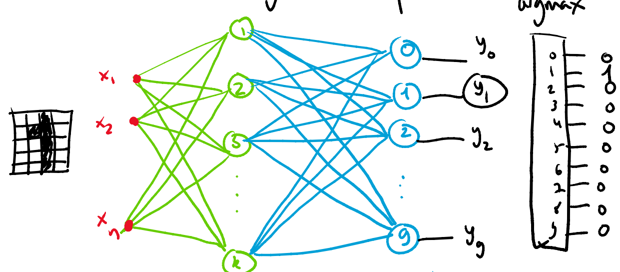
ECE/CS 559 Lecture 4 9/15

Last time: Feed-Forward Neural Networks:

Linear Algebraic Representation

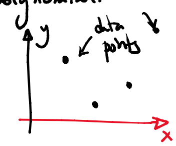


$x^n \in \mathbb{R}$ $W \in \mathbb{R}^{k \times n}$ $b \in \mathbb{R}^k$ $U \in \mathbb{R}^{9 \times k}$ $c \in \mathbb{R}^9$

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

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

  x y (target output)



$\varphi(U\varphi(Wx+b)+c) = y(x)$ (neural network output)

params

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

params

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

- Typical Algorithm processes data gradually:
  1. Initialize weights randomly
  2. For each (x, y) in data:
     - Feed x to NN and get $y(x)$
     - Update weights to bring $y(x)$ closer to y.
  3. Repeat 2 until weights converge

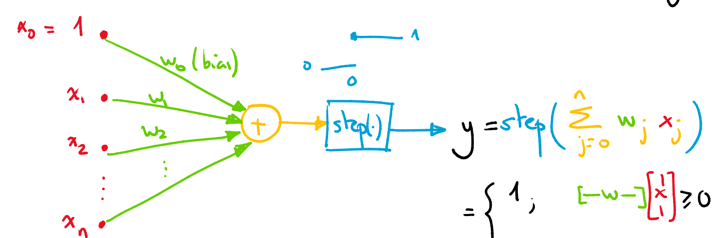
epoch: 1-pass over data

① 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.

② Perceptrons

- Consider the McCulloch-Pitts (1943) neuron, with bias as weight:



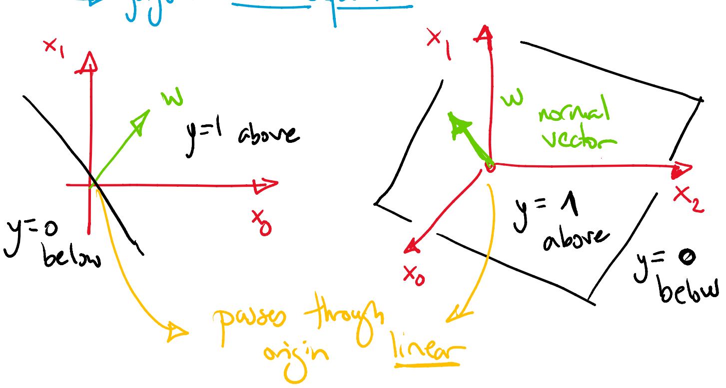
$$y = \text{step}\left(\sum_{j=0}^{n} w_j x_j\right) = \begin{cases} 1; & [-w-]\begin{bmatrix}1\\x\end{bmatrix} \geq 0 \\ 0; & < 0 \end{cases}$$

- What can it do?
- Recall $[w_0\ w_1]\begin{bmatrix}x_0\\x_1\end{bmatrix} = 0$ line $[w_0\ w_1\ w_2]\begin{bmatrix}x_0\\x_1\\x_2\end{bmatrix} = 0$ plane
- $[-w-]\begin{bmatrix}1\\x\end{bmatrix} = 0$ is a hyperplane.

  ≥ 0 is a half-space.

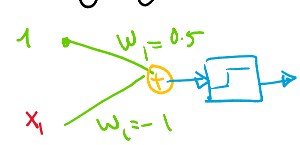
- This single neuron was used later by Rosenblatt (1961) who referred to it as "perceptron".
- It could classify using a hyperplane as decision boundary:

  → jargon: linear separator



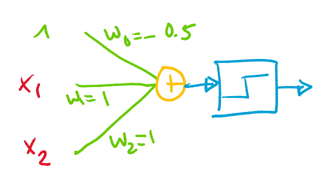
- Can model logic gate: 0 = False 1 = True

NOT:



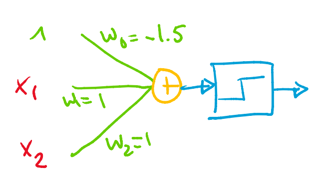
NOT x_1

OR:



x_1 OR x_2

AND:



x_1 AND x_2