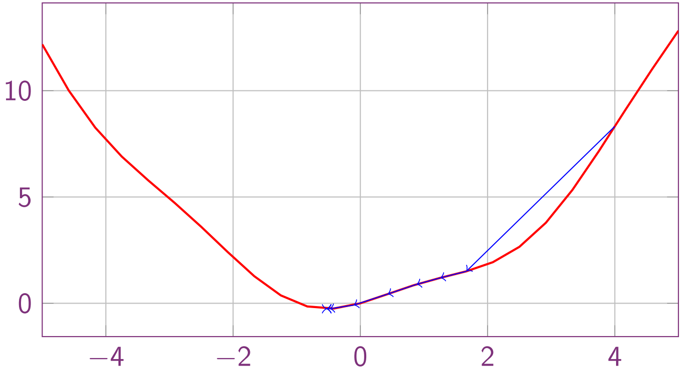
Neural Networks and Backpropagation Lecture

Grégoire Payen de La Garanderie

Durham University

February 4, 2019

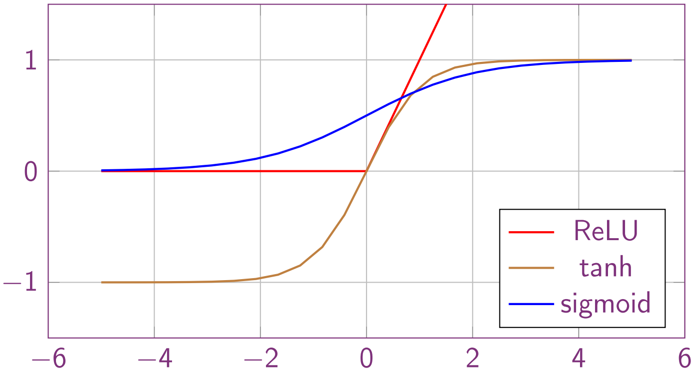
Neural Networks and Function Minimisation



Function of one real-valued variable

$$\begin{aligned} f : \mathbb{R} &\to \mathbb{R} \\ x &\mapsto f(x) \end{aligned}$$

Activation functions



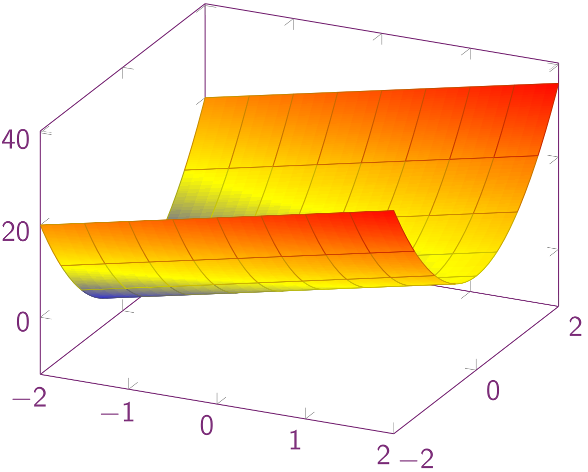
Durham University

Function of two real-valued variables

$$\begin{aligned} f : \mathbb{R}^2 &\to \mathbb{R} \\ x, y &\mapsto f(x, y) \end{aligned}$$

Durham University

Example: $f(x, y) = 4x + 7y^2$



Function of n real-valued variables

$$\begin{aligned} f : \mathbb{R}^n &\to \mathbb{R} \\ x_1, \ldots, x_n &\mapsto f(x_1, \ldots, x_n) \end{aligned}$$

Jacobian

$$
\begin{aligned}
f : \mathbb{R}^n &\to \mathbb{R}^m \\
x_1, \ldots, x_n &\mapsto (y_1, \ldots, y_m) = f(x_1, \ldots, x_n)
\end{aligned}
$$

Jacobian

$$J_f = \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Chain Rule

Given $h(x) = f(g(x))$

$$h'(x) = f'(g(x)) \times g'(x)$$

Example: A simple network

$$\boldsymbol{x}$$

$$\downarrow$$

$$h(\boldsymbol{x})$$

$$\downarrow$$

$$(g \circ h)(\boldsymbol{x})$$

$$\downarrow$$

$$(f \circ g \circ h)(\boldsymbol{x})$$

Durham University

Function of multiple vectors

$$\begin{aligned} f : \mathbb{R}^s \times \mathbb{R}^t &\to \mathbb{R}^u \\ x, y &\mapsto f(x, y) \end{aligned}$$

Example: Multilinear Map

$$
\begin{aligned}
f : \mathbb{R}^n \times \mathbb{R}^{m \times n} \times \mathbb{R}^m &\to \mathbb{R}^m \\
x, W, b &\mapsto Wx + b
\end{aligned}
$$

Durham University

Example: A weighted neural network

$$
\begin{tikzcd}
\mathbf{x} \arrow[d] & \mathbf{w_h} \arrow[dl] \\
h(\mathbf{x}, \mathbf{w_h}) \arrow[d] & \mathbf{w_g} \arrow[dl] \\
g(h(\mathbf{x}, \mathbf{w_f}), \mathbf{w_g}) \arrow[d] & \mathbf{w_f} \arrow[dl] \\
f(g(h(\mathbf{x}, \mathbf{w_h}), \mathbf{w_g}), \mathbf{w_f}) &
\end{tikzcd}
$$

Example: A simple network

$$\boldsymbol{x}$$

$$\downarrow$$

$$h(\boldsymbol{x})$$

$$\downarrow$$

$$(g \circ h)(\boldsymbol{x})$$

$$\downarrow$$

$$(f \circ g \circ h)(\boldsymbol{x})$$

```
function Forward(input)
    return f(input)
end function
```

function Backward(input,grad_output)
    return grad_output $\cdot f'$(input)
end function

```
function ConvBackward(input,grad_output)
    #Compute the Jacobian matrix
    J ← zero matrix of size n × n
    for i ∈ [1, n] do
        for j ∈ [1, n] do
            k ← j − i
            if k ∈ [−1, 1] then
                J_{i,j} ← h_{k+2}
            end if
        end for
    end for

    #Multiply the Jacobian with grad_output
    v ← zero vector of size n
    for i ∈ [1, n] do
        for j ∈ [1, n] do
            v_j ← v_j + grad_output_i · J_{i,j}
        end for
    end for
    return v
end function
```

```
function ConvBackward(input,grad_output)
    v ← zero vector of size n
    for k ∈ [−1, −1] do
        for j ∈ [1, n] do
            i ← j − k
            if i ∈ [1, n] then
                v_j ← v_j + grad_output_i · h_{k+2}
            end if
        end for
    end for
    return v
end function
```

Durham University