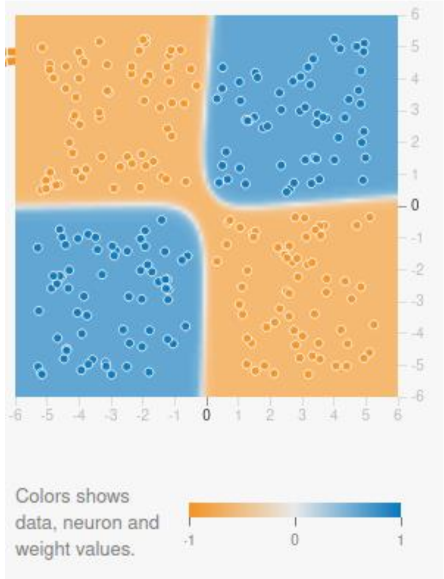
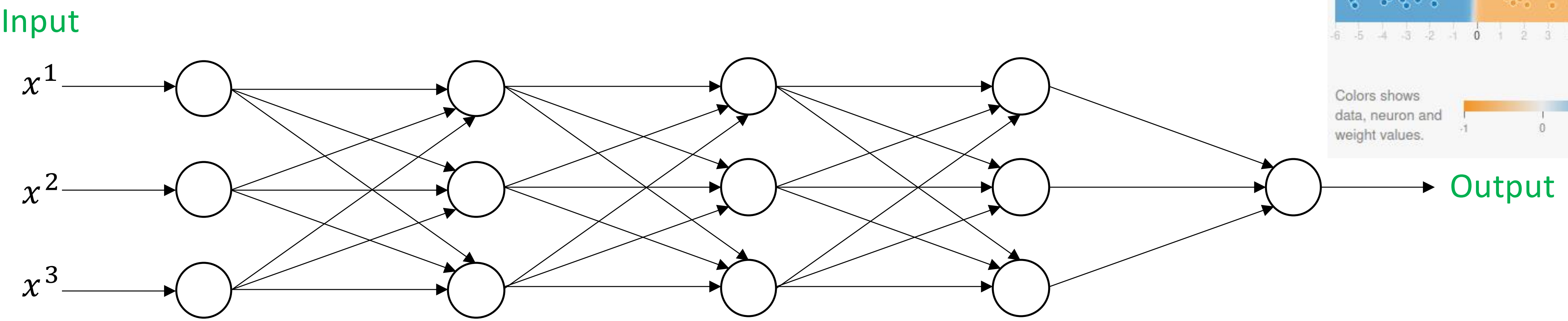
Neural Computation

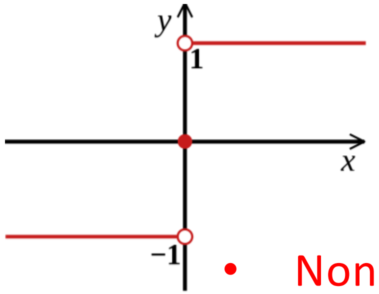
Perceptron II - the Chain Rule

Preview





This is a perceptron



output bias i'th weight

$$y = \mathrm{sgn}\left(b + \sum_i x_i w_i\right)$$

i'th input

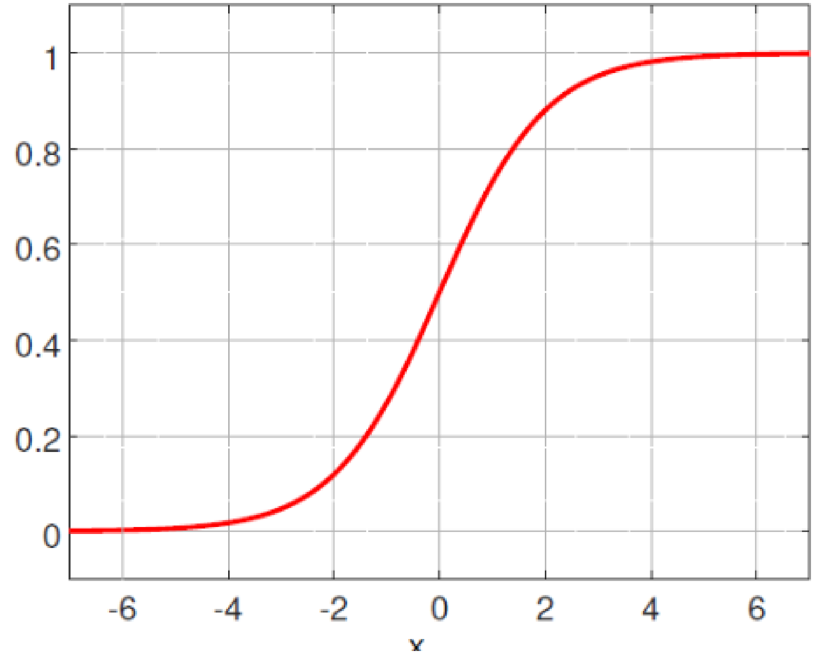
- Non-differentiable
- Problem

Build soft perceptron

How can we train this?

