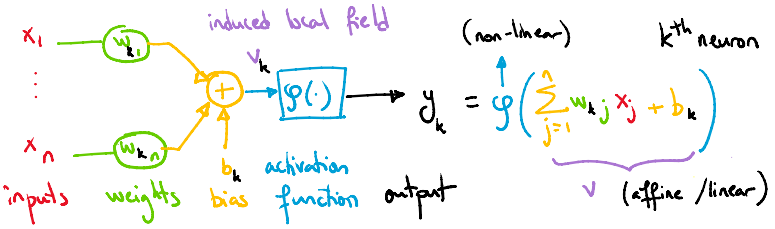
ECE/CS 559 Lecture 3 9/3

Last time: Biological Neuron → Mathematical Neuron



$\varphi(v) = \text{step}(v) = \begin{cases} 1 & v \geq 0 \\ 0 & v < 0 \end{cases}$ | $\text{sign}(v)$

$\varphi(v) = \sigma_a(v) = \frac{1}{1+e^{-av}}$ (sigmoid) | $\tanh(v)$

Also:

$\varphi(v) = \text{ReLU}(v) = \begin{cases} v & v \geq 0 \\ 0 & v < 0 \end{cases}$

Rectified Linear Unit

Linear Algebraic Representation

- All linear operations can be written as inner products!

$$\sum_{j=1}^{n} w_{1j} x_j + b_1 = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + b_1$$

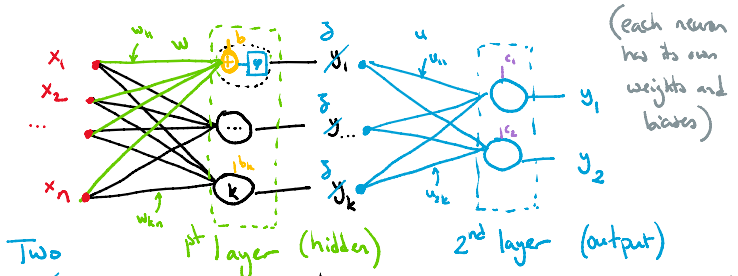
- Repeat this per neuron: $\begin{bmatrix} w_{21}, w_{22}, \cdots, w_{2n} \end{bmatrix} + b_2$; $\begin{bmatrix} w_{k1}, w_{k2}, \cdots, w_{kn} \end{bmatrix} + b_k$

- Single Layer: $y = \varphi(Wx + b)$ (applied per coordinate; matrix-vector multiplication)
- Two Layer:

$$y = \varphi(Uz + c) = \varphi(U\varphi(Wx + b) + c)$$

nested functions

① Feed-Forward Neural Networks:



Two ~~Single~~ Layer Feed Forward Fully Connected Neural Network

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & & w_{2n} \\ \vdots & & & \\ w_{k1} & w_{k2} & & w_{kn} \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix} \quad U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1k} \\ u_{21} & u_{22} & \cdots & u_{2k} \end{bmatrix} \quad c = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

vector | matrix | vector | matrix | vector | vector

"parameters"

- It's important to bookkeep dimensions:

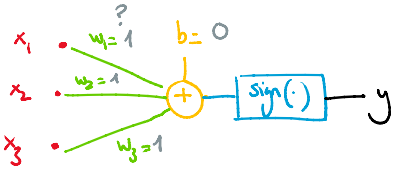
$x \in \mathbb{R}^n \quad W \in \mathbb{R}^{k \times n} \quad b \in \mathbb{R}^k \quad U \in \mathbb{R}^{2 \times k} \quad c \in \mathbb{R}^2 \quad y \in \mathbb{R}^2$

- Layer dimension: (input n), hidden k, output 2
- Naming: This is an n-k-2 fully connected FF network.
- If some weights are zero by design: not fully connected.
  Example: Convolutional Neural Networks:
- If some outputs feed back (after some delay) as inputs: not a feed forward neural network.
  Example: LSTMs (Long Short-Term Memory)

② Examples

(a) Majority operation:

- $n = 3$ inputs, assumed ± 1 $\quad x \in \{-1, +1\}^3$
- Activation function: sign $\quad \varphi(v) = \begin{cases} +1 & v > 0 \\ 0 & v = 0 \\ -1 & v < 0 \end{cases}$ (sign)
- Goal: Design a single neuron that outputs the value that most appears in the input.
  e.g. $x = (-1, -1, -1) \rightarrow y = -1$ $\quad x = (1, 1, -1) \rightarrow y = 1$.



$= \text{sign}(x_1 + x_2 + x_3)$

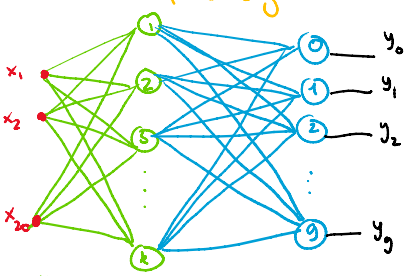
$= \text{majority}(x_1, x_2, x_3)$

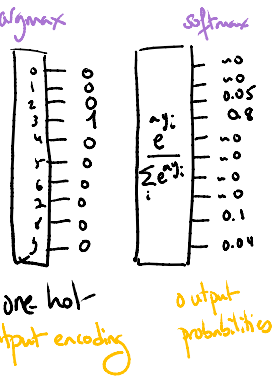
100% accurate.

(b) Logical Statements: Homework 1 + will revisit w/ Perceptrons

(c) Classification: $x =$  $\rightarrow y = 3$ $\quad x =$  $\rightarrow y = 1$

How? Vectorize $n=20$ input encoding





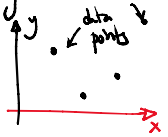
⑧ Approximation Theorems and Learning Algorithms

- Can we always find weights & biases (parameters)?
  - Given enough neurons & layers, almost any function can be approximated.
  - With logic (binary inputs/outputs) the DNF/CNF theorem shows that 2 layers are enough.

- But how do we set these parameters to achieve the desired behavior?
  - Manual adjustments → intractable except in simple cases
  - Instead, learn using guidance from data.

- This is not that different than fitting a polynomial.
- Data ( , 3) ( , 1) , etc.

  x y (target output)



$\varphi(u\varphi(Ax+b)+c) = \hat{y}(x)$ (neural network output)

Goal: minimize # of mistakes: $\sum_{\text{data}} \mathbb{1}\{\hat{y}(x) \neq y\}$

- Typical Algorithm processes data gradually:

epoch = 1 pass over data

1. Initialize weights randomly
2. For each (x, y) in data:
   - Feed x to NN and get $\hat{y}(x)$
   - Update weights to bring $\hat{y}(x)$ closer to y.
3. Repeat 2 until weights converge

$(y, \hat{y})$ gap = loss