Perceptron

INSTRUCTOR: HONGJIE CHEN

JUNE 6TH 2022

Imitate Human

- A neuron

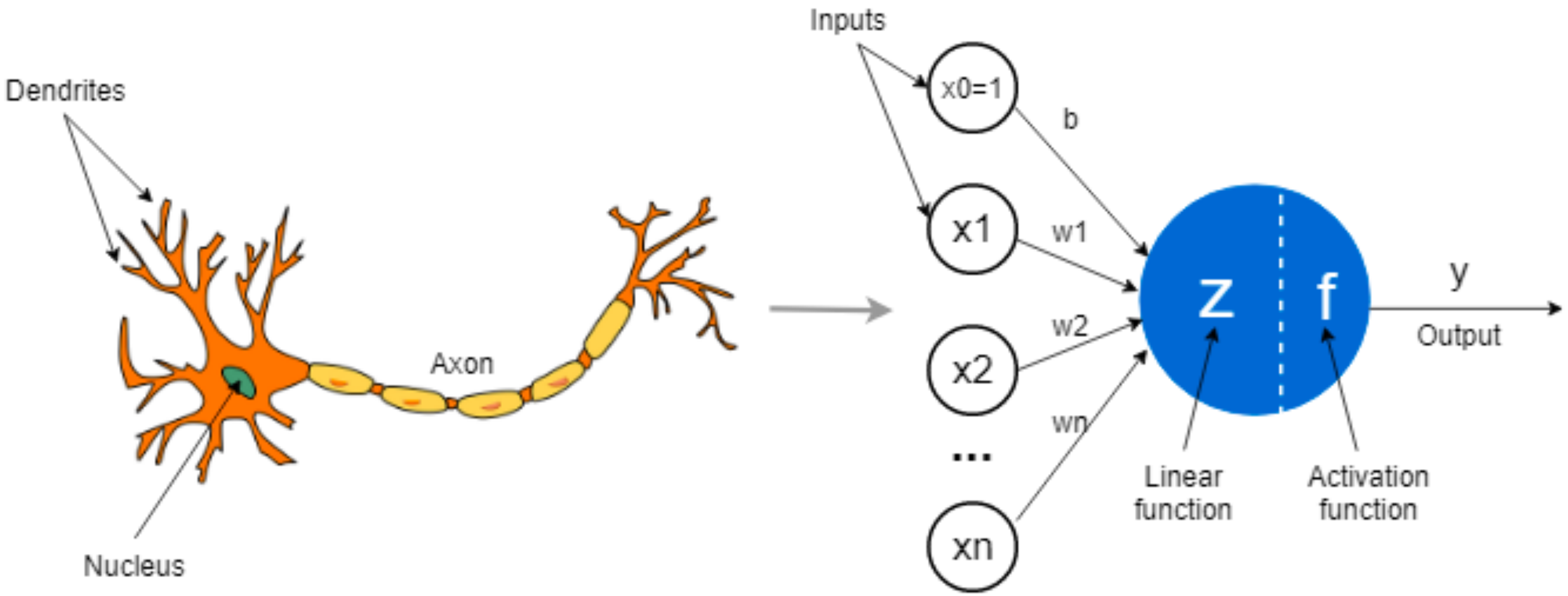


Figure credit: [link]

Brain v.s. Computer

- Brain
  - Network of neurons
  - Nerve signals propogate via neural network
  - Parallel computation
  - Robust: neurons grow and die.
- Computer
  - (Electronic) Gates
  - Electrical signals directed by gates
  - Sequential and parallel computation
  - Fragile: halt if there is no power

Artificial Neural Networks

- **Key idea: emulate biological neurons for computation**

- ANN
  - Units are called ***nodes*** and correspond to neurons
  - Connections between nodes correspond to synapses

- ANN v.s. Biological NN
  - Numerical signal transmitted between nodes corresponds to chemical signal between neurons
  - Nodes modifying numerical signal correspond to neurons activating gate

