Learning for vision II
Neural networks

Karel Zimmermann

http://cmp.felk.cvut.cz/~zimmerk/

Vision for Robotics and Autonomous Systems

https://cyber.felk.cvut.cz/vras/

Center for Machine Perception

https://cmp.felk.cvut.cz

CTU

CZECH TECHNICAL UNIVERSITY IN PRAGUE

Department for Cybernetics

Faculty of Electrical Engineering

Czech Technical University in Prague

Outline

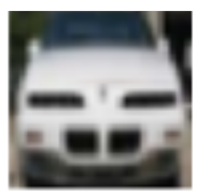
- Neuron+ computational graph
- Fully connected neural network

Linear classifier and neuron

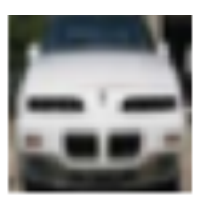
Labels | RGB images

$+1$

-1

def classify():

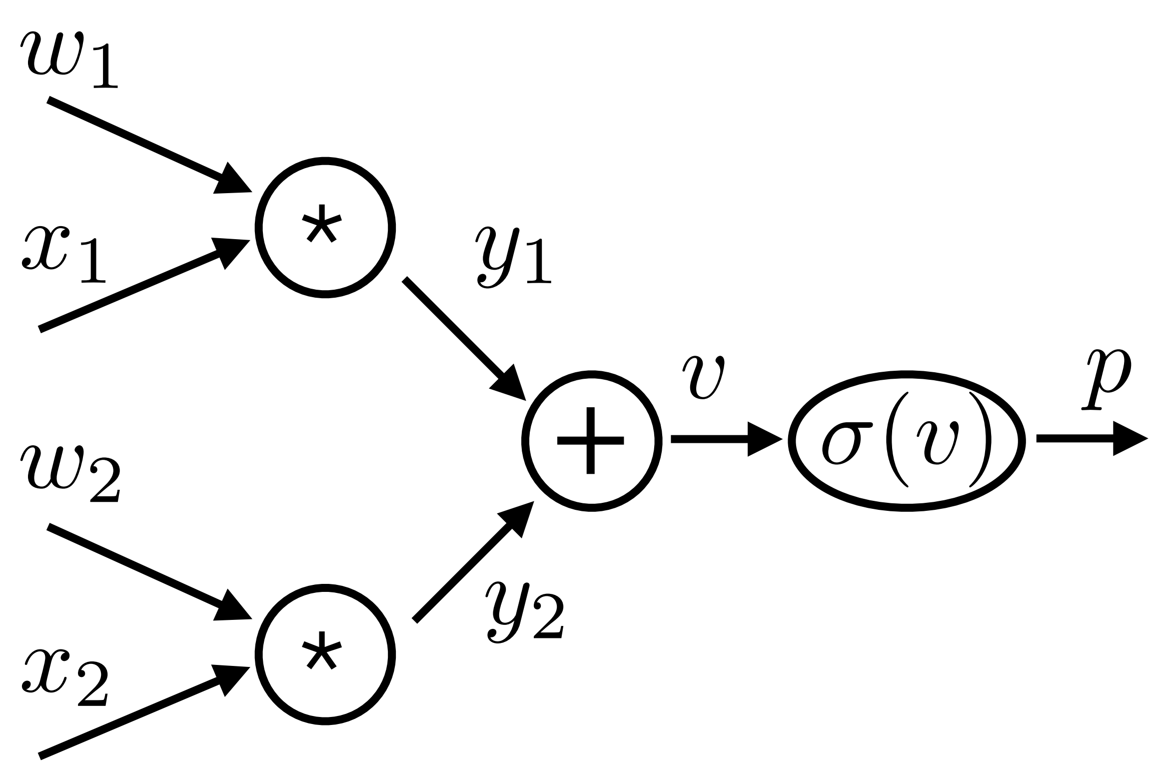
 # Linear classifier

$\mathbf{x} = \mathrm{vec}($  $)$

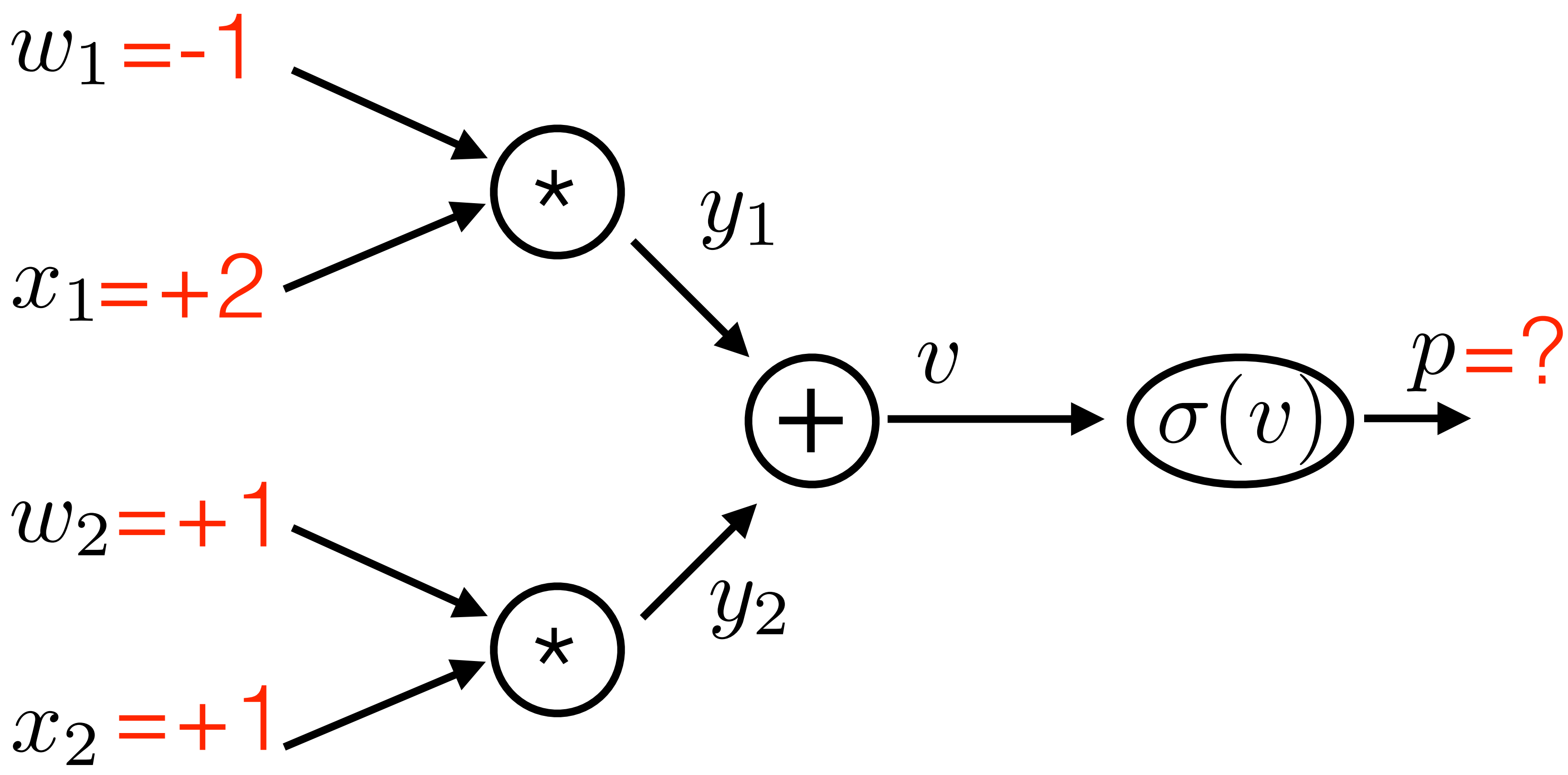
$p = \sigma\left(\mathbf{w}^\top \mathbf{x}\right)$

return p

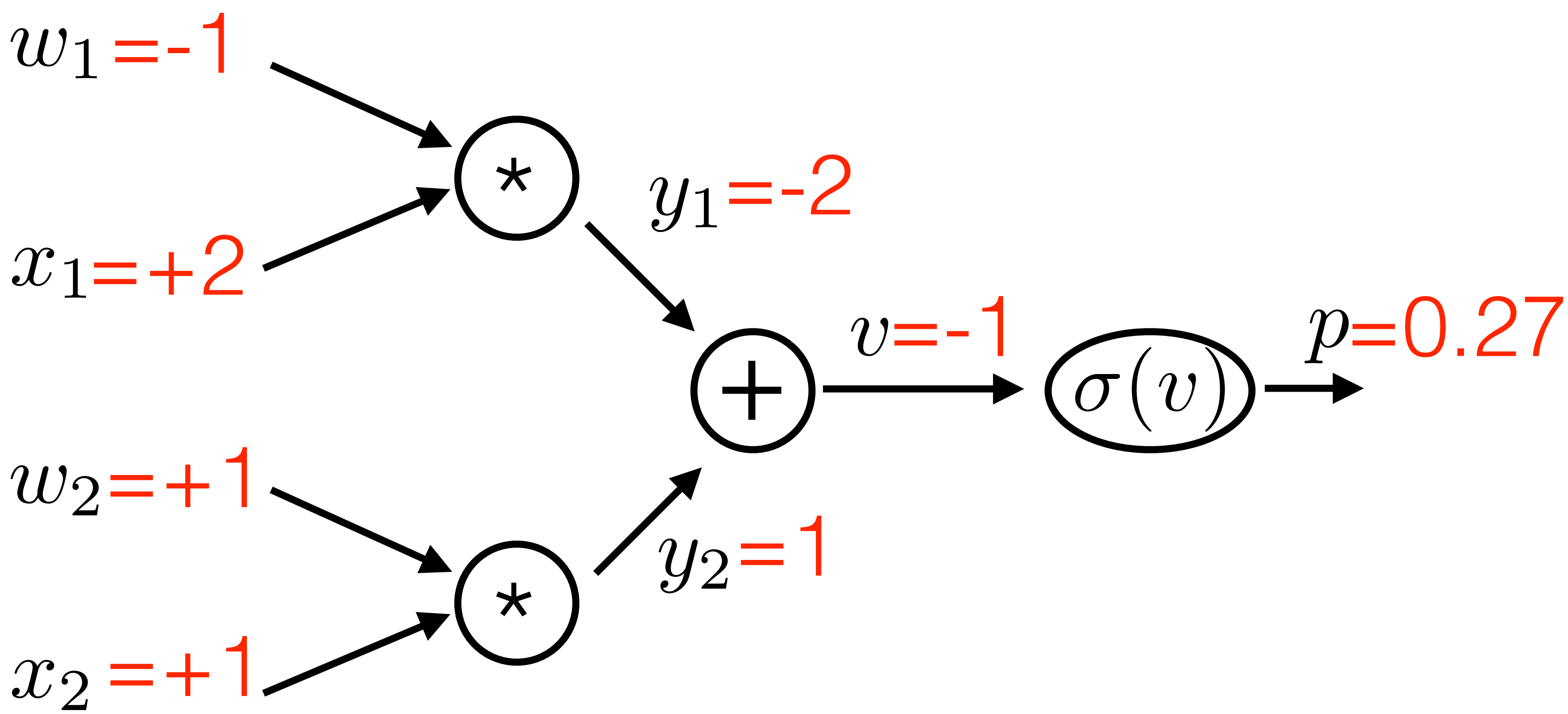
Computational graph of linear classifier



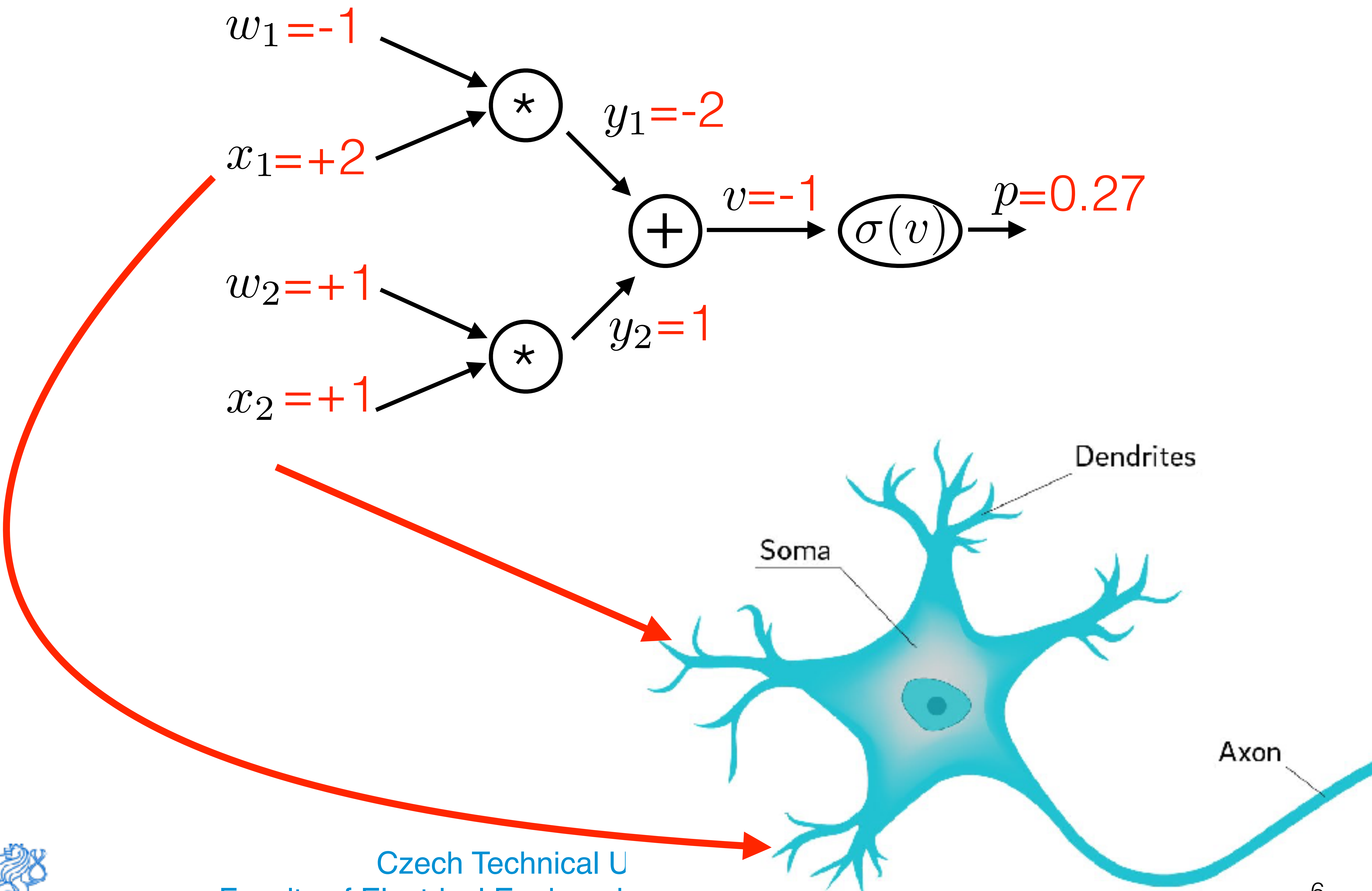
Example I: given trained neuron, and input, what is output?



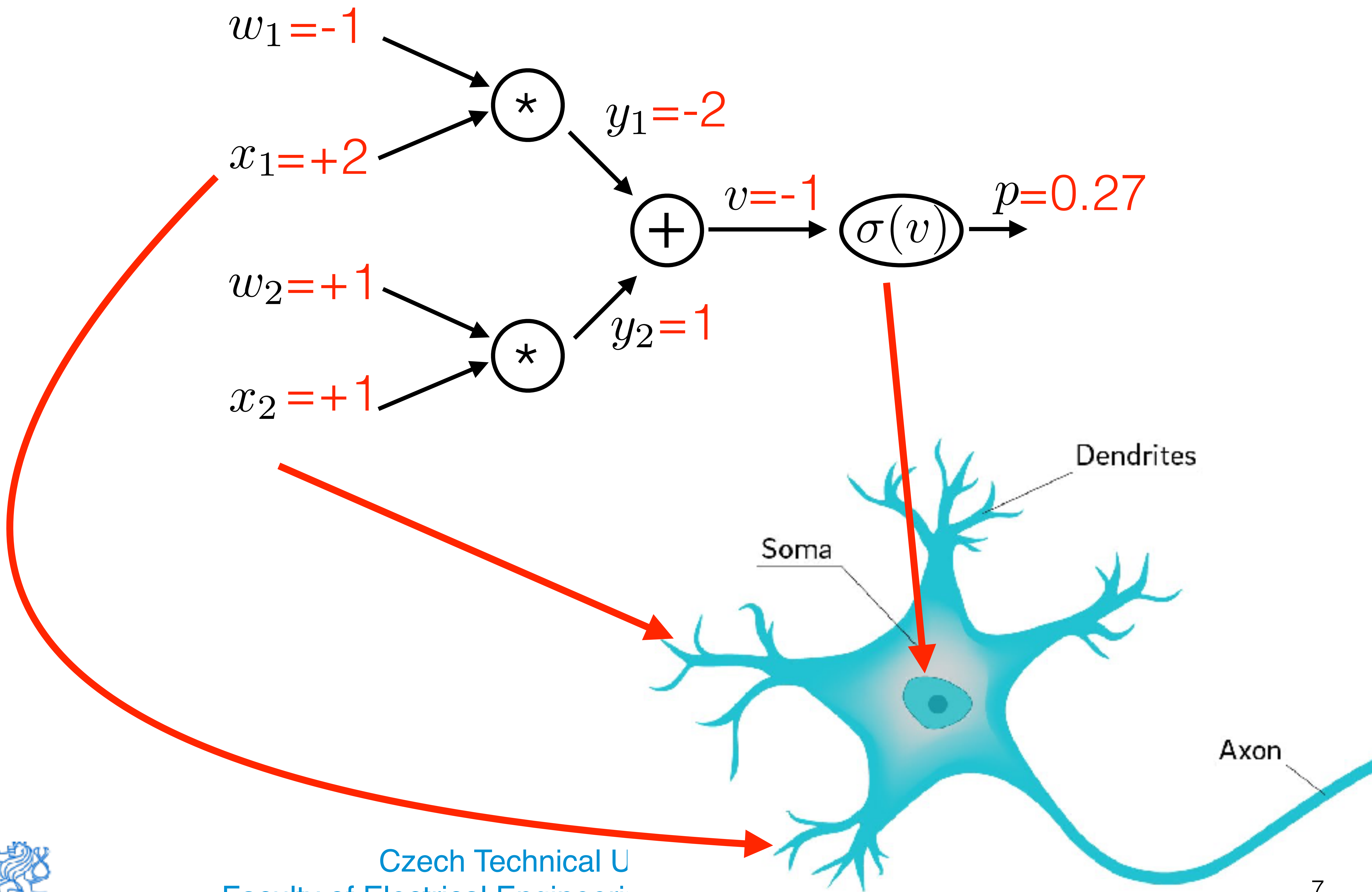
Example I: given trained classifier, and input, what is output?



