IT/PC/B/T/411

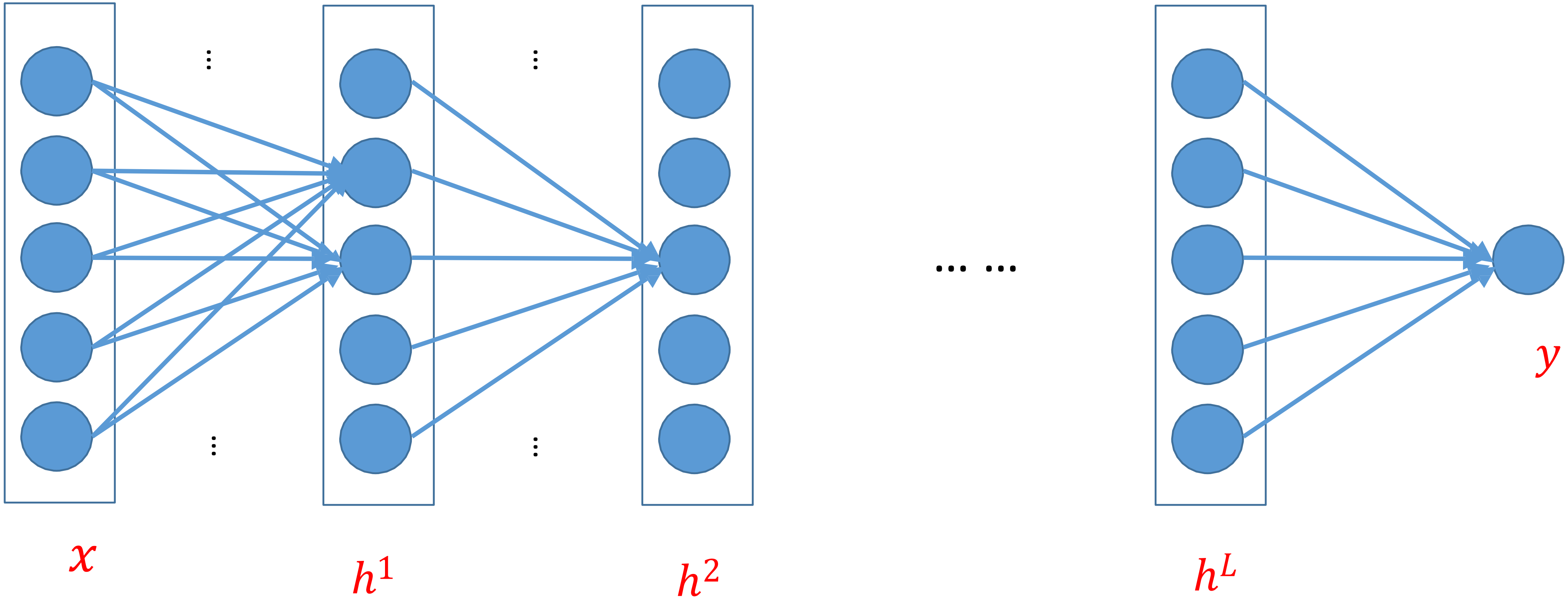
Machine Learning

Deep Learning Basics

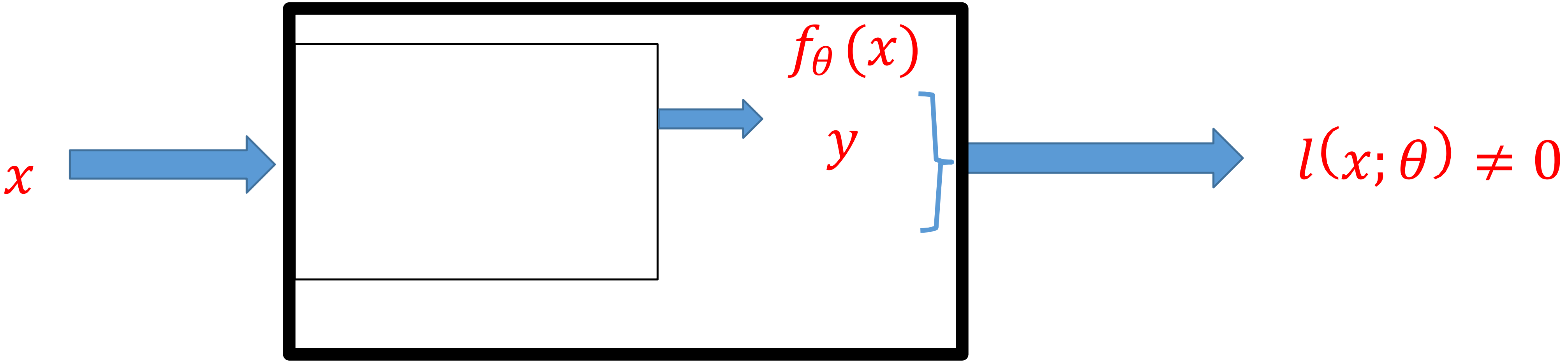
Lecture 02: Backpropagation

Dr. Pawan Kumar Singh
Department of Information Technology
Jadavpur University
pawankrsingh.cse@gmail.com
+91-6291555693

How to train the dragon?



How to get the expected output



Loss of the system

$$l(x;\theta) = l(f_\theta, x, y)$$

How to get the expected output

Find direction d so that:

x

$l(x; \theta + d) \approx 0$

Loss $l(x; \theta + d)$

How to get the expected output

How to find d: $l(x;\theta+\epsilon v) \approx l(x;\theta) + \nabla l(x;\theta) * \epsilon v$ for small scalar ϵ