ANN terms

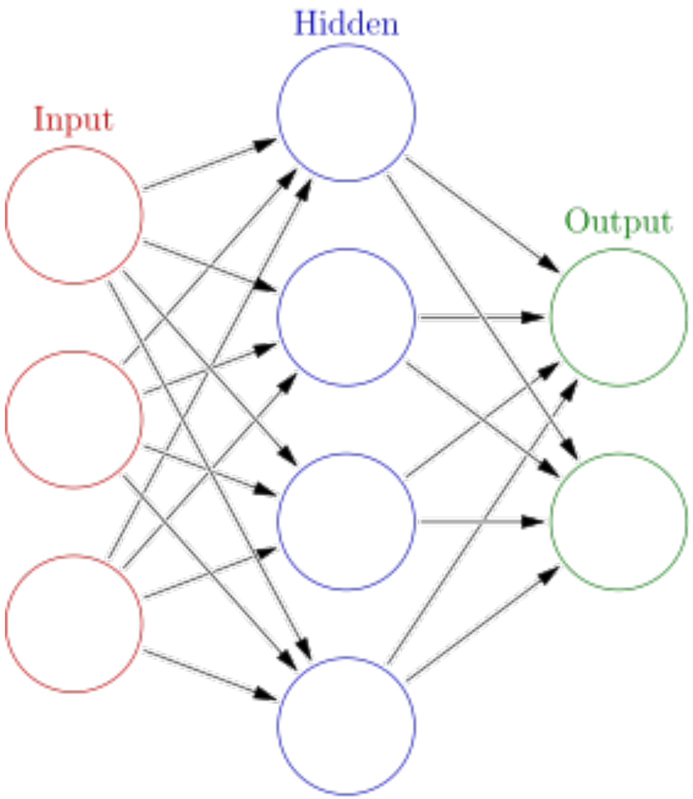
- Node: i
- Weights: W
  - Strength of the connection from node i to node j
  - Input signal x_i weighted by W_{ij} and linearly combined

$$a_j = \sum_i W_{ji} x_i + w_0 = \mathbf{W}_{ji}\mathbf{x}$$

- Activation function h
  - Produce numerical signal $y = h(a_j)$

Single-Layer Feed-forward Network

- Perceptron is the simplest type of ANN



Recall Supervised Learning

- Given a training sample $(\mathbf{x}, \mathbf{y})$
- Train perceptron, adjust weights $\mathbf{W}$ according to $(\mathbf{x}, \mathbf{y})$

VIRGINIA TECH.

Learning a Threshold Perceptron

- Learning is done separately for each output node j
  - Output nodes do not share weights
  - Assume output 1 or 0
- Perceptron learning for node j
  - For each $(\mathbf{x}, \mathbf{y})$ pair, repeat
    - Case 1: correct output
      - $\forall i W_{ji} \leftarrow W_{ji}$
    - Case 2: output produced 0(incorrect) instead of 1(correct)
      - $\forall i W_{ji} \leftarrow W_{ji} + x_i$
    - Case 3: output produced 1(incorrect) instead of 0(correct)
      - $\forall i W_{ji} \leftarrow W_{ji} - x_i$
  - Until correct output for all training samples.

Perceptron with a Threshold

b and w_0 are used interchangably

- Assume only one output node

$$f(x) = \mathbf{w}^T\mathbf{x} = \sum_i x_i w_i + b$$

Question: What if $f(x)$ equals to zero?