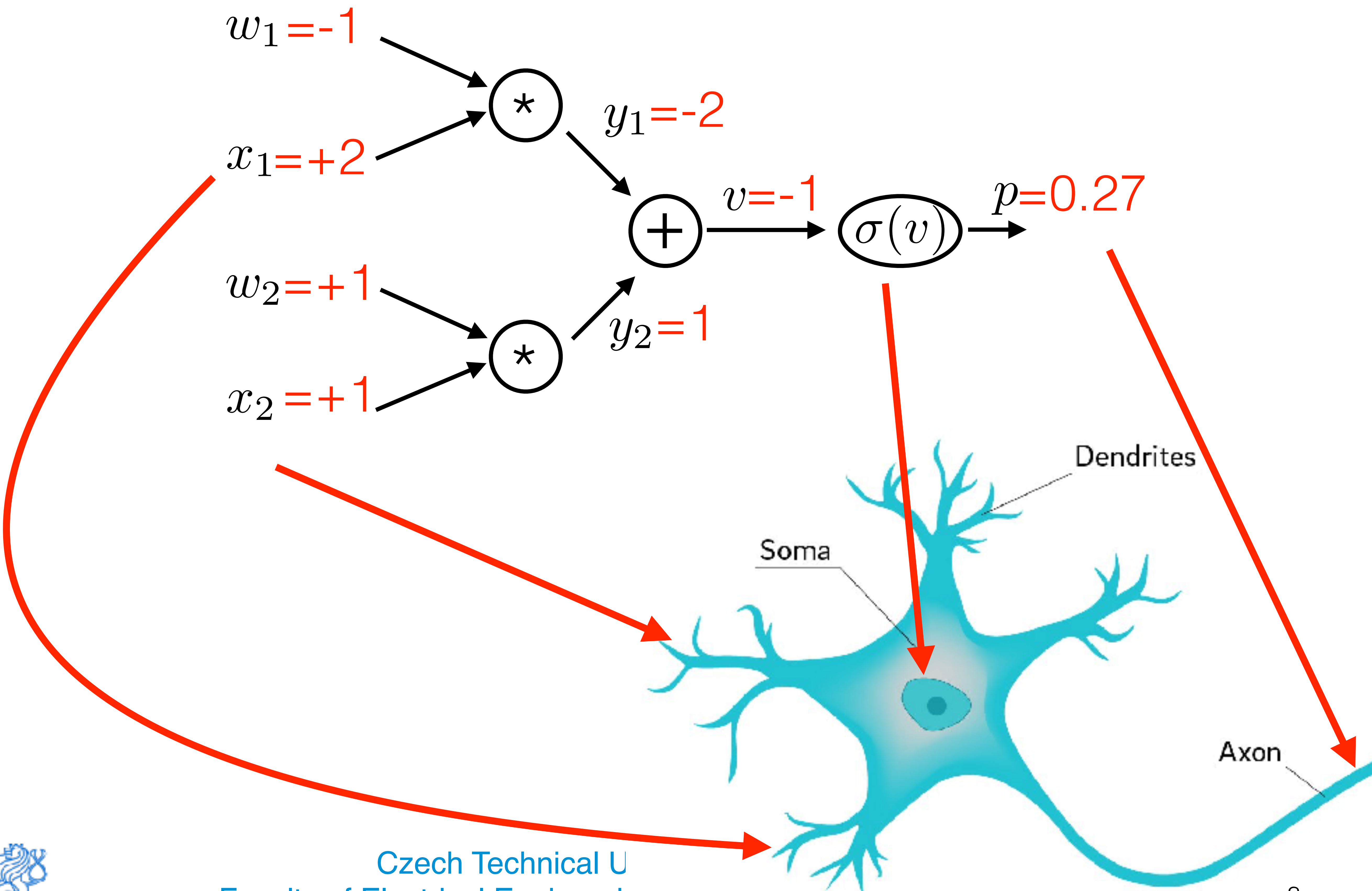
Relation to biological neuron



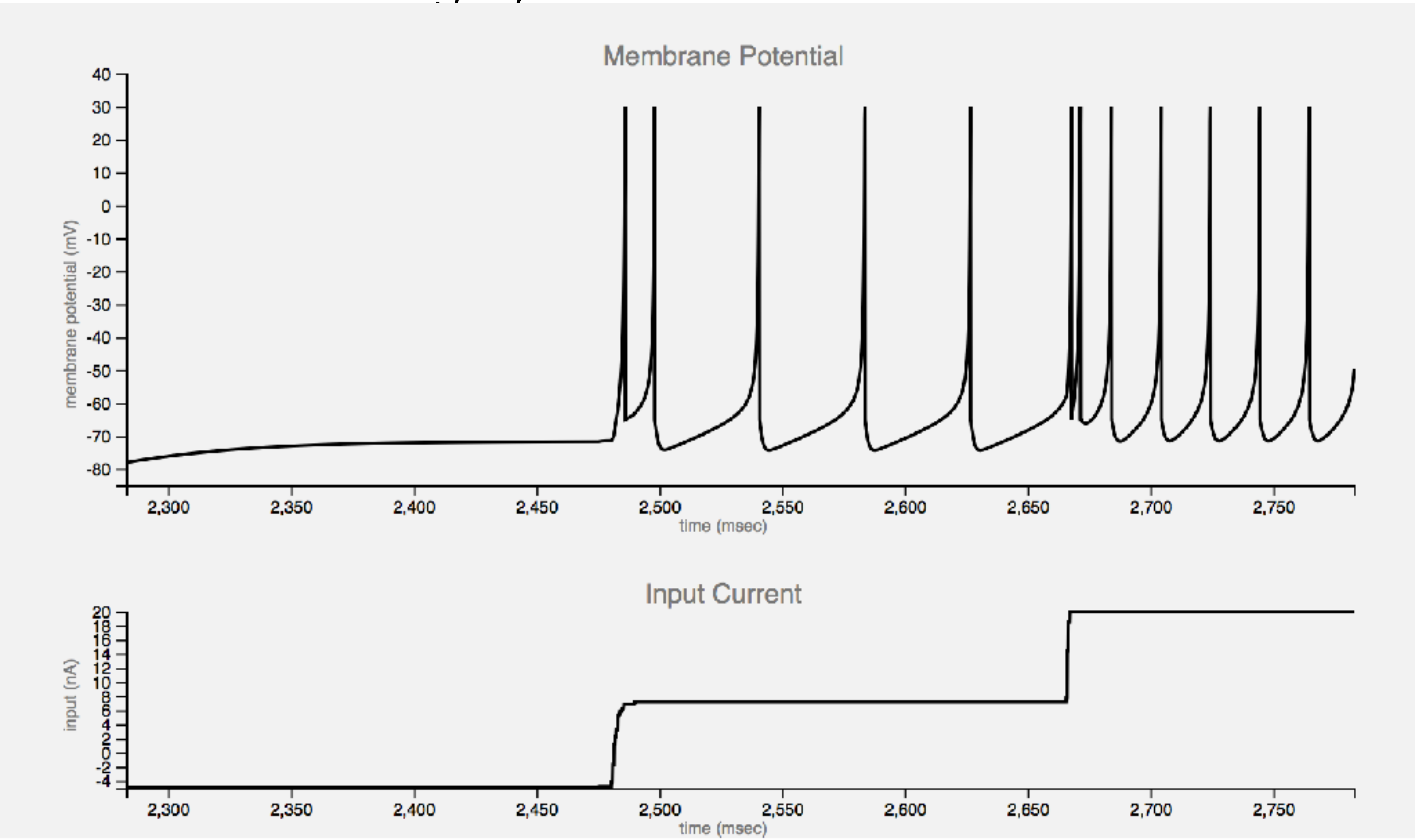
Relation to biological neuron



Relation to biological neuron



Modeling dynamic neuron behaviour



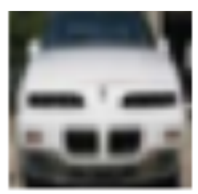
http://jackterwilliger.com/biological-neural-networks-part-i-spiking-neurons/

Linear classifier and neuron

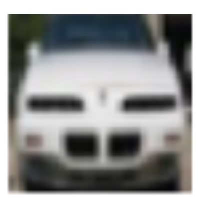
Labels | RGB images

$+1$

-1

def classify():

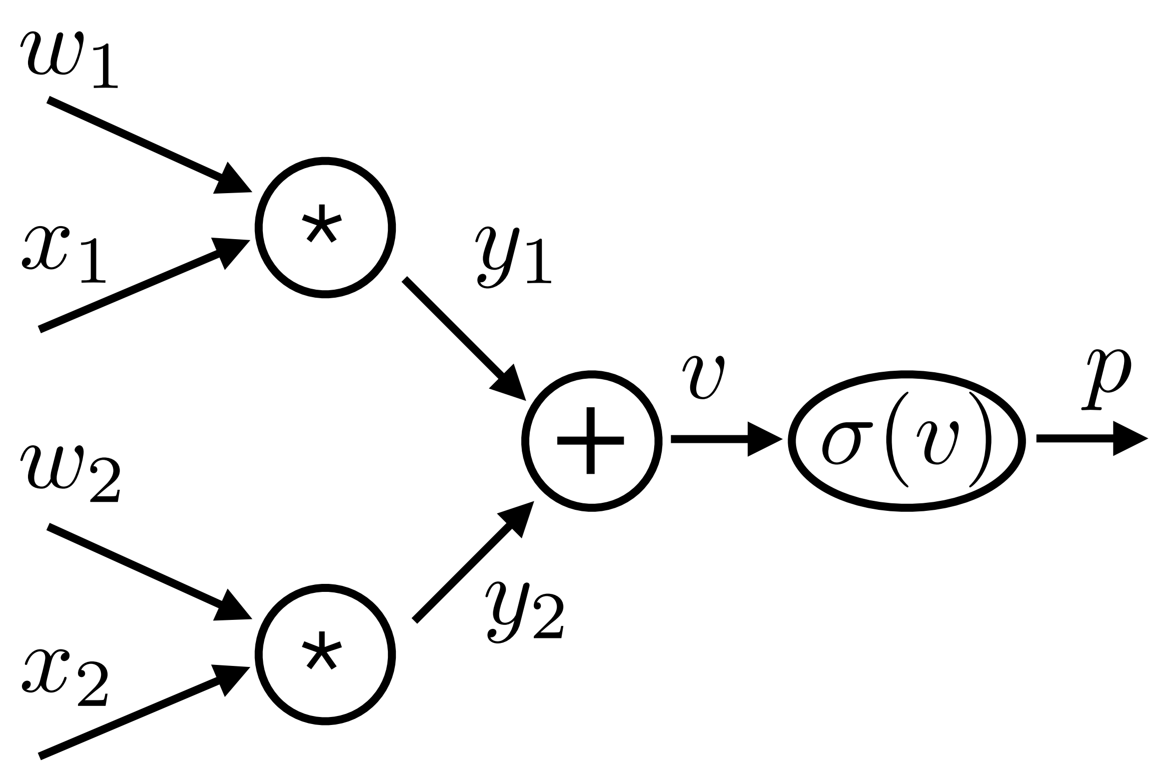
 # Linear classifier

$\mathbf{x} = \text{vec}($  $)$

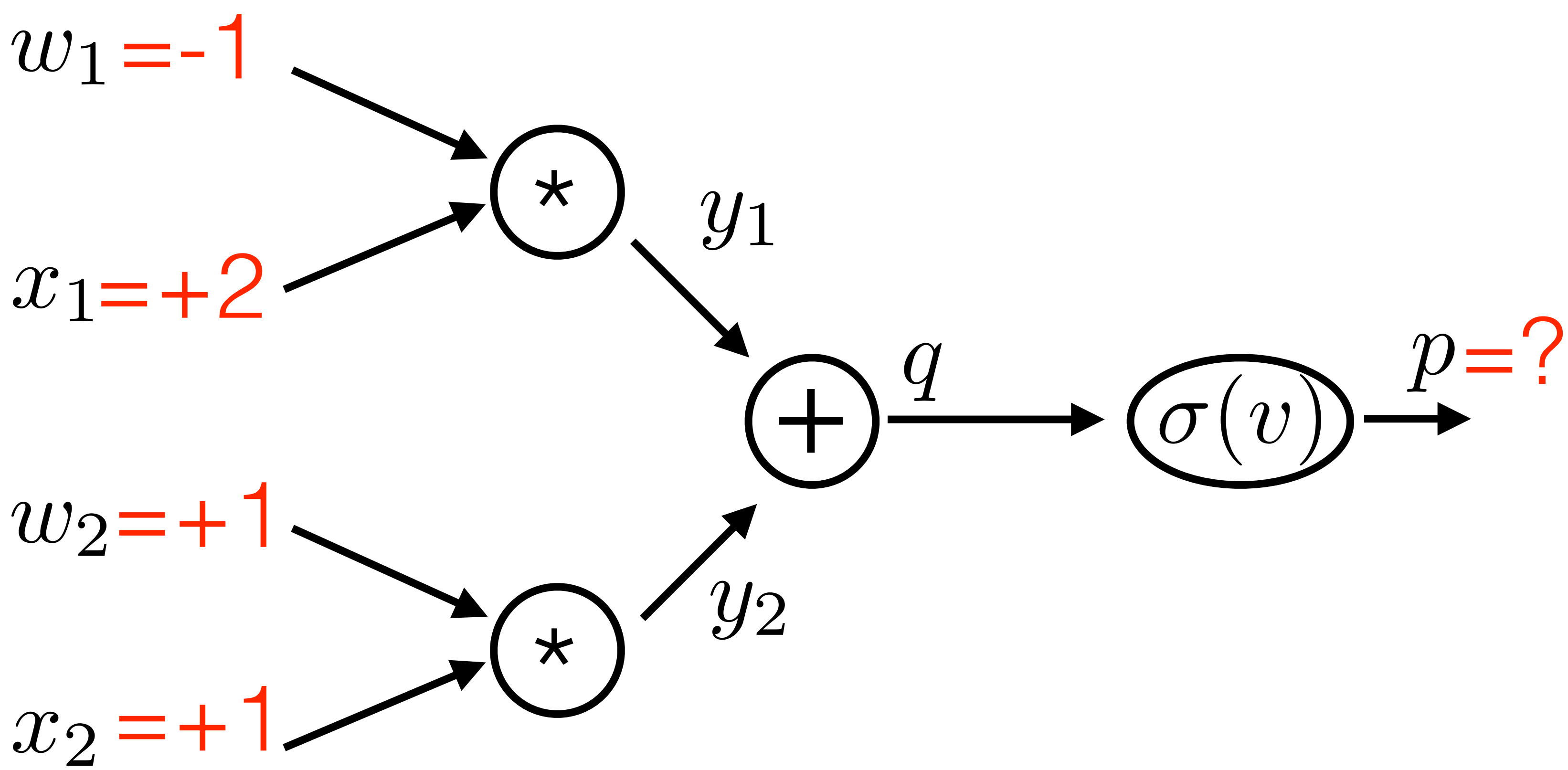
$p = \sigma\left(\mathbf{w}^\top \mathbf{x}\right)$

return p

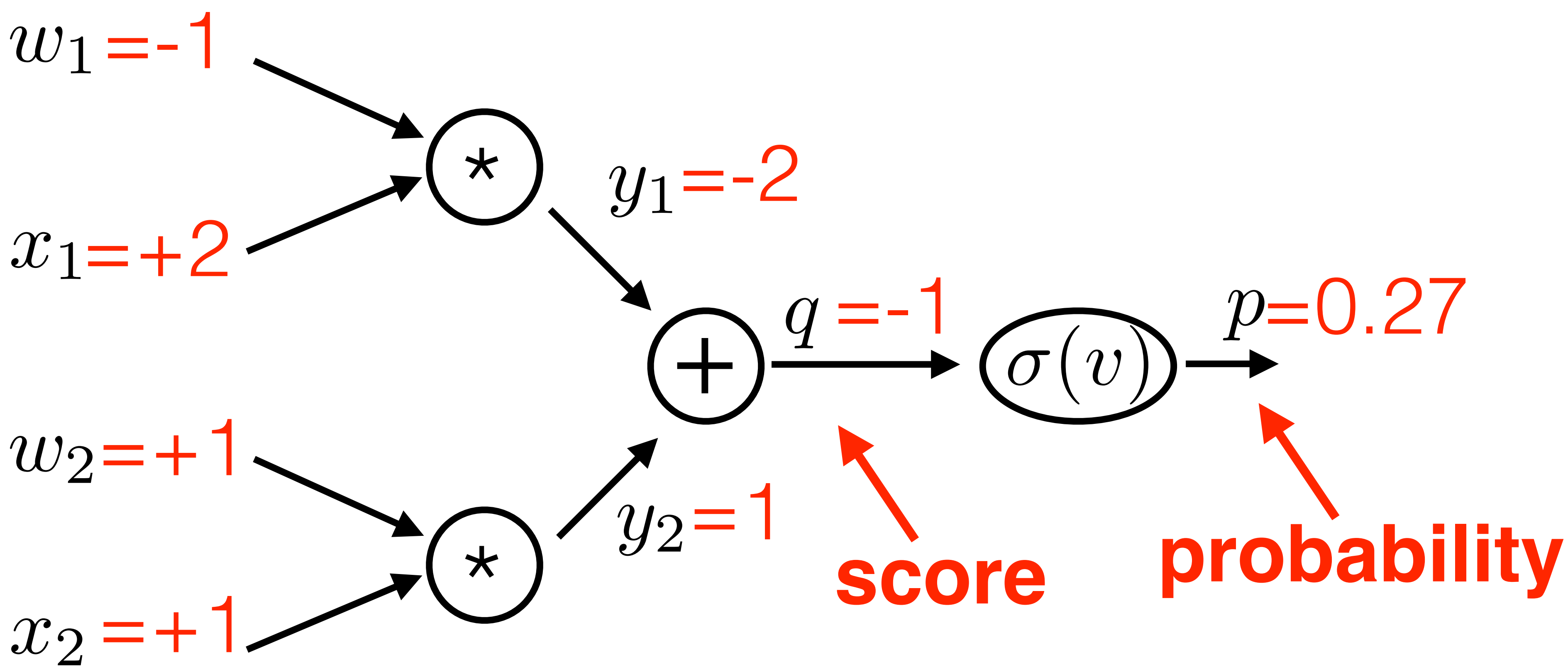
Computational graph of linear classifier



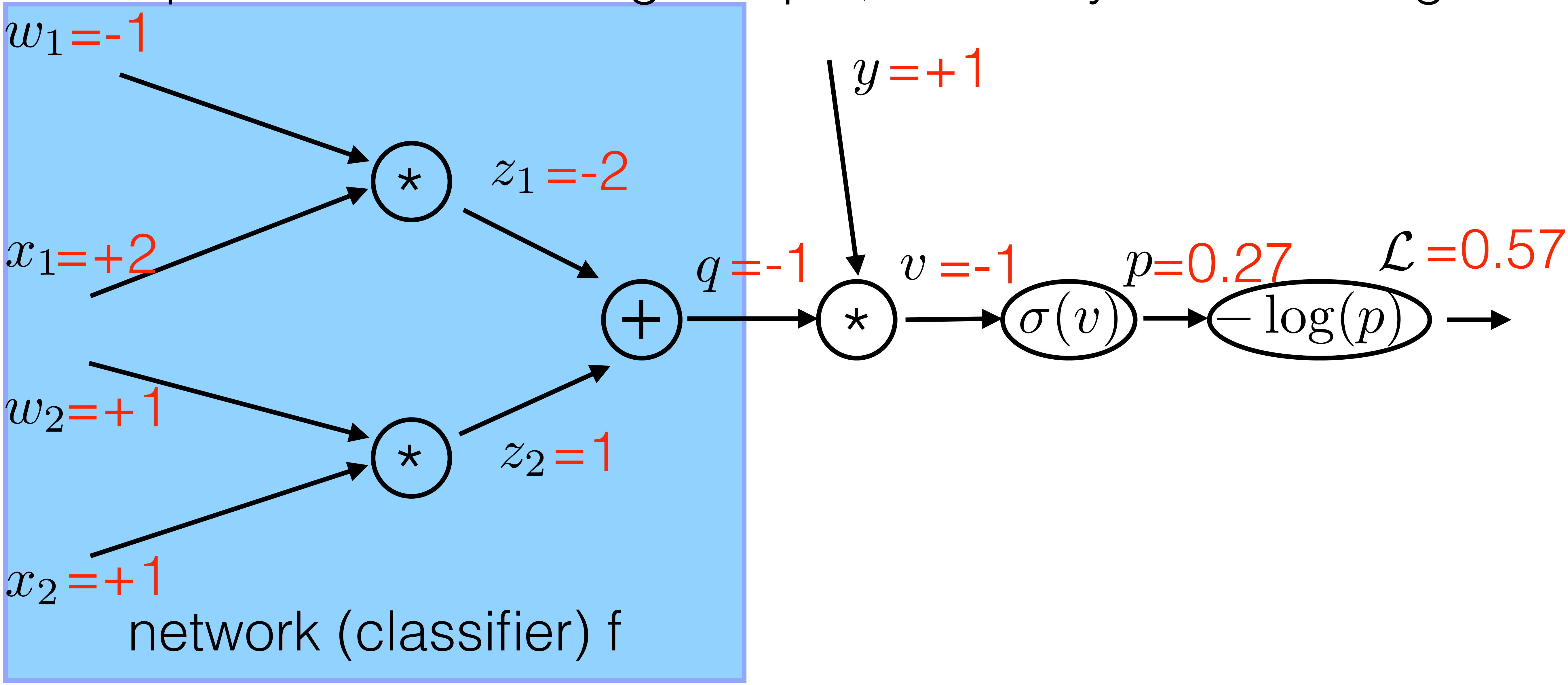
Example I: given trained neuron, and input, what is output?



Example I: given trained classifier, and input, what is output?

