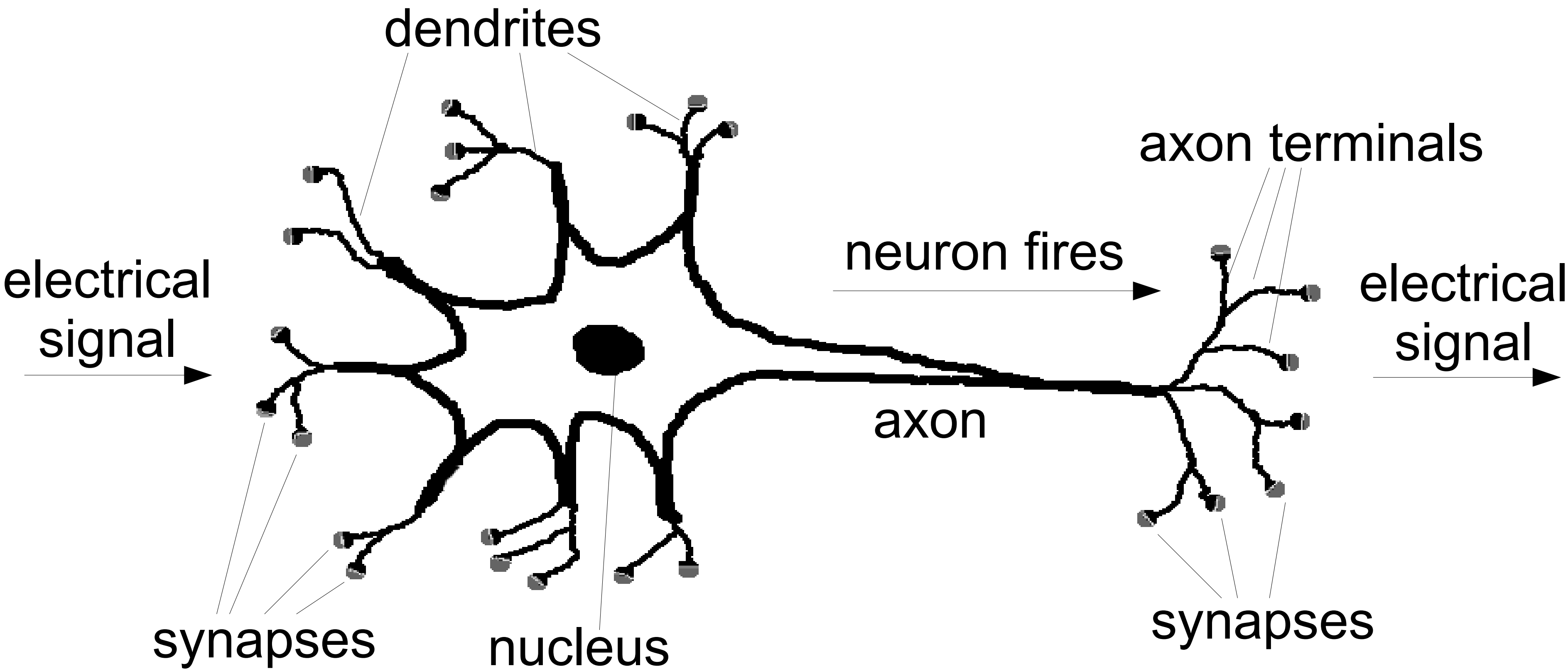
Introduction to Neural Networks 1

CS 486/686

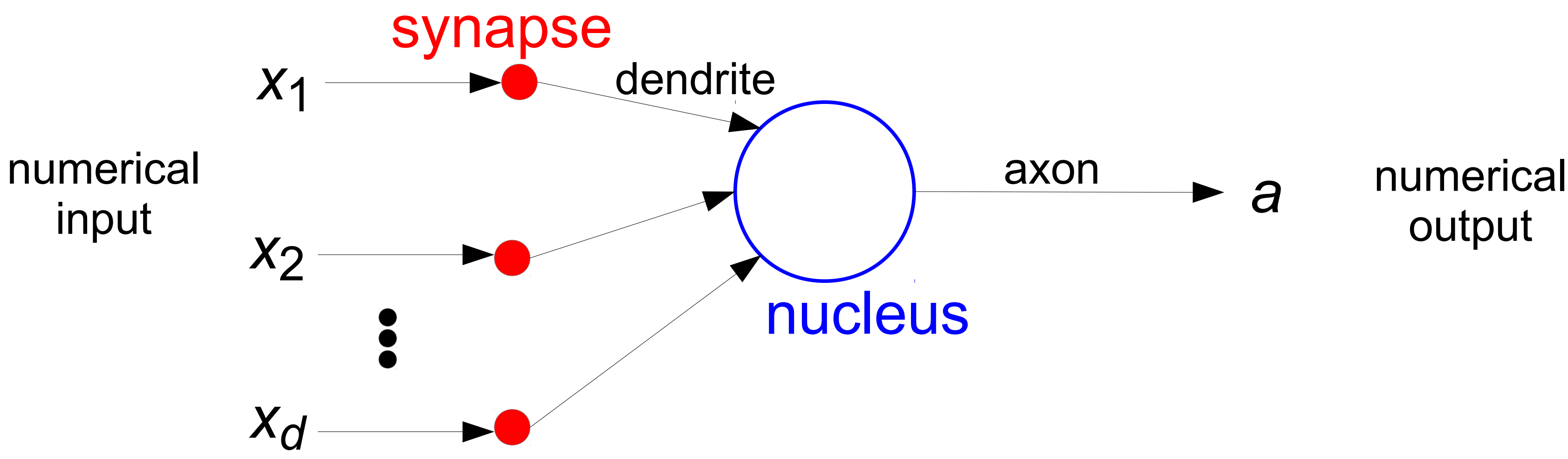
University of Waterloo

Lecture 19: July 7, 2015

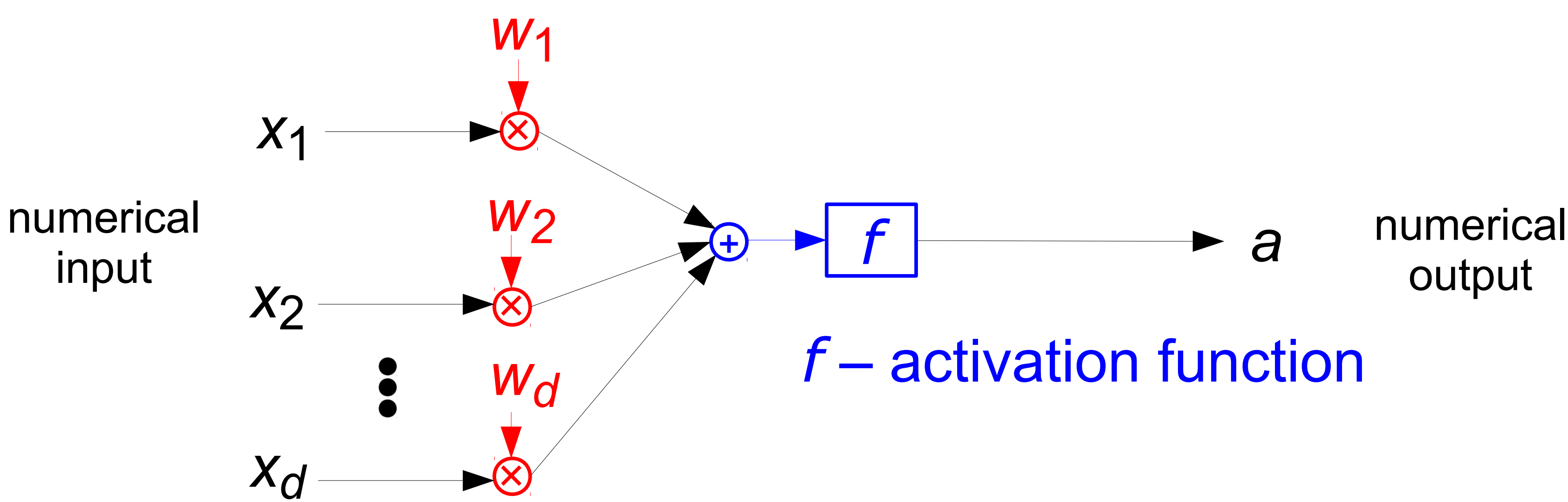
Biological Neuron



Computational Model of Neuron



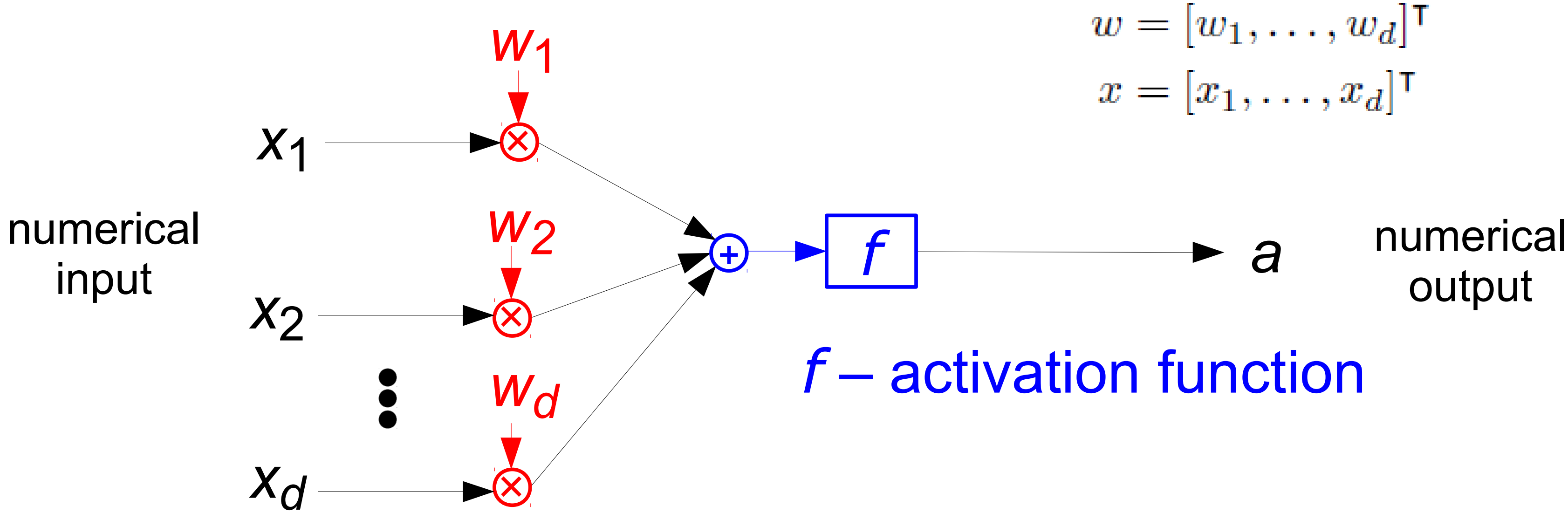
