Activation Functions

Natalie Parde

UIC CS 421

There are many different activation functions!

exponential linear unit (elu)

softmax

scaled exponential linear unit (selu)

softplus

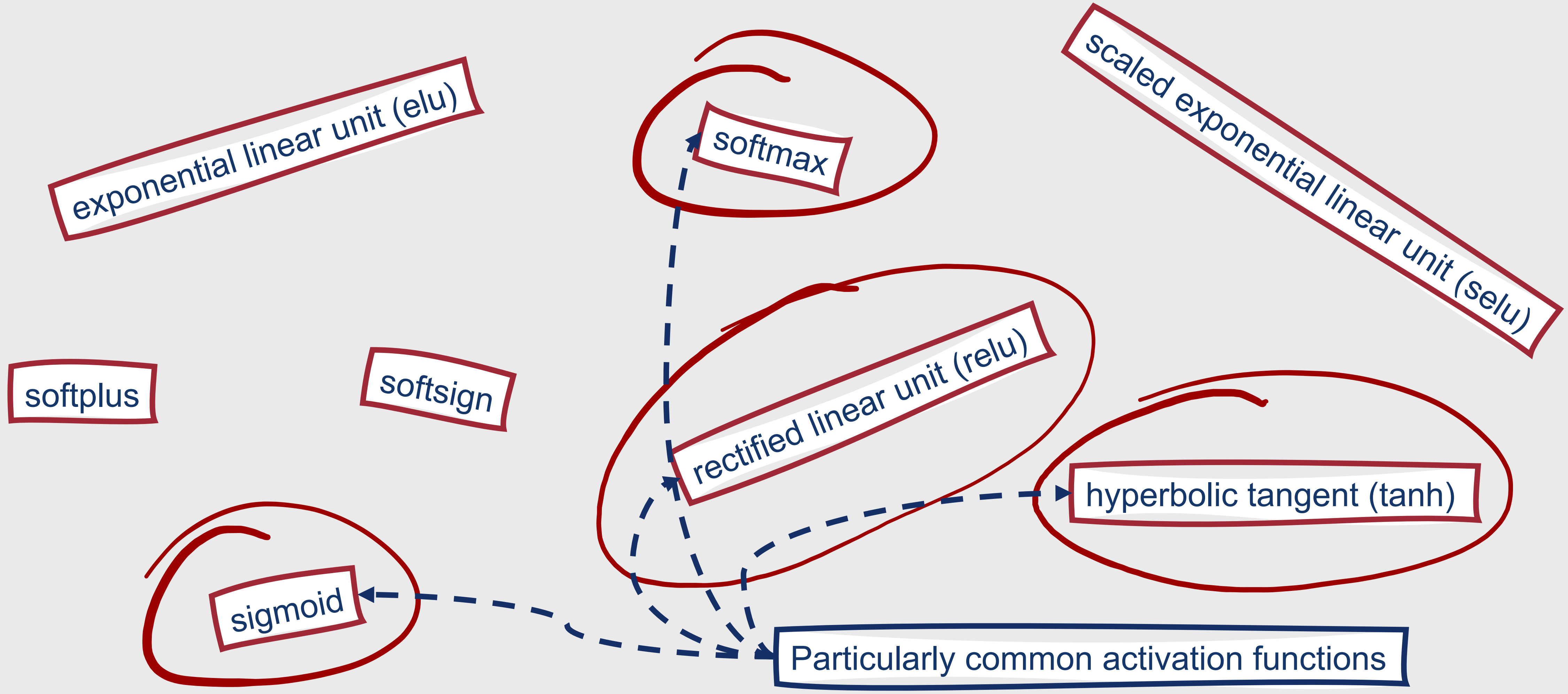
softsign

rectified linear unit (relu)

hyperbolic tangent (tanh)