x

$l(x;\theta+d) \approx 0$

Loss $l(x;\theta+d)$

How to get the expected output

Conclusion: Move θ along $-\nabla l(x;\theta)$ for a small amount

Loss $l(x;\theta+d)$

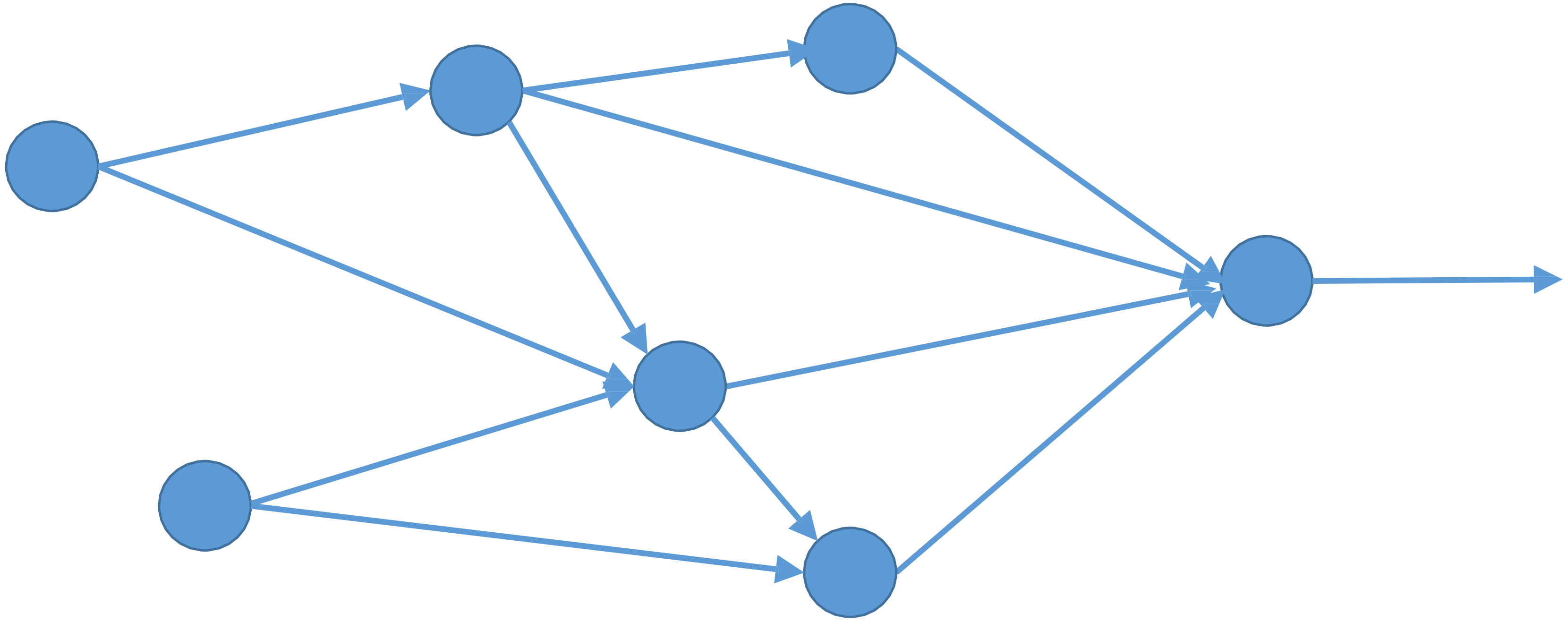
Neural Networks as real circuits

Pictorial illustration of gradient descent

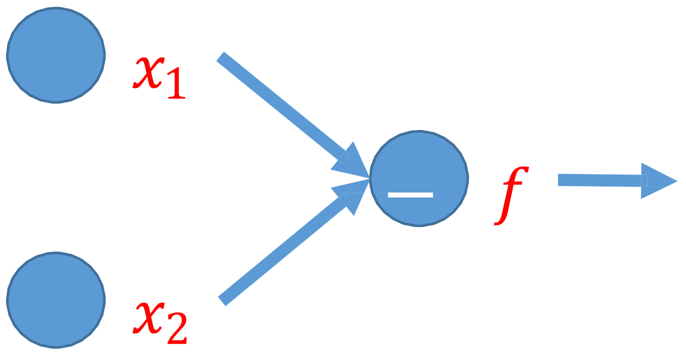
Gradient

- Gradient of the loss is simple
  - E.g., $l(f_\theta, x, y) = (f_\theta(x) - y)^2/2$
  - $\frac{\partial l}{\partial \theta} = (f_\theta(x) - y)\frac{\partial f_\theta}{\partial \theta}$
- Key part: gradient of the hypothesis

Open the box: real circuit

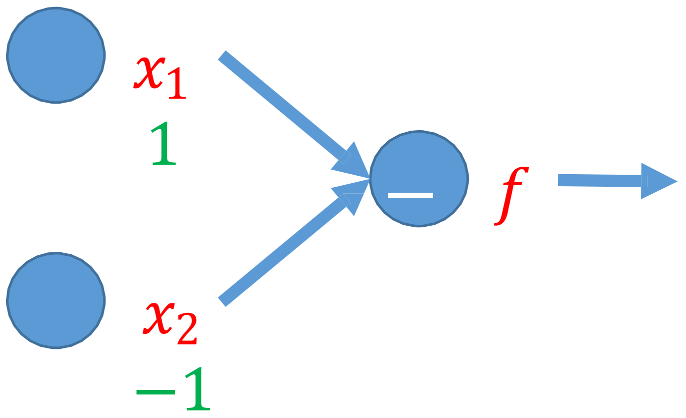


Single neuron



Function: $f = x_1 - x_2$

Single neuron

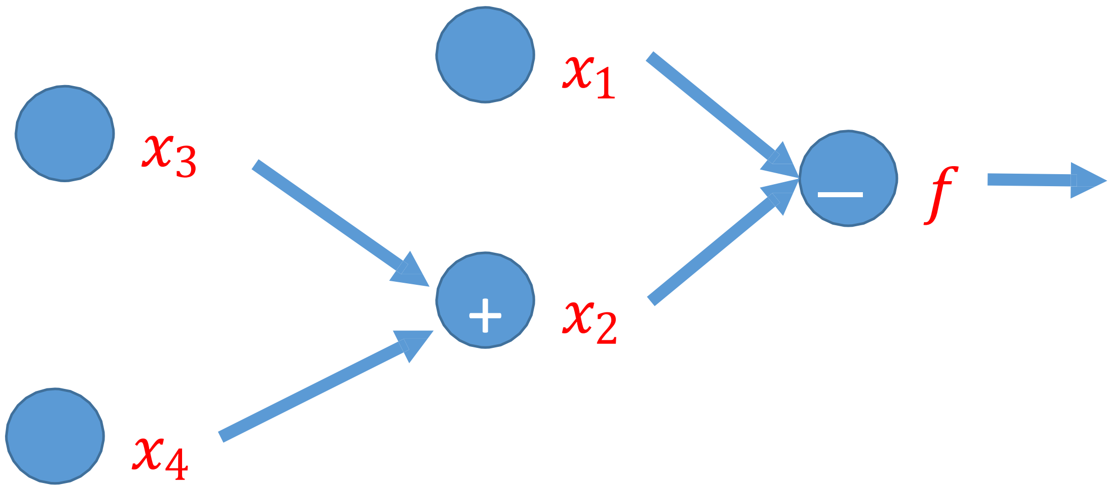


Function: $f = x_1 - x_2$

Gradient: $\frac{\partial f}{\partial x_1} = 1, \frac{\partial}{\partial x} = -1$

