UFV
Universidade Federal de Viçosa

Lecture 05 – Artificial neural networks deep learning

Prof. João Fernando Mari

joaofmari.github.io

joaof.mari@ufv.br

Agenda

- Multilayer perceptron
- Network architecture
- Backpropagation – Algorithm and equations
- Backpropagation – Functions and derivatives
- Backpropagation – Examples

MULTILAYER PERCEPTRON

Multilayer perceptron

NATURE VOL. 323 9 OCTOBER 1986 — LETTERS TO NATURE — 533

Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].

There have been many attempts to design self-organizing neural networks. The aim is to find a powerful synaptic modification rule that will allow an arbitrarily connected neural network to develop an internal structure that is appropriate for a particular task domain. The task is specified by giving the desired state vector of the output units for each state vector of the input units. If the input units are directly connected to the output units it is relatively easy to find learning rules that iteratively adjust the relative strengths of the connections so as to progressively reduce the difference between the actual and desired output vectors[2]. Learning becomes more interesting but more difficult when we introduce hidden units whose actual or desired states are not specified by the task. (In perceptrons, there are 'feature analysers' between the input and output that are not true hidden units because their input connections are fixed by hand, so their states are completely determined by the input vector: they do not learn representations.) The learning procedure must decide under what circumstances the hidden units should be active in order to help achieve the desired input–output behaviour. This amounts to deciding what these units should represent. We demonstrate that a general purpose and relatively simple procedure is powerful enough to construct appropriate internal representations.

The simplest form of the learning procedure is for layered networks which have a layer of input units at the bottom; any number of intermediate layers; and a layer of output units at the top. Connections within a layer or from higher to lower layers are forbidden, but connections can skip intermediate layers. An input vector is presented to the network by setting the states of the input units. Then the states of the units in each layer are determined by applying equations (1) and (2) to the connections coming from lower layers. All units within a layer have their states set in parallel, but different layers have their states set sequentially, starting at the bottom and working upwards until the states of the output units are determined.

The total input, x_j, to unit j is a linear function of the outputs, y_i, of the units that are connected to j and of the weights, w_{ji}, on these connections

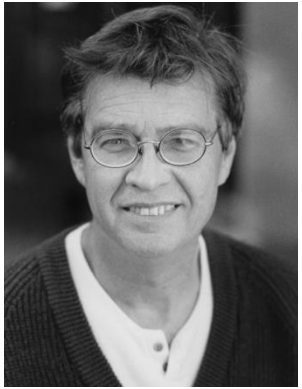
$$x_j = \sum_i y_i w_{ji} \qquad (1)$$

Units can be given biases by introducing an extra input to each unit which always has a value of 1. The weight on this extra input is called the bias and is equivalent to a threshold of the opposite sign. It can be treated just like the other weights.

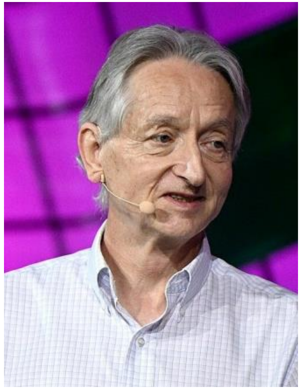
A unit has a real-valued output, y_j, which is a non-linear function of its total input