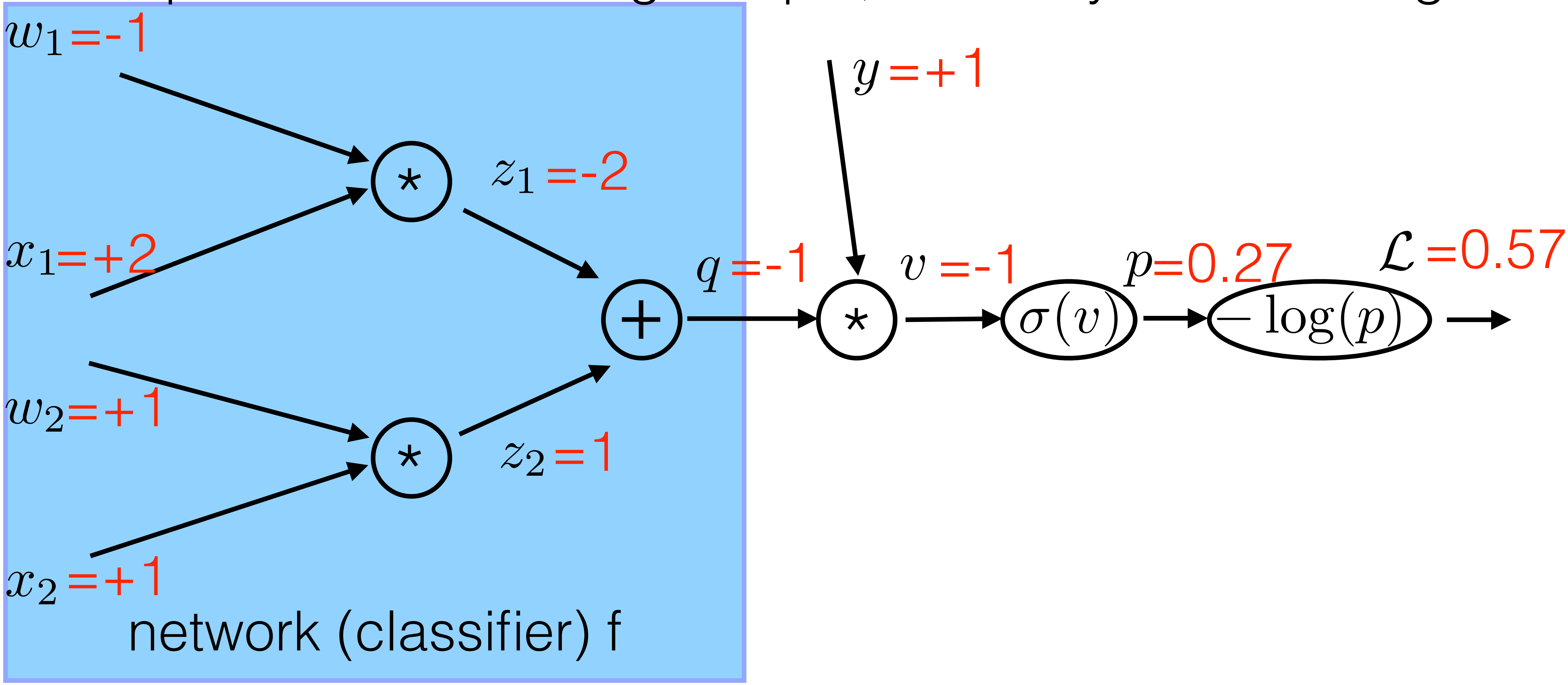


Example II: Given training sample, how do you learn weights?

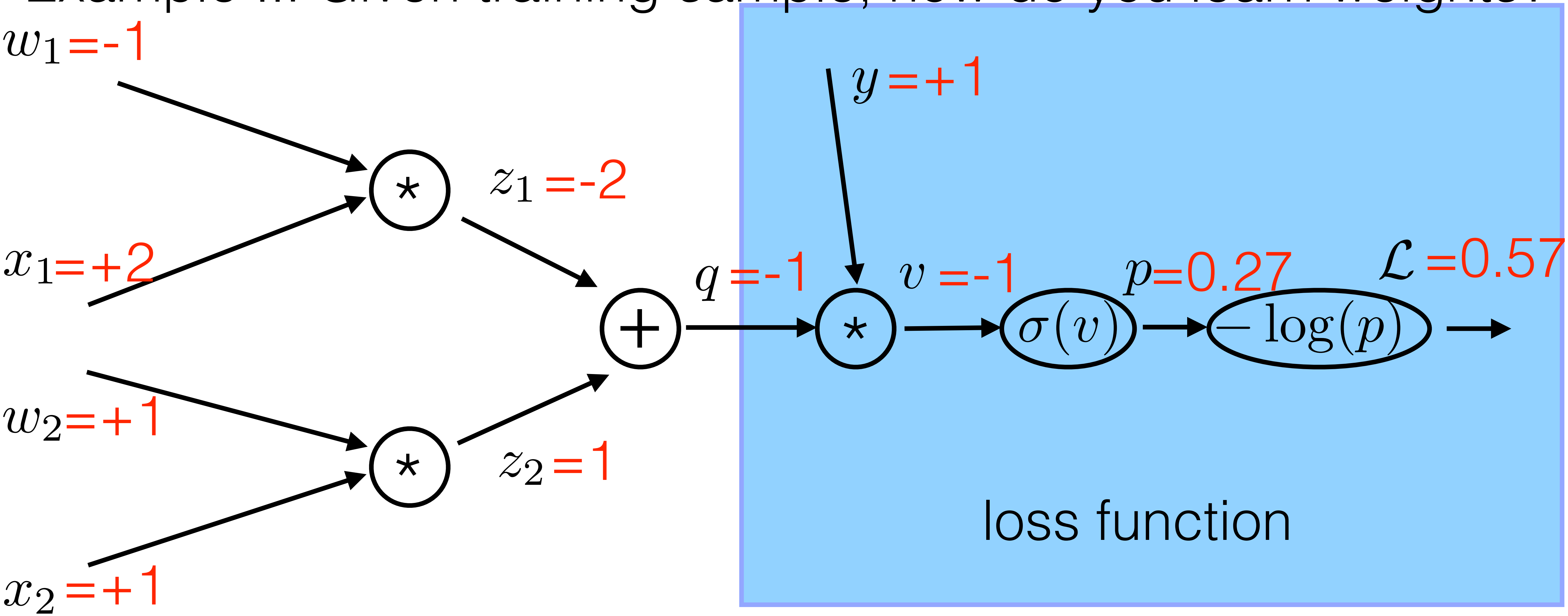


$$\arg\min_{\mathbf{w}} \Big(-\log\big[\sigma\big(y_i\, f(\mathbf{x}_i, \mathbf{w})\big)\big] \Big)$$

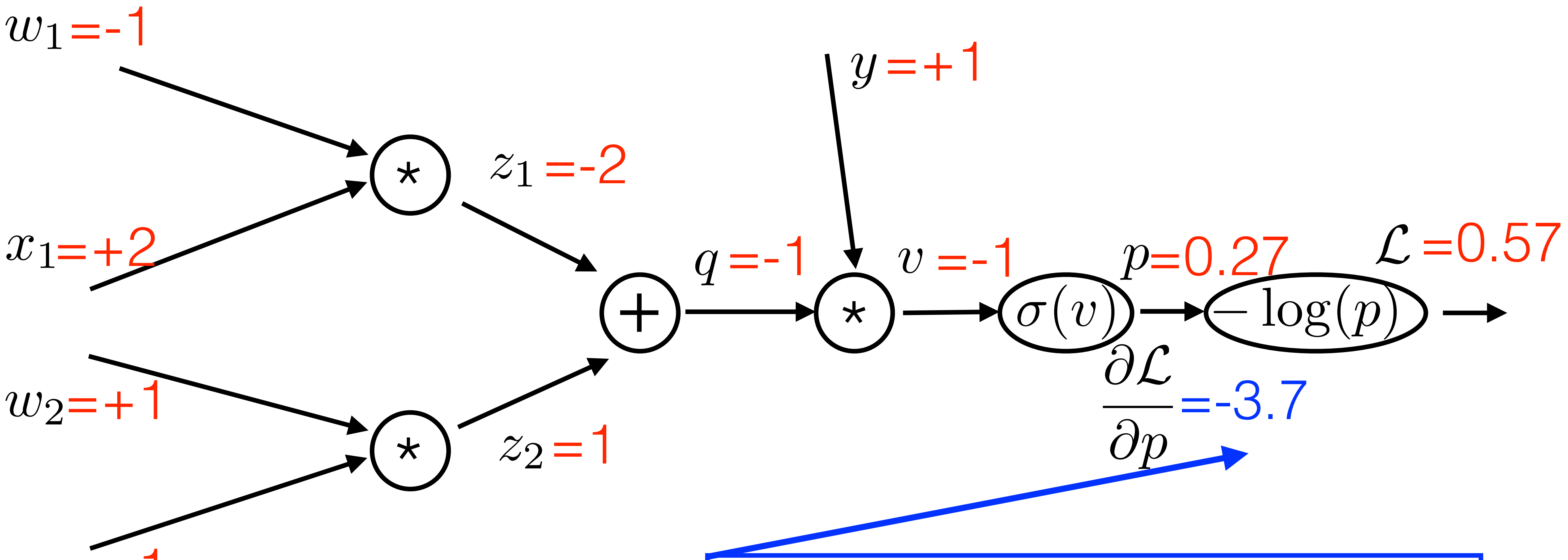
Example II: Given training sample, how do you learn weights?



Example II: Given training sample, how do you learn weights?



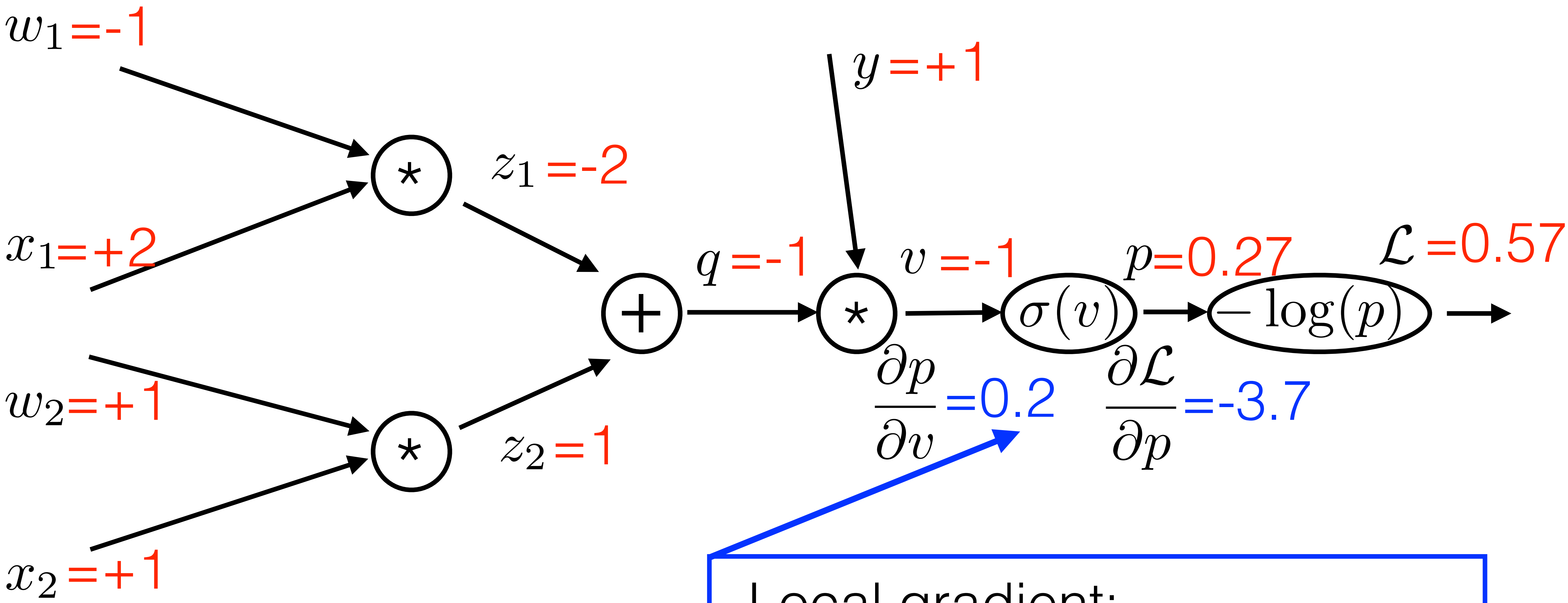
Example II: Given training sample, how do you learn weights?



Local gradient:

$$\frac{\partial \mathcal{L}}{\partial p} = \frac{\partial(-\log(p))}{\partial p} = -\frac{1}{p}$$

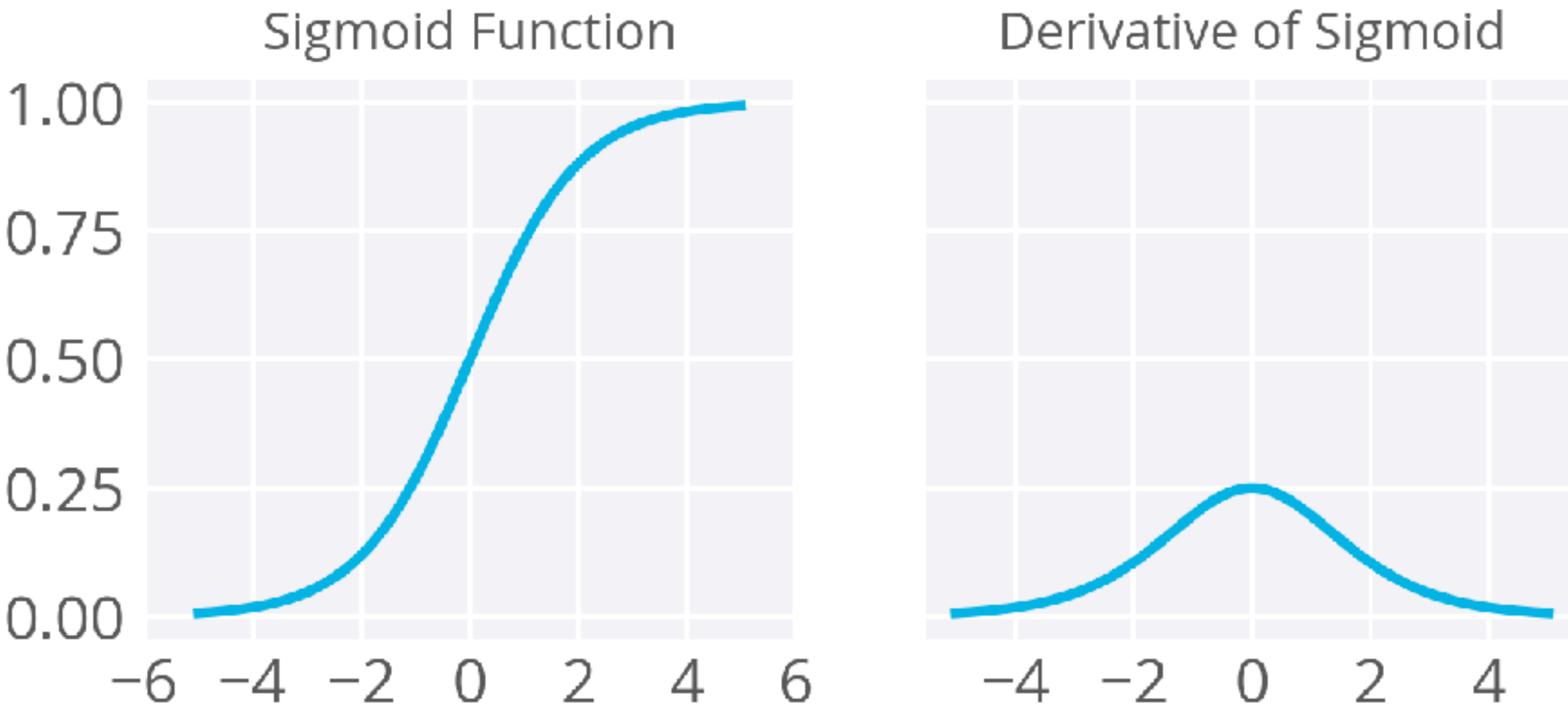
Example II: Given training sample, how do you learn weights?



Local gradient:

$$\frac{\partial p}{\partial v} = \frac{\partial \sigma(v)}{\partial v} = \sigma(v)(1 - \sigma(v))$$

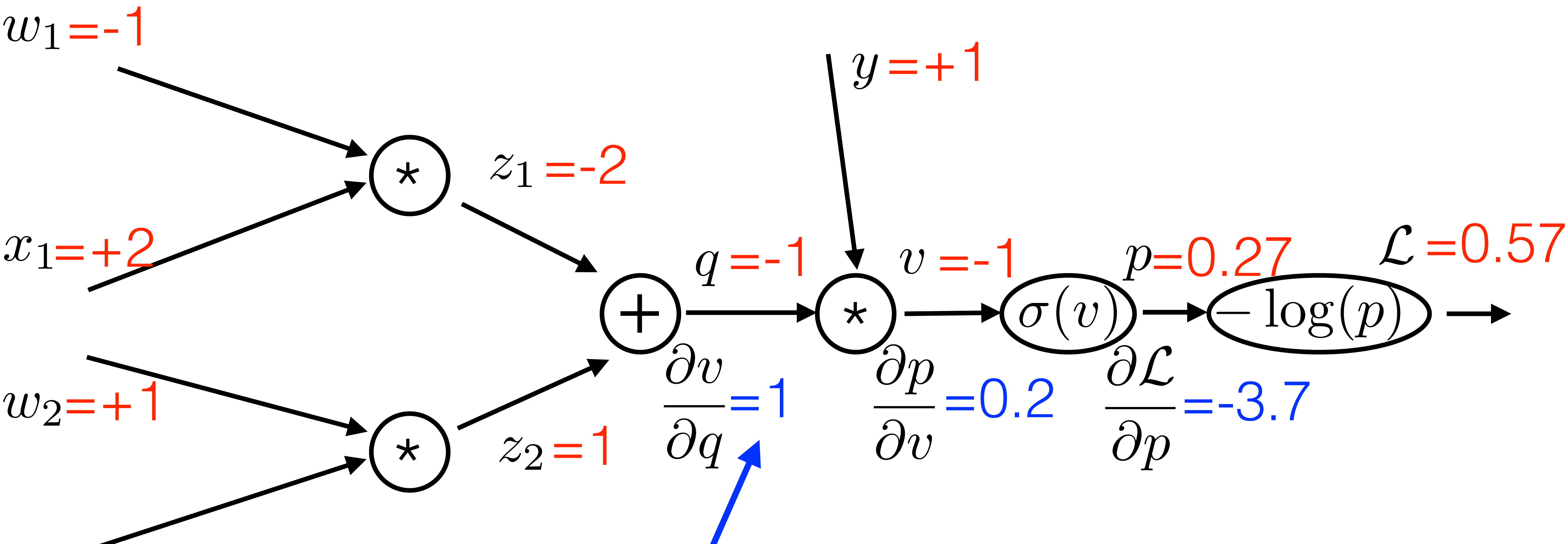
Example II: Given training sample, how do you learn weights?



Local gradient:

$$\frac{\partial p}{\partial v} = \frac{\partial \sigma(v)}{\partial v} = \sigma(v)(1 - \sigma(v))$$

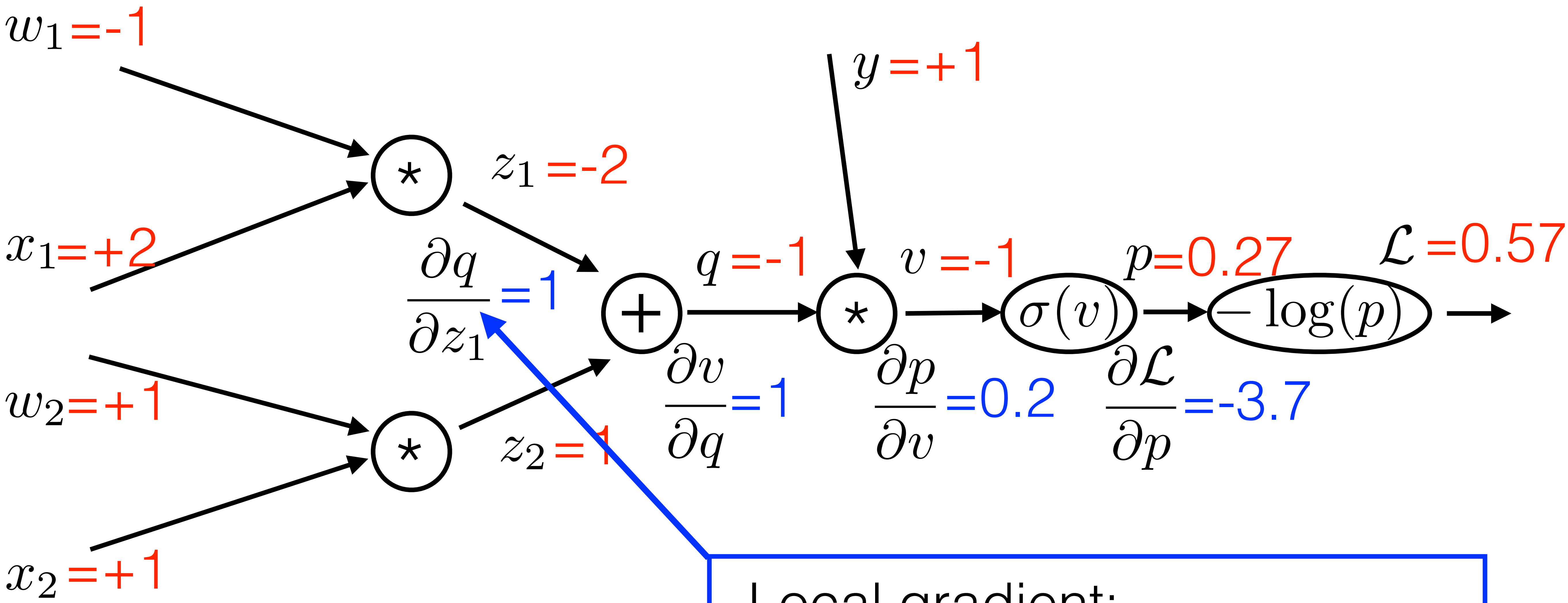
Example II: Given training sample, how do you learn weights?