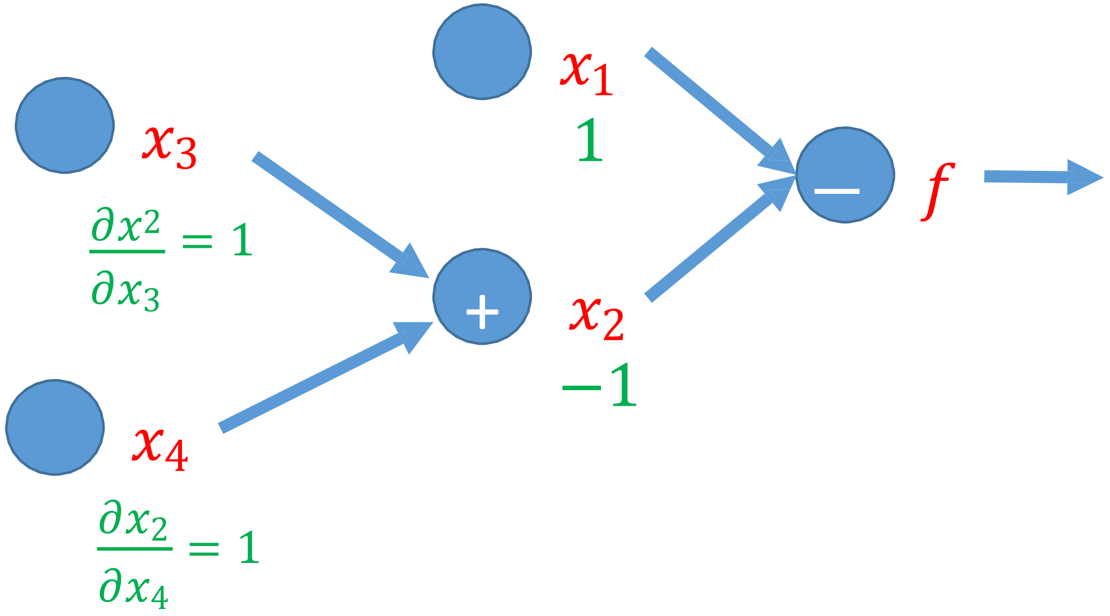
2

Two neurons



Function: $f = x_1 - x_2 = x_1 - (x_3 + x_4)$

Two neurons

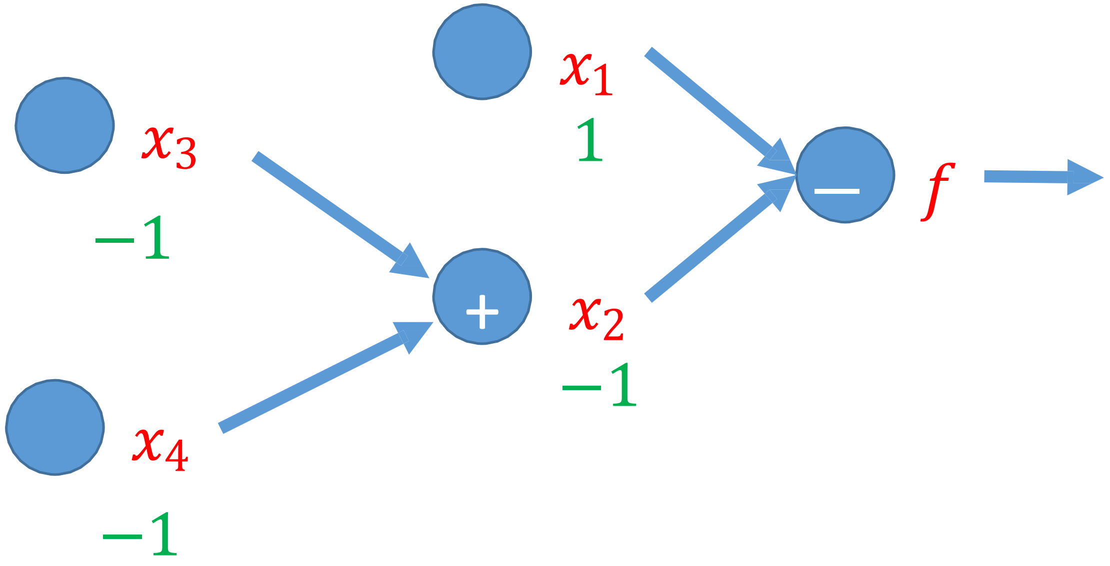


Function: $f = x_1 - x_2 = x_1 - (x_3 + x_4)$

Gradient: $\frac{\partial x_2}{\partial x_3} = 1, \frac{\partial x}{\partial x} = 1$. What about $\frac{\partial f}{\partial x}$?

4 3

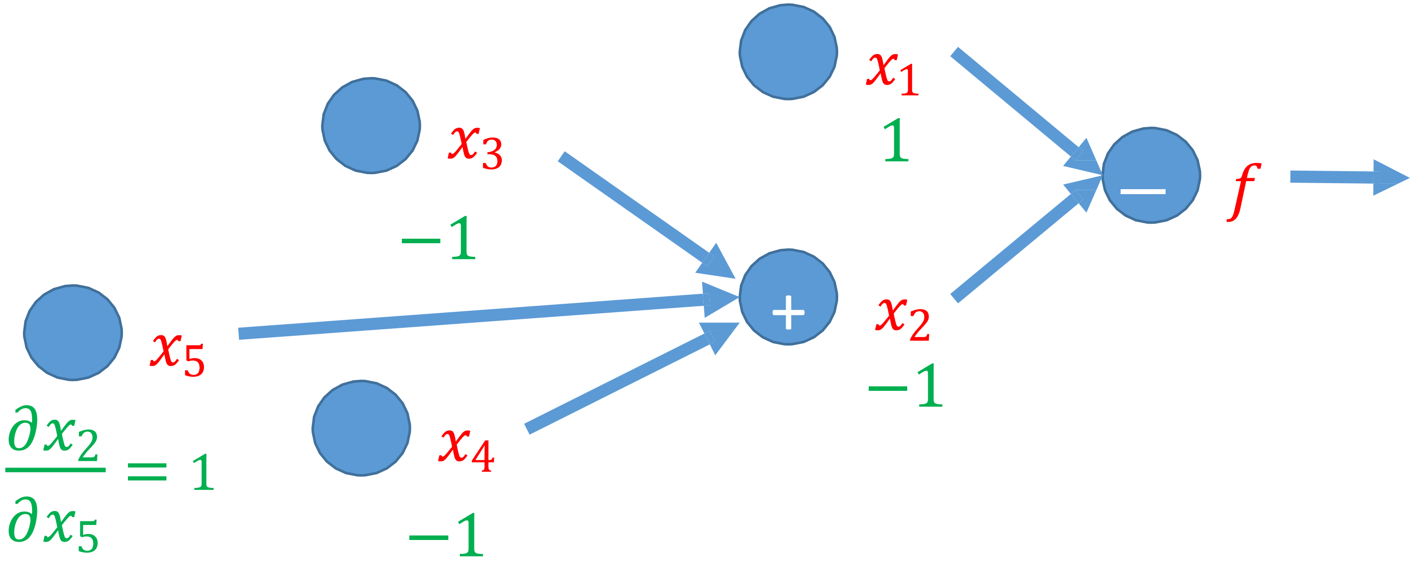
Two neurons



Function: $f = x_1 - x_2 = x_1 - (x_3 + x_4)$

Gradient: $\frac{\partial f}{\partial x_3} = \frac{\partial f}{\partial x_2}\frac{\partial x}{\partial x} = -1$