- With a threshold, a perceptron outputs $\begin{cases} 1, & f(x) > 0 \\ 0, & f(x) < 0 \end{cases}$
- Update $\mathbf{w}$
  - If output should be 1 instead of 0: $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}$ since $(\mathbf{w} + \mathbf{x})^T\mathbf{x} \geq \mathbf{w}^T\mathbf{x}$
  - If output should be 0 instead of 1: $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}$ since $(\mathbf{w} - \mathbf{x})^T\mathbf{x} \leq \mathbf{w}^T\mathbf{x}$

Alternative Approach

- Let $y \in \{-1,1\} \forall y$
- Let $M = \{\{\mathbf{x}_n, y_n\} \forall_n\}$ be set of misclassified samples.
  - i.e., $y_n * \mathbf{w}^T\mathbf{x} < 0$
- Misclassification error:
  - $E(\mathbf{w}) = -\sum_{(\mathbf{x}_n, y_n) \in M} y_n * \mathbf{w}^T\mathbf{x}$
- Now we can apply gradient descent algorithm
  - $\vec{w}^{(t+1)} \leftarrow \vec{w}^{(t)} - \eta \nabla E(\vec{w})$
  - $\nabla E = -\sum_{(\mathbf{x}_n, y_n) \in M} y_n * \mathbf{x}$

b is incorporated

Sequential Gradient Descent

- $$\nabla E = -\sum_{(\mathbf{x}_n, y_n) \in M} y_n * \mathbf{x}$$

- Adjust $\mathbf{w}$ based on one sample (x, y) at a time
- $\overrightarrow{w}^{(t+1)} \leftarrow \overrightarrow{w}^{(t)} - \eta y \mathbf{x}$

- When $\eta = 1$, turn into threshold perceptron algorithm

Threshold Perceptron Algorithm

- Let $y \in \{-1,1\} \forall y$
- Randomly Initialize weights $\mathbf{w}$

- Repeat until satisfied
  - For each training sample
    - $\hat{y} = sign(\mathbf{w}^T\mathbf{x})$ where $sign(a) = \begin{cases} 1, & x > 0 \\ -1, & otherwise \end{cases}$
    - If correct, no change
    - If wrong, update: $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$

Properties of Threshold Perceptron

- Binary classification
- A linear separator $\mathbf{w}^T\mathbf{x}$
- Converges *iff* the data are linearly separable