$$y_j = \frac{1}{1+e^{-x_j}} \qquad (2)$$

† To whom correspondence should be addressed

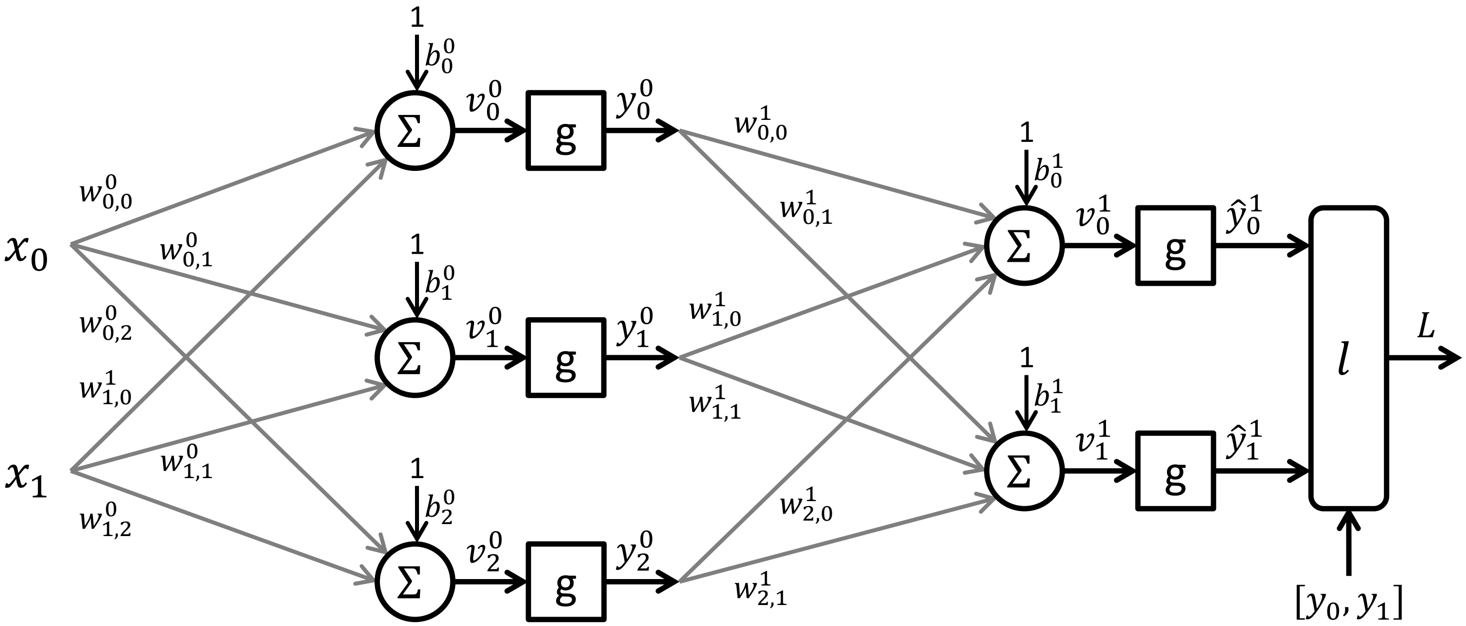


David E. Rumelhart
(1942 – 2011)
Cognitive psychology

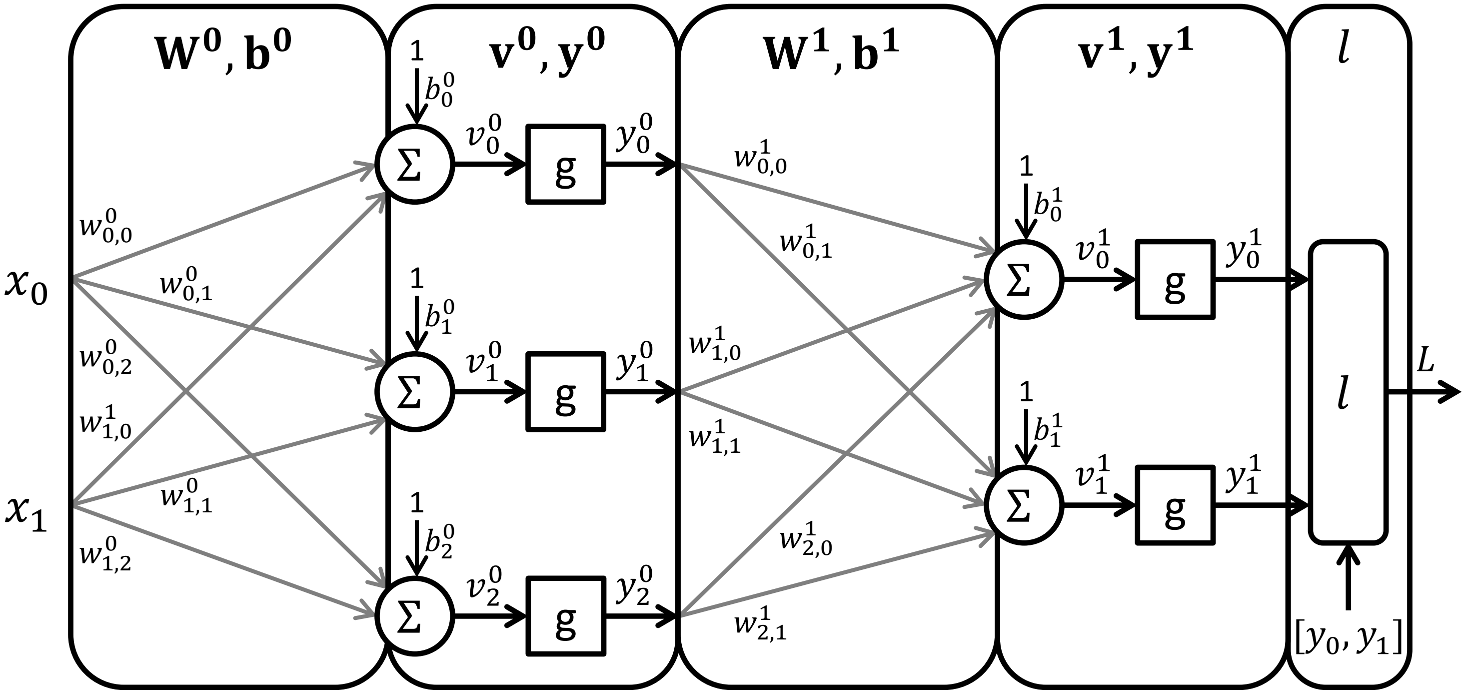


Geoffrey E. Hinton
(1947 -)
Cognitive psychology
Computer science

Multilayer perceptron

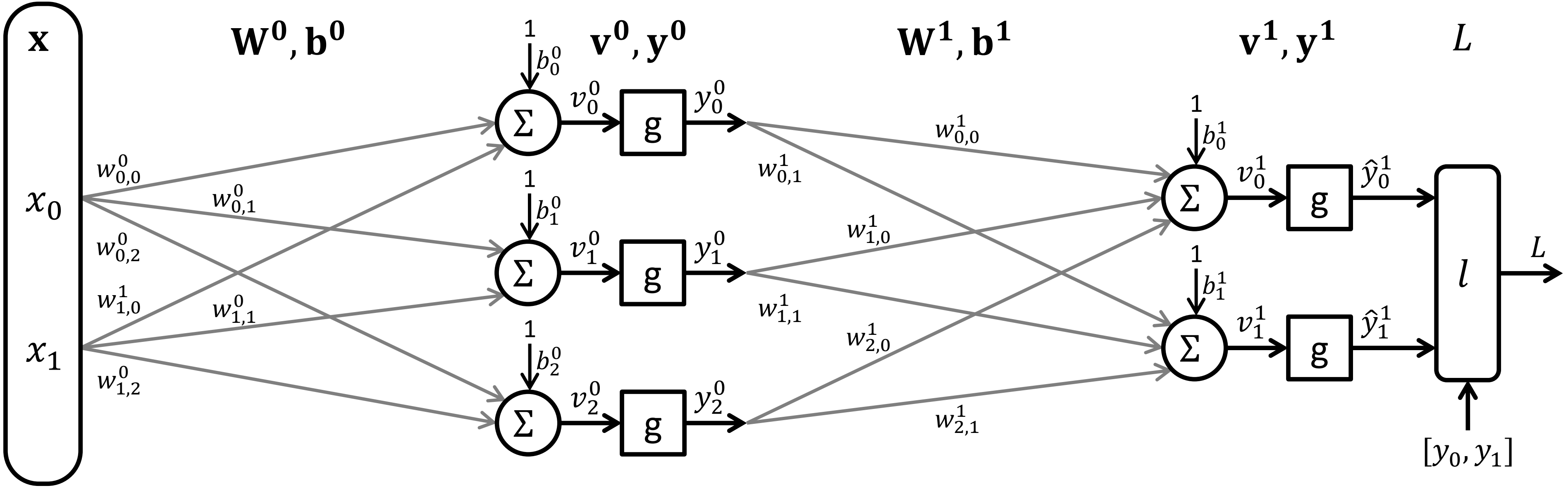


Multilayer perceptron



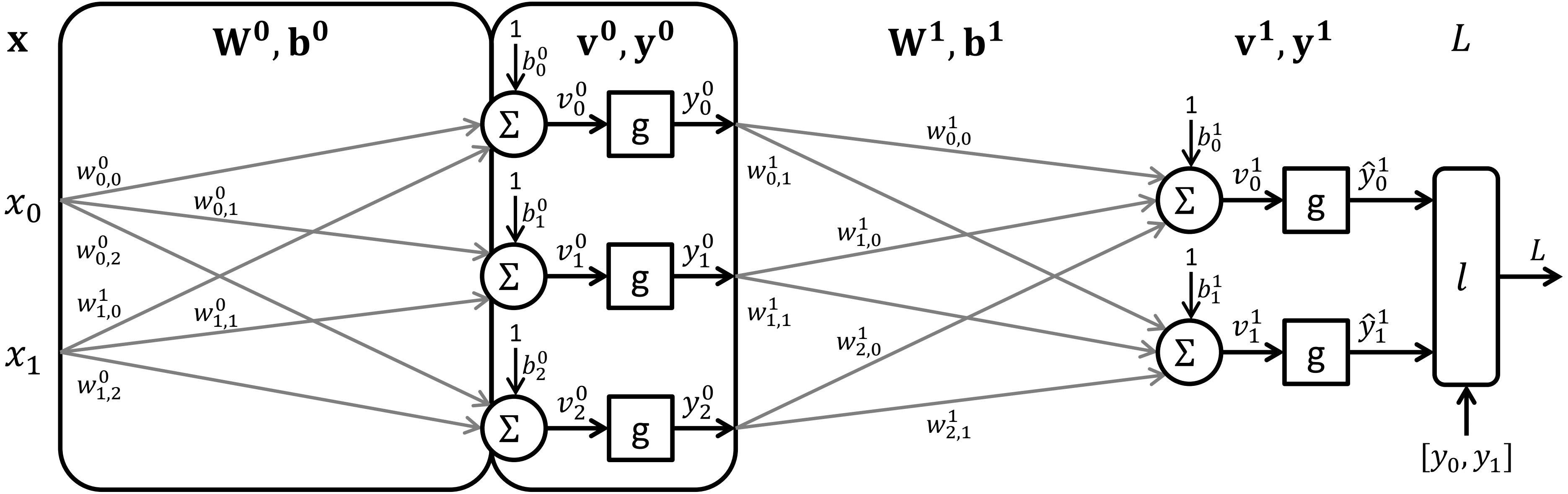
NETWORK ARCHITECTURE

Network architecture



$\mathbf{x} = [x_0 \quad x_1]$

Network architecture



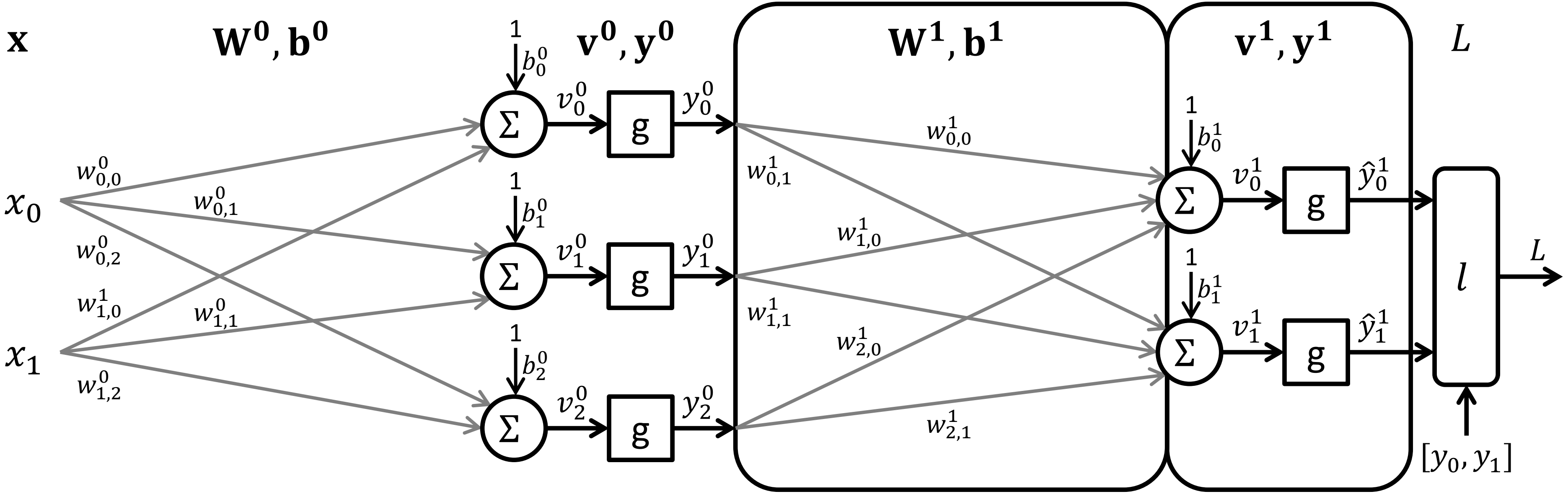
$$\mathbf{x} = [x_0 \quad x_1] \qquad \mathbf{W^0} = \begin{bmatrix} w^0_{0,0} & w^0_{0,1} & w^0_{0,2} \\ w^0_{1,0} & w^0_{1,1} & w^0_{1,2} \end{bmatrix}$$

$$\mathbf{v^0} = [v^0_0 \quad v^0_1 \quad v^0_2]$$

$$\mathbf{y^0} = [y^0_0 \quad y^0_1 \quad y^0_2]$$

$$\mathbf{b^0} = [b^0_0 \quad b^0_1 \quad b^0_2]$$

UFV
Universidade Federal de Viçosa



$\mathbf{x} = [x_0 \quad x_1]$ $\quad \mathbf{W^0} = \begin{bmatrix} w_{0,0}^0 & w_{0,1}^0 & w_{0,2}^0 \\ w_{1,0}^0 & w_{1,1}^0 & w_{1,2}^0 \end{bmatrix}$

$\mathbf{v^0} = [v_0^0 \quad v_1^0 \quad v_2^0]$

$\mathbf{y^0} = [y_0^0 \quad y_1^0 \quad y_2^0]$

$\mathbf{W^1} = \begin{bmatrix} w_{0,0}^1 & w_{0,1}^1 \\ w_{1,0}^1 & w_{1,1}^1 \\ w_{2,0}^1 & w_{2,1}^1 \end{bmatrix}$

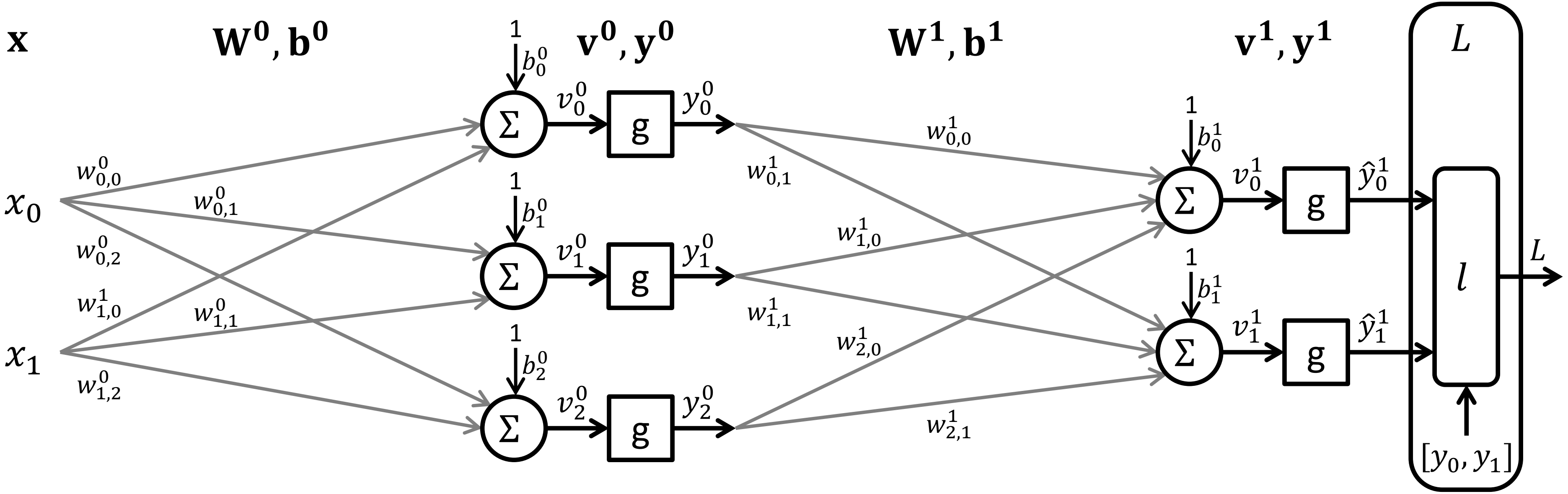
$\mathbf{v^1} = [v_0^1 \quad v_1^1]$

$\hat{\mathbf{y}}^1 = [\hat{y}_0^1 \quad \hat{y}_1^1]$

$\mathbf{b^0} = [b_0^0 \quad b_1^0 \quad b_2^0]$

$\mathbf{b^1} = [b_0^1 \quad b_1^1]$

Network architecture



$\mathbf{x} = [x_0 \quad x_1]$

$\mathbf{W^0} = \begin{bmatrix} w^0_{0,0} & w^0_{0,1} & w^0_{0,2} \\ w^0_{1,0} & w^0_{1,1} & w^0_{1,2} \end{bmatrix}$

$\mathbf{b^0} = [b^0_0 \quad b^0_1 \quad b^0_2]$

$\mathbf{v^0} = [v^0_0 \quad v^0_1 \quad v^0_2]$

$\mathbf{y^0} = [y^0_0 \quad y^0_1 \quad y^0_2]$

$\mathbf{W^1} = \begin{bmatrix} w^1_{0,0} & w^1_{0,1} \\ w^1_{1,0} & w^1_{1,1} \\ w^1_{2,0} & w^1_{2,1} \end{bmatrix}$

$\mathbf{b^1} = [b^1_0 \quad b^1_1]$

$\mathbf{v^1} = [v^1_0 \quad v^1_1]$

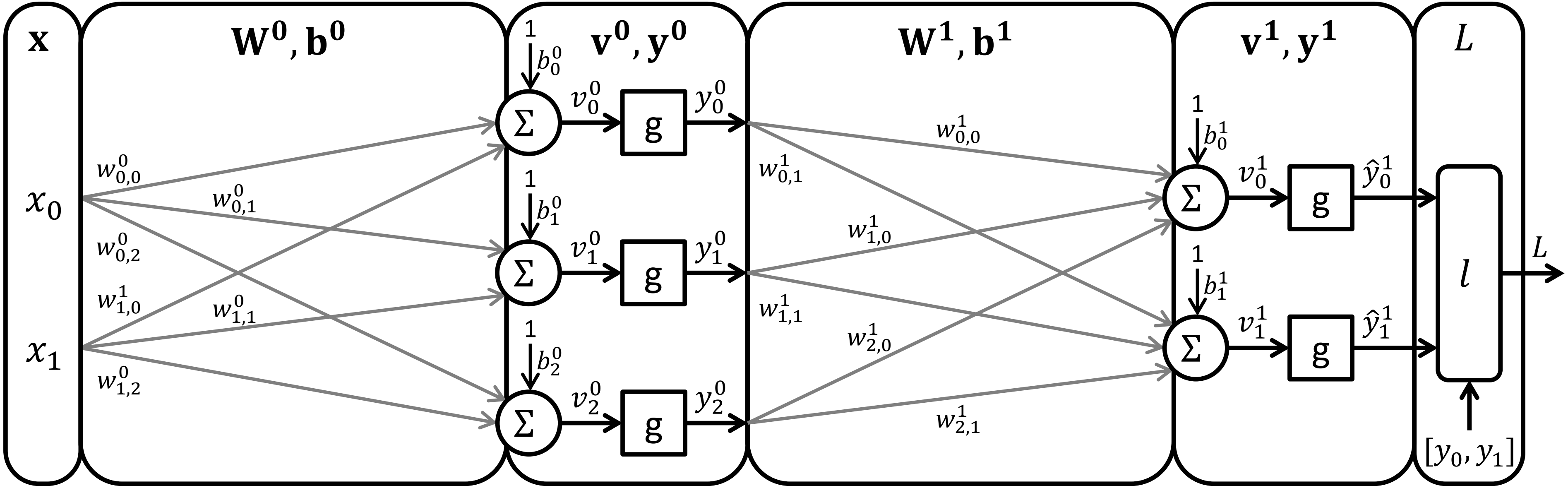
$\mathbf{\hat{y}^1} = [\hat{y}^1_0 \quad \hat{y}^1_1]$