Multiple input

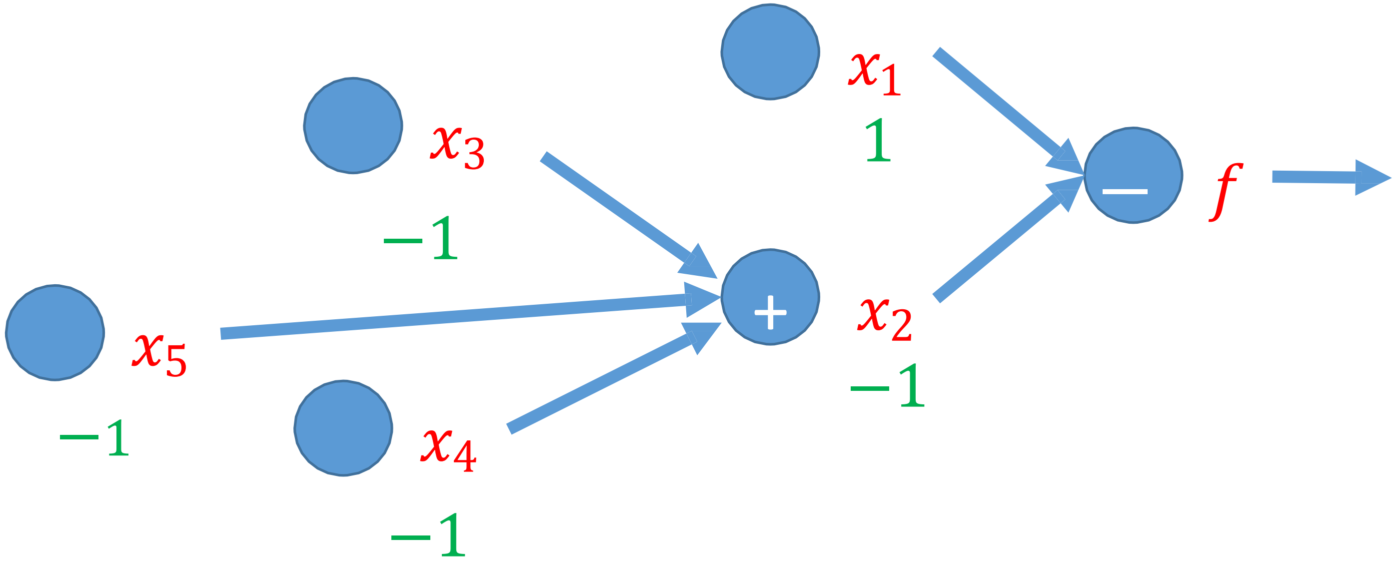


Function: $f = x_1 - x_2 = x_1 - (x_3 + x_5 + x_4)$

Gradient: $\frac{\partial x}{\partial x} = 1$

5

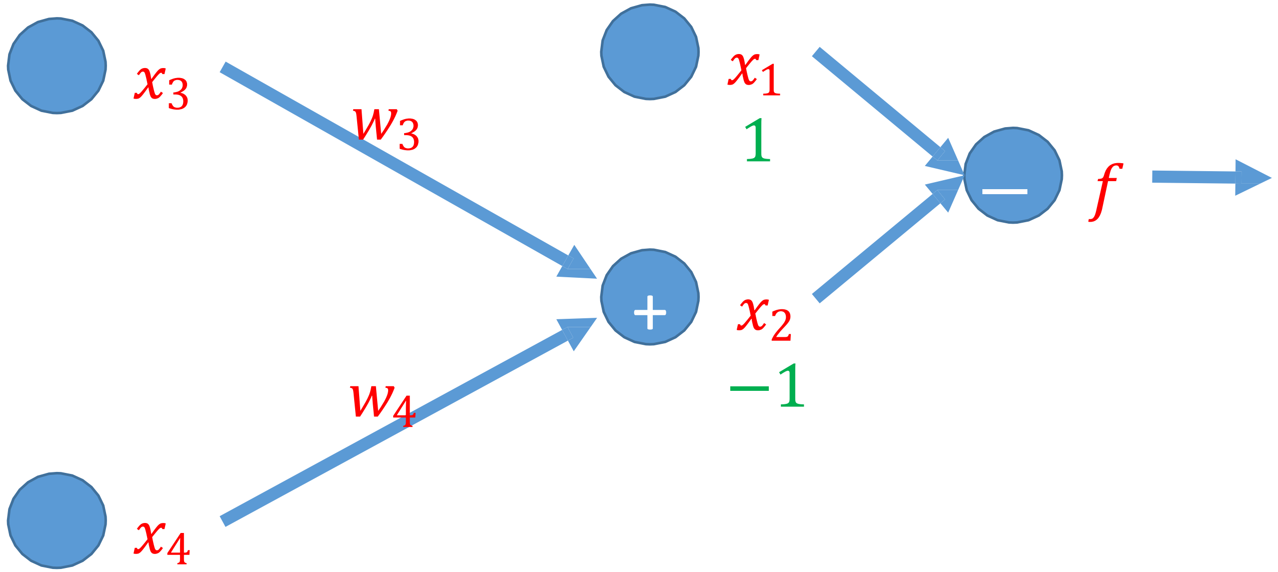
Multiple input



Function: $f = x_1 - x_2 = x_1 - (x_3 + x_5 + x_4)$

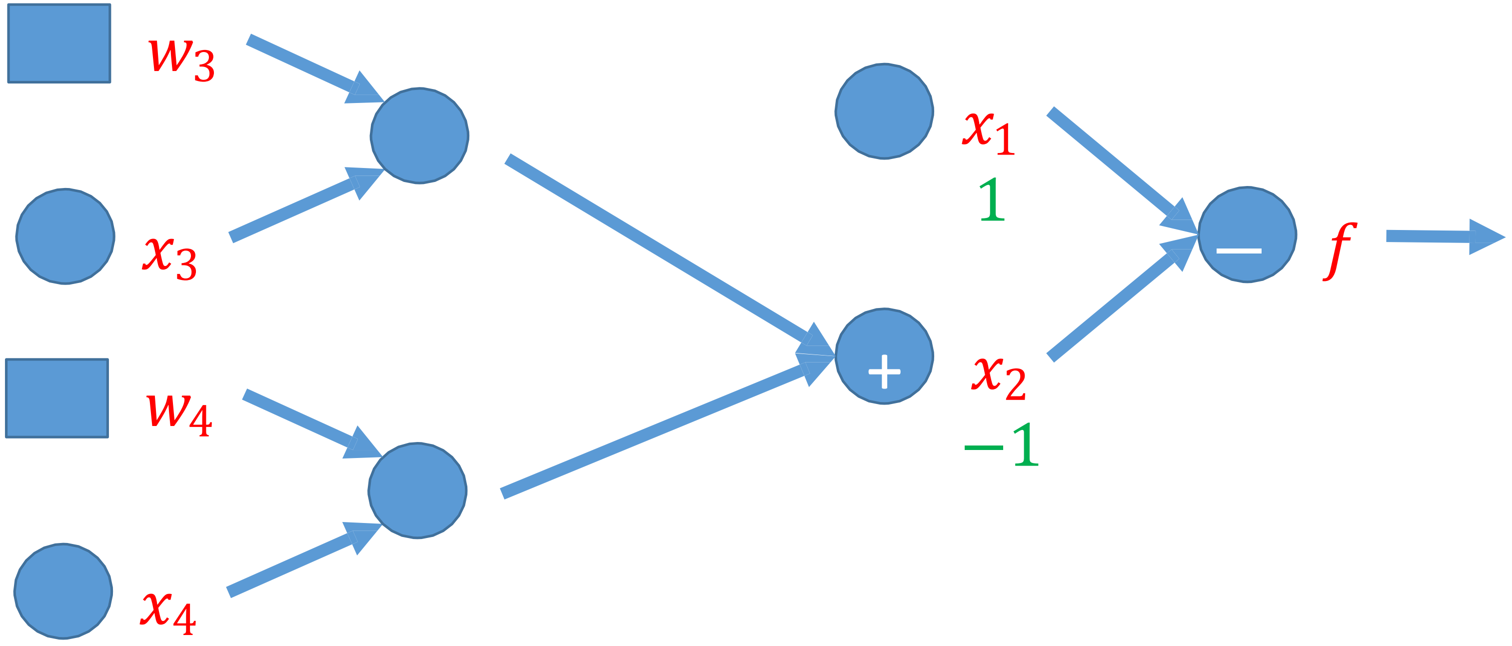
Gradient: $\frac{\partial f}{\partial x_5} = \frac{\partial f}{\partial x_3} \frac{\partial x_5}{\partial x} = -1$

Weights on the edges



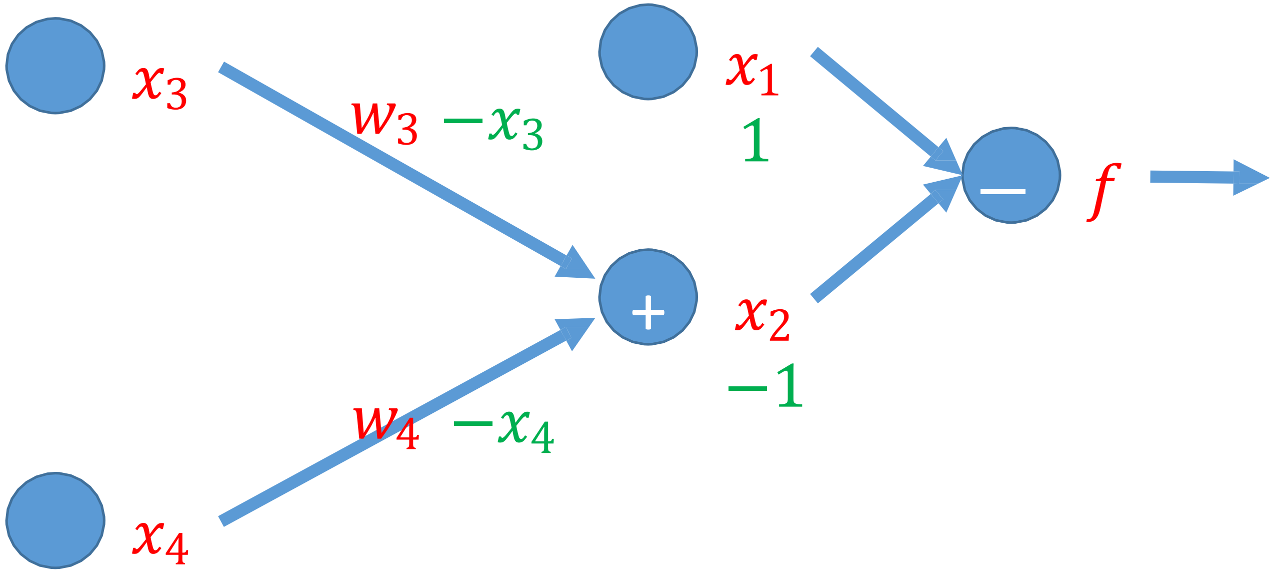
Function: $f = x_1 - x_2 = x_1 - (w_3 x_3 + w_4 x_4)$