Computational Model of Neuron



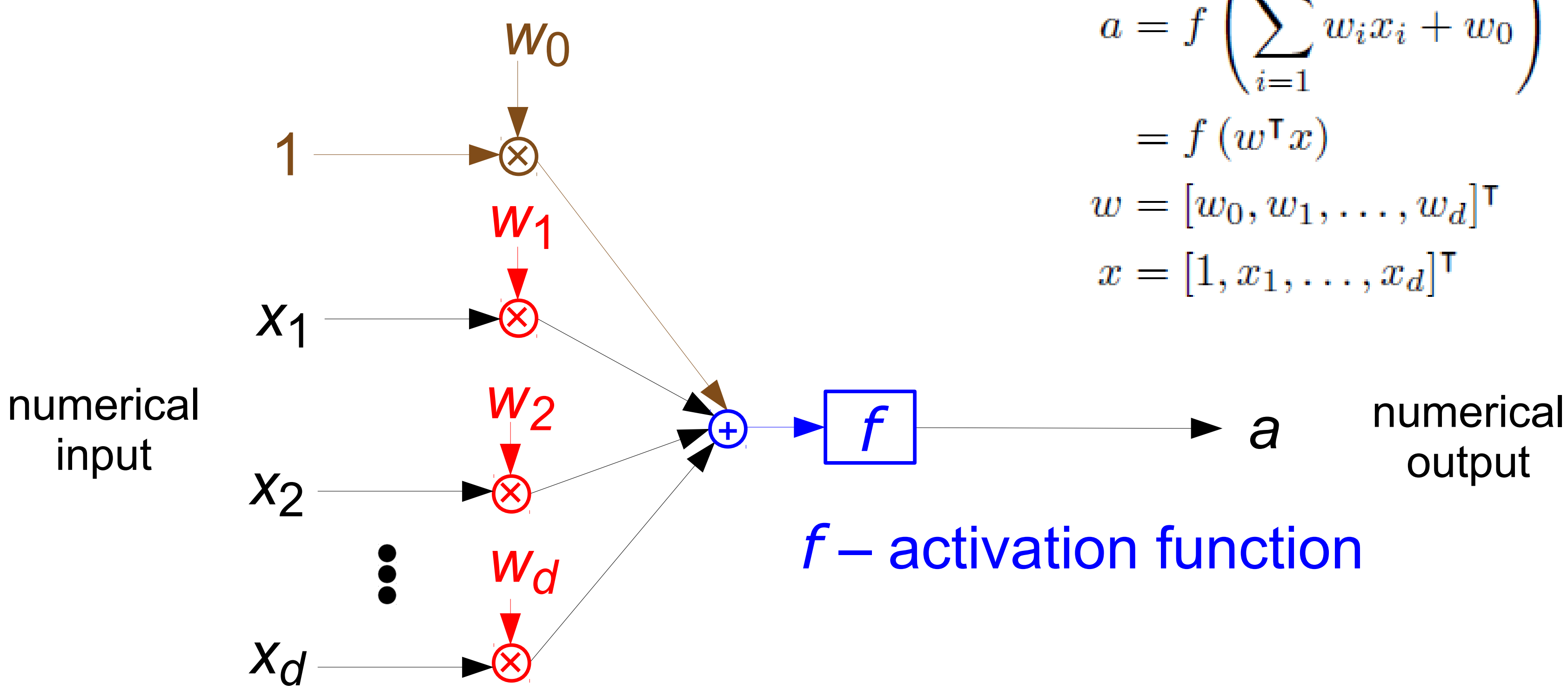
f – activation function

Computational Model of Neuron

$$a = f\left(\sum_{i=1}^{d} w_i x_i\right)$$
$$= f\left(w^\intercal x\right)$$
$$w = [w_1, \ldots, w_d]^\intercal$$
$$x = [x_1, \ldots, x_d]^\intercal$$