sigmoid

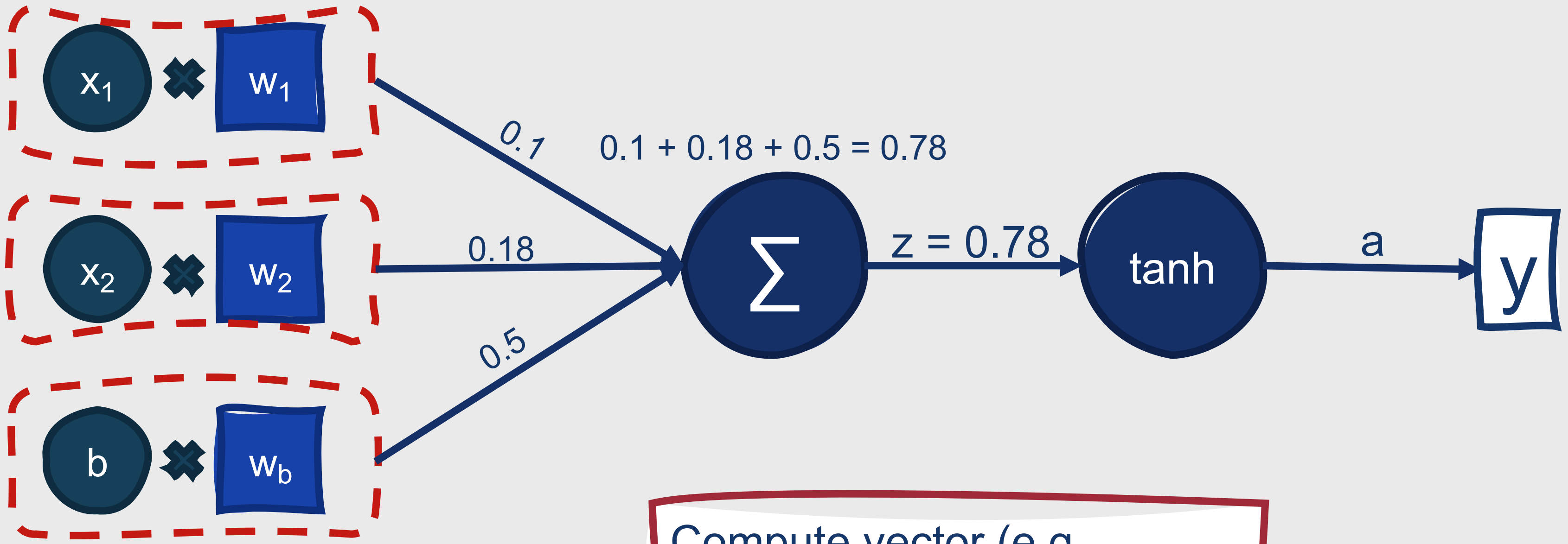
There are many different activation functions!


exponential linear unit (elu)
softmax
scaled exponential linear unit (selu)
softplus
softsign
rectified linear unit (relu)
hyperbolic tangent (tanh)
sigmoid
Particularly common activation functions

Activation: tanh

- Variant of sigmoid that ranges from -1 to +1
  - $y = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
- Once again differentiable
- Larger derivatives → generally faster convergence

Example: Computational Unit with tanh Activation



Input: “beautiful brutalist architecture”

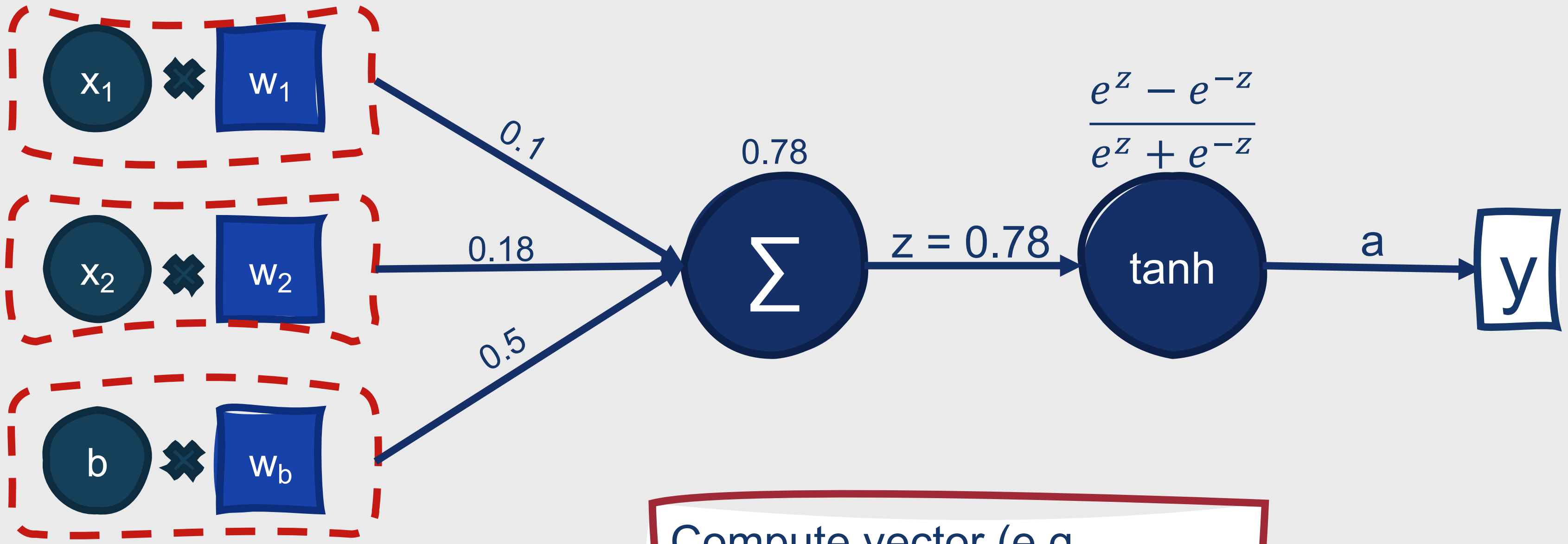
Compute vector (e.g., averaged Word2Vec embeddings for “beautiful,” “brutalist,” and “architecture”)