$\mathbf{y} = [y_0 \quad y_1]$

$\mathbf{l} = [l_0 \quad l_1]$

$L = l_0 + l_1$

Network architecture



$\mathbf{x} = [x_0 \quad x_1]$ $\quad \mathbf{W^0} = \begin{bmatrix} w^0_{0,0} & w^0_{0,1} & w^0_{0,2} \\ w^0_{1,0} & w^0_{1,1} & w^0_{1,2} \end{bmatrix}$

$\mathbf{b^0} = [b^0_0 \quad b^0_1 \quad b^0_2]$

$\mathbf{v^0} = [v^0_0 \quad v^0_1 \quad v^0_2]$

$\mathbf{y^0} = [y^0_0 \quad y^0_1 \quad y^0_2]$

$\mathbf{W^1} = \begin{bmatrix} w^1_{0,0} & w^1_{0,1} \\ w^1_{1,0} & w^1_{1,1} \\ w^1_{2,0} & w^1_{2,1} \end{bmatrix}$

$\mathbf{b^1} = [b^1_0 \quad b^1_1]$

$\mathbf{v^1} = [v^1_0 \quad v^1_1]$

$\hat{\mathbf{y}}^1 = [\hat{y}^1_0 \quad \hat{y}^1_1]$

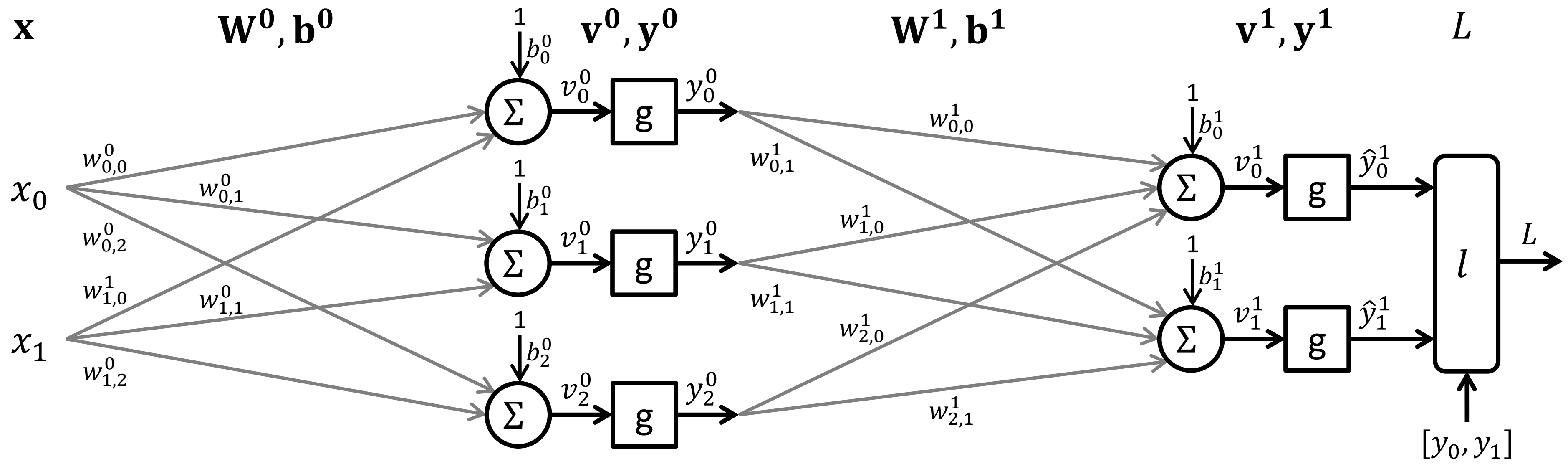
$\mathbf{y} = [y_0 \quad y_1]$

$\mathbf{l} = [l_0 \quad l_1]$

$L = l_0 + l_1$

BACKPROPAGATION – ALGORITHM AND EQUATIONS

Backpropagation – Algorithm and equations



$\mathbf{v}^0 = \mathbf{x}\mathbf{W}^0 + \mathbf{b}^0$ $\quad$ $\mathbf{y}^0 = \mathbf{g}^0(\mathbf{v}^0)$ $\quad$ $\mathbf{v}^1 = \mathbf{y}^0\mathbf{W}^1 + \mathbf{b}^1$ $\quad$ $\mathbf{y}^1 = \mathbf{g}^1(\mathbf{v}^1)$ $\quad$ $l = l(\mathbf{y}, \hat{\mathbf{y}}^1)$ $\quad$ $L = l_0 + l_1$

$$\mathbf{v^0} = \begin{bmatrix} x_0 & x_1 \end{bmatrix} \begin{bmatrix} w_{0,0}^0 & w_{0,1}^0 & w_{0,2}^0 \\ w_{1,0}^0 & w_{1,1}^0 & w_{1,2}^0 \end{bmatrix} + \begin{bmatrix} b_0^0 & b_1^0 & b_2^0 \end{bmatrix}$$

$$\mathbf{v^1} = \begin{bmatrix} y_0^0 & y_1^0 & y_2^0 \end{bmatrix} \begin{bmatrix} w_{0,0}^1 & w_{0,1}^1 \\ w_{1,0}^1 & w_{1,1}^1 \\ w_{2,0}^1 & w_{2,1}^1 \end{bmatrix} + \begin{bmatrix} b_0^1 & b_1^1 \end{bmatrix}$$

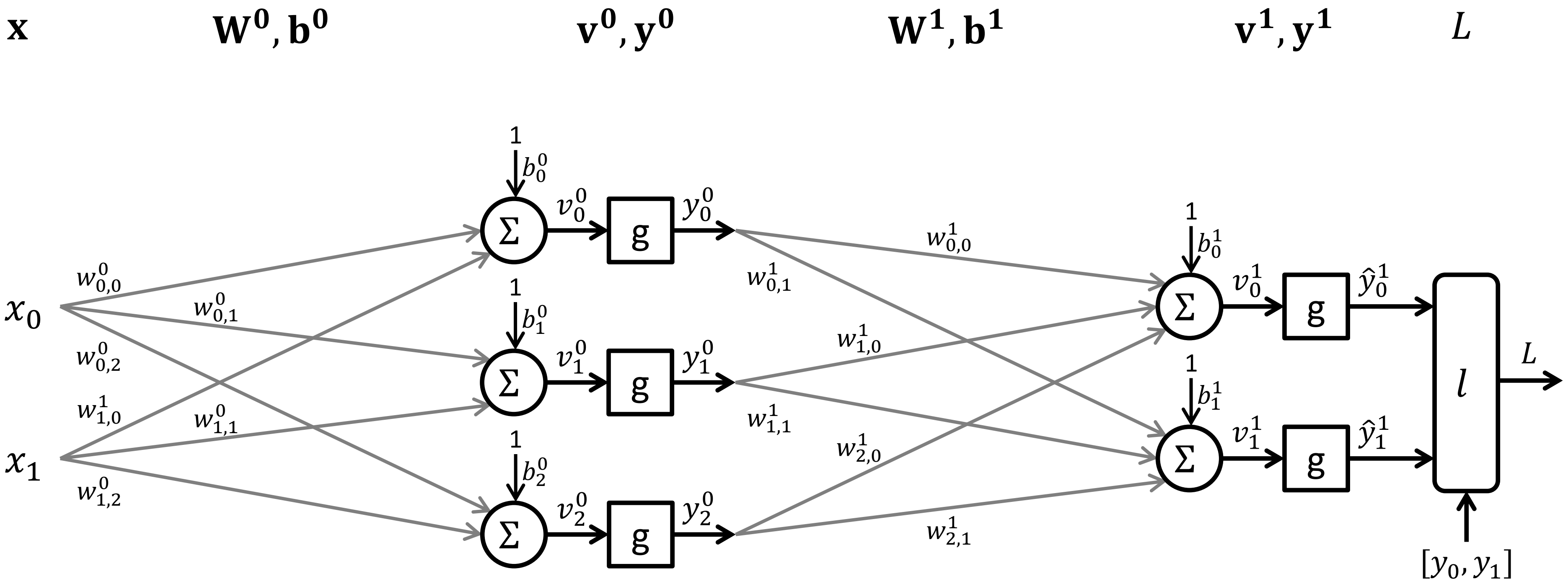
$$\mathbf{v^0} = \begin{bmatrix} x_0 w_{0,0}^0 + x_1 w_{1,0}^0 + b_0^0 & x_0 w_{0,1}^0 + x_1 w_{1,1}^0 + b_1^0 & x_0 w_{0,2}^0 + x_1 w_{1,2}^0 + b_2^0 \end{bmatrix}$$

$$\mathbf{y^0} = \begin{bmatrix} g(v_0^0) & g(v_1^0) & g(v_2^0) \end{bmatrix}$$

$$\mathbf{v^1} = \begin{bmatrix} y_0^0 w_{0,0}^1 + y_1^0 w_{1,0}^1 + y_2^0 w_{2,0}^1 + b_0^1 & y_0^0 w_{0,1}^1 + y_1^0 w_{1,1}^1 + y_2^0 w_{2,1}^1 + b_1^1 \end{bmatrix}$$

$$\hat{\mathbf{y}}^{\mathbf{1}} = \begin{bmatrix} g(v_0^1) & g(v_1^1) \end{bmatrix}$$

Backpropagation – Algorithm and equations



$$\frac{\partial L}{\partial \mathbf{W}^0} = \frac{\partial L}{\partial \mathbf{y}^0} \times \frac{\partial \mathbf{y}^0}{\partial \mathbf{v}^0} \times \frac{\partial \mathbf{v}^0}{\partial \mathbf{W}^0}$$