Sigmoid Perceptron

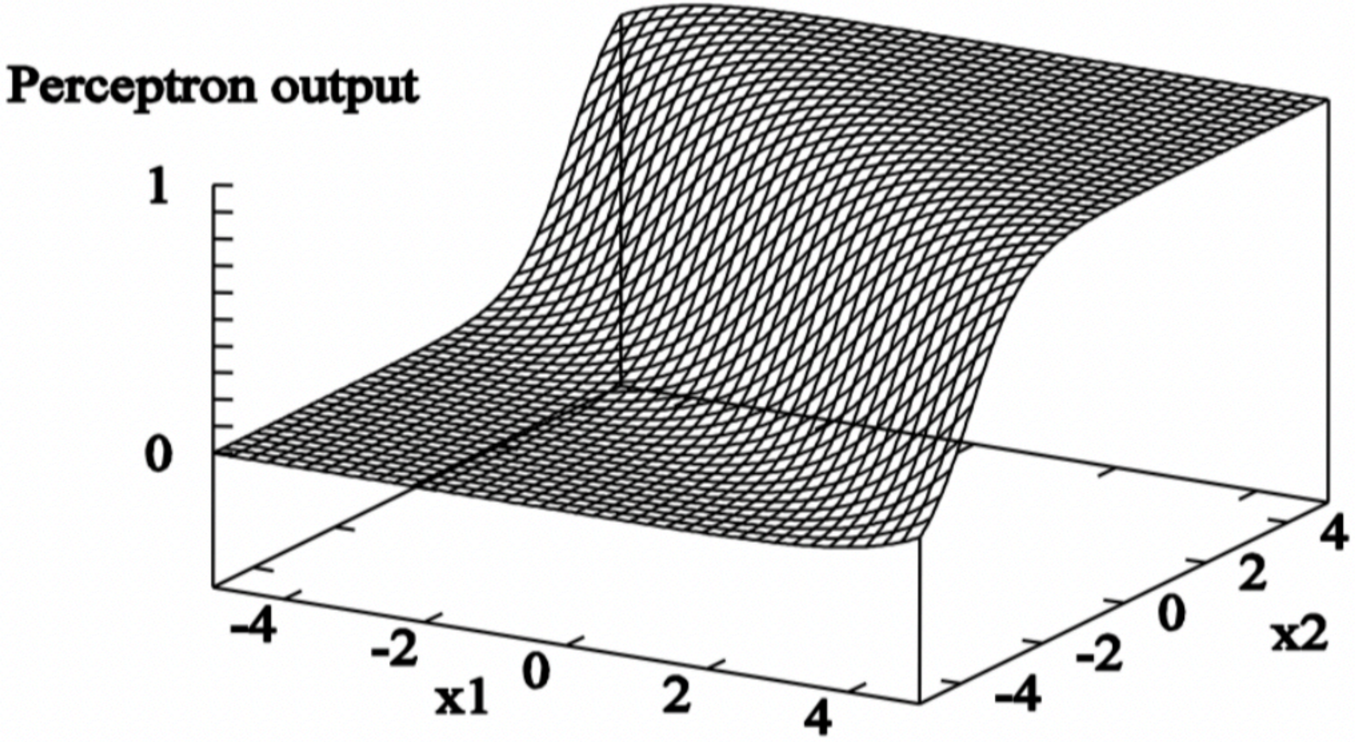


Figure credit: [link]()