[0.5, 0.6]

Weights (Input): [0.2, 0.3]
Weight (Bias): [0.5]

Bias: 1.0

Example: Computational Unit with tanh Activation



Input: "beautiful brutalist architecture"

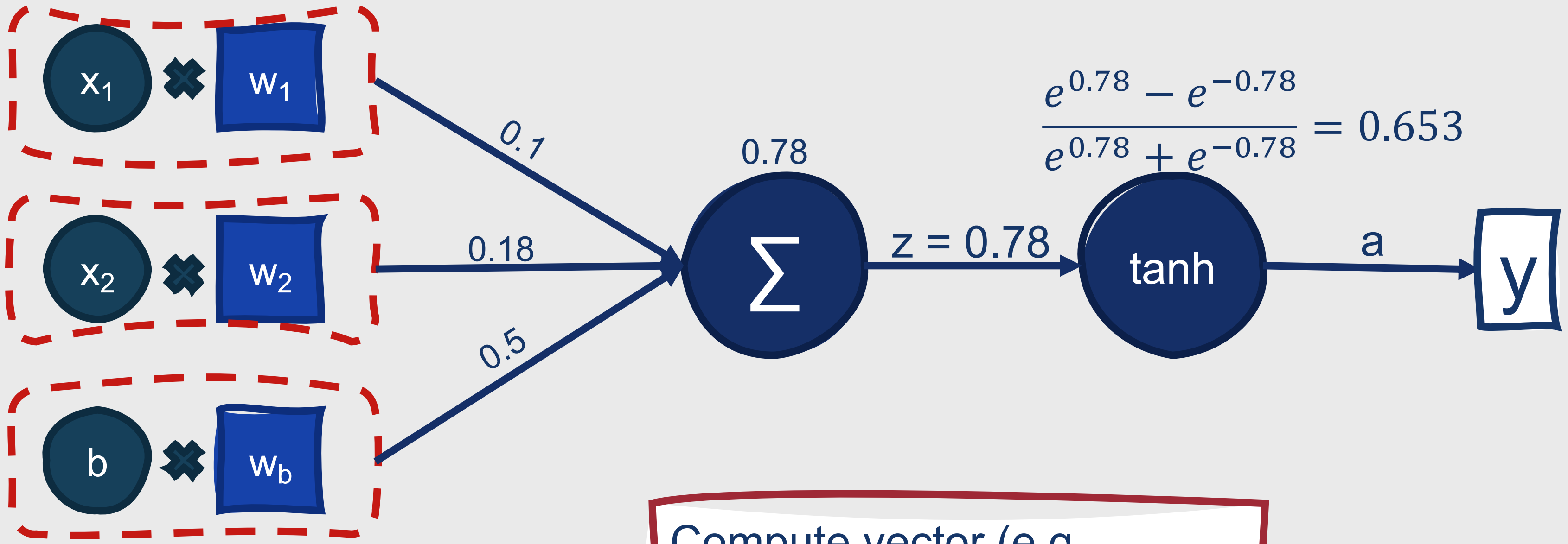
Compute vector (e.g., averaged Word2Vec embeddings for "beautiful," "brutalist," and "architecture")

[0.5, 0.6]

Weights (Input): [0.2, 0.3]
Weight (Bias): [0.5]

Bias: 1.0

Example: Computational Unit with tanh Activation



Input: "beautiful brutalist architecture"