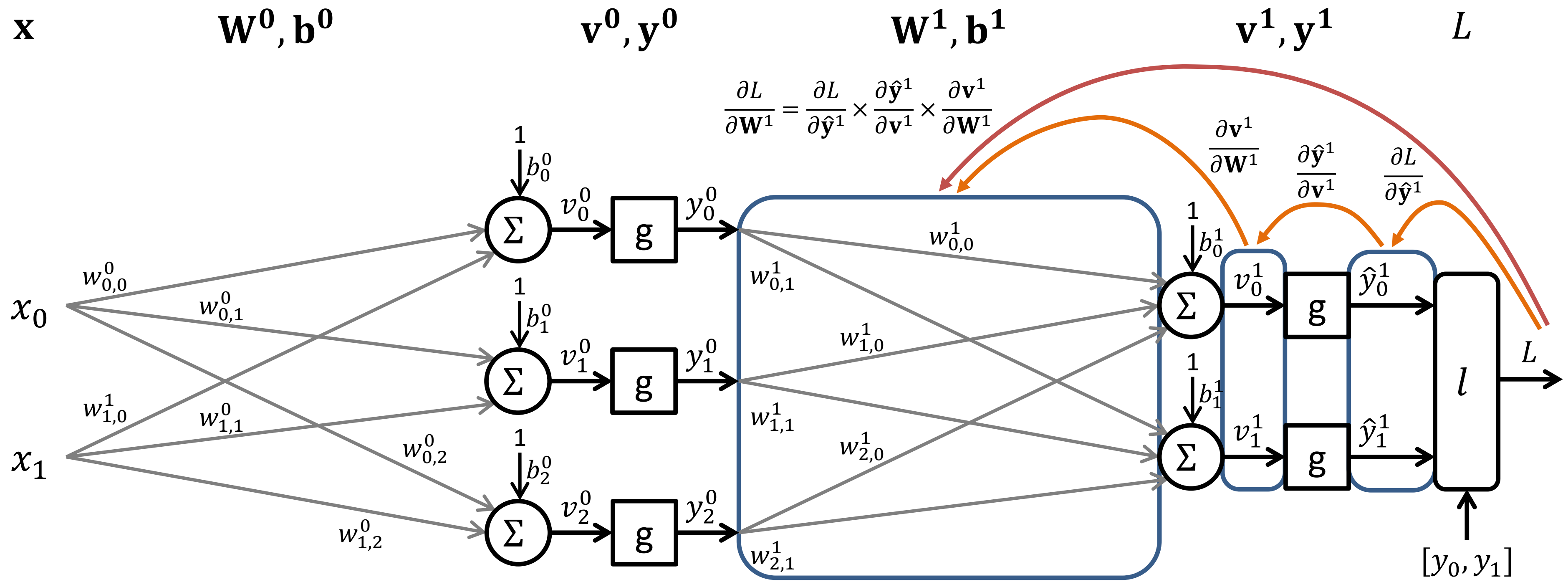
$$\frac{\partial L}{\partial \mathbf{y}^0} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^0}{\partial \mathbf{y}^0}$$

$$\frac{\partial L}{\partial \mathbf{b}^0} = \frac{\partial L}{\partial \mathbf{y}^0} \times \frac{\partial \mathbf{y}^0}{\partial \mathbf{v}^0} \times \frac{\partial \mathbf{v}^0}{\partial \mathbf{b}^0}$$

$$\frac{\partial L}{\partial \mathbf{W}^1} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^1}{\partial \mathbf{W}^1}$$

$$\frac{\partial L}{\partial \mathbf{b}^1} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^1}{\partial \mathbf{b}^1}$$

Backpropagation – Algorithm and equations



$$\frac{\partial L}{\partial \mathbf{W}^0} = \frac{\partial L}{\partial \mathbf{y}^0} \times \frac{\partial \mathbf{y}^0}{\partial \mathbf{v}^0} \times \frac{\partial \mathbf{v}^0}{\partial \mathbf{W}^0}$$

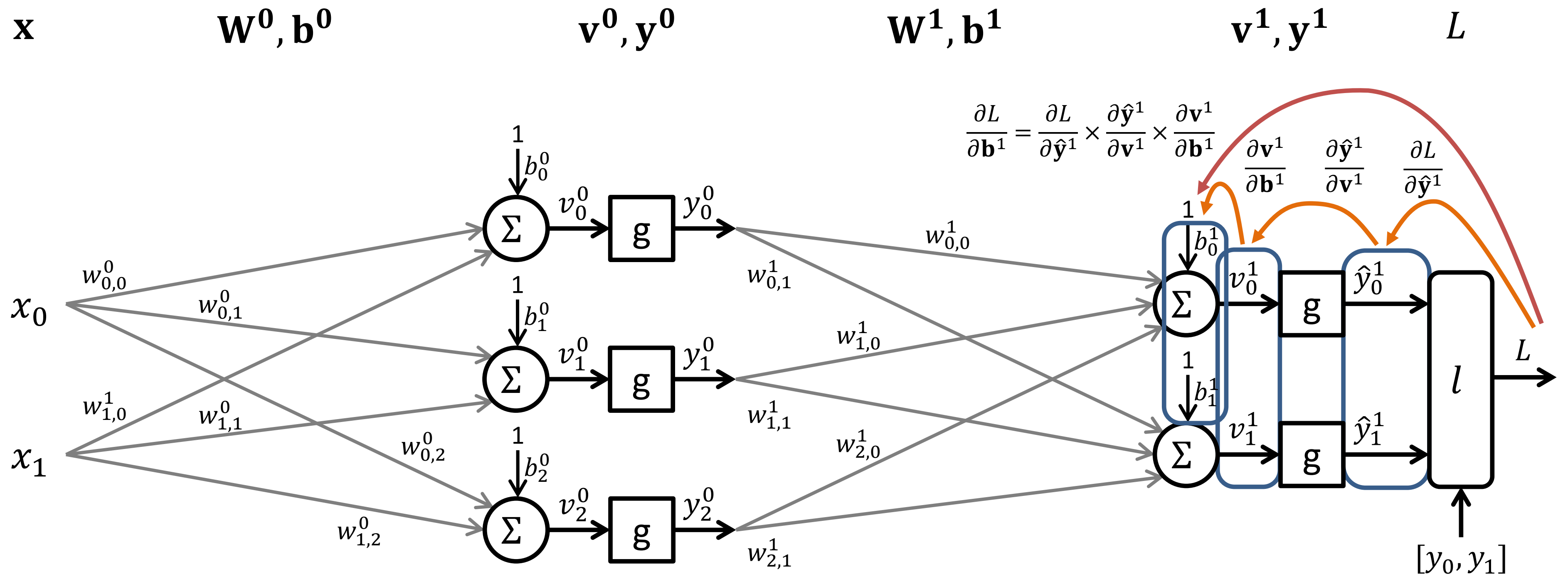
$$\frac{\partial L}{\partial \mathbf{y}^0} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^0}{\partial \mathbf{y}^0}$$

$$\frac{\partial L}{\partial \mathbf{b}^0} = \frac{\partial L}{\partial \mathbf{y}^0} \times \frac{\partial \mathbf{y}^0}{\partial \mathbf{v}^0} \times \frac{\partial \mathbf{v}^0}{\partial \mathbf{b}^0}$$

$$\boxed{\frac{\partial L}{\partial \mathbf{W}^1} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^1}{\partial \mathbf{W}^1}}$$

$$\frac{\partial L}{\partial \mathbf{b}^1} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^1}{\partial \mathbf{b}^1}$$

Backpropagation – Algorithm and equations



$$\frac{\partial L}{\partial \mathbf{W}^0} = \frac{\partial L}{\partial \mathbf{y}^0} \times \frac{\partial \mathbf{y}^0}{\partial \mathbf{v}^0} \times \frac{\partial \mathbf{v}^0}{\partial \mathbf{W}^0}$$

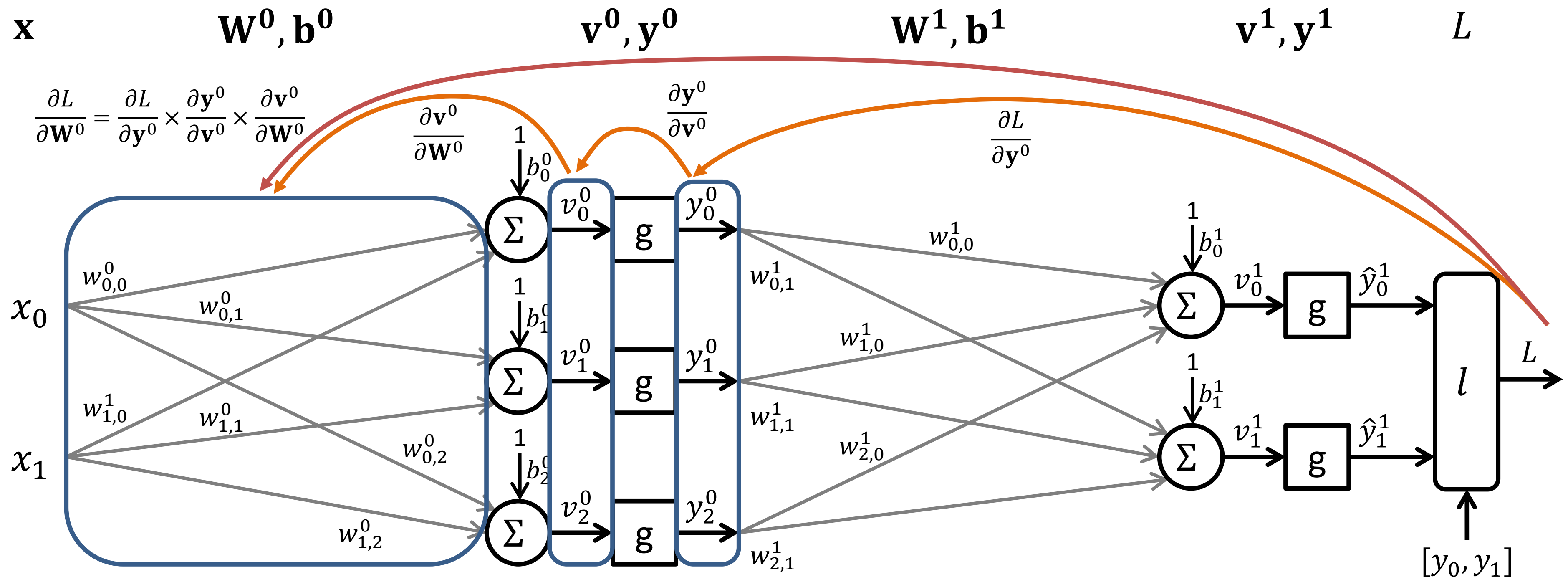
$$\frac{\partial L}{\partial \mathbf{y}^0} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^0}{\partial \mathbf{y}^0}$$

$$\frac{\partial L}{\partial \mathbf{b}^0} = \frac{\partial L}{\partial \mathbf{y}^0} \times \frac{\partial \mathbf{y}^0}{\partial \mathbf{v}^0} \times \frac{\partial \mathbf{v}^0}{\partial \mathbf{b}^0}$$

$$\frac{\partial L}{\partial \mathbf{W}^1} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^1}{\partial \mathbf{W}^1}$$

$$\boxed{\frac{\partial L}{\partial \mathbf{b}^1} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^1}{\partial \mathbf{b}^1}}$$

Backpropagation – Algorithm and equations



$$\boxed{\frac{\partial L}{\partial \mathbf{W}^0} = \frac{\partial L}{\partial \mathbf{y}^0} \times \frac{\partial \mathbf{y}^0}{\partial \mathbf{v}^0} \times \frac{\partial \mathbf{v}^0}{\partial \mathbf{W}^0}}$$

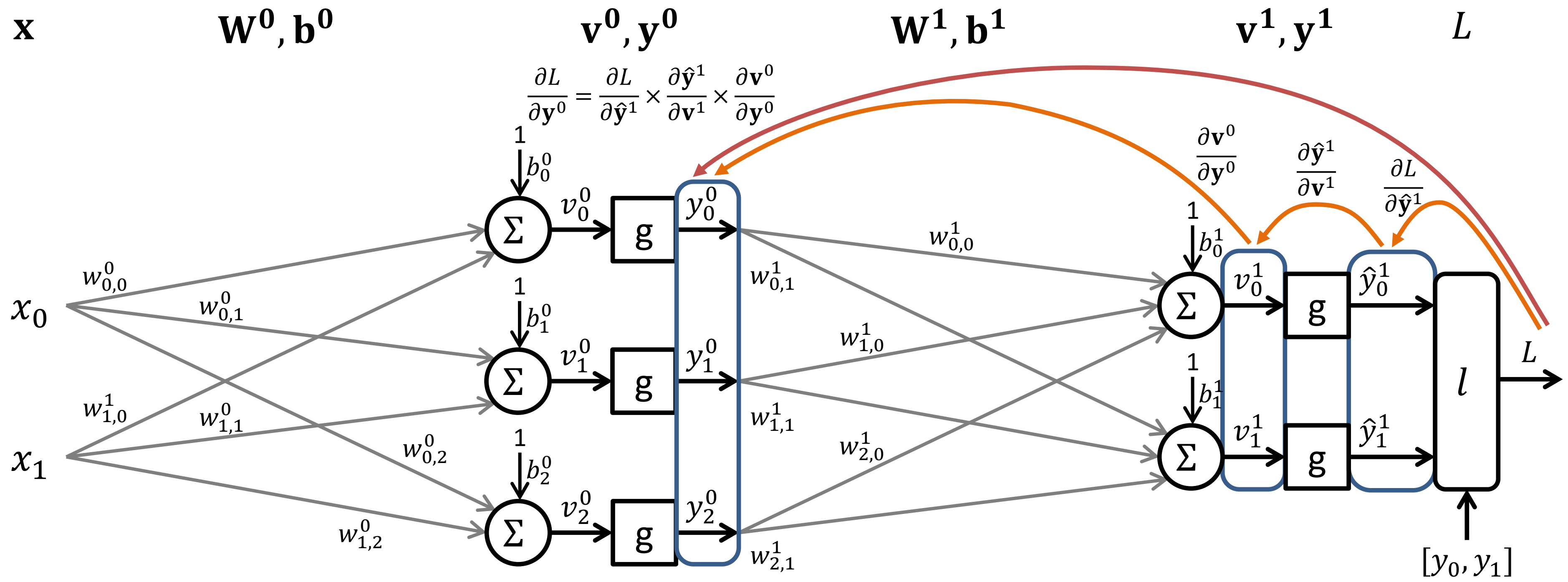
$$\frac{\partial L}{\partial \mathbf{y}^0} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^1}{\partial \mathbf{y}^0}$$

$$\frac{\partial L}{\partial \mathbf{b}^0} = \frac{\partial L}{\partial \mathbf{y}^0} \times \frac{\partial \mathbf{y}^0}{\partial \mathbf{v}^0} \times \frac{\partial \mathbf{v}^0}{\partial \mathbf{b}^0}$$

$$\frac{\partial L}{\partial \mathbf{W}^1} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^1}{\partial \mathbf{W}^1}$$

$$\frac{\partial L}{\partial \mathbf{b}^1} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^1}{\partial \mathbf{b}^1}$$