Weights on the edges



Function: $f = x_1 - x_2 = x_1 - (w_3 x_3 + w_4 x_4)$

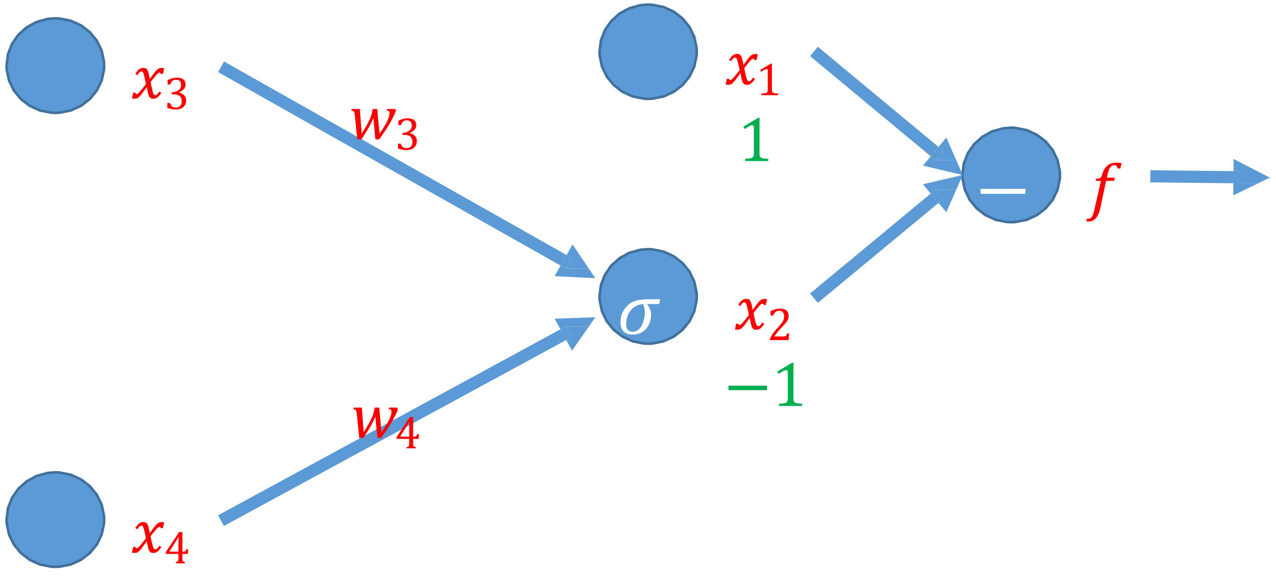
Weights on the edges



Function: $f = x_1 - x_2 = x_1 - (w_3 x_3 + w_4 x_4)$

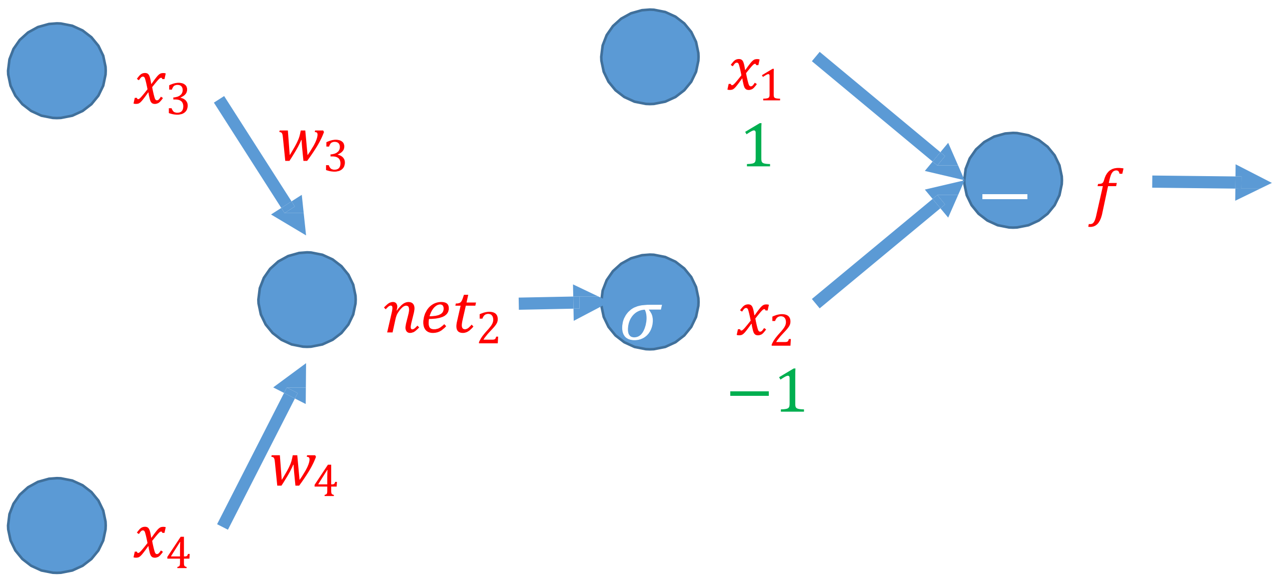
Gradient: $\frac{\partial f}{\partial w_3} = \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial w_3} = -1 \times x_3 = -x_3$