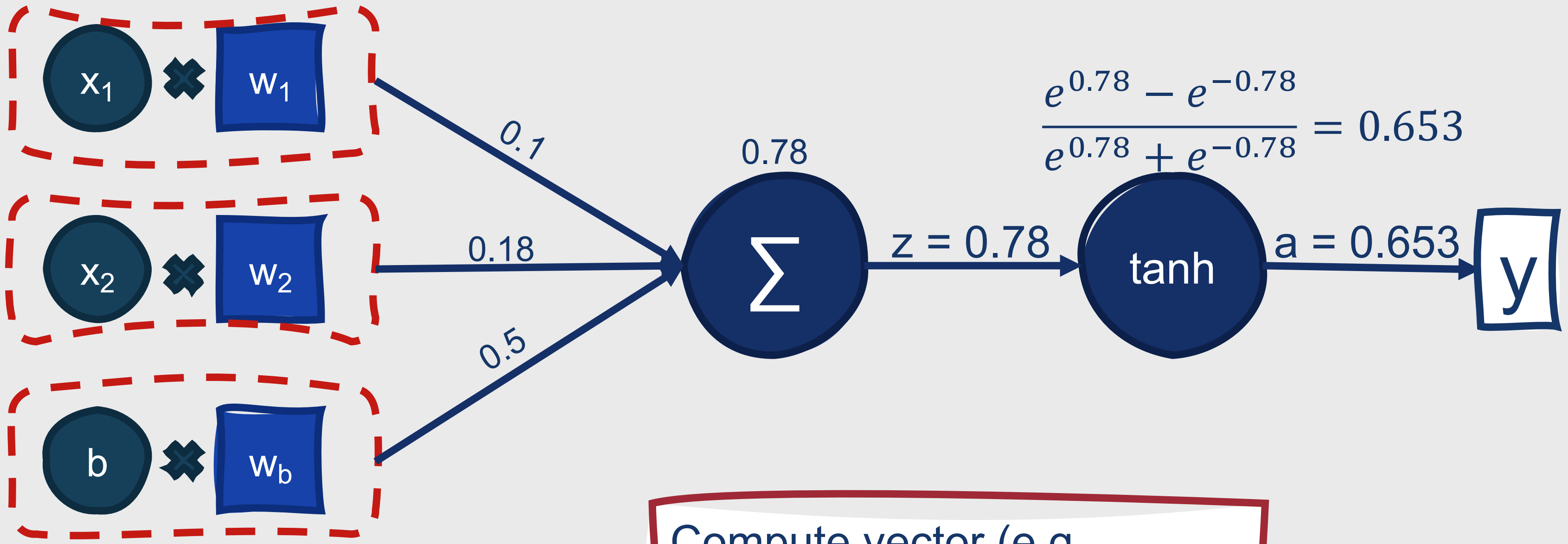
Compute vector (e.g., averaged Word2Vec embeddings for "beautiful," "brutalist," and "architecture")

[0.5, 0.6]

Weights (Input): [0.2, 0.3]
Weight (Bias): [0.5]

Bias: 1.0

Example: Computational Unit with tanh Activation

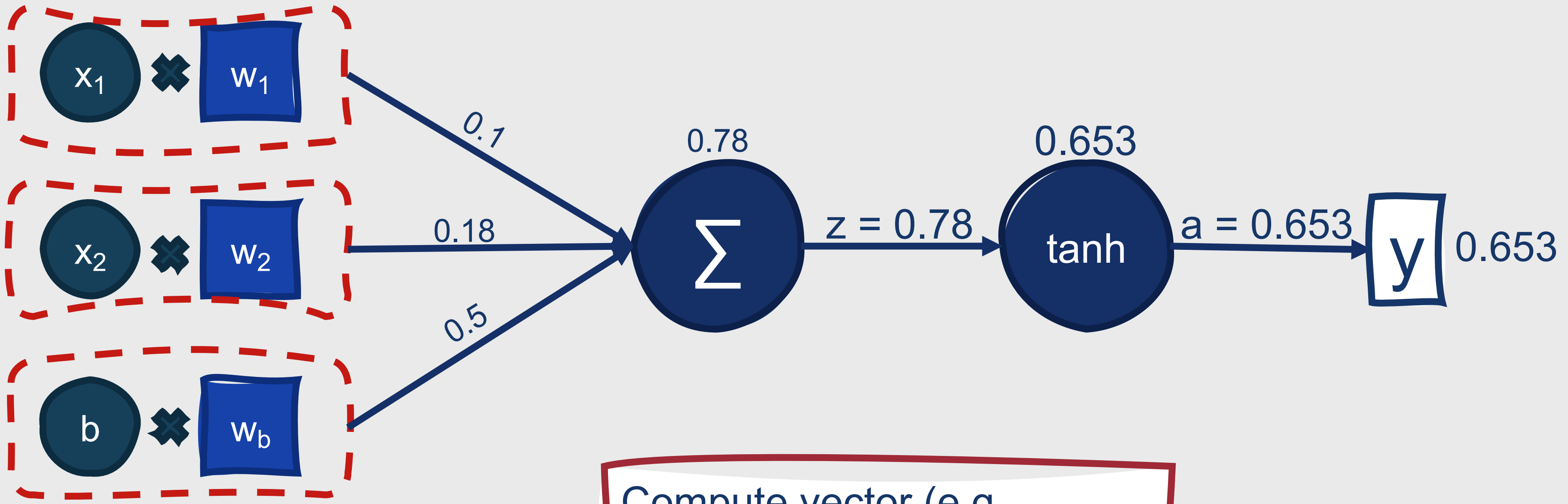


Input: “beautiful brutalist architecture” → Compute vector (e.g., averaged Word2Vec embeddings for “beautiful,” “brutalist,” and “architecture”) → [0.5, 0.6]

Weights (Input): [0.2, 0.3]
Weight (Bias): [0.5]

Bias: 1.0

Example: Computational Unit with tanh Activation



Input: “beautiful brutalist architecture” → Compute vector (e.g., averaged Word2Vec embeddings for “beautiful,” “brutalist,” and “architecture”) → [0.5, 0.6]