UFV
Universidade Federal de Viçosa



$$\frac{\partial L}{\partial \mathbf{W}^0} = \frac{\partial L}{\partial \mathbf{y}^0} \times \frac{\partial \mathbf{y}^0}{\partial \mathbf{v}^0} \times \frac{\partial \mathbf{v}^0}{\partial \mathbf{W}^0}$$

$$\boxed{\frac{\partial L}{\partial \mathbf{y}^0} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^0}{\partial \mathbf{y}^0}}$$

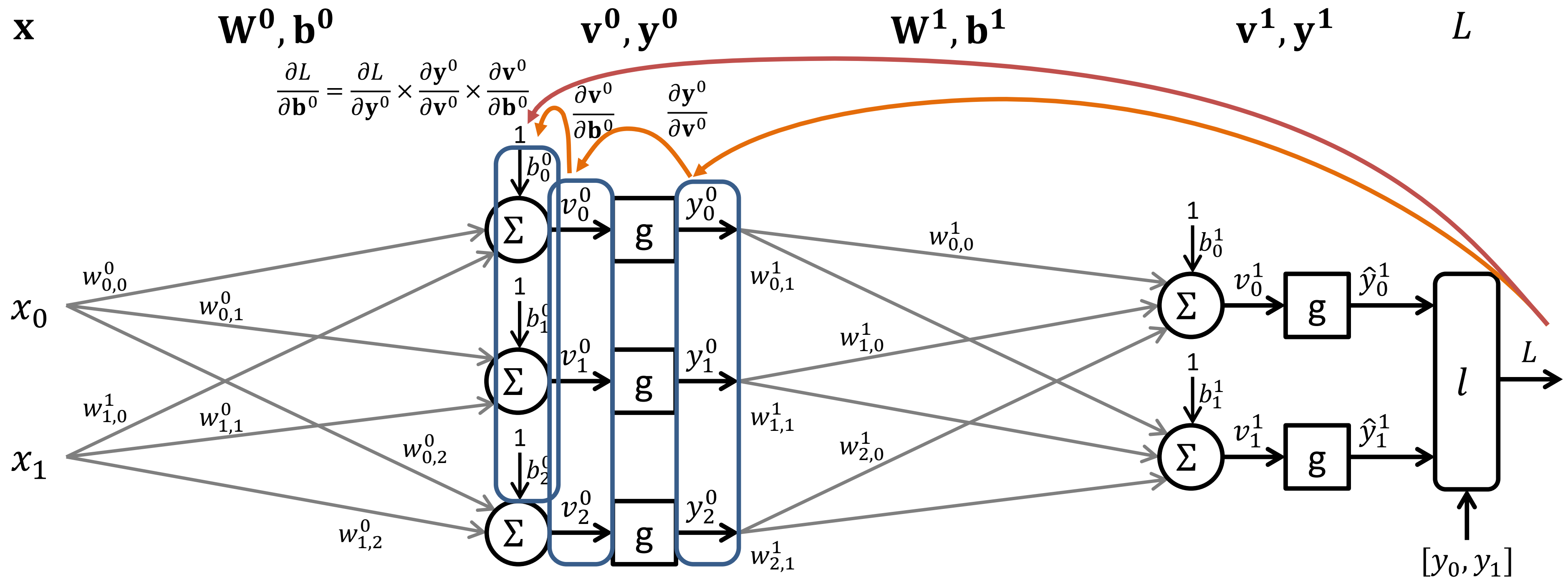
$$\frac{\partial L}{\partial \mathbf{b}^0} = \frac{\partial L}{\partial \mathbf{y}^0} \times \frac{\partial \mathbf{y}^0}{\partial \mathbf{v}^0} \times \frac{\partial \mathbf{v}^0}{\partial \mathbf{b}^0}$$

$$\frac{\partial L}{\partial \mathbf{W}^1} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^1}{\partial \mathbf{W}^1}$$

$$\frac{\partial L}{\partial \mathbf{b}^1} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^1}{\partial \mathbf{b}^1}$$

Backpropagation – Algorithm and equations

UFV
Universidade Federal de Viçosa



$$\frac{\partial L}{\partial \mathbf{W}^0} = \frac{\partial L}{\partial \mathbf{y}^0} \times \frac{\partial \mathbf{y}^0}{\partial \mathbf{v}^0} \times \frac{\partial \mathbf{v}^0}{\partial \mathbf{W}^0}$$

$$\frac{\partial L}{\partial \mathbf{y}^0} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^0}{\partial \mathbf{y}^0}$$

$$\boxed{\frac{\partial L}{\partial \mathbf{b}^0} = \frac{\partial L}{\partial \mathbf{y}^0} \times \frac{\partial \mathbf{y}^0}{\partial \mathbf{v}^0} \times \frac{\partial \mathbf{v}^0}{\partial \mathbf{b}^0}}$$

$$\frac{\partial L}{\partial \mathbf{W}^1} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^1}{\partial \mathbf{W}^1}$$

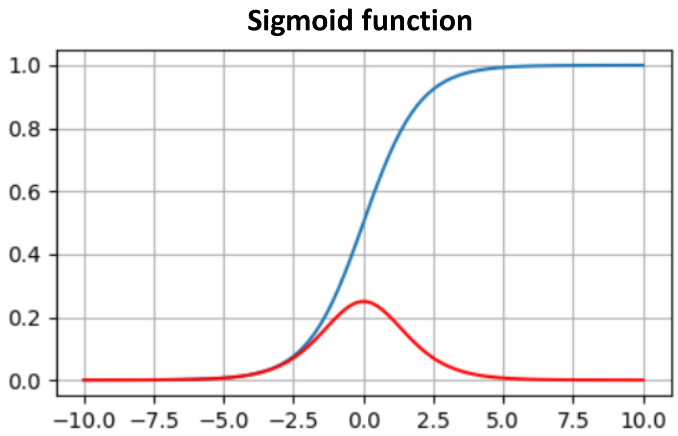
$$\frac{\partial L}{\partial \mathbf{b}^1} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^1}{\partial \mathbf{b}^1}$$

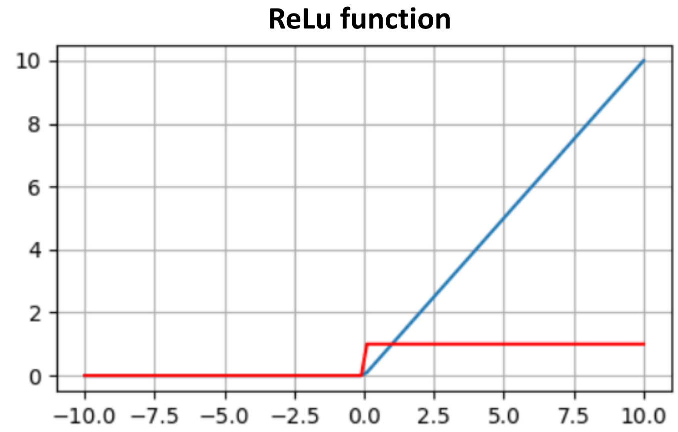
BACKPROPAGATION – FUNCTIONS AND DERIVATIVES

Backpropagation – Functions and derivatives

- Derivative of a function:
  - $\frac{d}{dx} f(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$

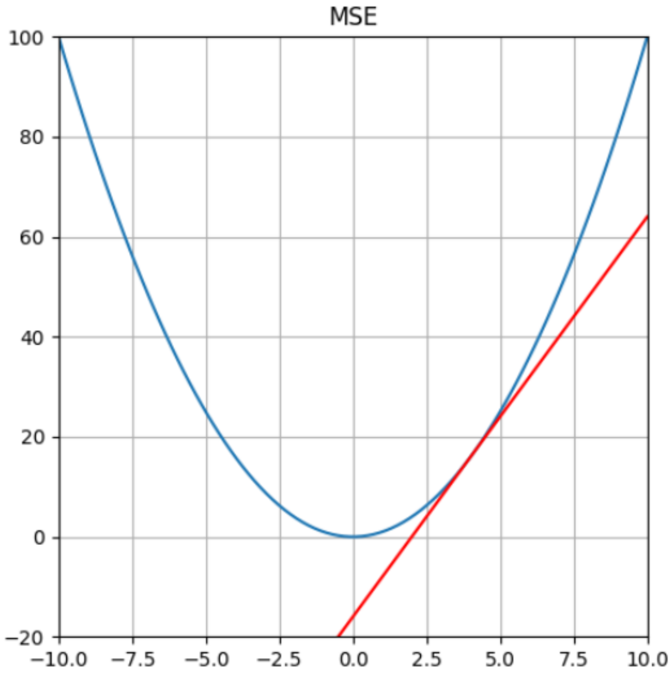
- Sigmoid function:
  - $g(v) = \frac{1}{1+e^{-v}}$
- Derivative of sigmoid function:
  - $\frac{d}{dv} g(v) = g(v)(1 - g(v))$
- ReLu function:
  - $g(v) = \max(0, v)$ ou $g(v) = \begin{cases} v, & v > 0 \\ 0, & v \leq 0 \end{cases}$
- Derivative of the ReLu function:
  - $\frac{d}{dv} g(v) = \begin{cases} 1, & v > 0 \\ 0, & v \leq 0 \end{cases}$





Backpropagation – Functions and derivatives

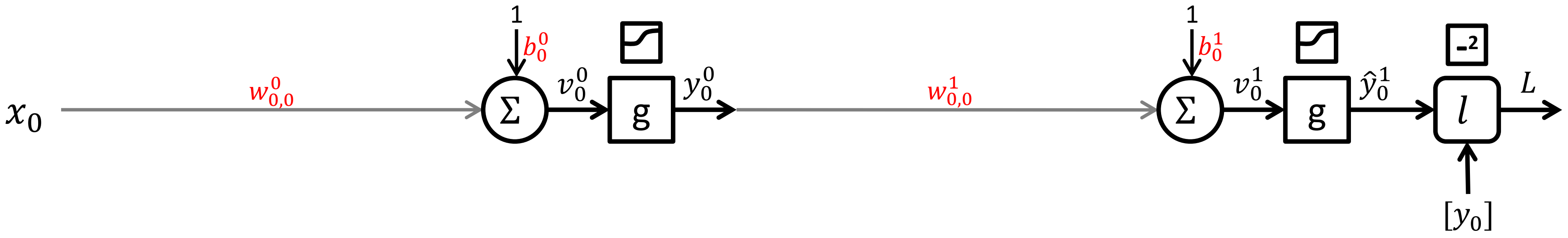
- Mean squared error (MSE):
  - $L(\hat{y}, y) = \frac{1}{N}(y - \hat{y})^{2}$, onde N é o número de classes.
- Partial derivative of the error function w.r.t. y :
  - $\frac{\partial L}{\partial y} = -\frac{2}{N}(y - \hat{y})$,