Local gradient:

$$\frac{\partial v}{\partial q} = \frac{\partial (yq)}{\partial q} = y$$

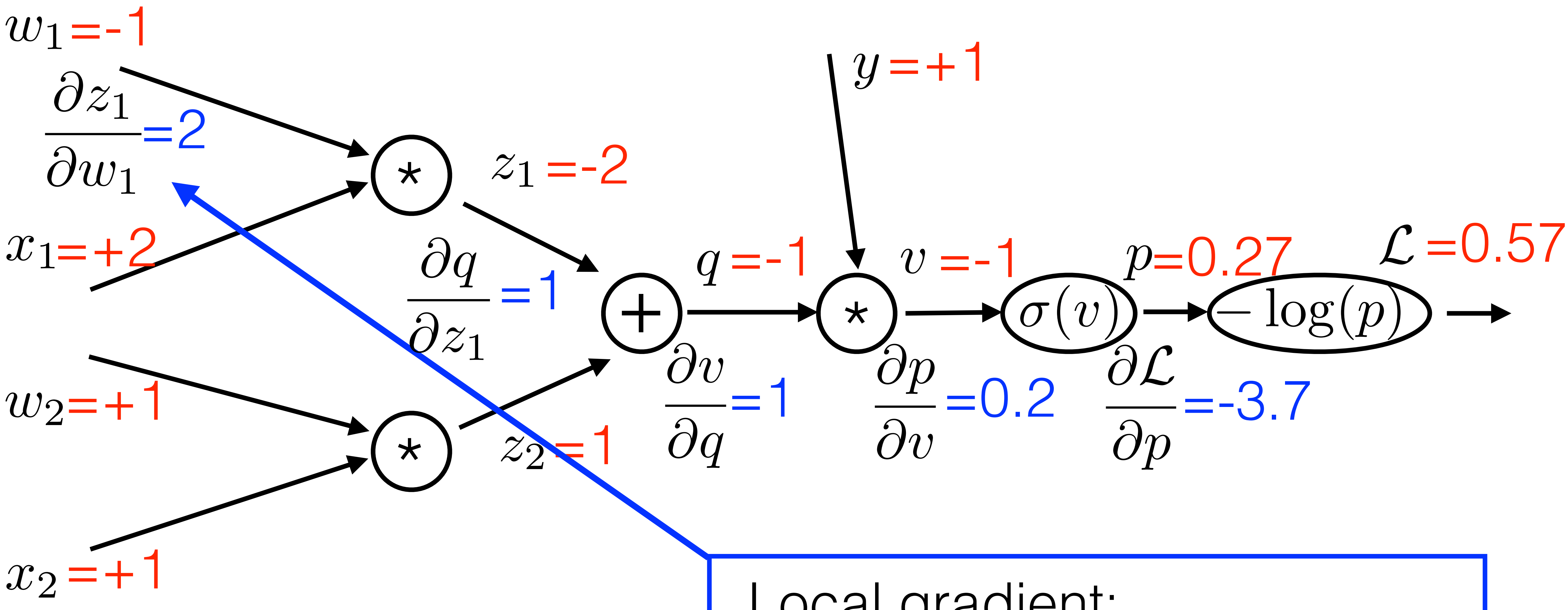
Example II: Given training sample, how do you learn weights?



Local gradient:

$$\frac{\partial q}{\partial z_1} = \frac{\partial(z_1 + z_2)}{\partial z_1} = 1$$

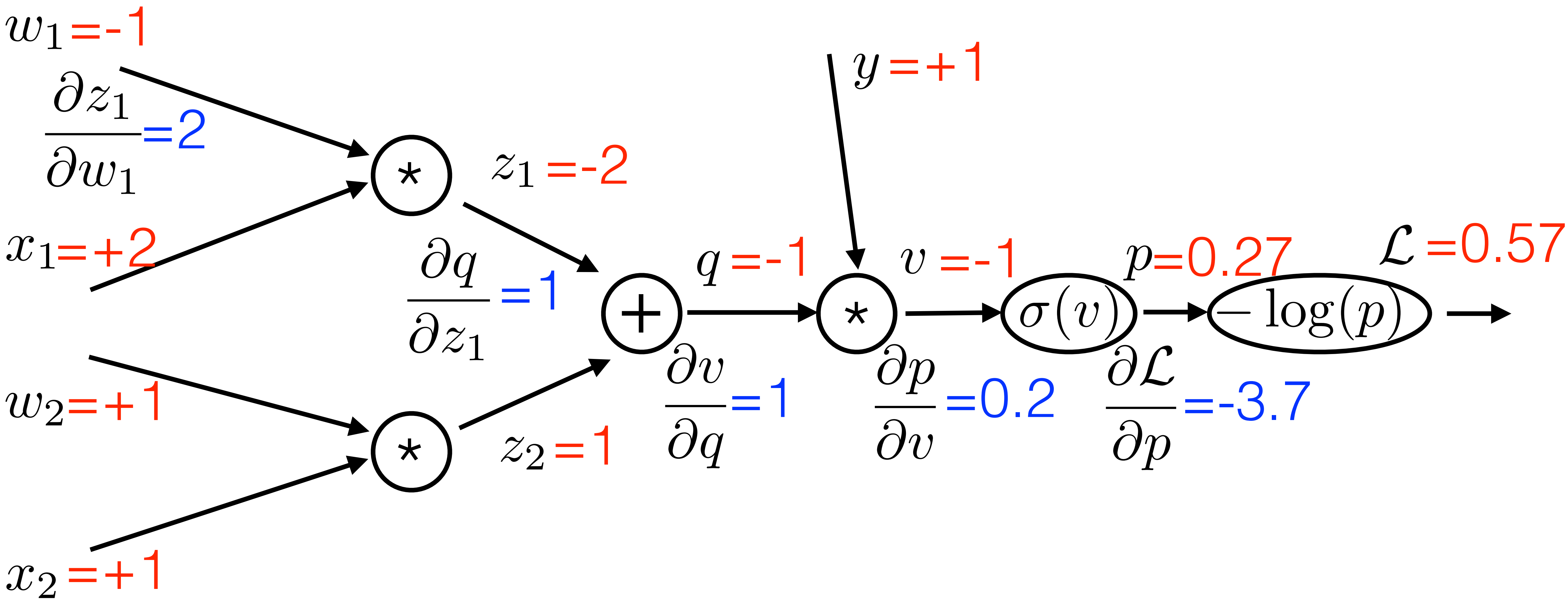
Example II: Given training sample, how do you learn weights?



Local gradient:

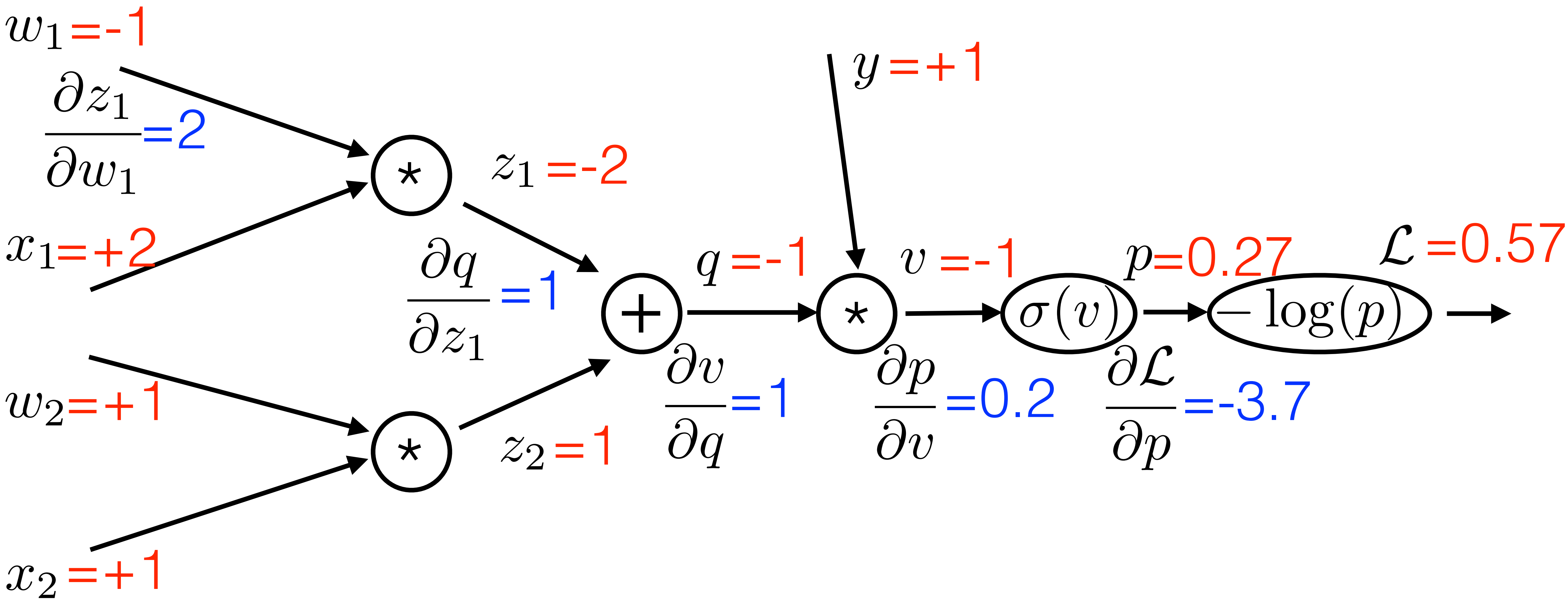
$$\frac{\partial z_1}{\partial w_1} = \frac{\partial(w_1 x_1)}{\partial w_1} = x_1$$

Example II: Given training sample, how do you learn weights?



$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial p}\frac{\partial p}{\partial v}\frac{\partial v}{\partial q}\frac{\partial q}{\partial z_1}\frac{\partial z_1}{\partial w_1} = -3.7*0.2*1*1*2 = -1.48$$

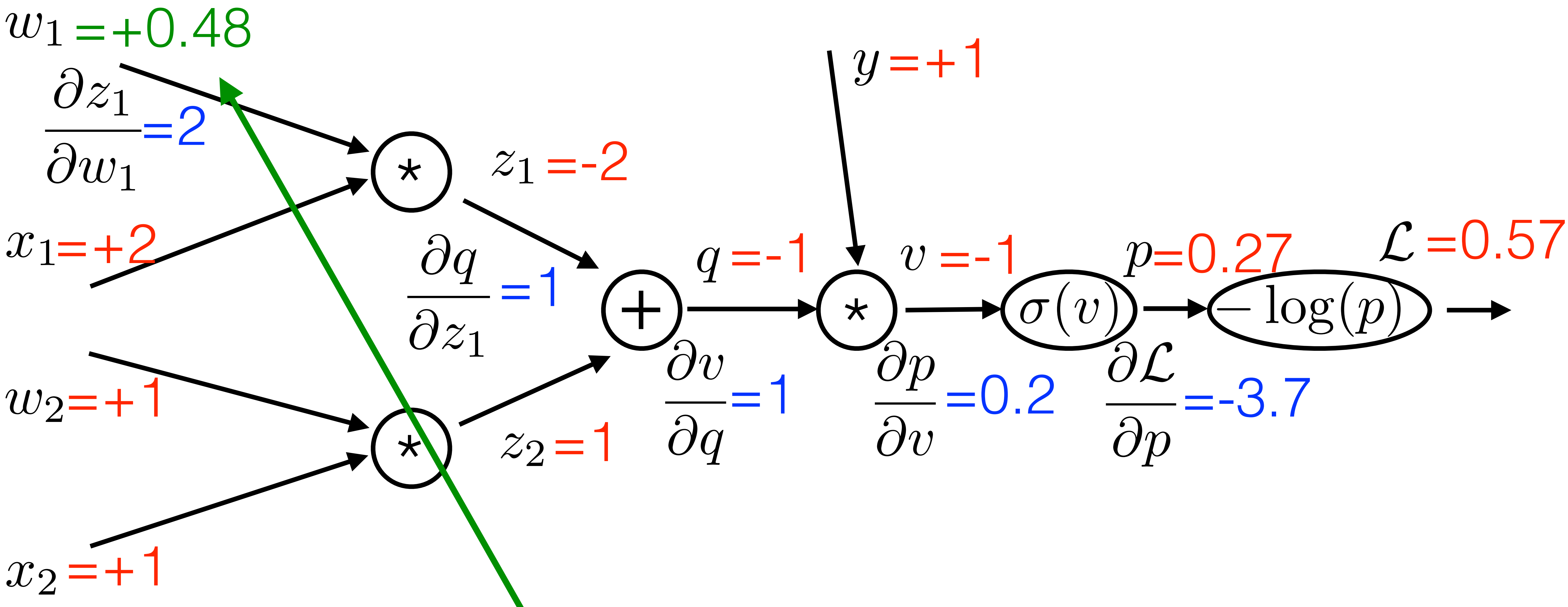
Example II: Given training sample, how do you learn weights?



$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial p}\frac{\partial p}{\partial v}\frac{\partial v}{\partial q}\frac{\partial q}{\partial z_1}\frac{\partial z_1}{\partial w_1} = -3.7*0.2*1*1*2 = -1.48$$

$$w_1 = w_1 - \alpha \frac{\partial \mathcal{L}}{\partial w1} = +0.48$$

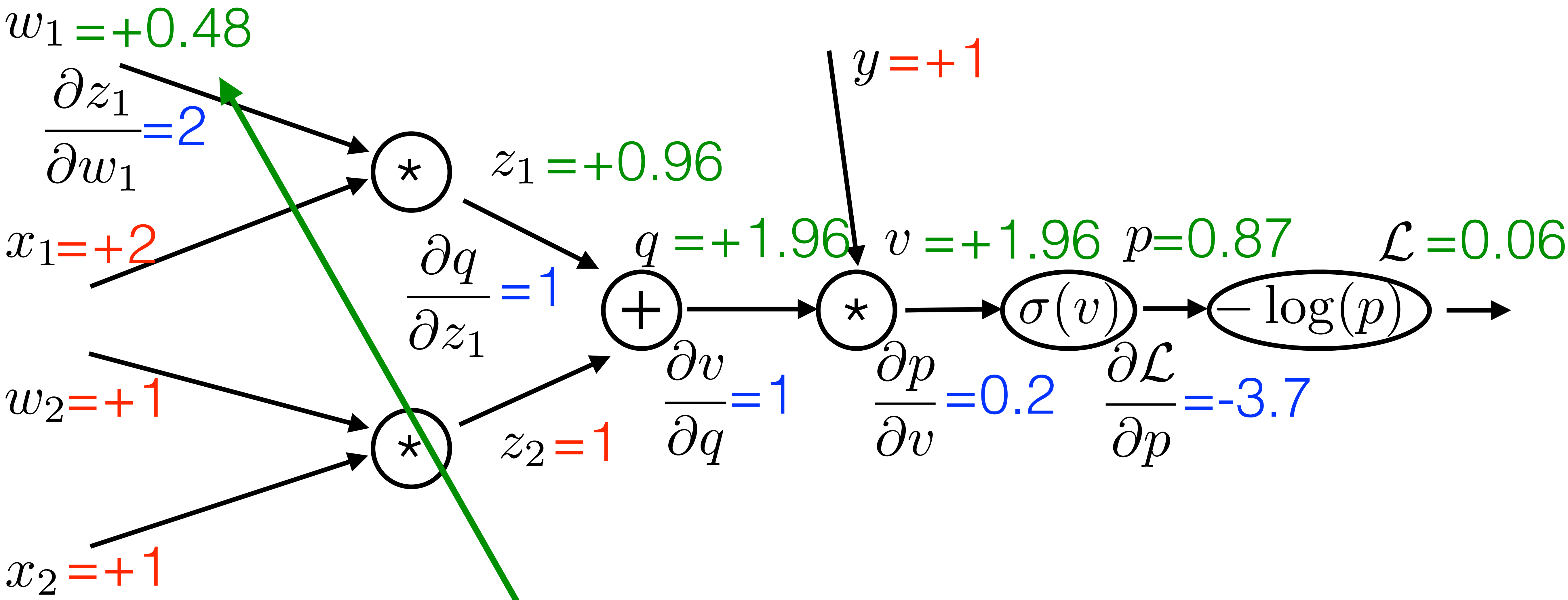
Example II: Given training sample, how do you learn weights?



$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial p}\frac{\partial p}{\partial v}\frac{\partial v}{\partial q}\frac{\partial q}{\partial z_1}\frac{\partial z_1}{\partial w_1} = -3.7*0.2*1*1*2 = -1.48$$

$$w_1 = w_1 - \alpha\frac{\partial \mathcal{L}}{\partial w1} = +0.48$$

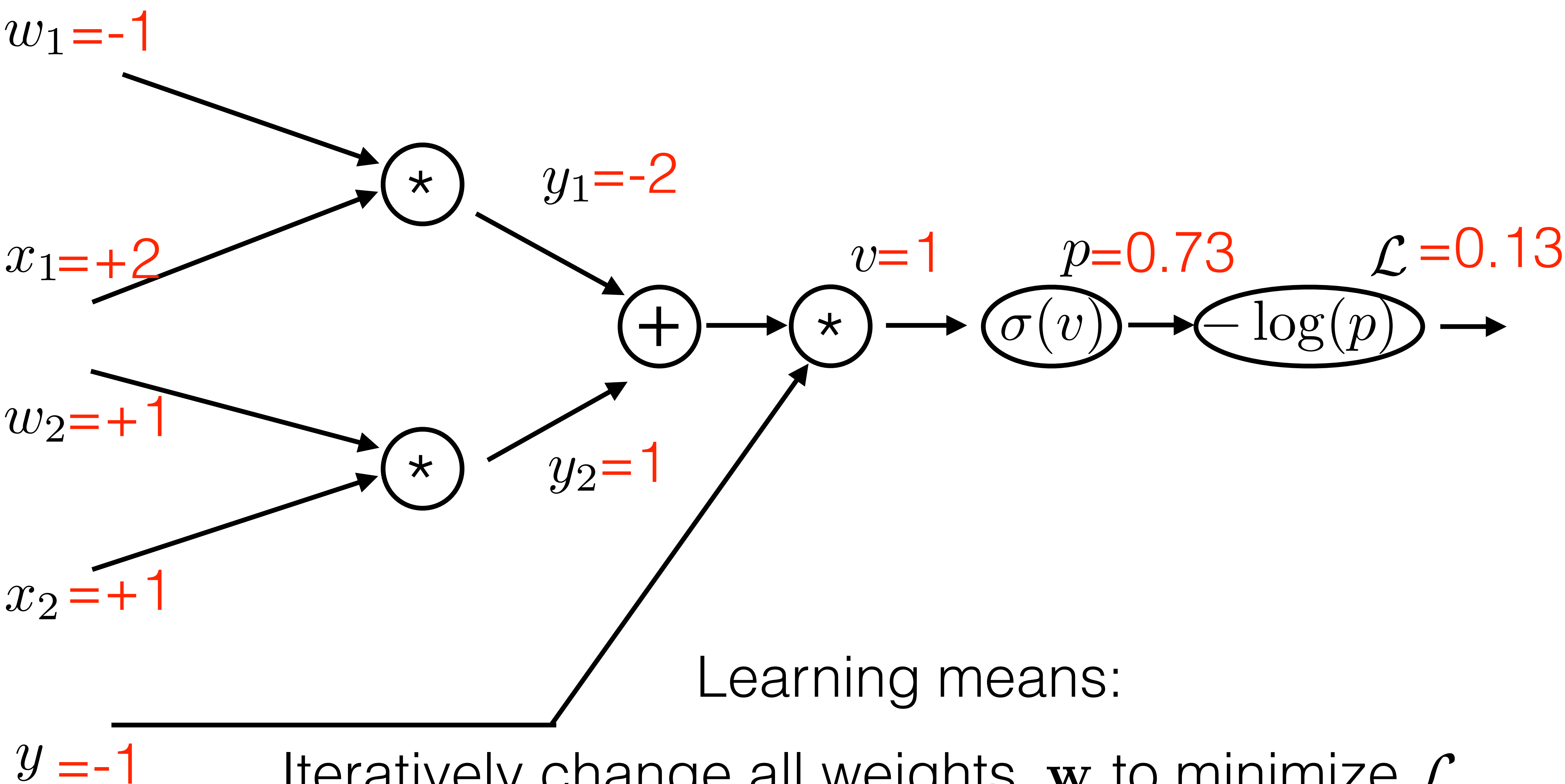
Example II: Given training sample, how do you learn weights?