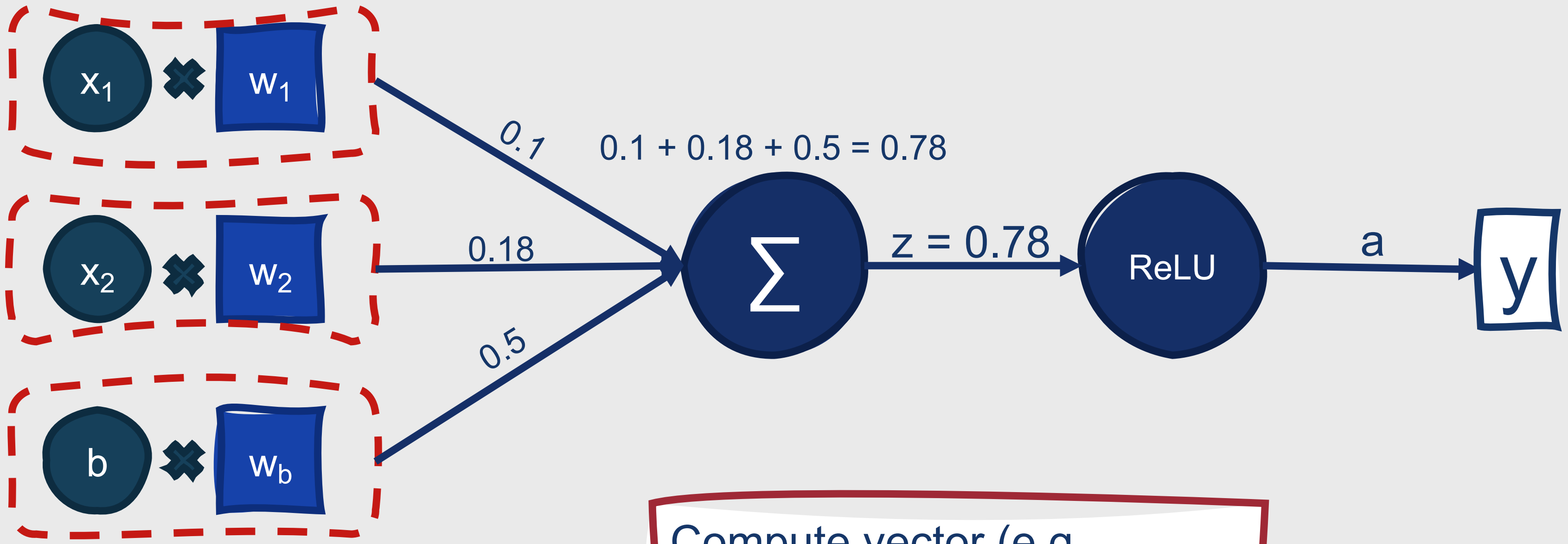
Weights (Input): [0.2, 0.3]
Weight (Bias): [0.5]

Bias: 1.0

Activation: ReLU

- Ranges from 0 to ∞
- Simplest activation function:
  - $y = \max(z, 0)$
- Very close to a linear function!
- Quick and easy to compute