BACKPROPAGATION – EXAMPLES

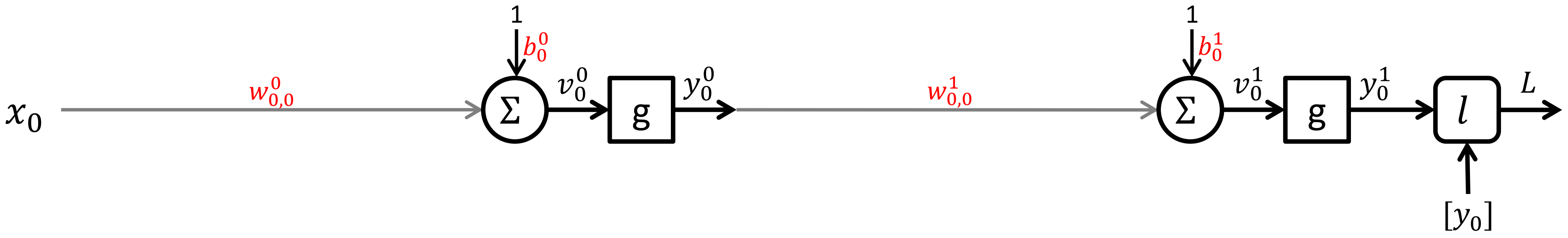
Backpropagation – Example 1

$\mathbf{x}$ $\mathbf{W}^0, \mathbf{b}^0$ $\mathbf{v}^0, \mathbf{y}^0$ $\mathbf{W}^1, \mathbf{b}^1$ $\mathbf{v}^1, \mathbf{y}^1$ L

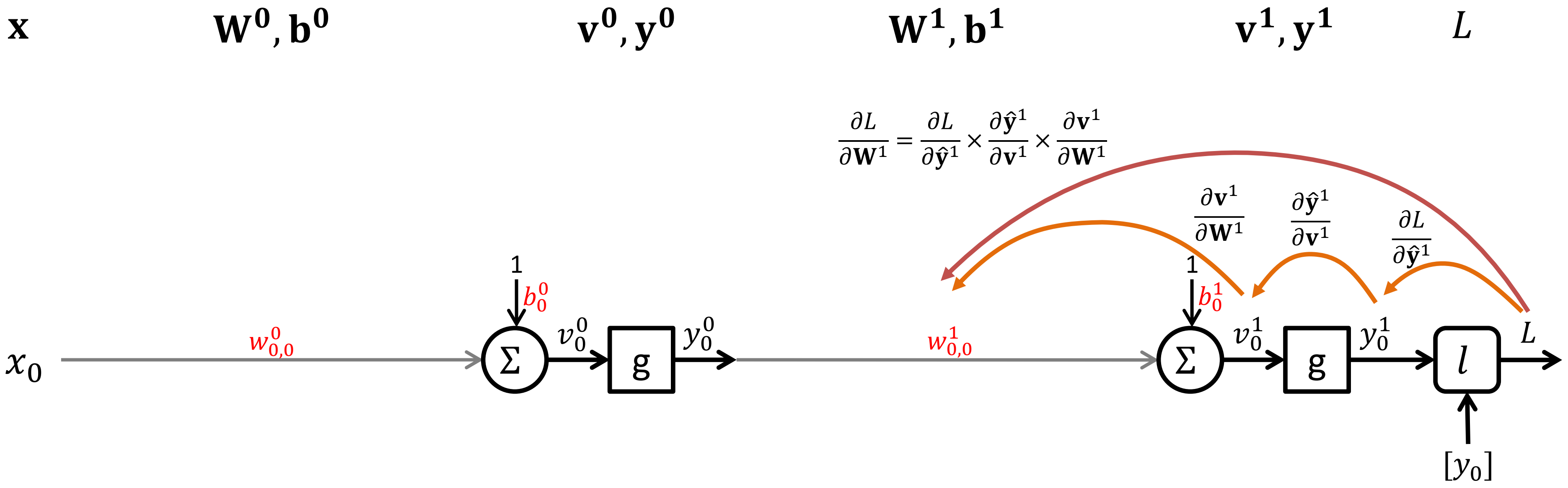


Backpropagation – Example 1

$\mathbf{x}$ $\mathbf{W}^0, \mathbf{b}^0$ $\mathbf{v}^0, \mathbf{y}^0$ $\mathbf{W}^1, \mathbf{b}^1$ $\mathbf{v}^1, \mathbf{y}^1$ L



Backpropagation – Example 1

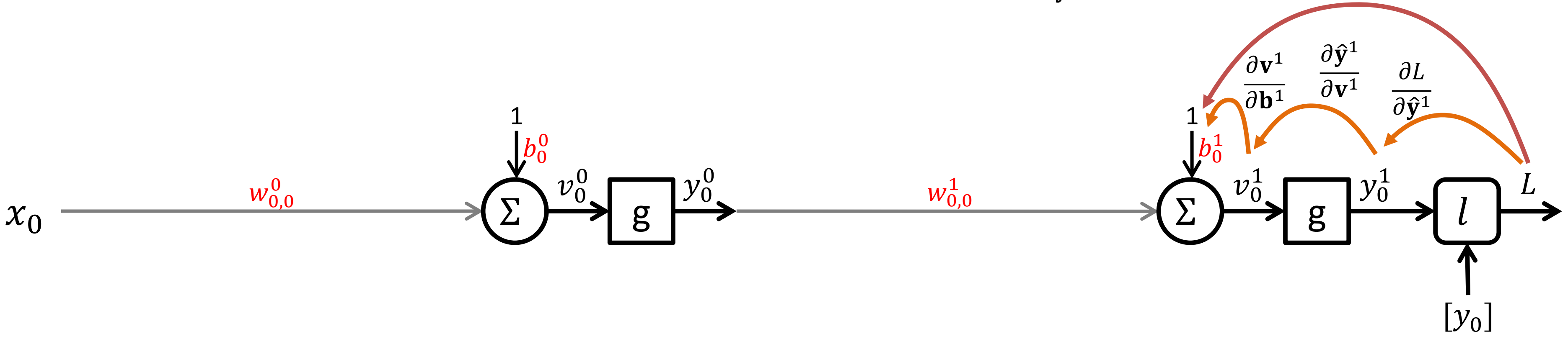


Backpropagation – Example 1

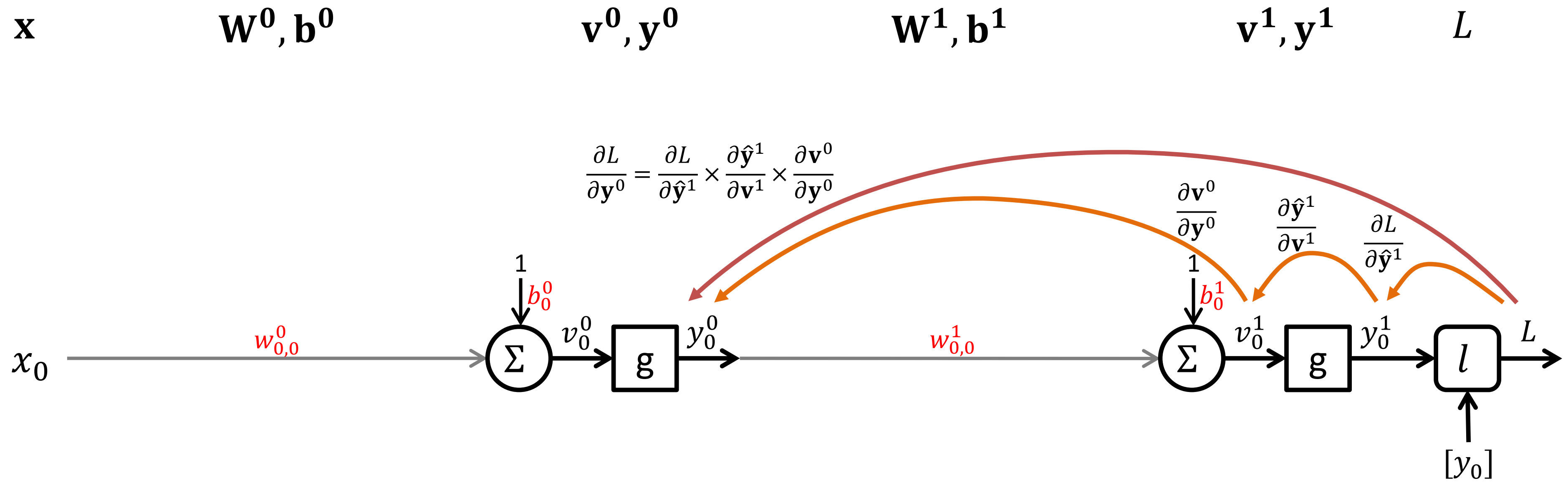
UFV
Universidade Federal de Viçosa

$\mathbf{x}$ $\mathbf{W}^0, \mathbf{b}^0$ $\mathbf{v}^0, \mathbf{y}^0$ $\mathbf{W}^1, \mathbf{b}^1$ $\mathbf{v}^1, \mathbf{y}^1$ L

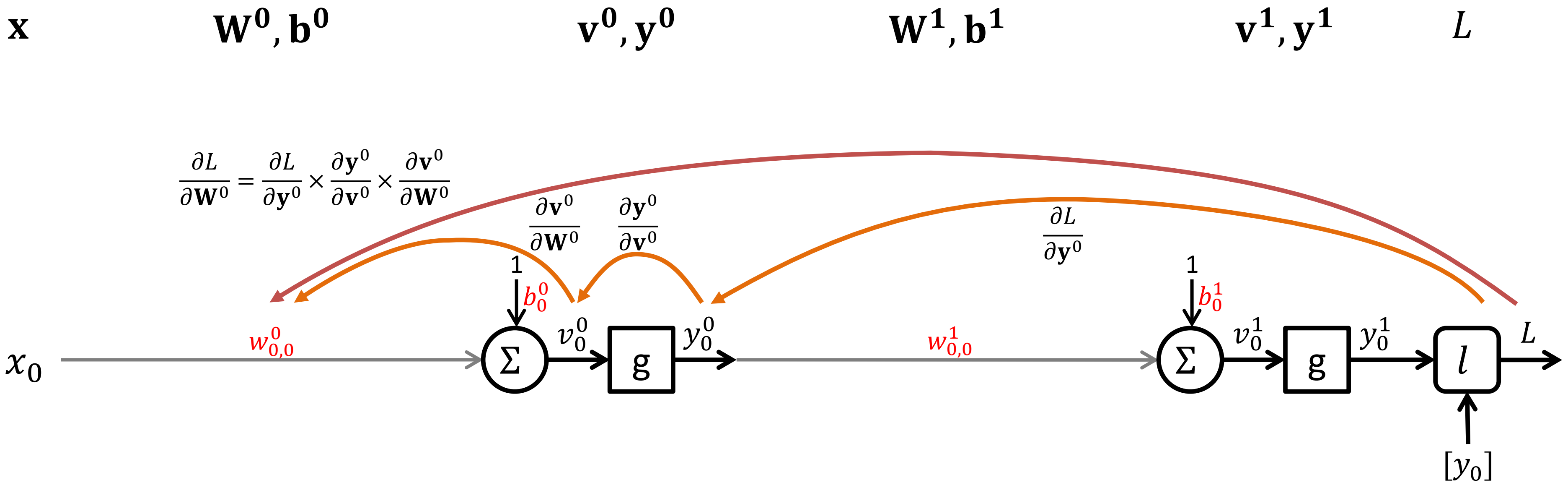
$$\frac{\partial L}{\partial \mathbf{b}^1} = \frac{\partial L}{\partial \hat{\mathbf{y}}^1} \times \frac{\partial \hat{\mathbf{y}}^1}{\partial \mathbf{v}^1} \times \frac{\partial \mathbf{v}^1}{\partial \mathbf{b}^1}$$