- Perceptron algorithm no longer works
- Use gradient descent

The idea is to replace the sgn function with a differentiable non-linear function

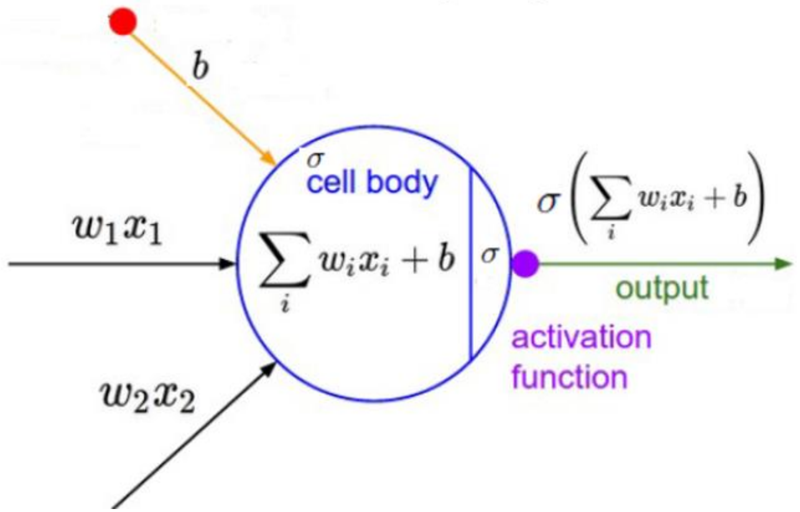


e.g., sigmoid function

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

Mapping: $(-\infty, +\infty) \mapsto (0, 1)$

$\sigma'(x) = \sigma(x)(1 - \sigma(x))$ (exercise)



$$f(\mathbf{x}) = \sigma\left(\sum_i w_i x_i + b\right)$$

Interpret as probability

Training

Dataset: n input/output pairs $S = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(n)}, y^{(n)})\}$

Mean-Square Error

$$C(\mathbf{w}) = \frac{1}{2n} \sum_{i=1}^{n} \Big(\underbrace{\sigma(\mathbf{w}^\top \mathbf{x}^{(i)} + b)}_{\text{predicted output}} - \underbrace{y^{(i)}}_{\text{output}} \Big)^2.$$

How to compute?

$$\boxed{\frac{\partial(\sigma(\mathbf{w}^\top \mathbf{x}+b)-y)^2}{\partial w_i}}$$

No closed form solution for the minimizer of $C(\mathbf{w})$

Use Gradient Descent to train a $\mathbf{w} \in \mathbb{R}^d$ with a small $C(\mathbf{w})$

$$\mathbf{w}^{\text{new}} = \mathbf{w} - \eta \nabla C(\mathbf{w}) \iff w_i^{\text{new}} = w_i - \eta \frac{\partial C(\mathbf{w})}{\partial w_i}.$$

Chain Rule

- Consider

$$f = f(g),$$

where

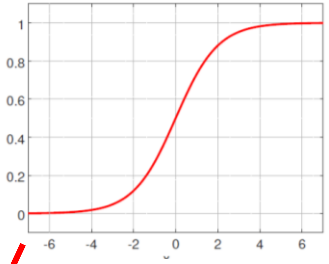
$g = g(t)$ is a function of t

- We can compute the derivative as

$$\frac{\partial}{\partial t} f = \frac{\partial f}{\partial g} \frac{\partial g}{\partial t}$$

$$t \longrightarrow g \longrightarrow f$$

Perceptron



$$\frac{\partial}{\partial t} f = \frac{\partial f}{\partial g} \frac{\partial g}{\partial t}$$

Consider $C = \frac{1}{2}\left(\sigma\left(\sum_{i}^{m} w_i x_i + b\right) - y\right)^2$

Goal compute $\frac{\partial}{\partial w_i} C$

Reformulate as

$$C(p) = \frac{1}{2}p^2 \qquad p(q) = \sigma(q) - y \qquad q(w_i) = \sum_{j}^{m} w_j x_j + b$$

$$\frac{\partial}{\partial w_i} C = \frac{\partial C}{\partial p}\frac{\partial p}{\partial w_i} = \underbrace{\frac{\partial C}{\partial p}}_{p}\underbrace{\frac{\partial p}{\partial q}}_{\sigma'(q)}\underbrace{\frac{\partial q}{\partial w_i}}_{x_i} = \left(\sigma\left(\sum_{j}^{m} w_j x_j + b\right) - y\right) \quad \sigma'\left(\sum_{j}^{m} w_j x_j + b\right) \quad x$$

Gradient Descent for One-Layer NN

Initialize $\mathbf{w}^{(1)} = 0$, $b^{(1)} = 0$

for $t = 1, 2, \ldots, T$ **do** $\quad \triangleright$ T is the number of iterations

Use (3), (2) to compute gradients

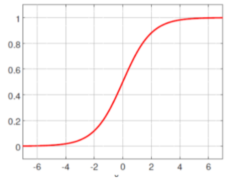
$$\nabla_{\mathbf{w}} = \frac{1}{n} \sum_{i=1}^{n} \frac{\partial C_i(\mathbf{w}^{(t)})}{\partial \mathbf{w}^{(t)}}, \qquad \nabla_b = \frac{1}{n} \sum_{i=1}^{n} \frac{\partial C_i(\mathbf{w}^{(t)})}{\partial b^{(t)}}$$

Update the model

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta_t \nabla_{\mathbf{w}}, \qquad b^{(t+1)} = b^{(t)} - \eta_t \nabla_b.$$

Summary

- Soft perceptron



- Chain rule

$$\frac{\partial}{\partial t} f = \frac{\partial f}{\partial g} \frac{\partial g}{\partial t}$$

- Gradient descent

$$\frac{\partial}{\partial w_i} C = \left(\sigma\left(\sum_j^m w_j x_j + b \right) - y \right) \sigma'\left(\sum_j^m w_j x_j + b \right) x$$

- Limitations remain
  - Need to combine to network