Example: Computational Unit with ReLU Activation

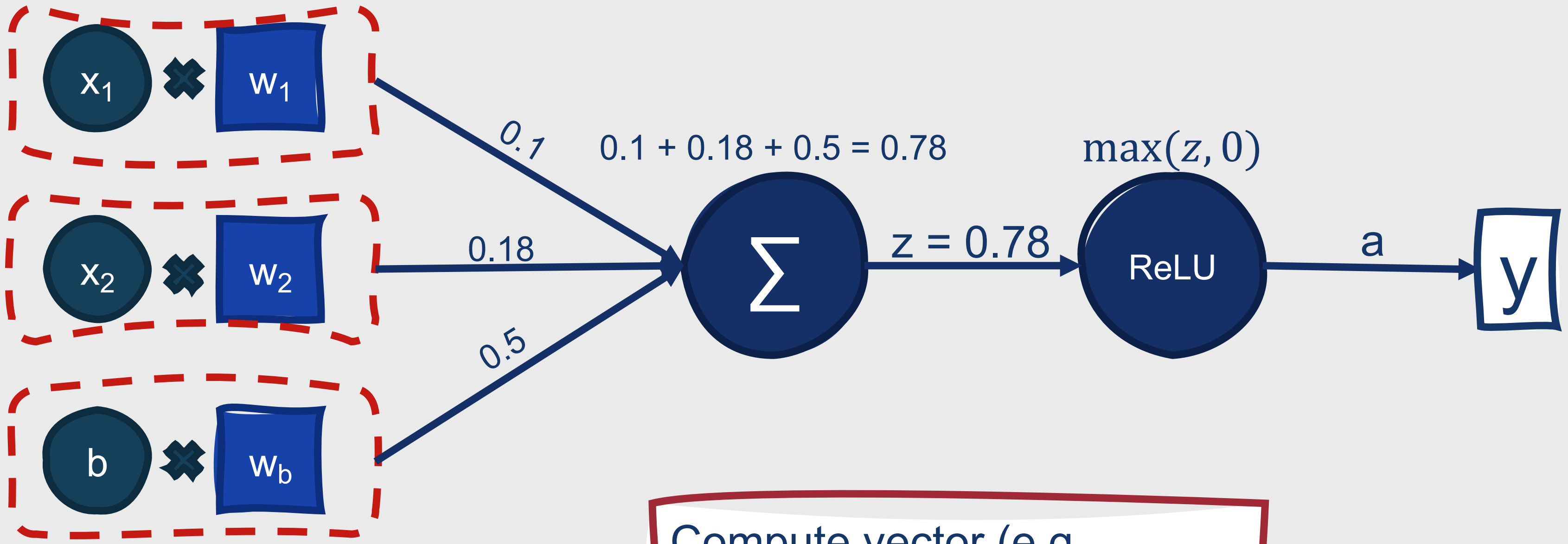


Input: “beautiful brutalist architecture” → Compute vector (e.g., averaged Word2Vec embeddings for “beautiful,” “brutalist,” and “architecture”) → [0.5, 0.6]

Weights (Input): [0.2, 0.3]
Weight (Bias): [0.5]

Bias: 1.0

Example: Computational Unit with ReLU Activation



Input: “beautiful brutalist architecture”

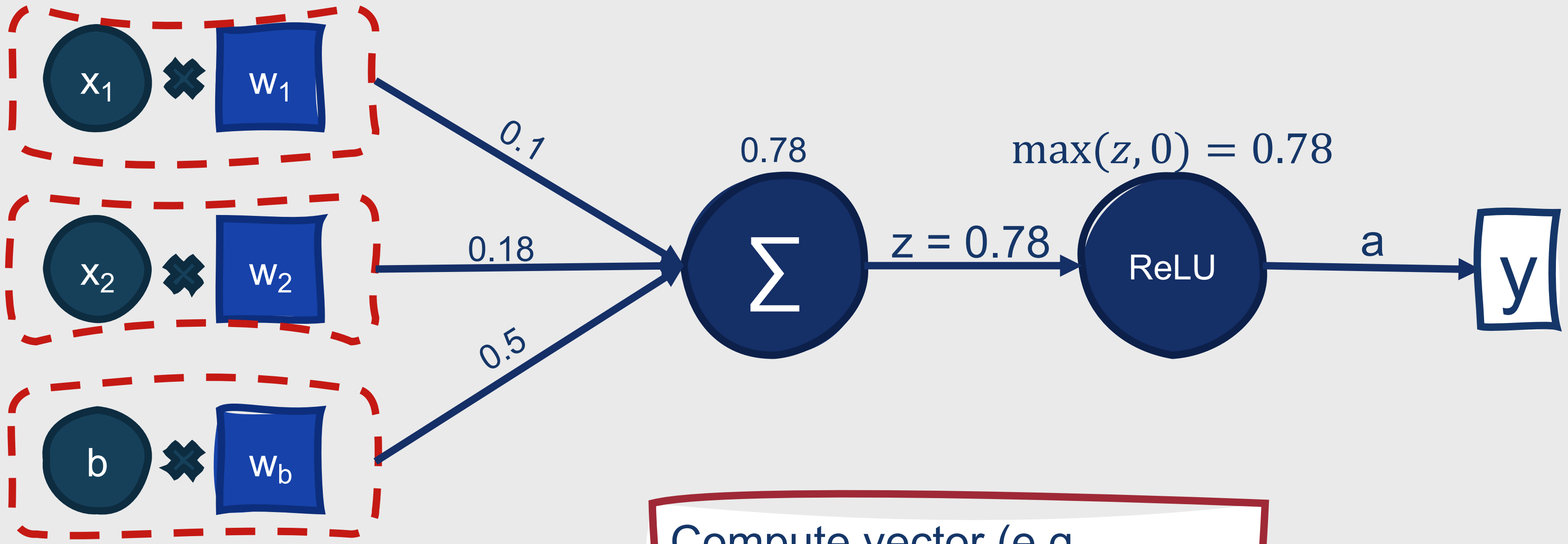
Compute vector (e.g., averaged Word2Vec embeddings for “beautiful,” “brutalist,” and “architecture”)

[0.5, 0.6]

Weights (Input): [0.2, 0.3]
Weight (Bias): [0.5]