Backpropagation – Example 1



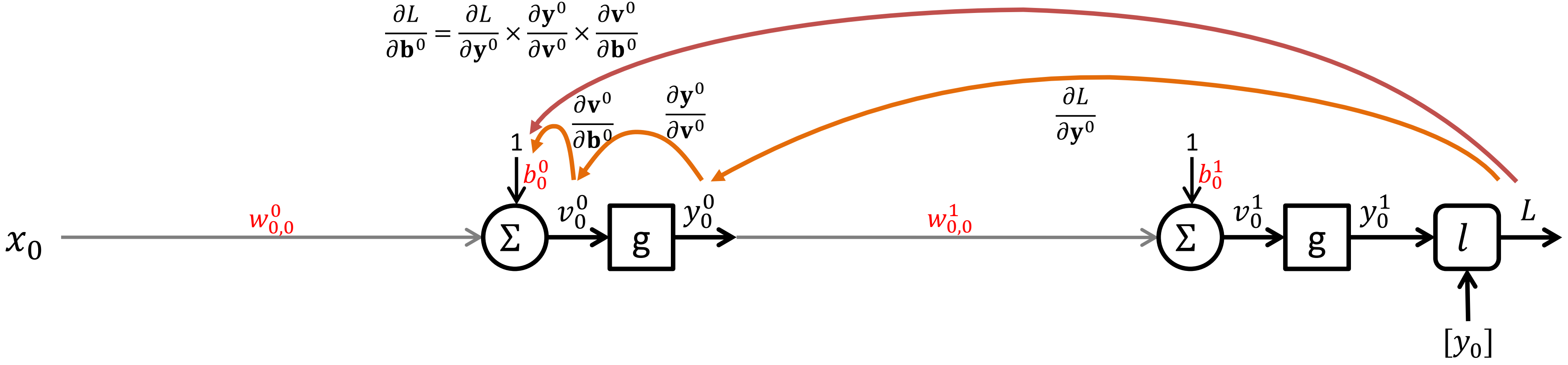
Backpropagation – Example 1



Backpropagation – Example 1

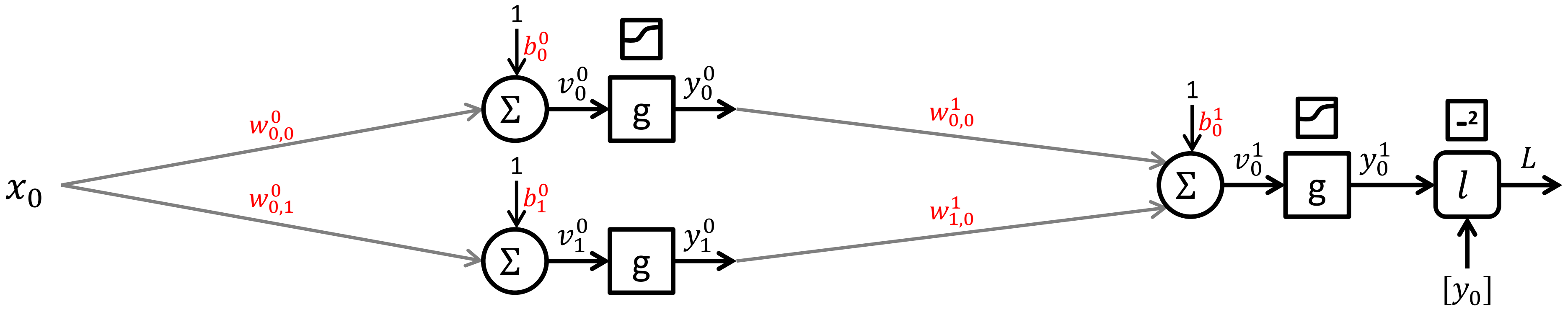
$\mathbf{x}$ $\quad\quad$ $\mathbf{W^0}, \mathbf{b^0}$ $\quad\quad$ $\mathbf{v^0}, \mathbf{y^0}$ $\quad\quad$ $\mathbf{W^1}, \mathbf{b^1}$ $\quad\quad$ $\mathbf{v^1}, \mathbf{y^1}$ $\quad\quad$ L

$$\frac{\partial L}{\partial \mathbf{b}^0} = \frac{\partial L}{\partial \mathbf{y}^0} \times \frac{\partial \mathbf{y}^0}{\partial \mathbf{v}^0} \times \frac{\partial \mathbf{v}^0}{\partial \mathbf{b}^0}$$



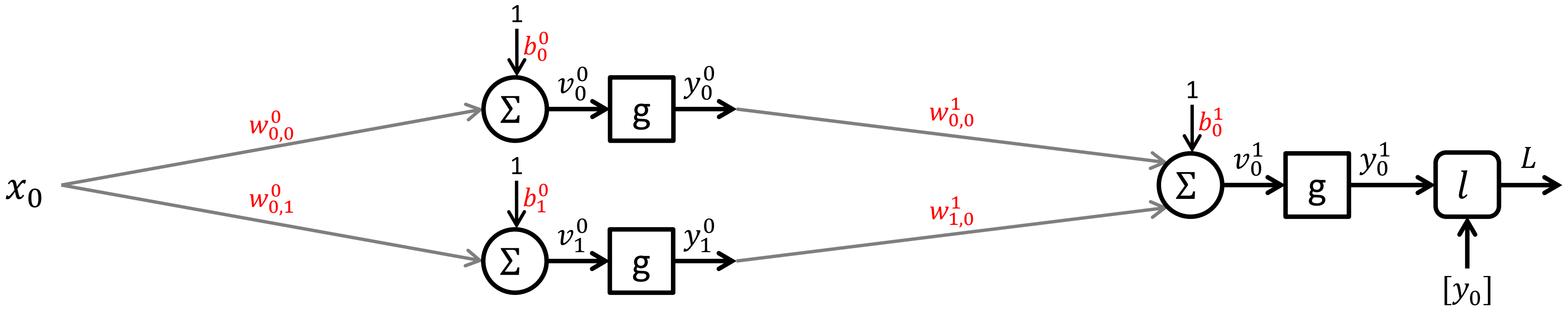
Backpropagation – Example 2

$\mathbf{x}$ $\mathbf{W}^0, \mathbf{b}^0$ $\mathbf{v}^0, \mathbf{y}^0$ $\mathbf{W}^1, \mathbf{b}^1$ $\mathbf{v}^1, \mathbf{y}^1$ L

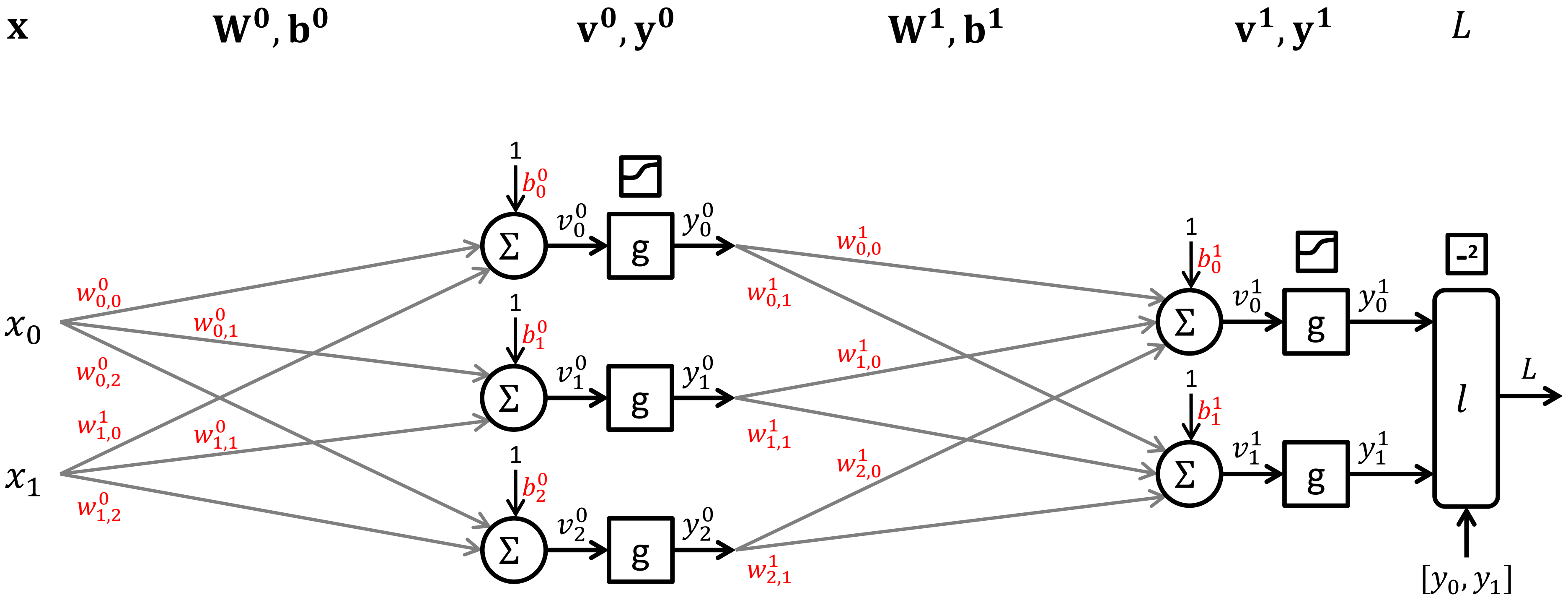


Backpropagation – Example 2

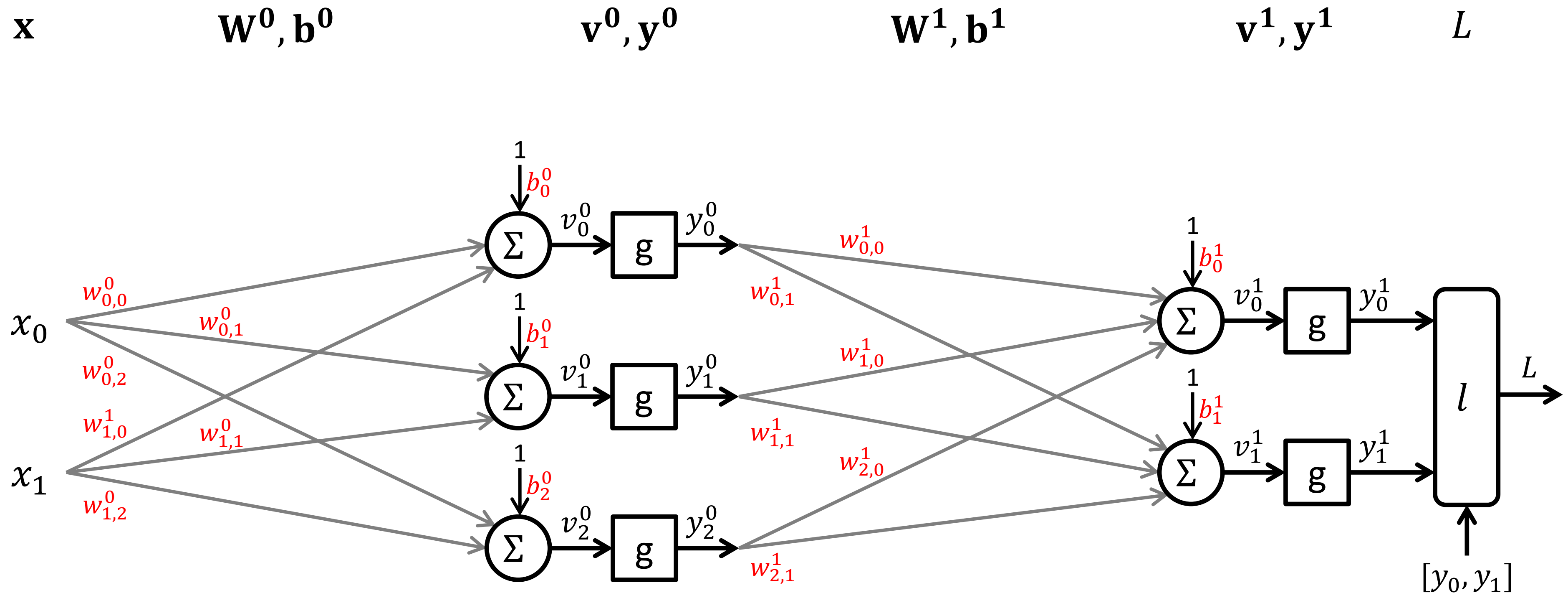
$\mathbf{x}$ $\mathbf{W^0}, \mathbf{b^0}$ $\mathbf{v^0}, \mathbf{y^0}$ $\mathbf{W^1}, \mathbf{b^1}$ $\mathbf{v^1}, \mathbf{y^1}$ L