$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial p}\frac{\partial p}{\partial v}\frac{\partial v}{\partial q}\frac{\partial q}{\partial z_1}\frac{\partial z_1}{\partial w_1} = -3.7*0.2*1*1*2 = -1.48$$

$$w_1 = w_1 - \alpha\frac{\partial \mathcal{L}}{\partial w1} = +0.48$$

Example III: vector representation

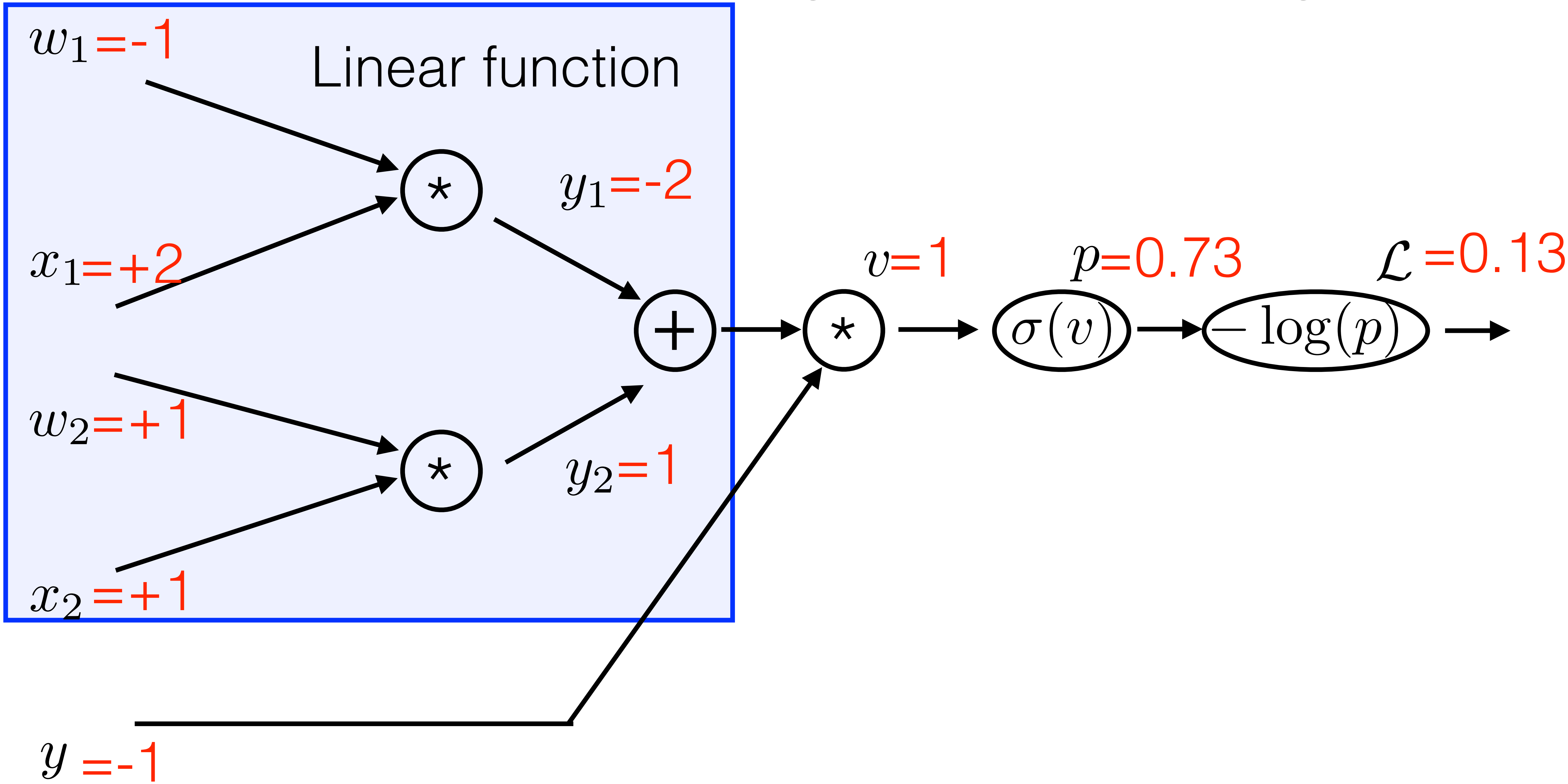


Learning means:

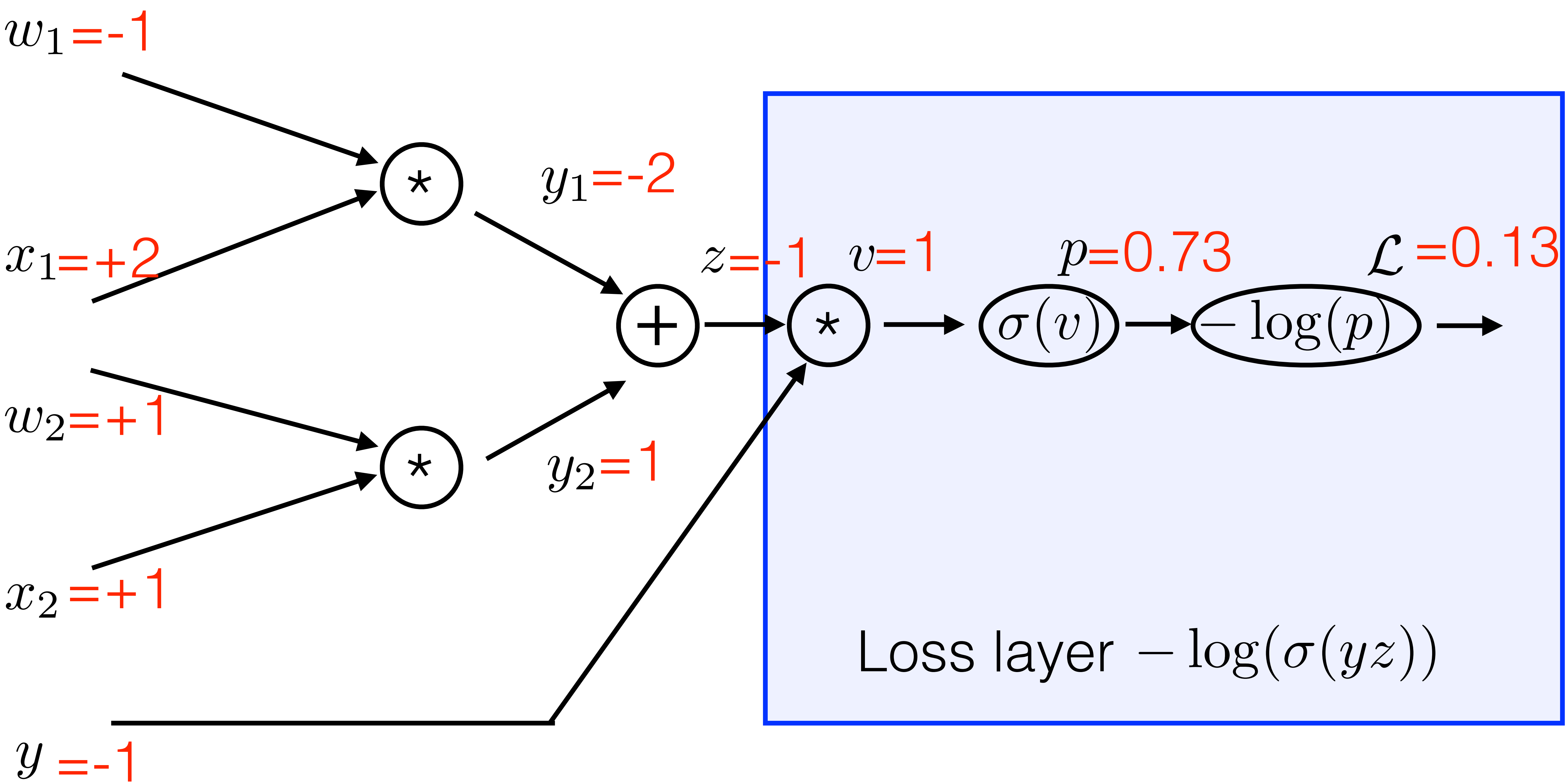
Iteratively change all weights $\mathbf{w}$ to minimize $\mathcal{L}$

$$\mathbf{w} = \mathbf{w} - \alpha \left[\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}}\right]^\top \quad \text{where} \quad \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \left[\frac{\partial \mathcal{L}}{\partial w_1}, \frac{\partial \mathcal{L}}{\partial w_2}, \dots\right]$$

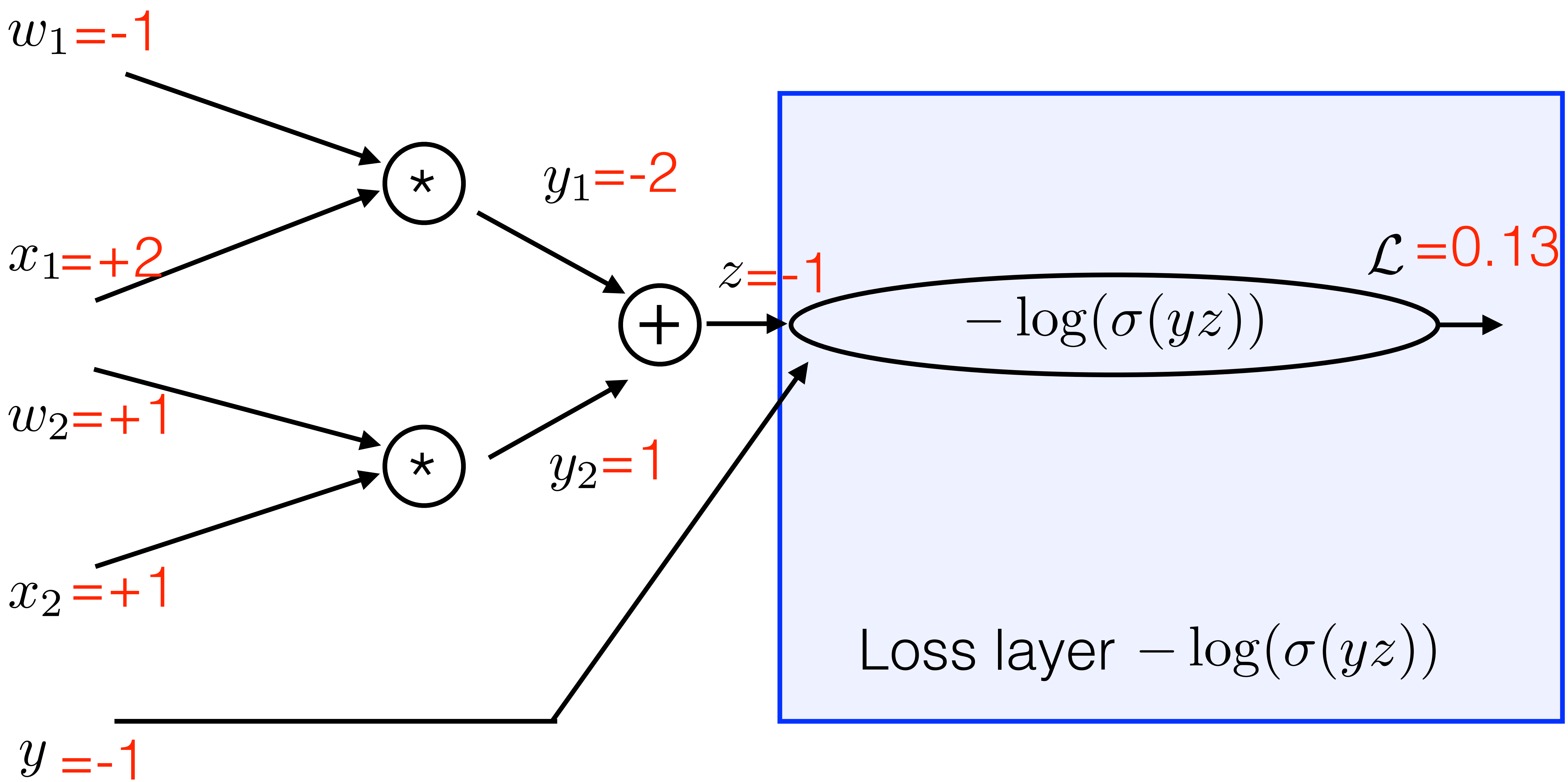
Computational graph of the learning



Computational graph of the learning



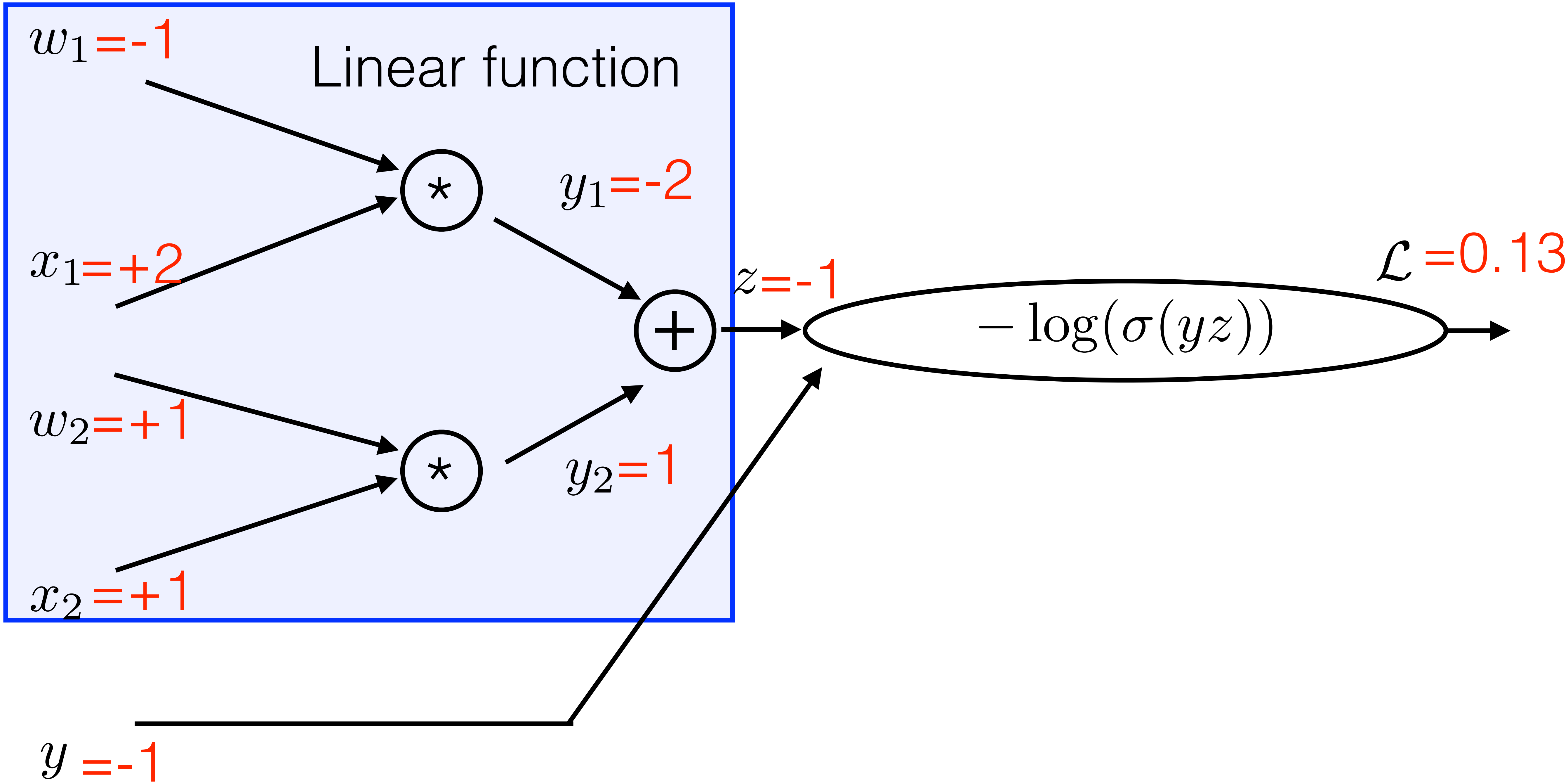
Backprop in vector representation



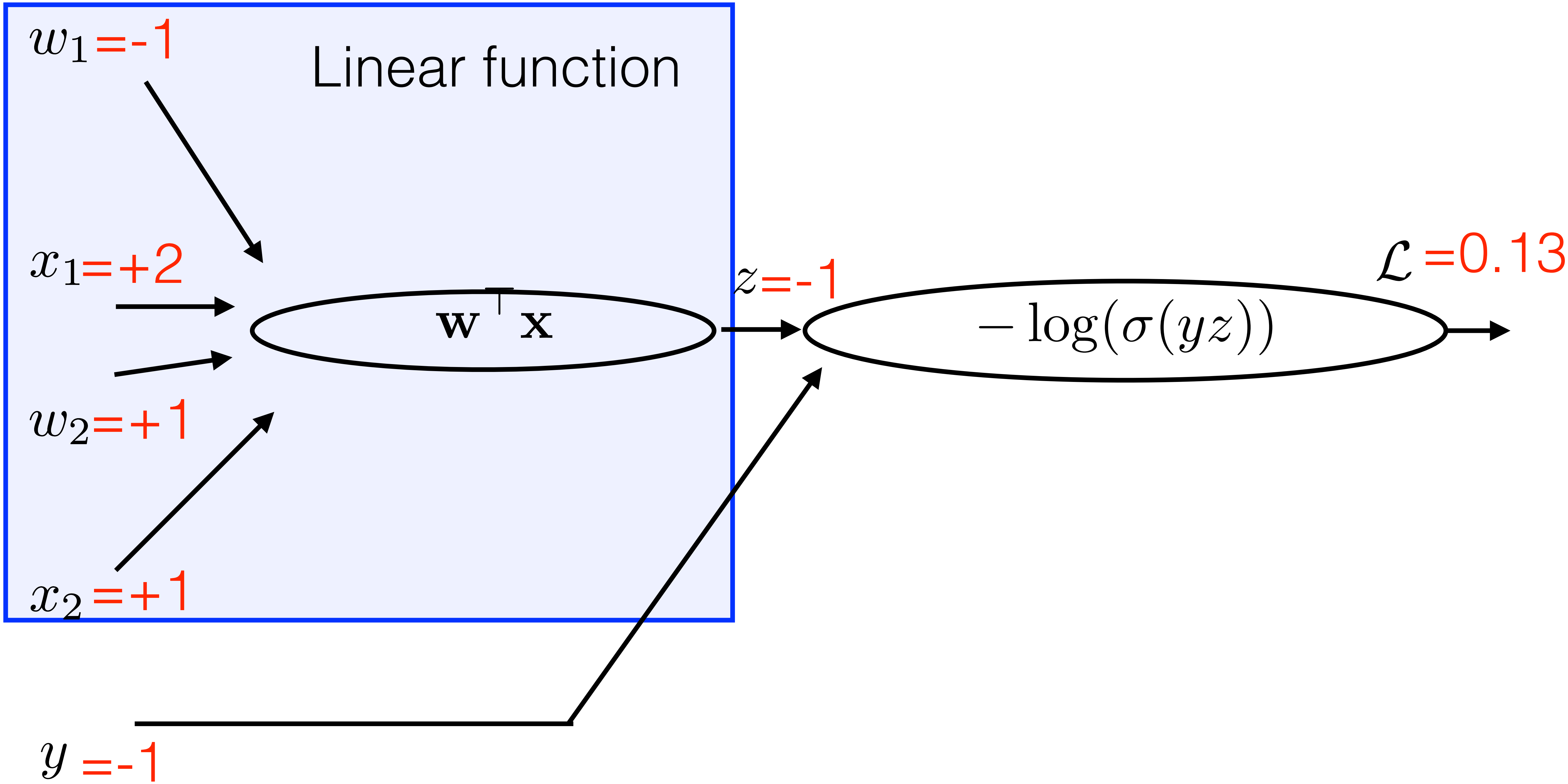
This is the logistic loss!

