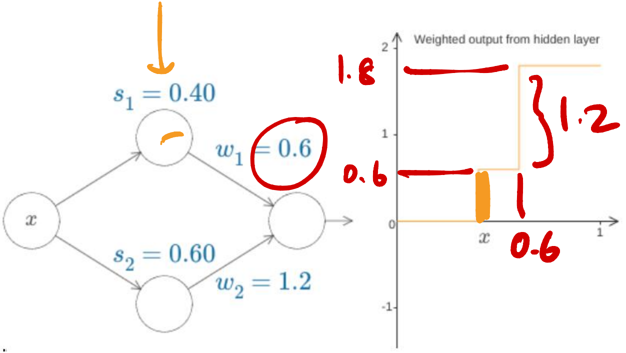
- Create a step at $x = -b/w$
  - Use s to denote $-b/w$
  - Step position



Universality

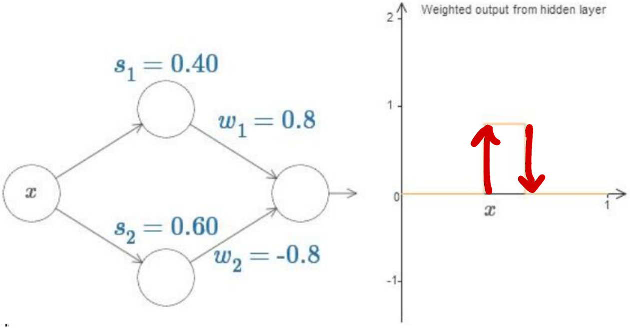
Two parallel nodes

- Create a step at $x = -b/w$
  - Use s to denote $-b/w$
  - Step position
- Cascade steps



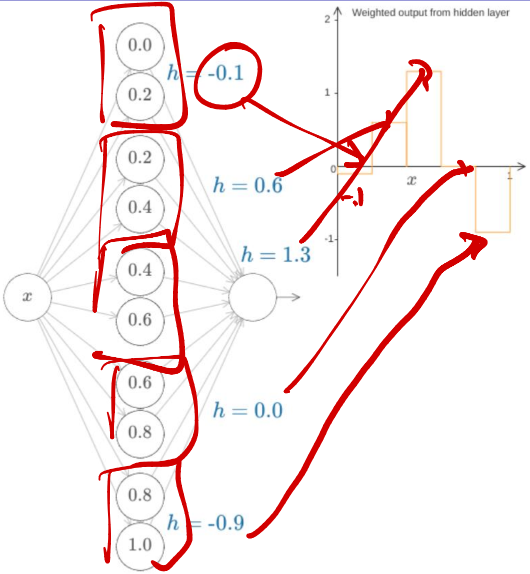
Universality

- Create a step at $x = -b/w$
  - Use s to denote $-b/w$
  - Step position
- Cascade steps
- Subtract steps to create a box

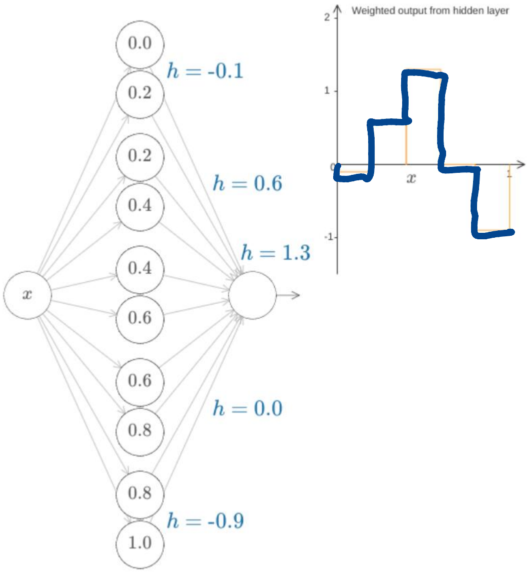


Universality

- Create a step at $x = -b/w$
  - Use s to denote $-b/w$
  - Step position
- Cascade steps
- Subtract steps to create a box
- Create many boxes

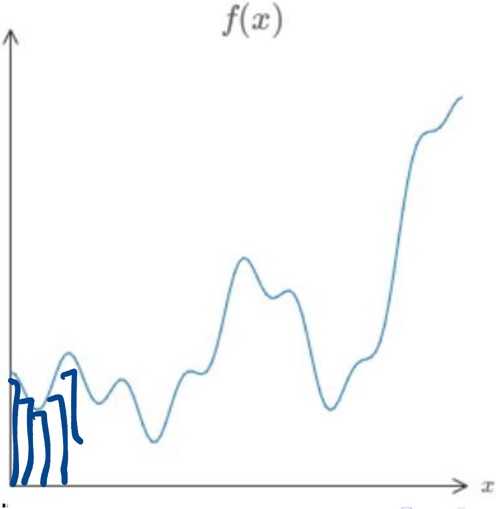


- Create a step at $x = -b/w$
  - Use s to denote $-b/w$
  - Step position
- Cascade steps
- Subtract steps to create a box
- Create many boxes



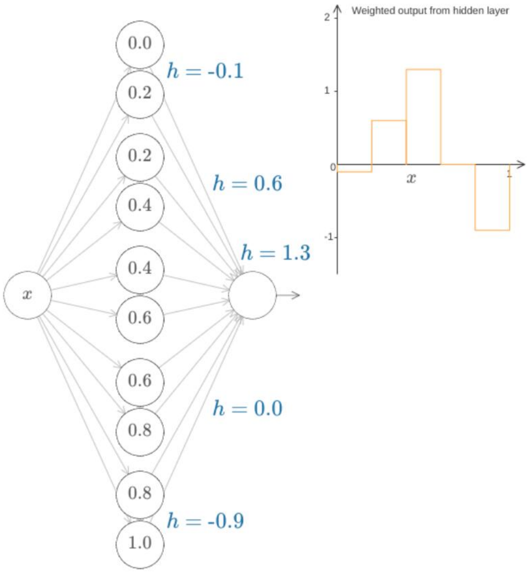
Universality

- Create a step at $x = -b/w$
  - Use s to denote $-b/w$
  - Step position
- Cascade steps
- Subtract steps to create a box
- Create many boxes
- Approximate any function



Universality

- Create a step at $x = -b/w$
  - Use s to denote $-b/w$
  - Step position
- Cascade steps
- Subtract steps to create a box
- Create many boxes
- Approximate any function
- Need only one hidden layer!

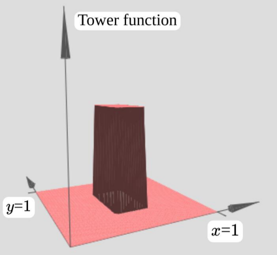


Non-linear activation

- With non-linear activation, network of neurons can approximate any function

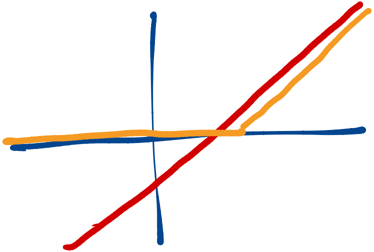
Non-linear activation

- With non-linear activation, network of neurons can approximate any function
  - Can build “rectangular” blocks



Non-linear activation

- With non-linear activation, network of neurons can approximate any function
  - Can build "rectangular" blocks
  - Combine blocks to capture any classification boundary



Rectified Linear Unit

RELU