Backpropagation – Example 2



Backpropagation – Example 2



Bibliography

Universidade Federal de Viçosa

- Ponti et al. **Everything You Wanted to Know about Deep Learning for Computer Vision but Were Afraid to Ask**. Sibgrapi 2017.
  - https://sites.icmc.usp.br/moacir/p17sibgrapi-tutorial/
- Martin Görner. **Learn TensorFlow and deep learning, without a Ph.D**. Google Cloud.
  - https://cloud.google.com/blog/products/gcp/learn-tensorflow-and-deep-learning-without-a-phd
- CS231n: Convolutional Neural Networks for Visual Recognition. Stanford University.
  - http://cs231n.stanford.edu/
  - http://cs231n.github.io/
- Goodfellow, Bengio and Courville. **Deep Learning**. MIT Press, 2016
  - https://www.deeplearningbook.org/

Bibliography

- Rabindra Lamsal. A step by step forward pass and backpropagation example
  - https://theneuralblog.com/forward-pass-backpropagation-example/
- Matt Mazur. A Step by Step Backpropagation Example
  - https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

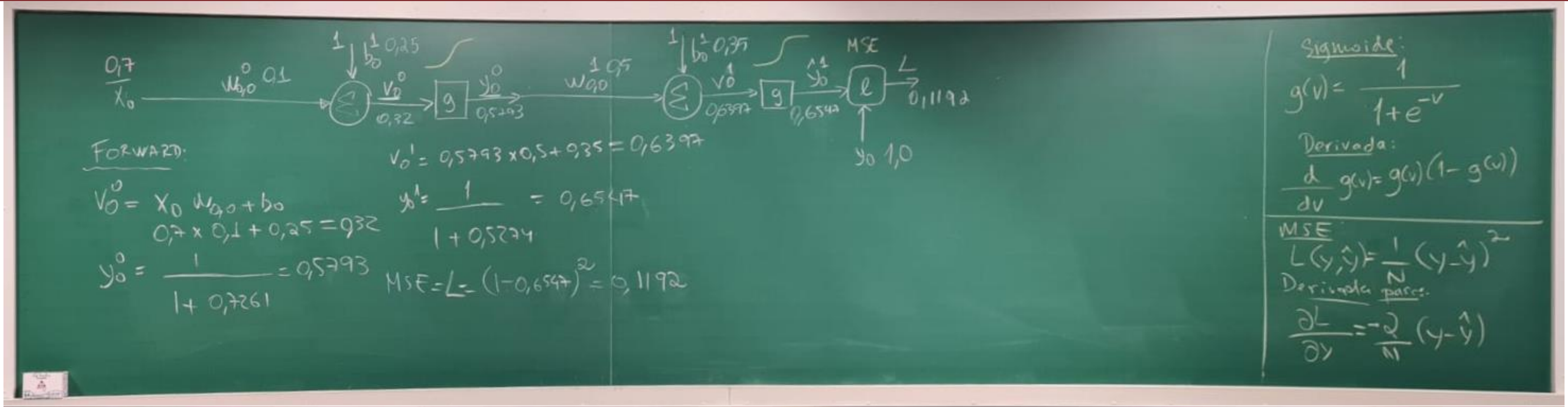
Bibliography

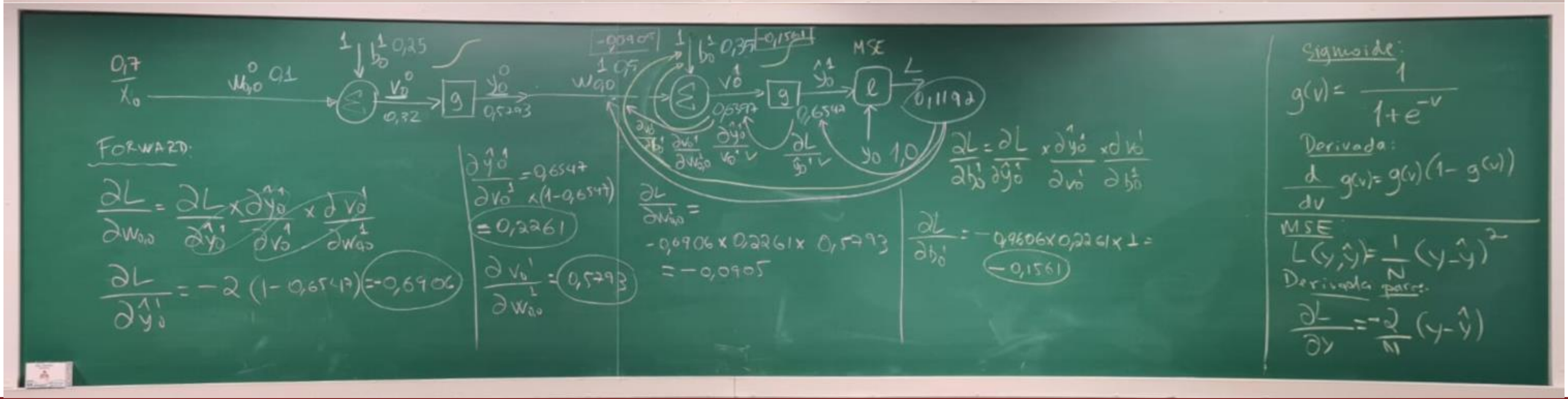
UFV
Universidade Federal de Viçosa

- Back-Propagation is very simple. Who made it Complicated ?
  - https://medium.com/@14prakash/back-propagation-is-very-simple-who-made-it-complicated-97b794c97e5c
- Chapter 7: Artificial neural networks with Math.
  - https://medium.com/deep-math-machine-learning-ai/chapter-7-artificial-neural-networks-with-math-bb711169481b
- The Matrix Calculus You Need For Deep Learning
  - http://explained.ai/matrix-calculus/index.html
- How backpropagation works, and how you can use Python to build a neural network
  - https://medium.freecodecamp.org/build-a-flexible-neural-network-with-backpropagation-in-python-acffeb7846d0
- All the Backpropagation derivatives
  - https://medium.com/@pdquant/all-the-backpropagation-derivatives-d5275f727f60
- Brent Scarff. **Understanding Backpropagation**.
  - https://towardsdatascience.com/understanding-backpropagation-abcc509ca9d0

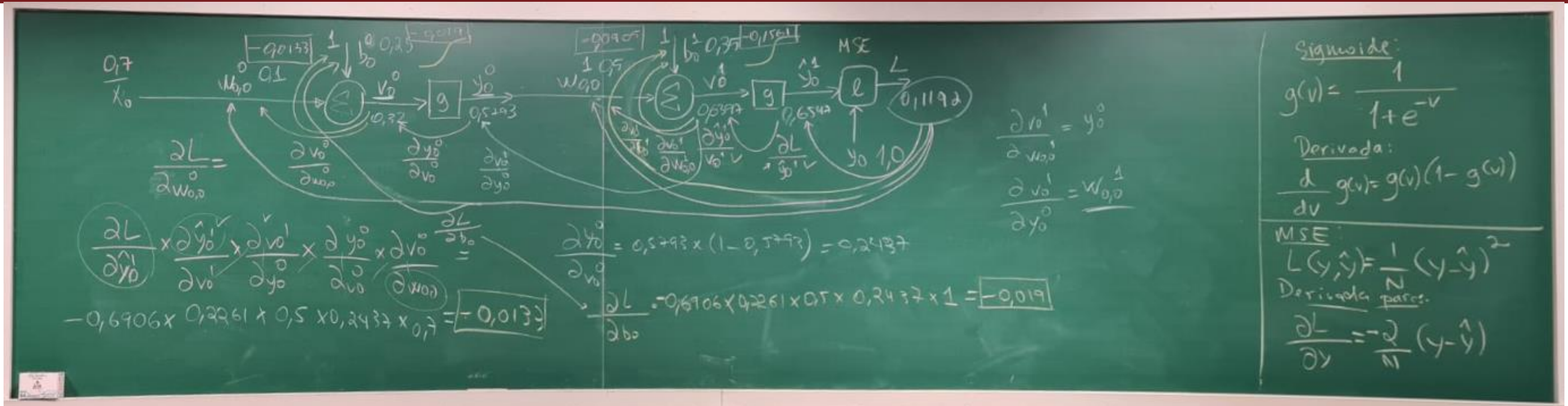
APPENDIX: EXAMPLE

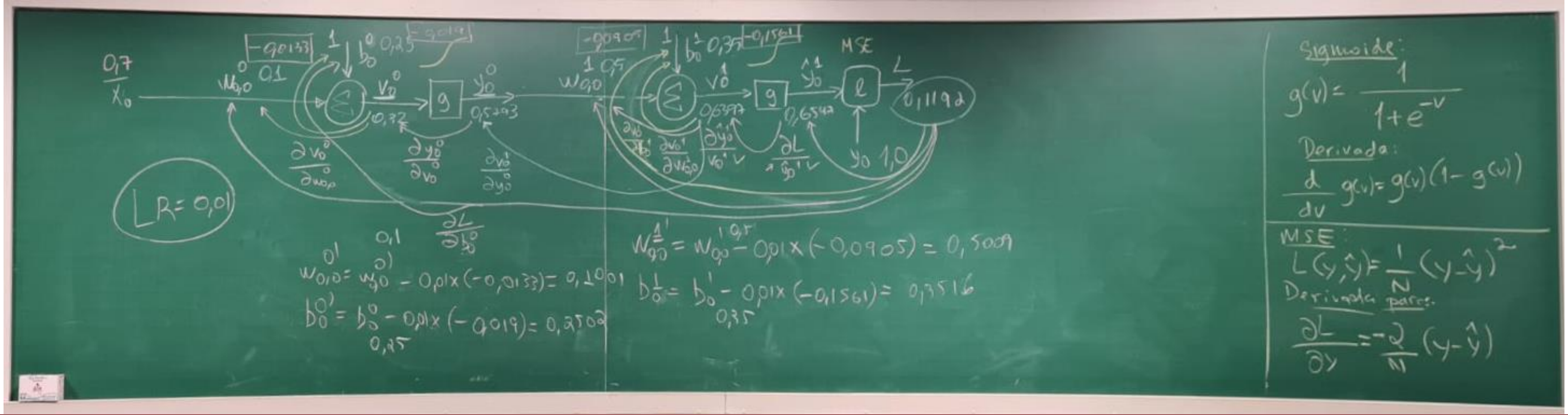
Example 1


FORWARD:
V0 = X0 W0,0 + b0
0,7 x 0,1 + 0,25 = 0,32
y0 = 1 / (1 + 0,7261) = 0,5793
V0' = 0,5793 x 0,5 + 0,35 = 0,6397
y0' = 1 / (1 + 0,5274) = 0,6547
MSE = L = (1 - 0,6547)² = 0,1192
Sigmoide:
g(v) = 1 / (1 + e^-v)
Derivada:
d/dv g(v) = g(v)(1 - g(v))
MSE
L(y, ŷ) = 1/N (y - ŷ)²
Derivada parcial:
∂L/∂ŷ = -2/N (y - ŷ)


FORWARD:
∂L/∂W0,0 = ∂L/∂ŷ0' x ∂ŷ0'/∂V0' x ∂V0'/∂W0,0
∂L/∂ŷ0' = -2 (1 - 0,6547) = -0,6906
∂ŷ0'/∂V0' = 0,6547 x (1 - 0,6547) = 0,2261
∂V0'/∂W0,0 = 0,5793
∂L/∂W0,0 = -0,6906 x 0,2261 x 0,5793 = -0,0905
∂L/∂b0' = ∂L/∂ŷ0' x ∂ŷ0'/∂V0' x ∂V0'/∂b0'
∂L/∂b0' = -0,6906 x 0,2261 x 1 = -0,1561

Example 1





THE END