$$\mathcal{L}(y,z) = -\log(\sigma(yz)) = \log(1+\exp(-yz))$$

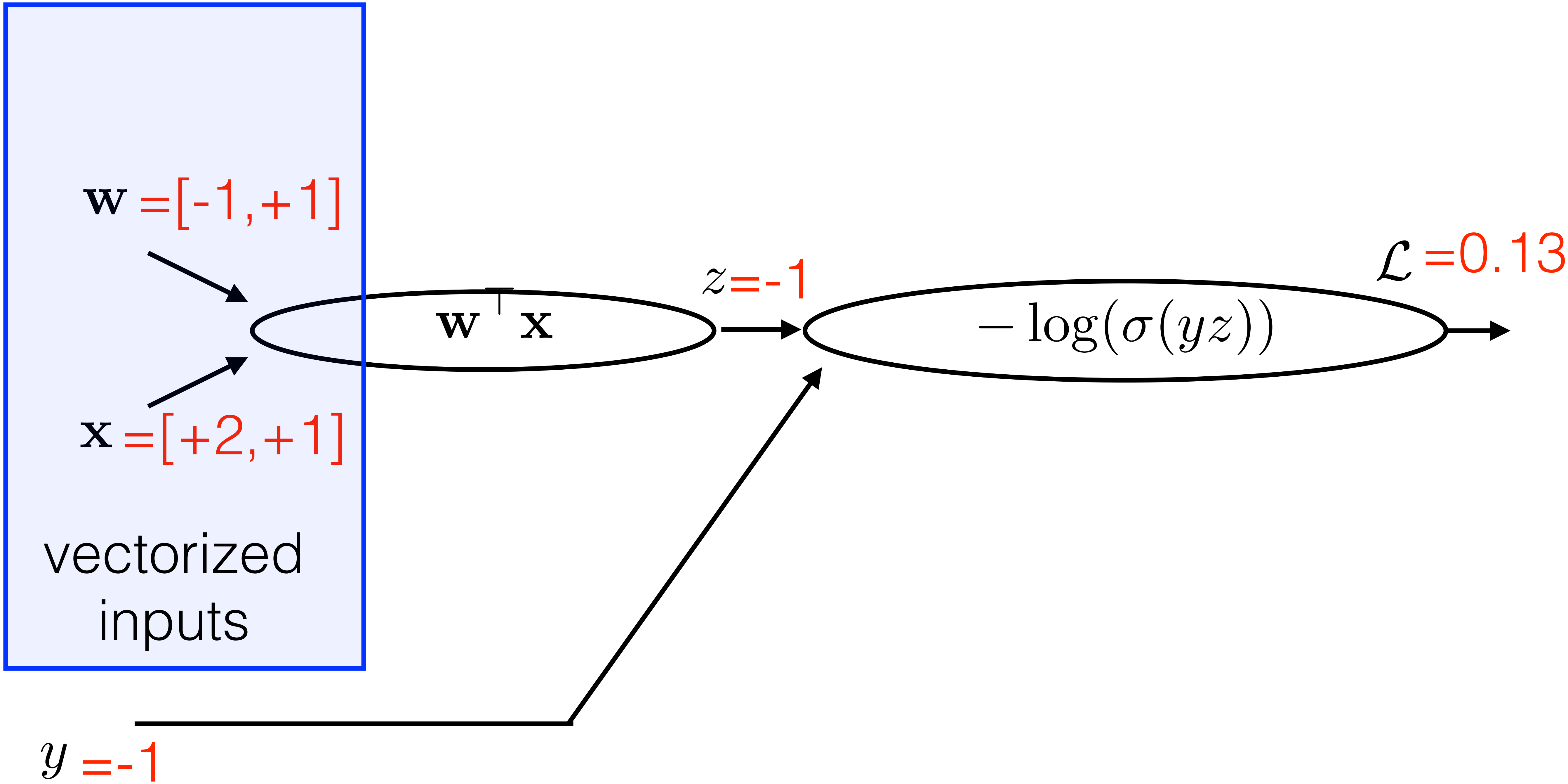
Backprop in vector representation



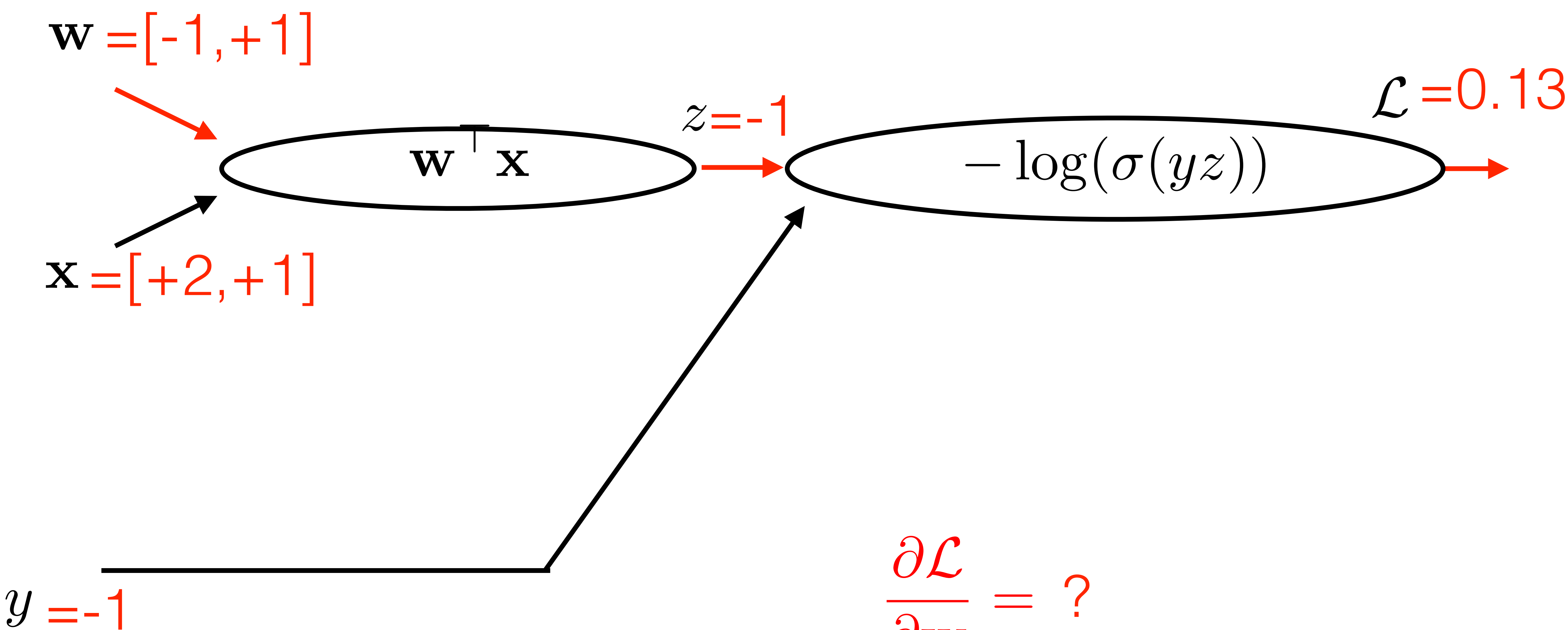
Backprop in vector representation



Backprop in vector representation

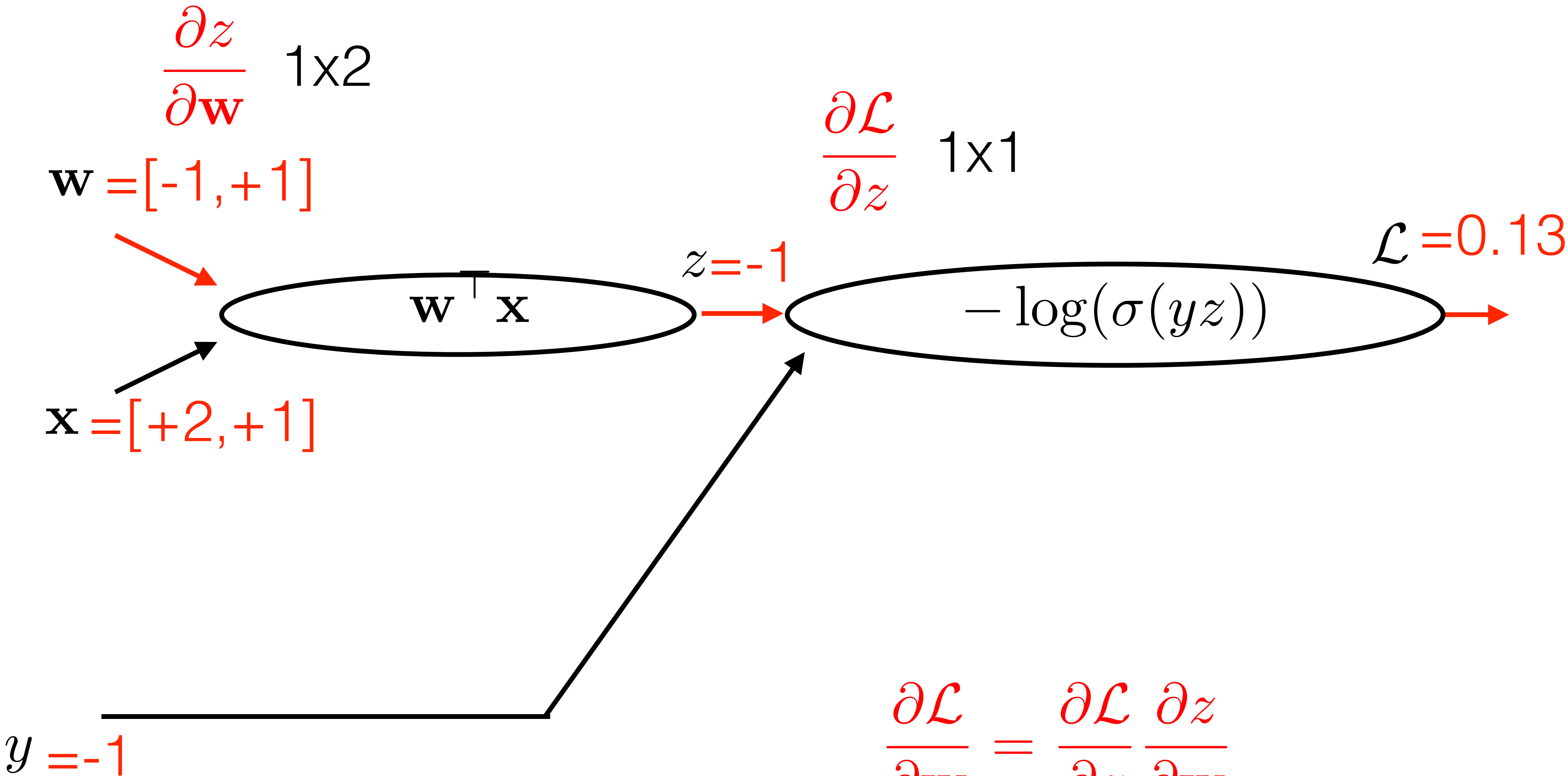


Backprop in vector representation



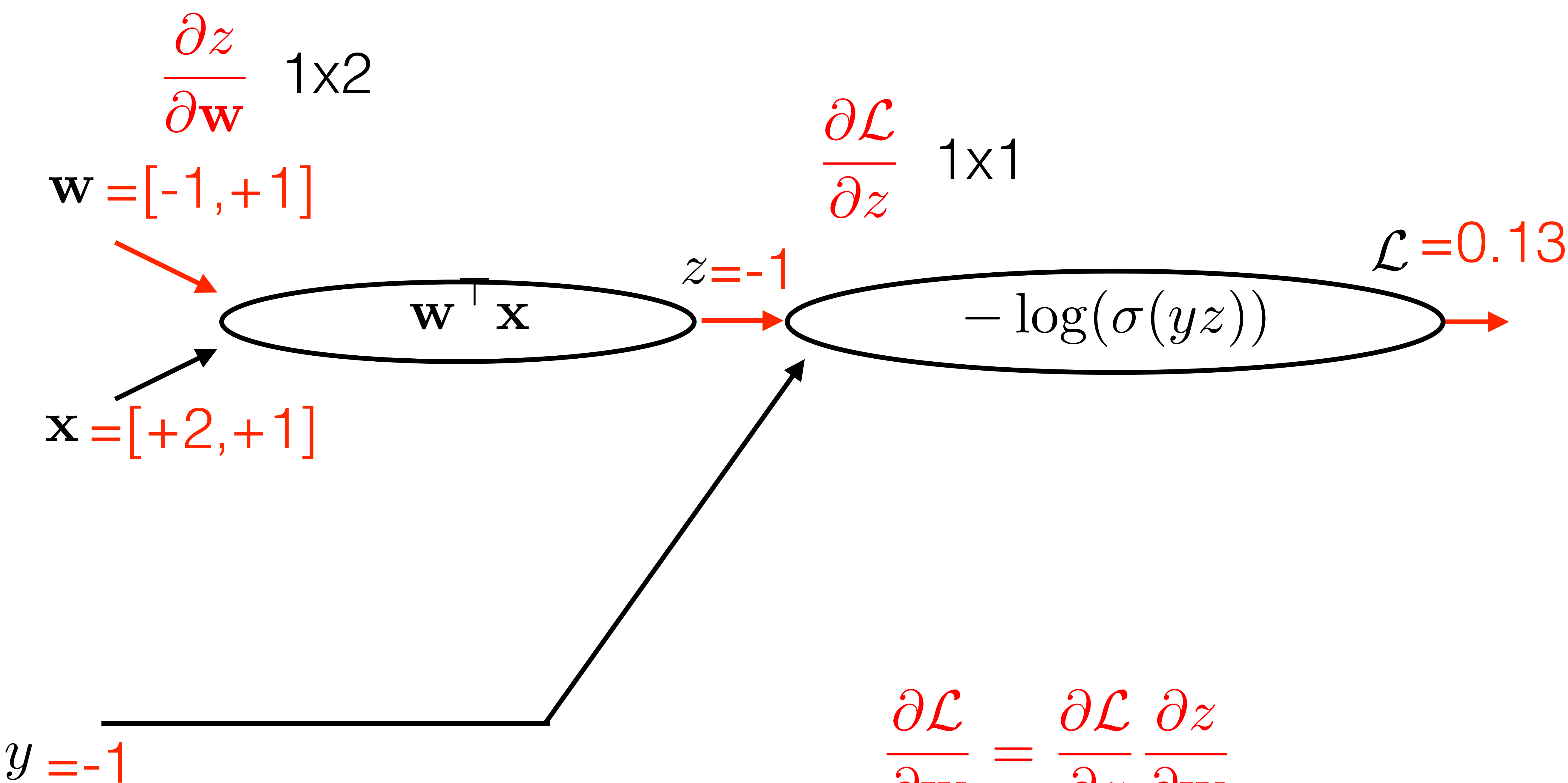
$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = ?$$

Backprop in vector representation



$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}}{\partial z} \frac{\partial z}{\partial \mathbf{w}}$$

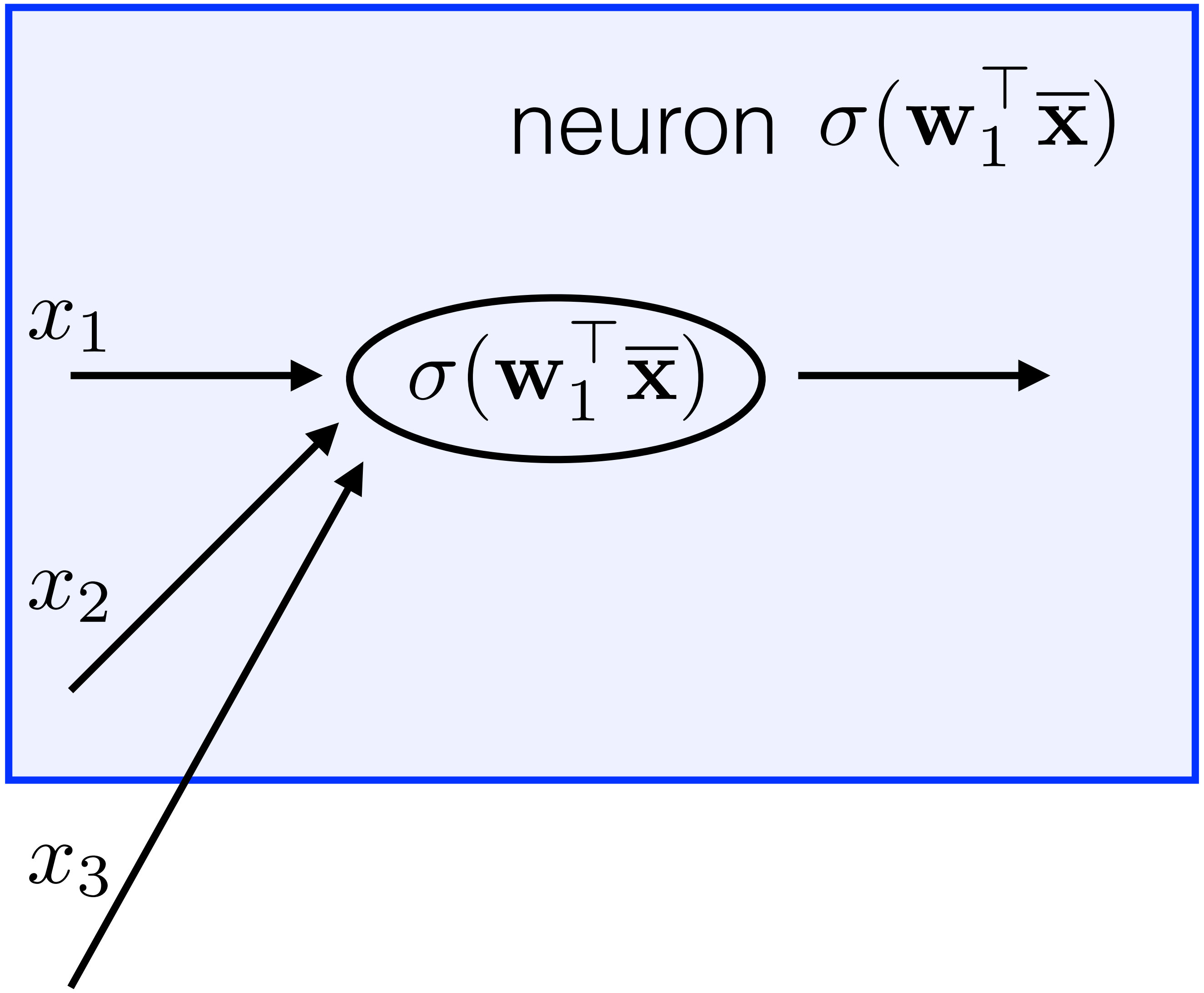
Backprop in vector representation



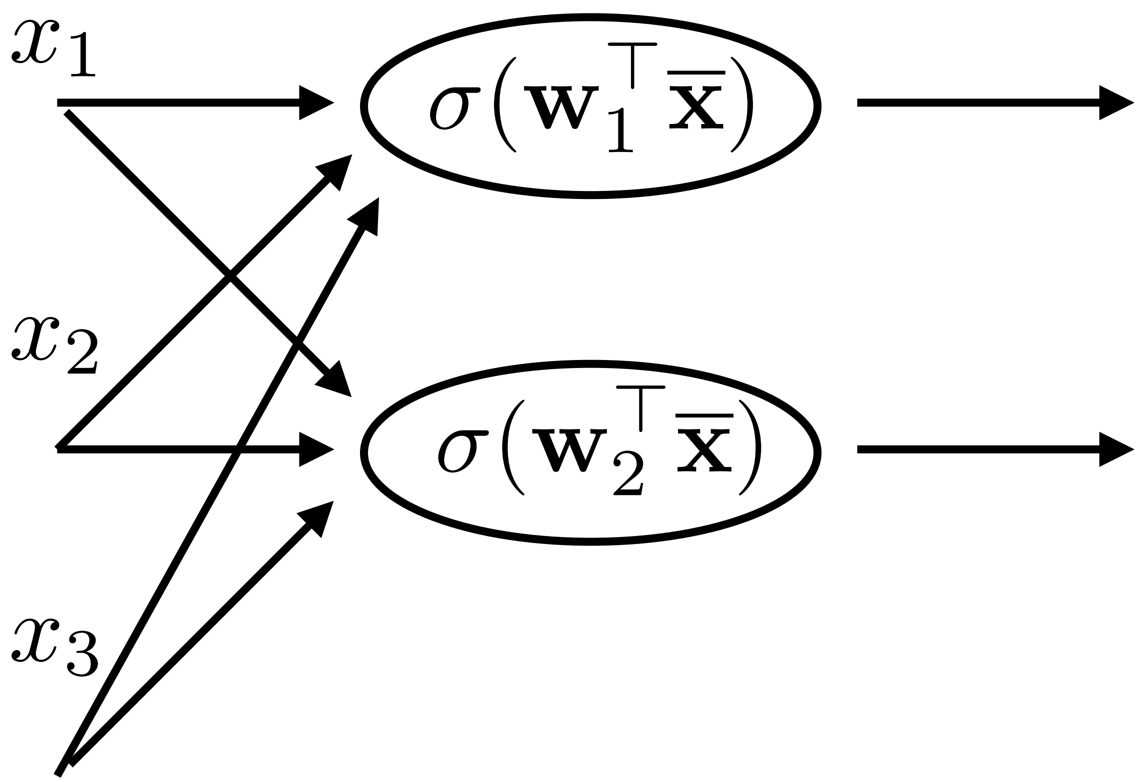
$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}}{\partial z} \frac{\partial z}{\partial \mathbf{w}}$$

Learning from multiple training samples means summing up the gradient over all samples