Activation



Function: $f = x_1 - x_2 = x_1 - \sigma\left(w_3x_3 + w_4x_4\right)$

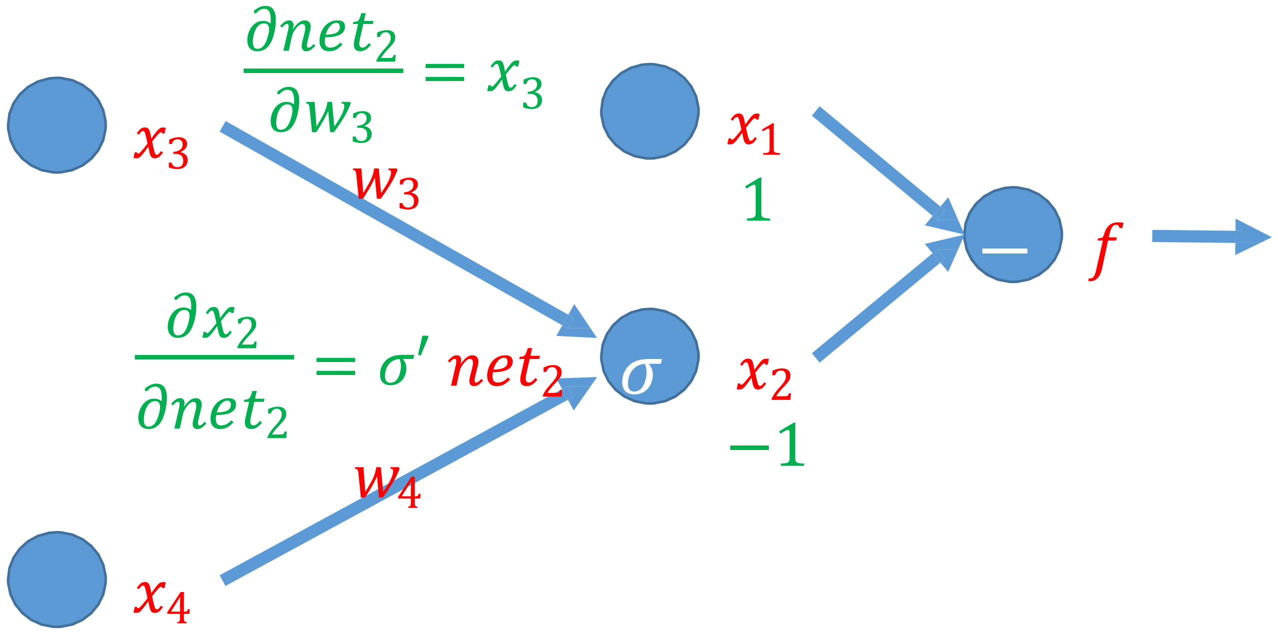
Activation



Function: $f = x_1 - x_2 = x_1 - \sigma(w_3 x_3 + w_4 x_4)$

Let $net_2 = w_3 x_3 + w_4 x_4$

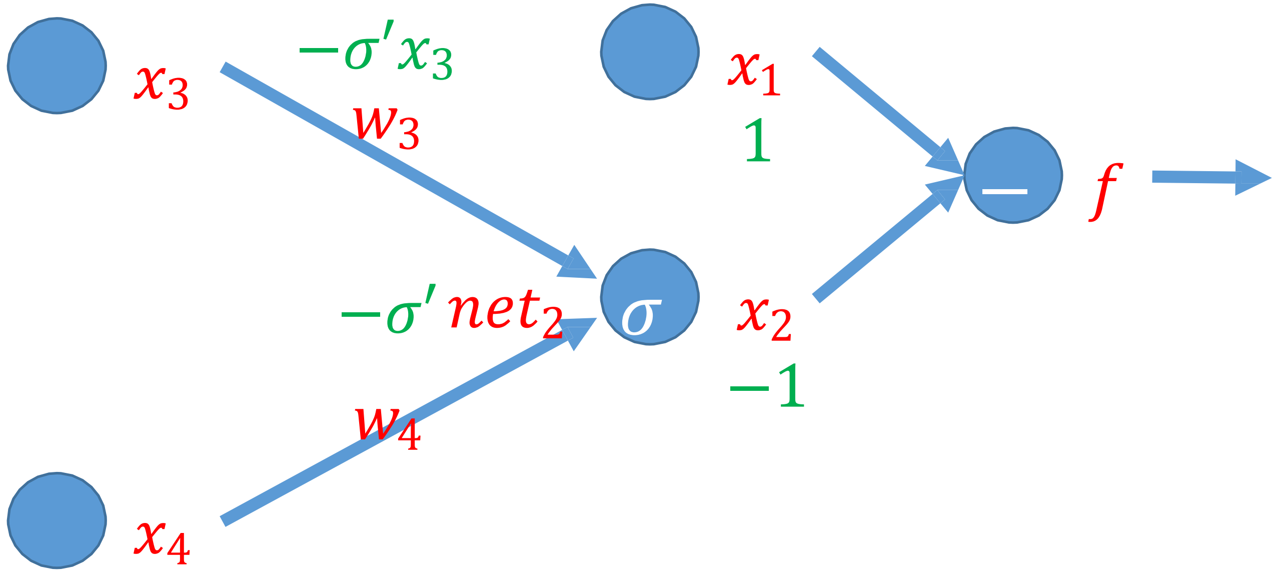
Activation



Function: $f = x_1 - x_2 = x_1 - \sigma(w_3 x_3 + w_4 x_4)$

Gradient: $\frac{\partial f}{\partial w_3} = \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial net_2} \frac{\partial net}{\partial w_3} = -1 \times \sigma' \times x_3 = -\sigma' x_3$

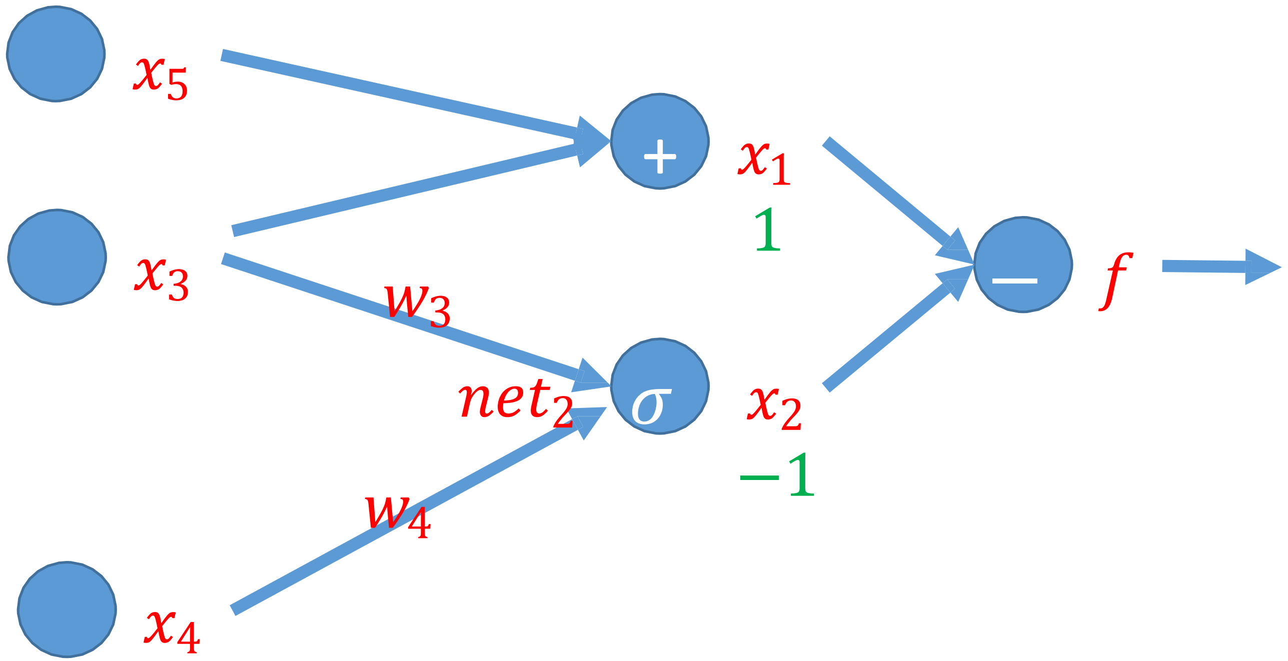
Activation



Function: $f = x_1 - x_2 = x_1 - \sigma(w_3 x_3 + w_4 x_4)$

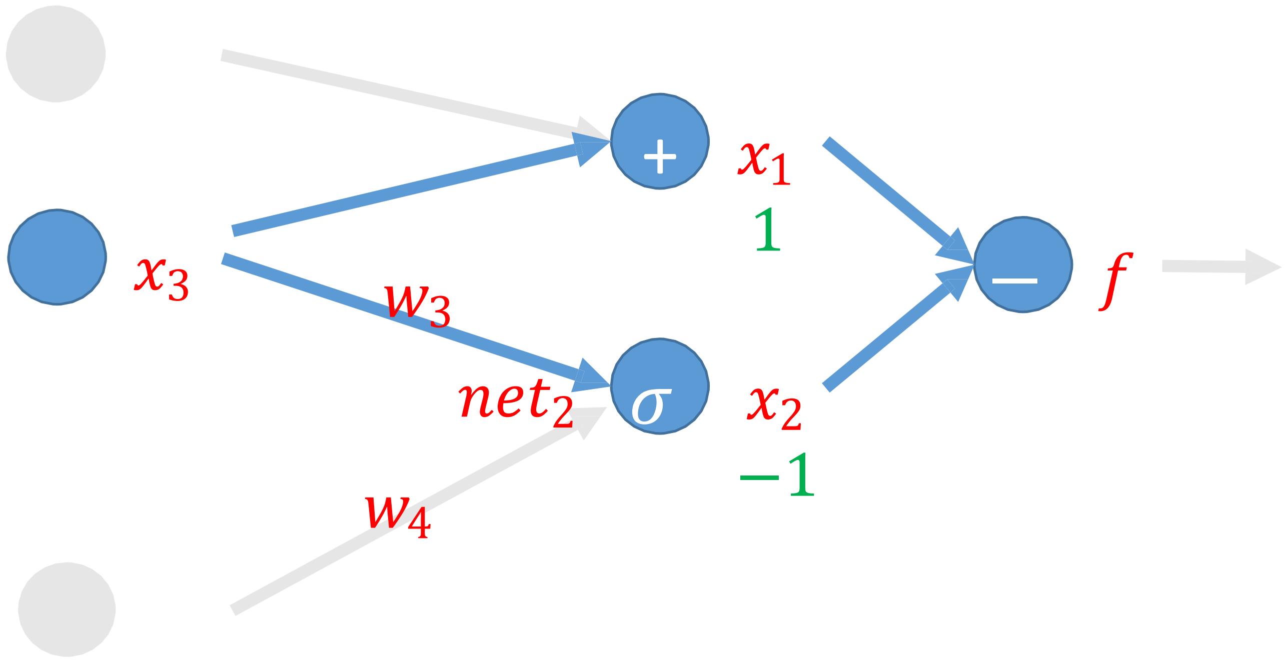
Gradient: $\frac{\partial f}{\partial w_3} = \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial net_2}\frac{\partial net}{\partial w} = -1 \times \sigma' \times x_3 = -\sigma' x_3$