Bias: 1.0

Example: Computational Unit with ReLU Activation

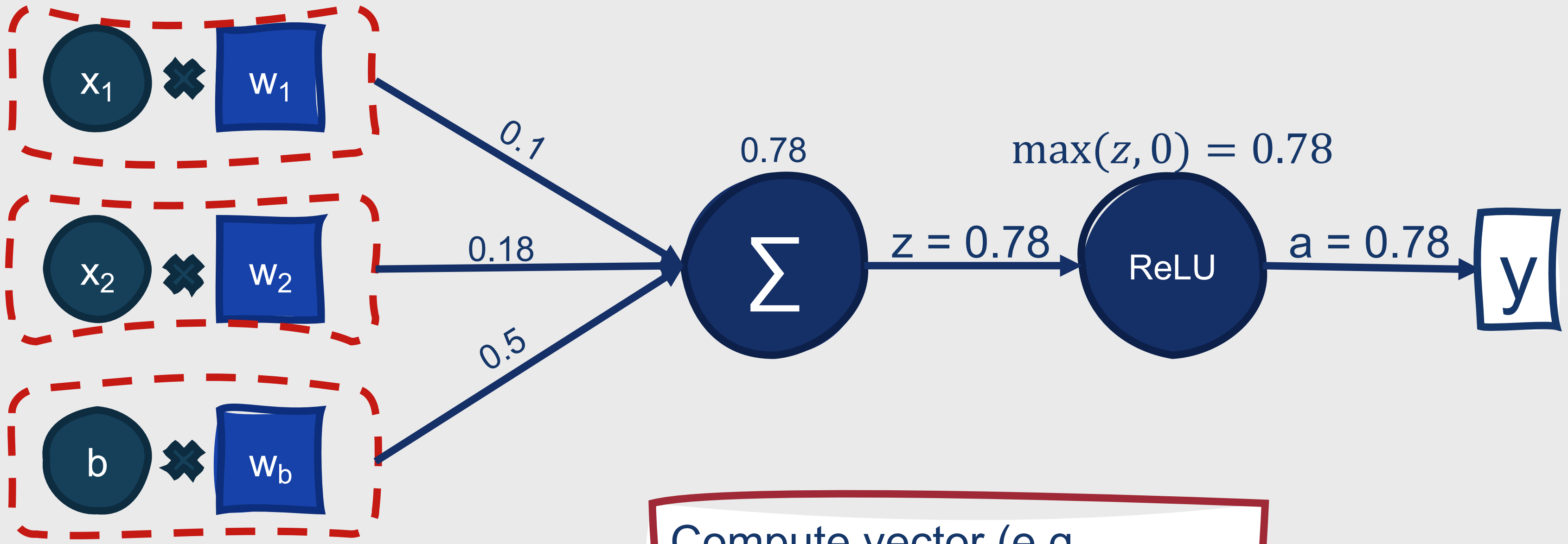


Input: "beautiful brutalist architecture" → Compute vector (e.g., averaged Word2Vec embeddings for "beautiful," "brutalist," and "architecture") → [0.5, 0.6]

Weights (Input): [0.2, 0.3]
Weight (Bias): [0.5]