Computational Model of Neuron



$$
\begin{aligned}
a &= f\left(\sum_{i=1}^{d} w_i x_i + w_0\right) \\
&= f\left(w^{\mathsf{T}} x\right) \\
w &= [w_0, w_1, \ldots, w_d]^{\mathsf{T}} \\
x &= [1, x_1, \ldots, x_d]^{\mathsf{T}}
\end{aligned}
$$

f – activation function

Neuron as Binary Classifier

- Step activation function

$$f(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

(threshold perceptron)

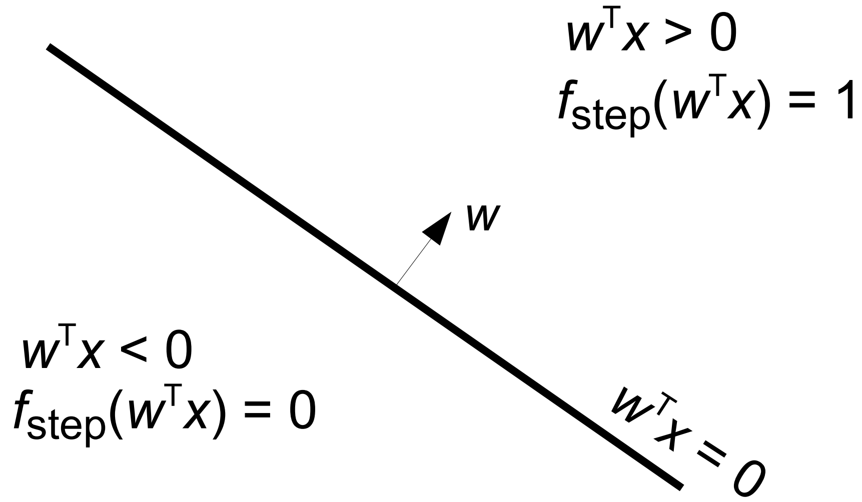
- Sigmoid activation function

$$f(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

(sigmoid perceptron / logistic regression)