Multiple paths



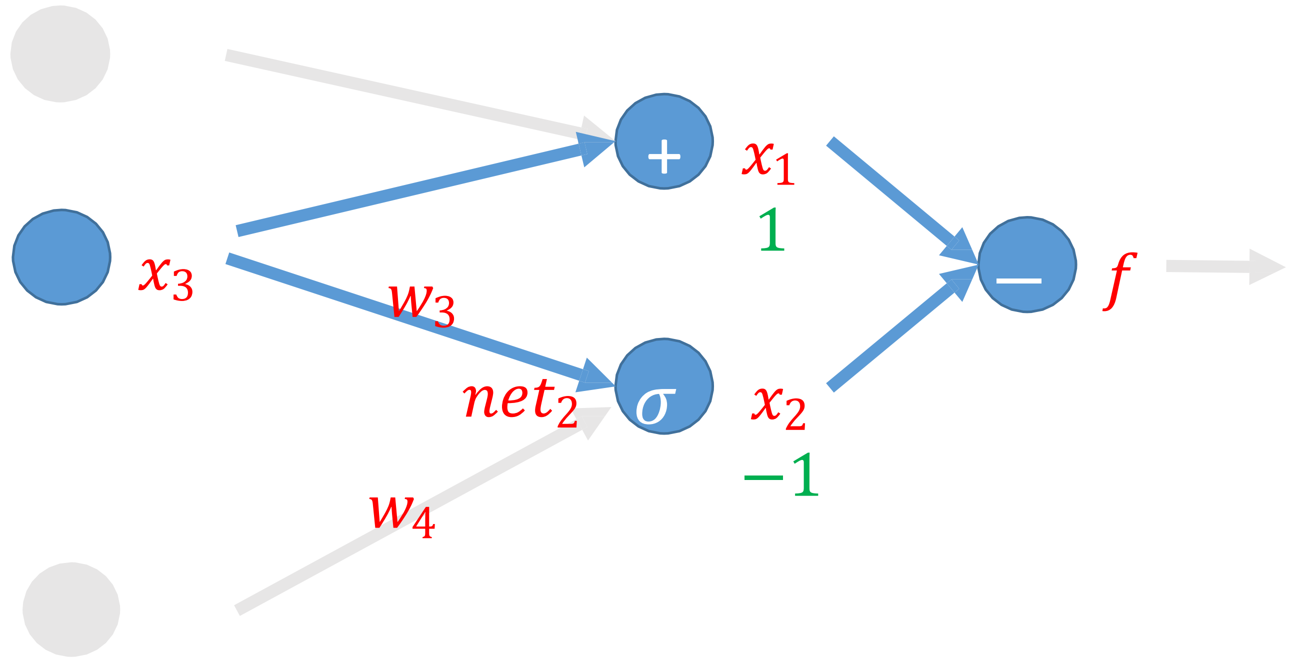
Function: $f = x_1 - x_2 = (x_1 + x_5) - \sigma\left(w_3 x_3 + w_4 x_4\right)$

Multiple paths



Function: $f = x_1 - x_2 = (x_1 + x_5) - \sigma\left(w_3 x_3 + w_4 x_4\right)$

Multiple paths

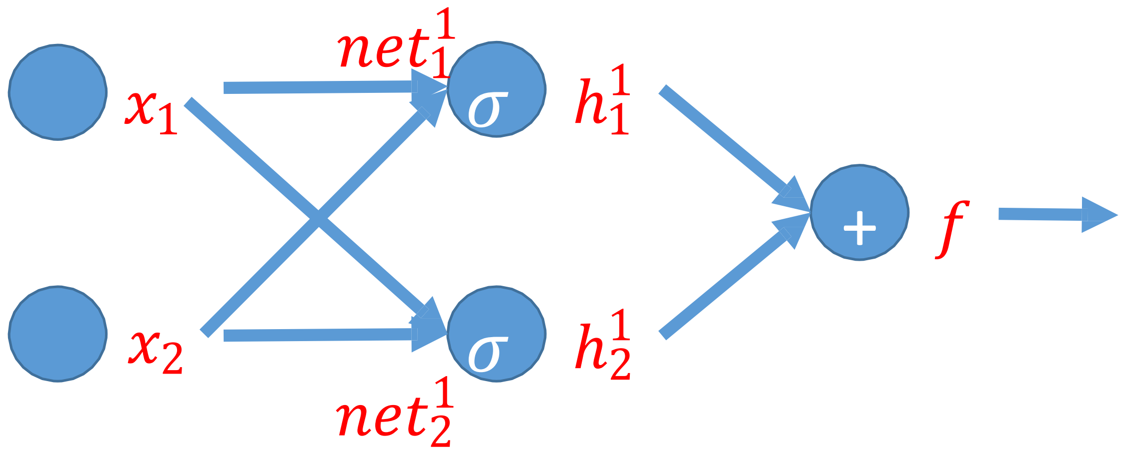


Function: $f = x_1 - x_2 = (x_3 + x_5) - \sigma\left(w_3 x_3 + w_4 x_4\right)$

Gradient: $\frac{\partial f}{\partial x_3} = \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial net_2}\frac{\partial net_2}{\partial x_3} + \frac{\partial f}{\partial x_1}\frac{\partial x}{\partial x} = -1 \times \sigma' \times w_3 + 1 \times 1 \;= -\sigma' w_3 + 1$

Summary

- Forward to compute f
- Backward to compute the gradients



Math form

Gradient descent

- Minimize loss $\mathcal{L}(\theta)$, where the hypothesis is parametrized by θ

- Gradient descent
  - Initialize θ_0
  - $\theta_{t+1} = \theta_t - \eta_t \nabla \mathcal{L}(\theta_t)$

Stochastic gradient descent (SGD)

- Suppose data points arrive one by one

- $L(\theta) = \frac{1}{n}\sum_{t=1}^{n} l(\theta, x_t, y_t)$, but we only know $l(\theta, x_t, y_t)$ at time t

- Idea: simply do what you can based on local information
  - Initialize θ_0
  - $\theta_{t+1} = \theta_t - \eta_t \nabla l(\theta_t, x_t, y_t)$

Mini-batch

- Instead of one data point, work with a small batch of b points

$$(x_{tb+1}, y_{tb+1}), \ldots, (x_{tb+b}, y_{tb+b})$$

- Update rule

$$\theta_{t+1} = \theta_t - \eta_t \nabla \left(\frac{1}{b} \sum_{1 \le i \le b} l(\theta_t, x_{tb+i}, y_{tb+}) \right)$$

- Typical batch size: $b = 128$