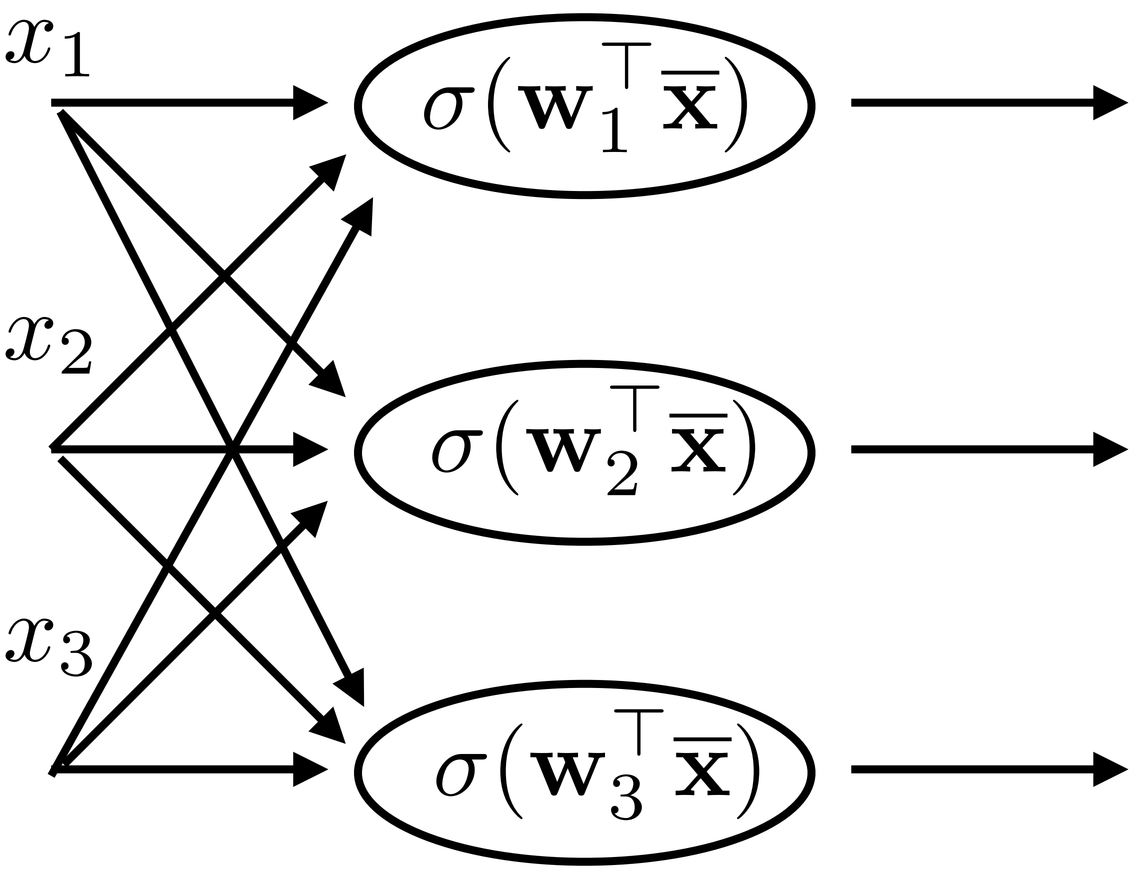
Fully connected neural network



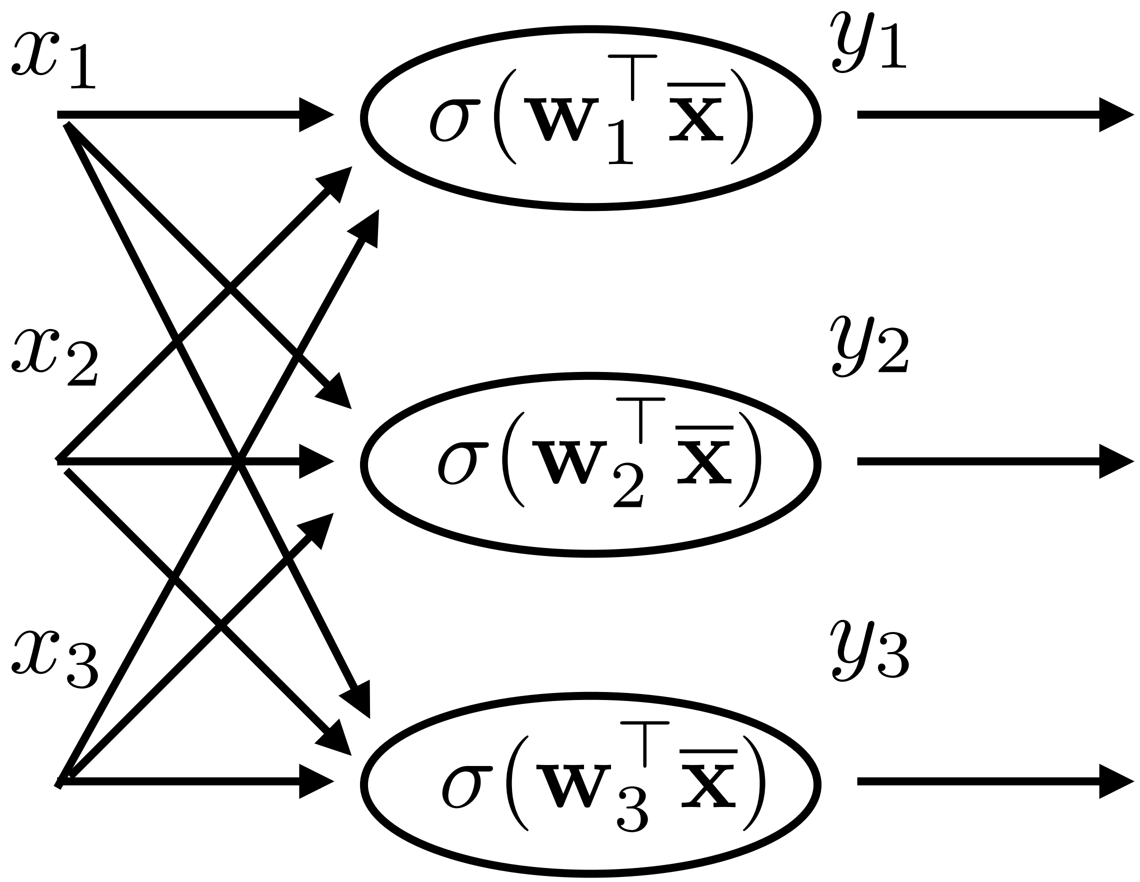
Fully connected neural network



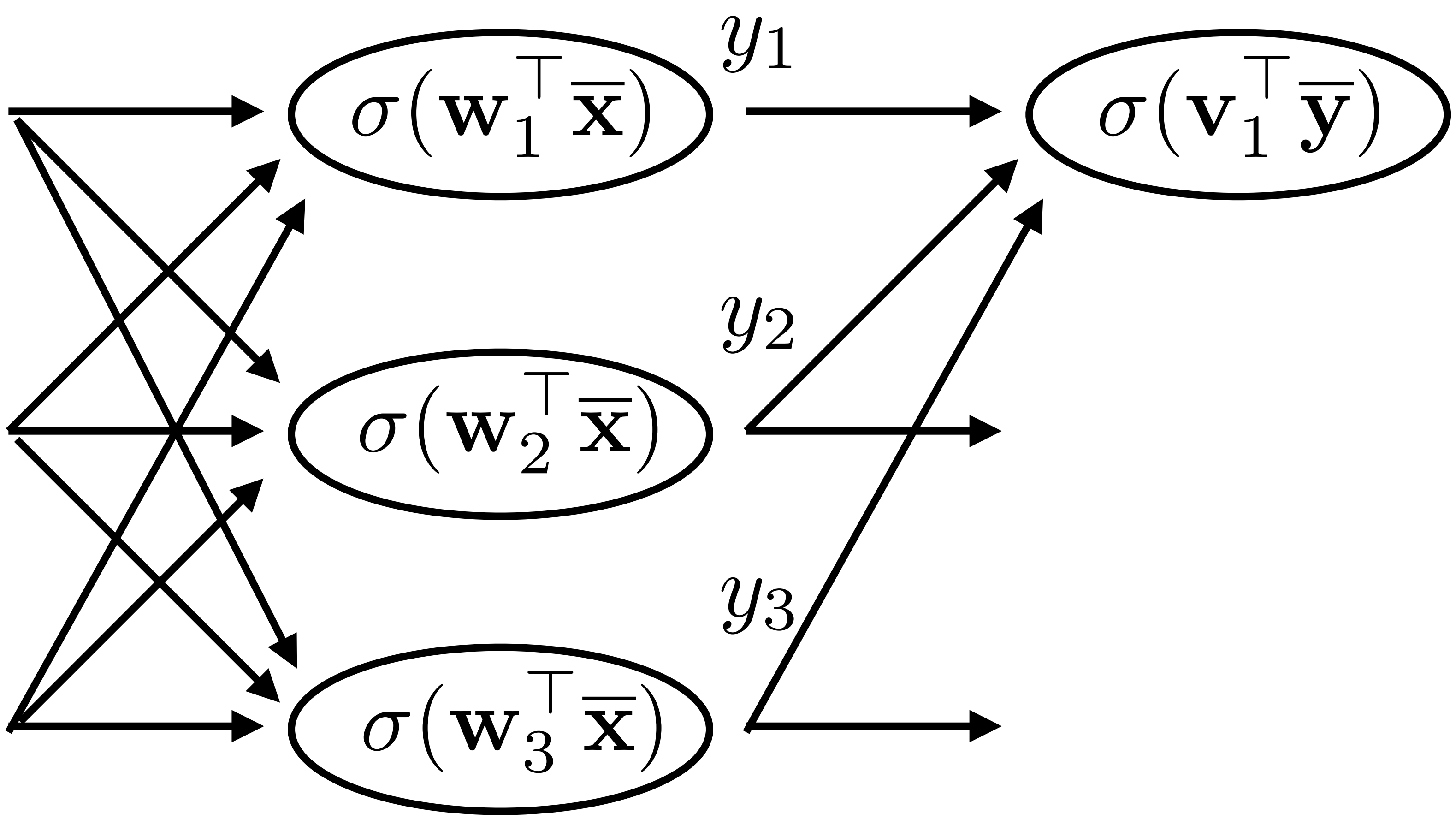
Fully connected neural network



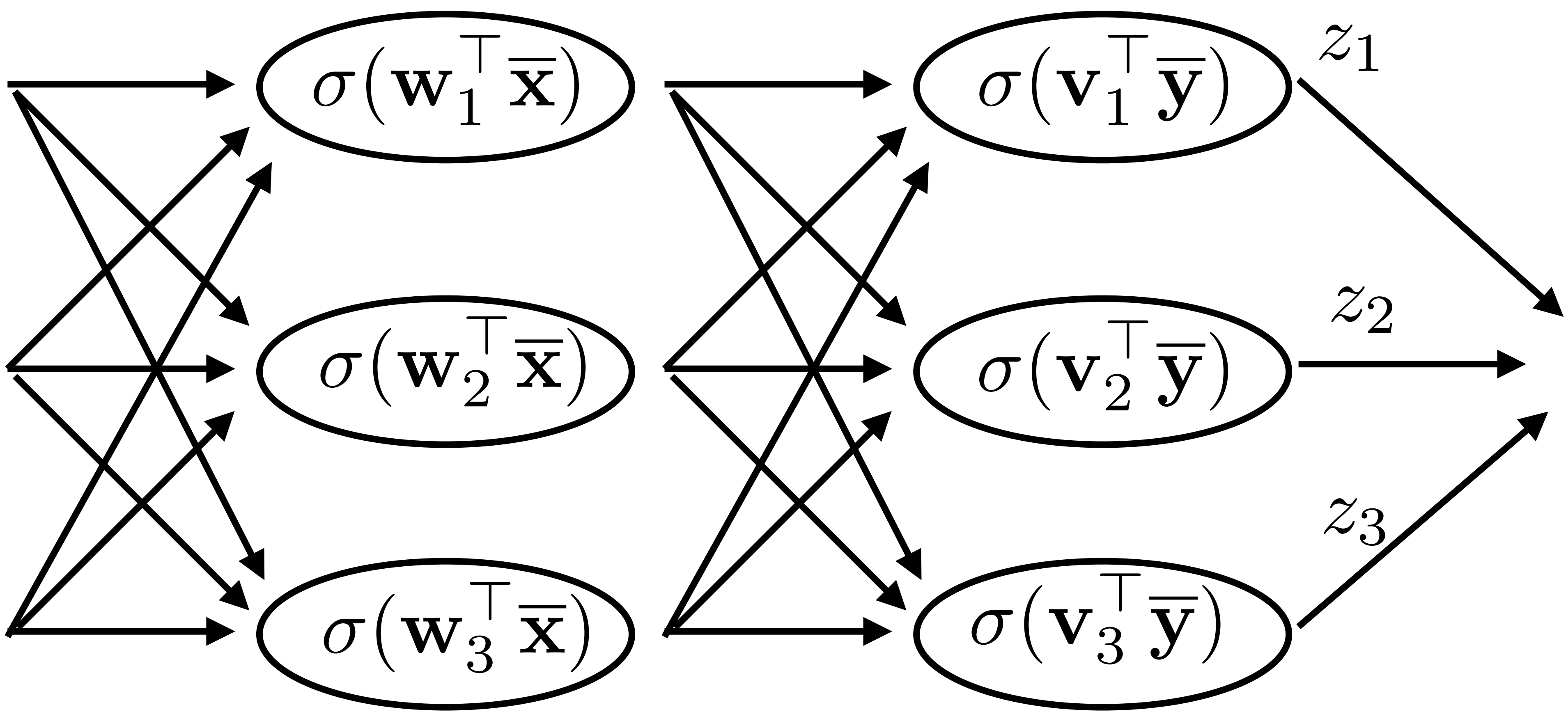
Fully connected neural network



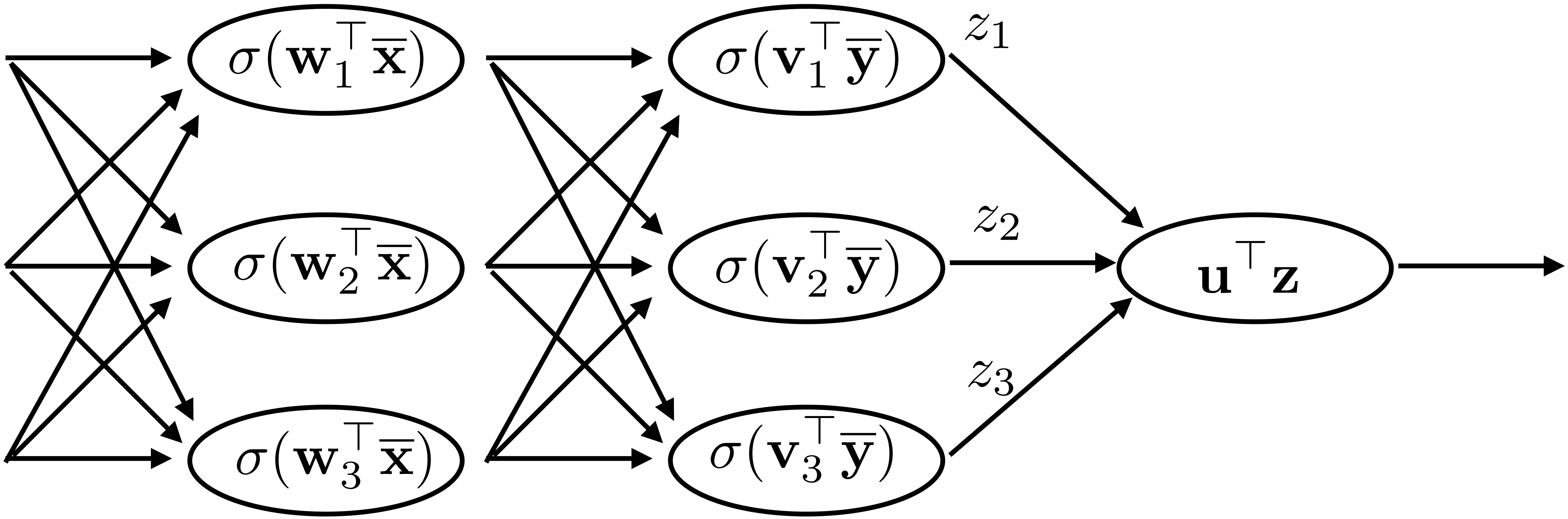
Fully connected neural network



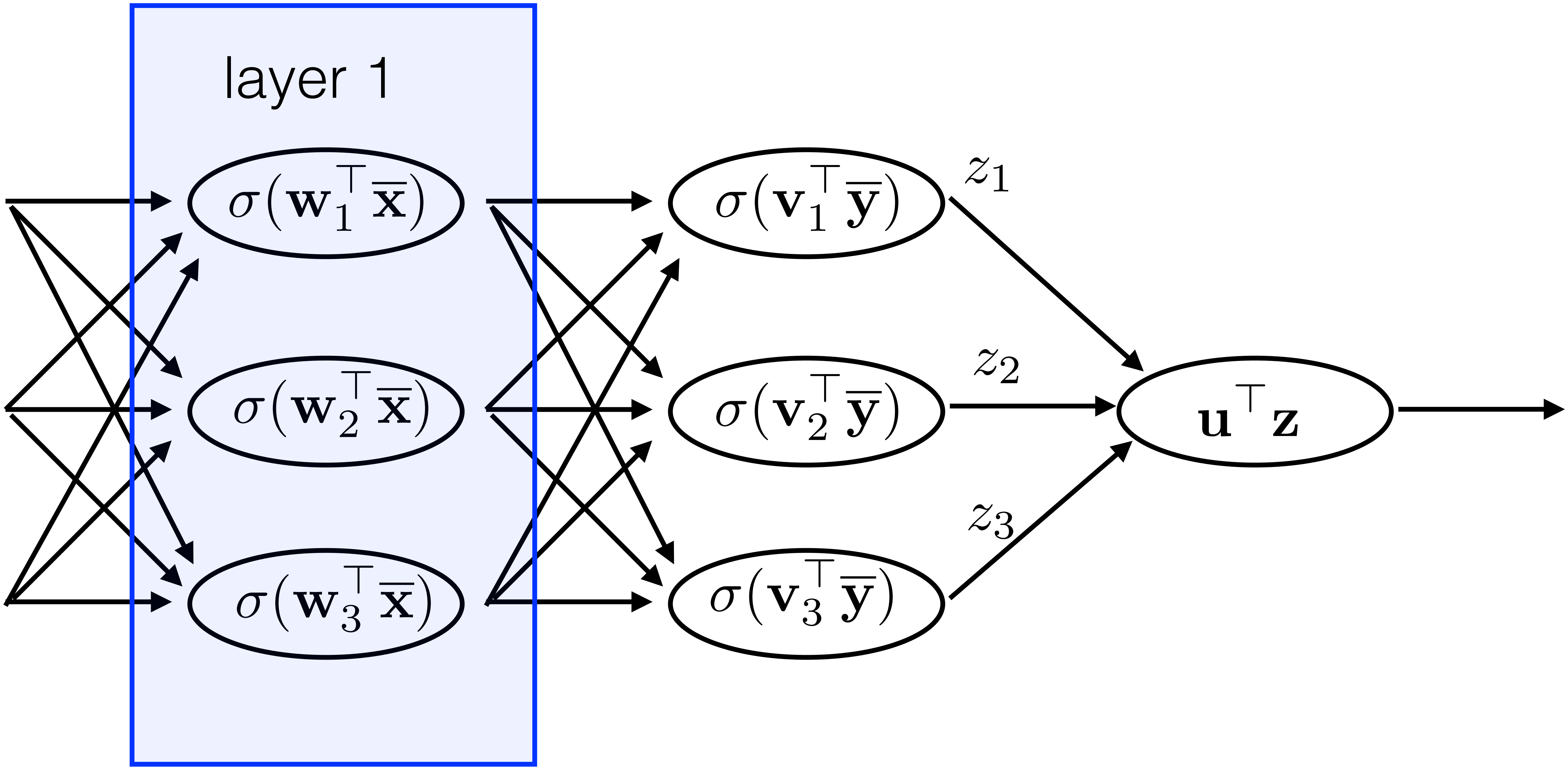
Fully connected neural network



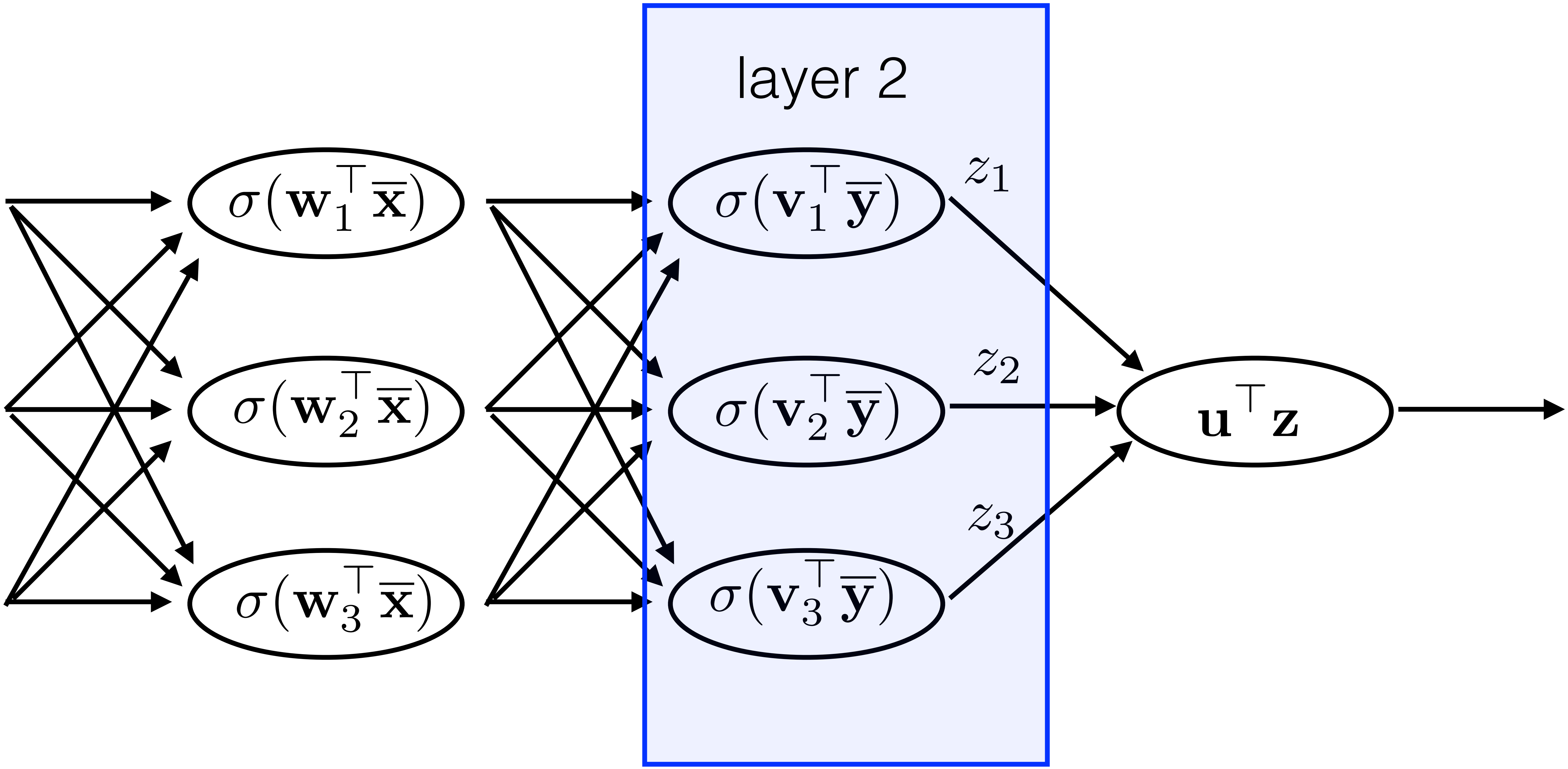
Fully connected neural network