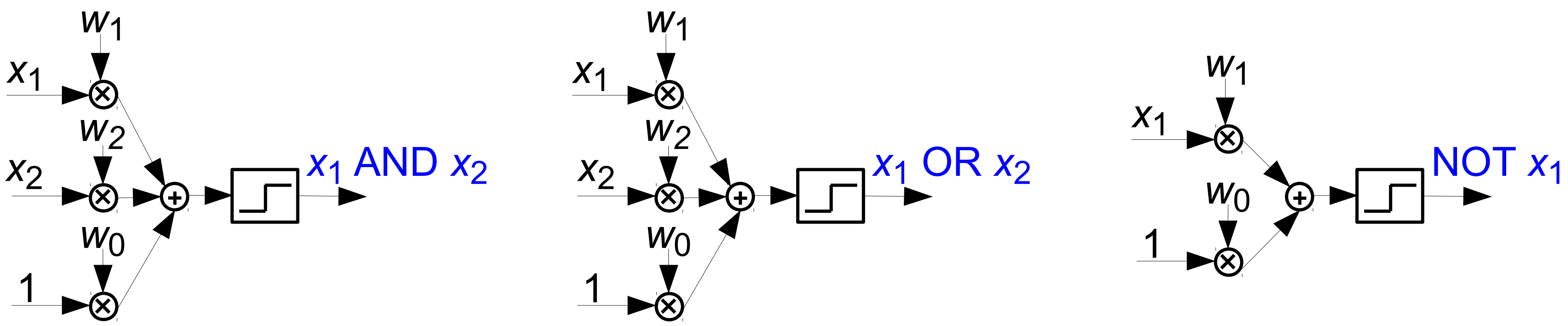
Hypothesis Space of Perceptron

- Linear separator



Threshold Perceptron as Logic Gate

- What should the weights be to encode the Boolean functions AND, OR, and NOT?



- How about XOR?

Threshold Perceptron Learning

```
1: Initialize weights w
2: for each training example (x, y) do
3:     a ← f_step(wᵀx)
4:     w ← w + (y − a)x
5: end for
6: Repeat 2–5 until every example is classified correctly
```

Threshold Perceptron Learning

1: Initialize weights $\boldsymbol{w}$
2: **for** each training example $(\boldsymbol{x}, y)$ **do**
3:     $a \leftarrow f_{\text{step}}(\boldsymbol{w}^{\intercal}\boldsymbol{x})$
4:     $\boldsymbol{w} \leftarrow \boldsymbol{w} + (y - a)\boldsymbol{x}$
5: **end for**
6: Repeat 2–5 until every example is classified correctly

Threshold perceptron learning converges if data is linearly separable.

Sigmoid Perceptron Learning

- Minimize squared error:

$$E(w, x, y) = \frac{1}{2}\left[y - \sigma\left(w^{\intercal} x\right)\right]^2$$

$$\hat{w} = \arg\min_{w} E(w, x, y)$$

- Solution: gradient descent

Gradient Descent

- Gradient
  - $\nabla E(w_1, \ldots, w_d) = \left[\frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_d}\right]$
  - Points in the direction of steepest slope
- Gradient descent: move in the direction of negative gradient to find minimum

$$w \leftarrow w - \alpha \nabla E(w)$$

$$\alpha > 0: \text{ learning rate}$$

Sigmoid Perceptron Learning with Gradient Descent

1: Initialize weights $\boldsymbol{w}$
2: **for** each training example $(\boldsymbol{x}, y)$ **do**
3: $\quad z \leftarrow \boldsymbol{w}^\mathsf{T}\boldsymbol{x}$
4: $\quad \sigma'(z) \leftarrow \sigma(z)(1 - \sigma(z))$
5: $\quad \boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha \cdot (y - \sigma(z))\sigma'(z)\boldsymbol{x}$
6: **end for**
7: Repeat 2–6 until stopping criteria satisfied

Next: Multi-Layer Neural Networks

- Perceptron can only represent linear separators

- Need multiple layers to represent more complicated separators