Bias: 1.0

Example: Computational Unit with ReLU Activation

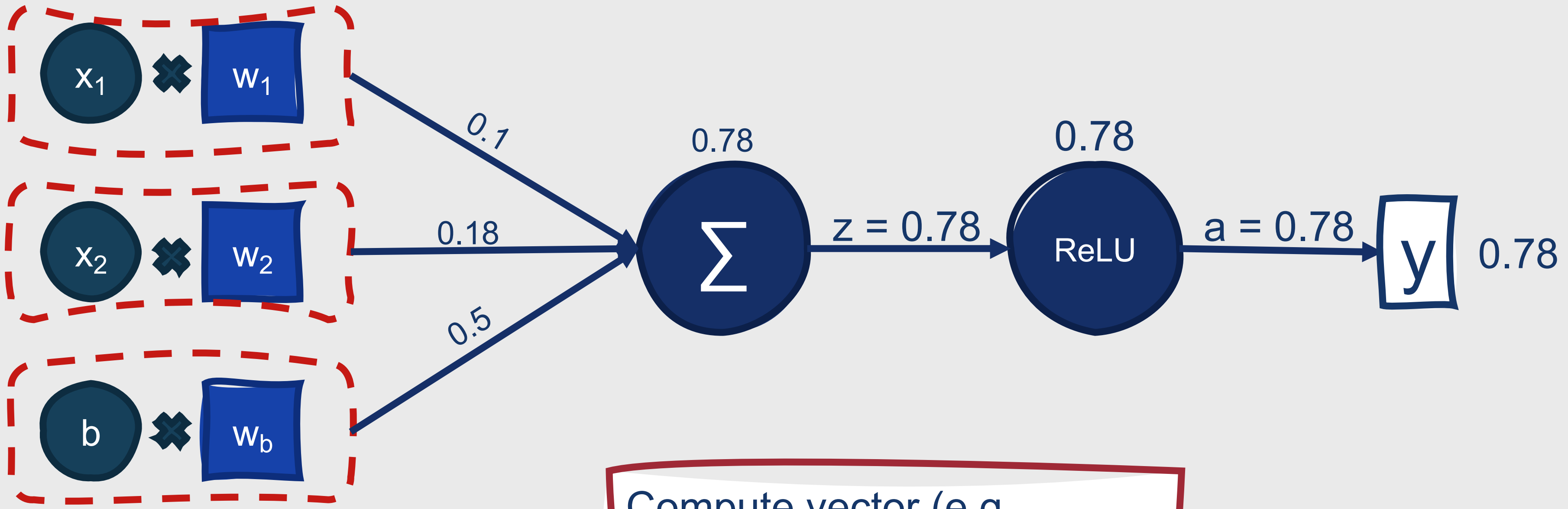


Input: “beautiful brutalist architecture” → Compute vector (e.g., averaged Word2Vec embeddings for “beautiful,” “brutalist,” and “architecture”) → [0.5, 0.6]

Weights (Input): [0.2, 0.3]
Weight (Bias): [0.5]

Bias: 1.0

Example: Computational Unit with ReLU Activation



Input: “beautiful brutalist architecture” → Compute vector (e.g., averaged Word2Vec embeddings for “beautiful,” “brutalist,” and “architecture”) → [0.5, 0.6]

Weights (Input): [0.2, 0.3]
Weight (Bias): [0.5]

Bias: 1.0

Comparing sigmoid, tanh, and ReLU

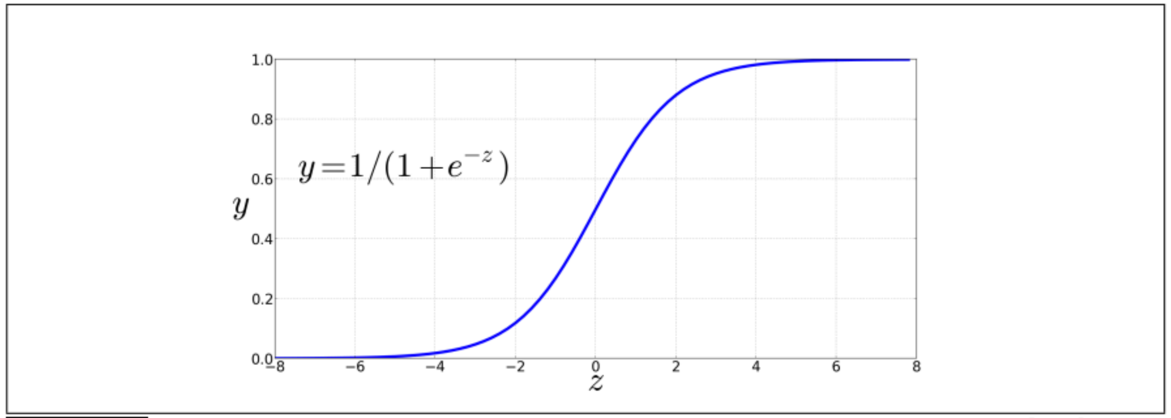


Figure 7.1 The sigmoid function takes a real value and maps it to the range [0,1]. It is nearly linear around 0 but outlier values get squashed toward 0 or 1.

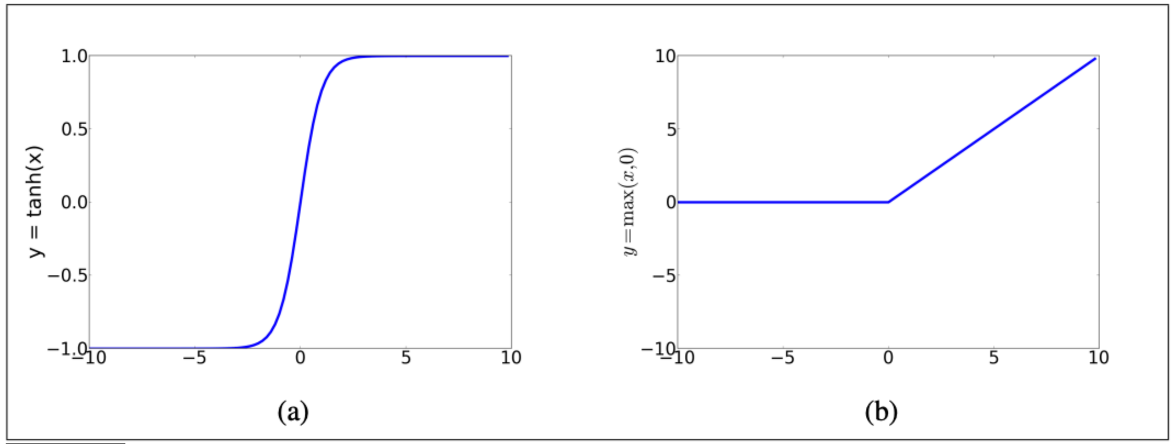


Figure 7.3 The tanh and ReLU activation functions.