Multilayer Networks

Ridge

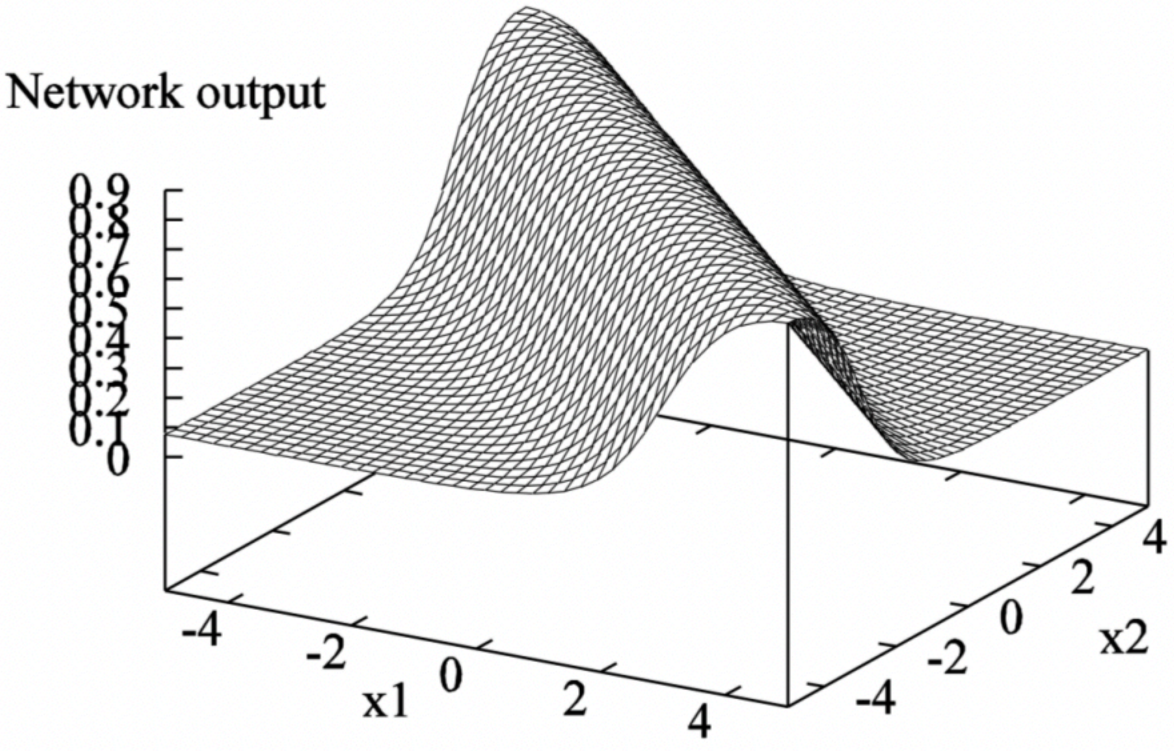
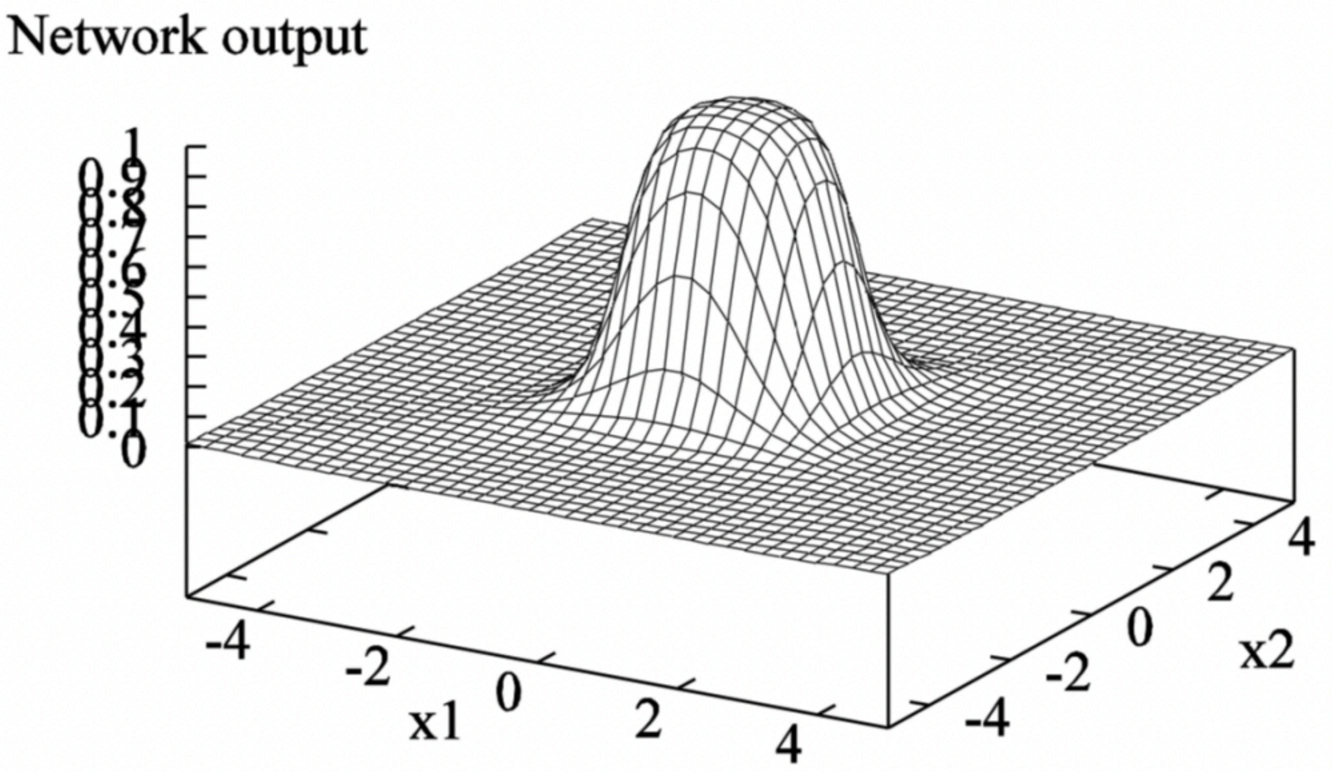


Figure credit: [link]

Multilayer Networks

Bump



Separate with Bumps

- A bump can classify linearly non-separable data points
- By tiling bumps of various heights together, we can approximate any function
- Demo: https://playground.tensorflow.org