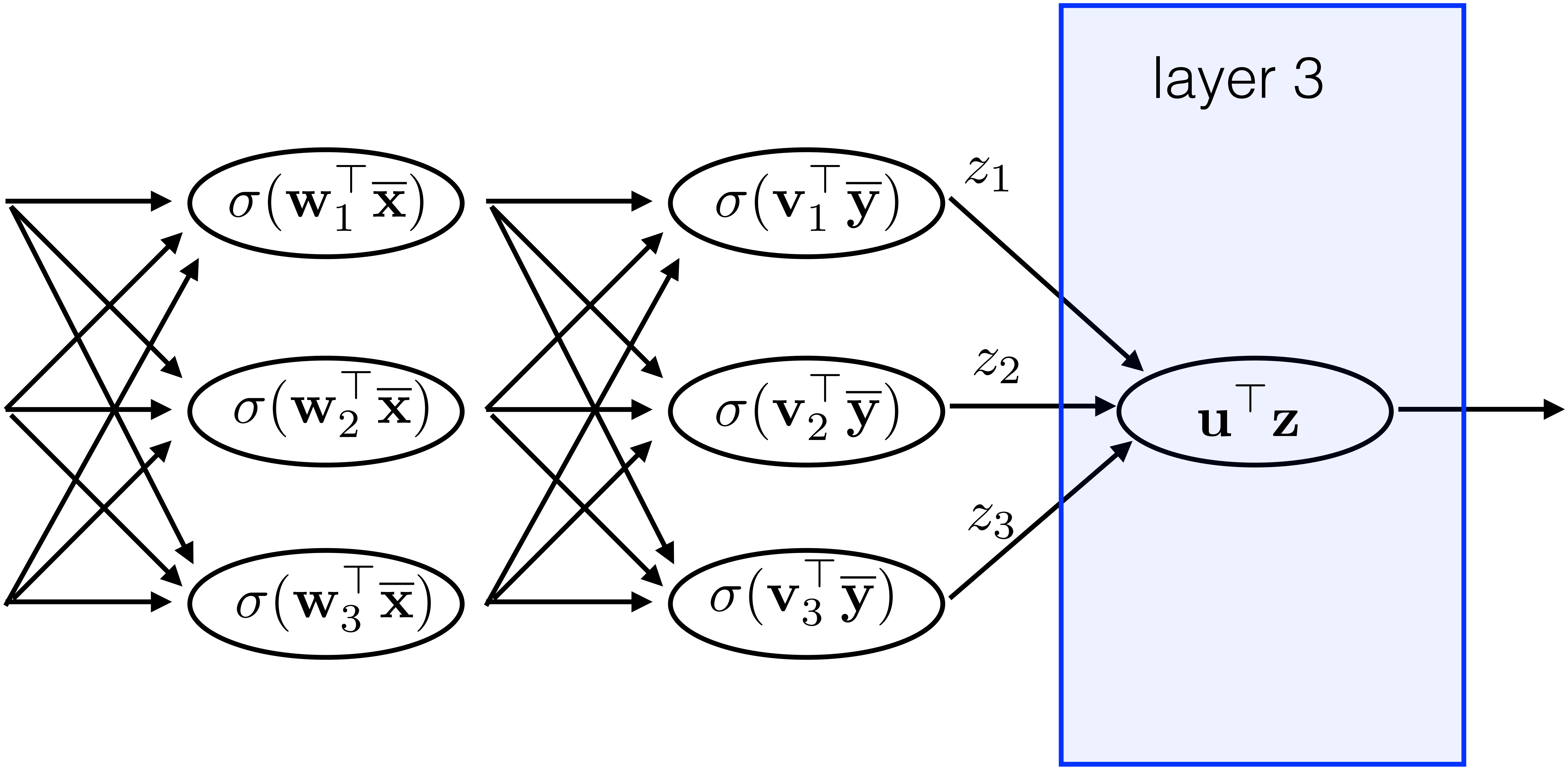
Fully connected neural network



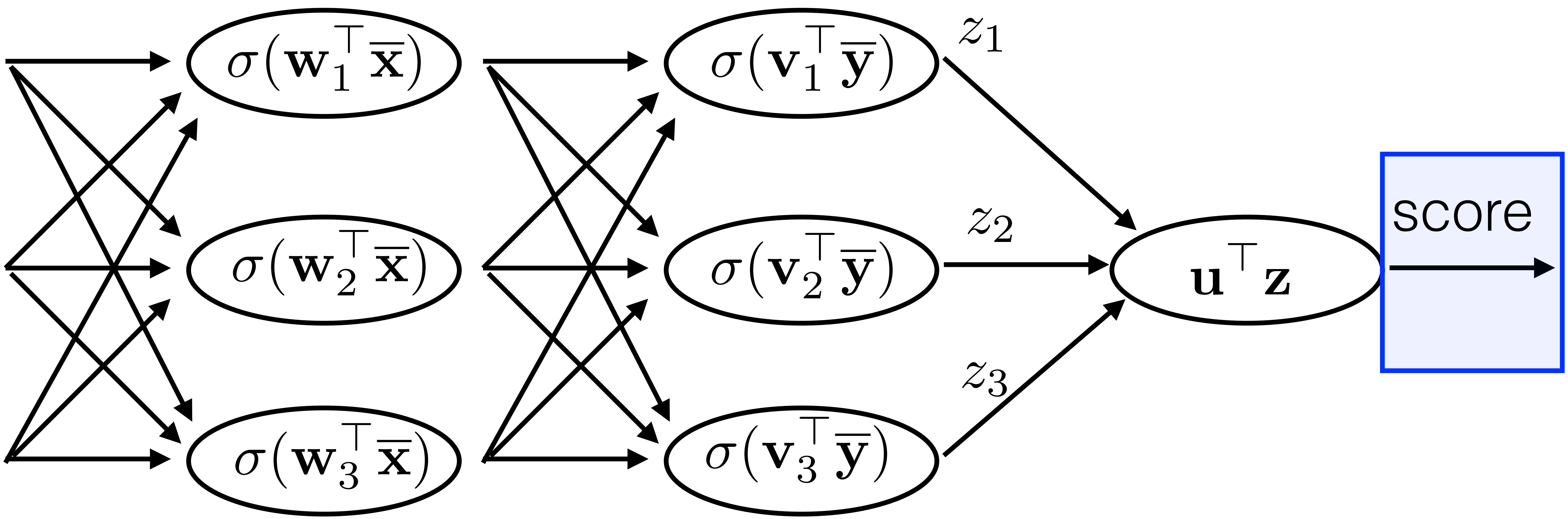
Fully connected neural network



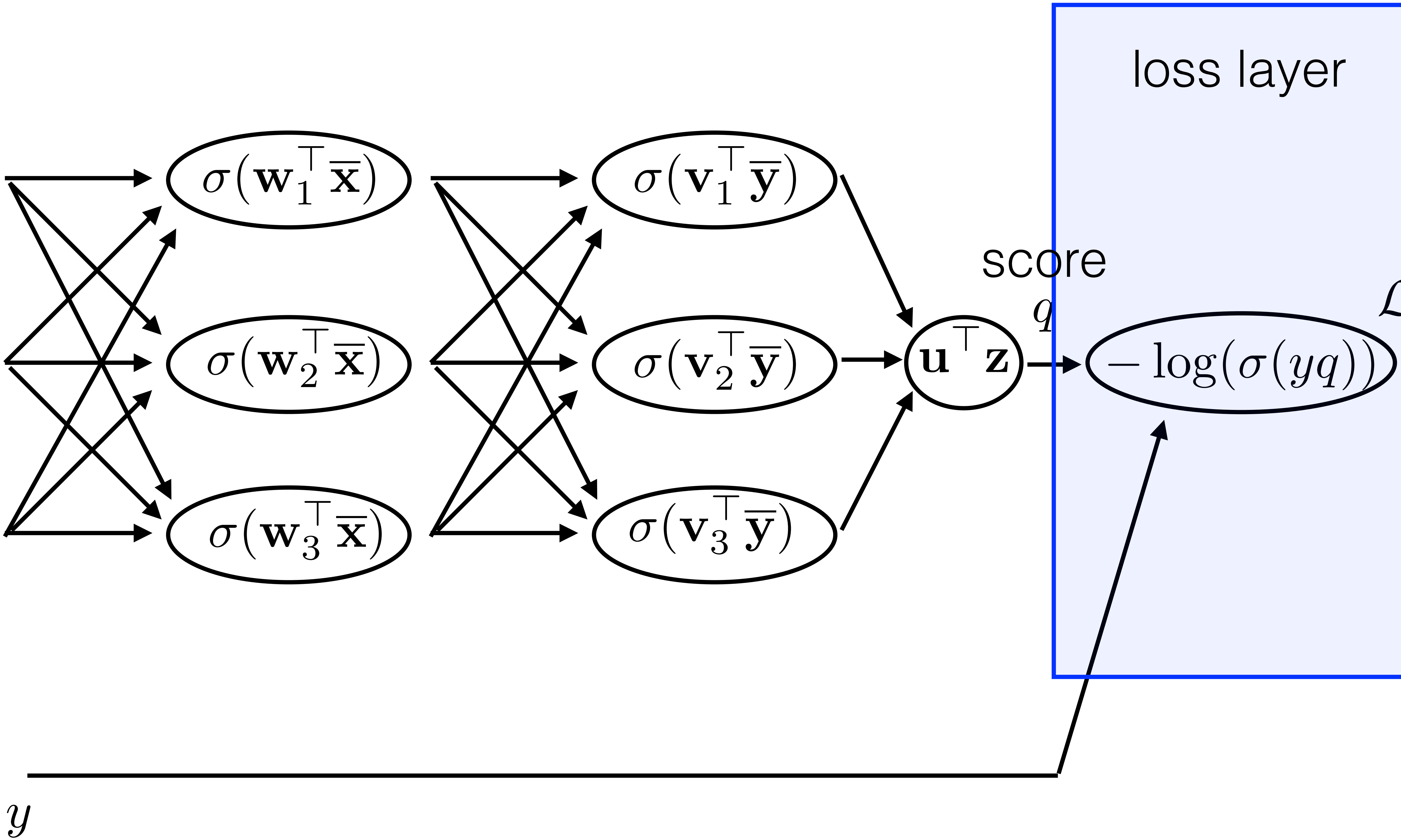
Fully connected neural network



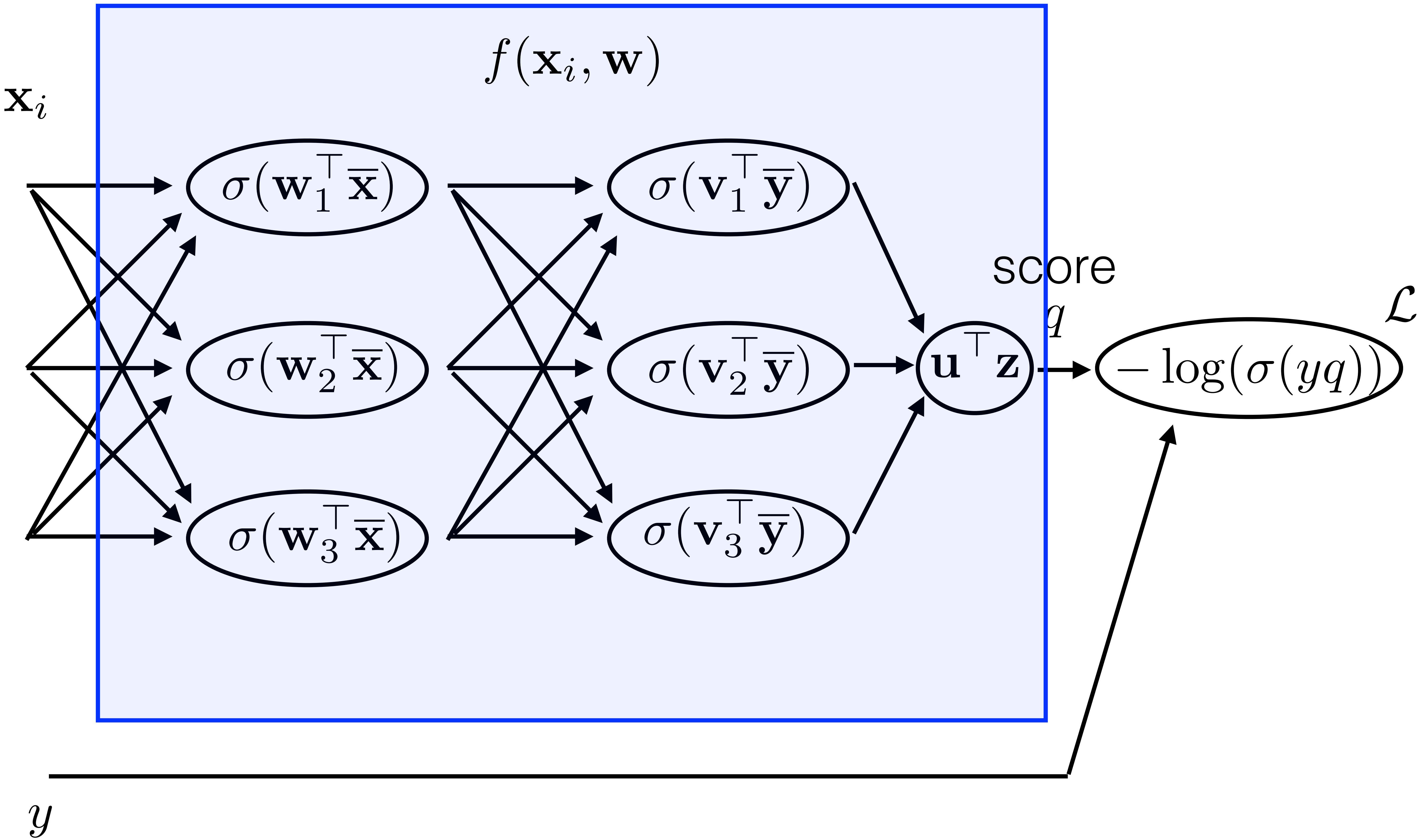
Fully connected neural network



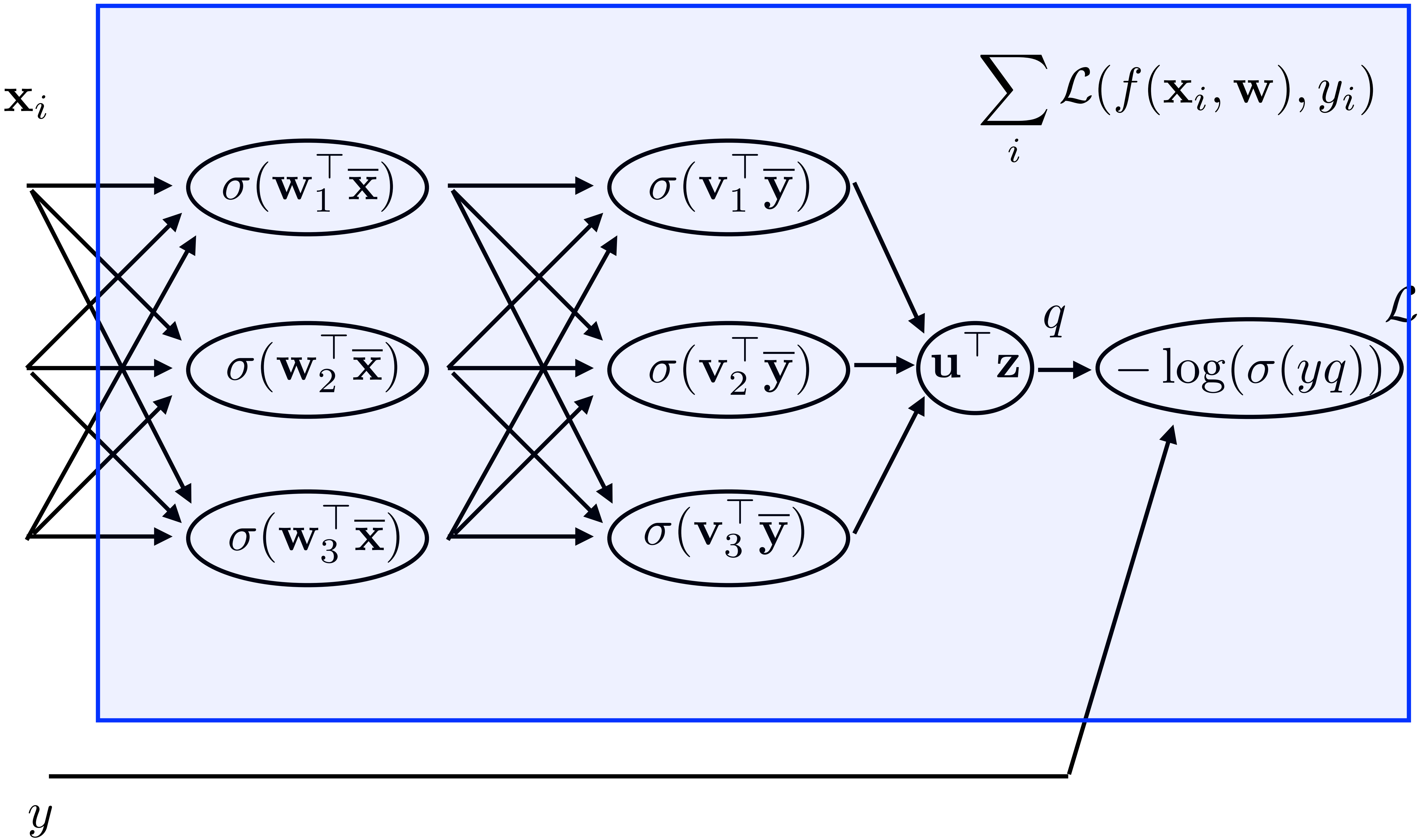
Fully connected neural network



Fully connected neural network



Fully connected neural network



Learning of fully connected neural network

1. Estimate gradient

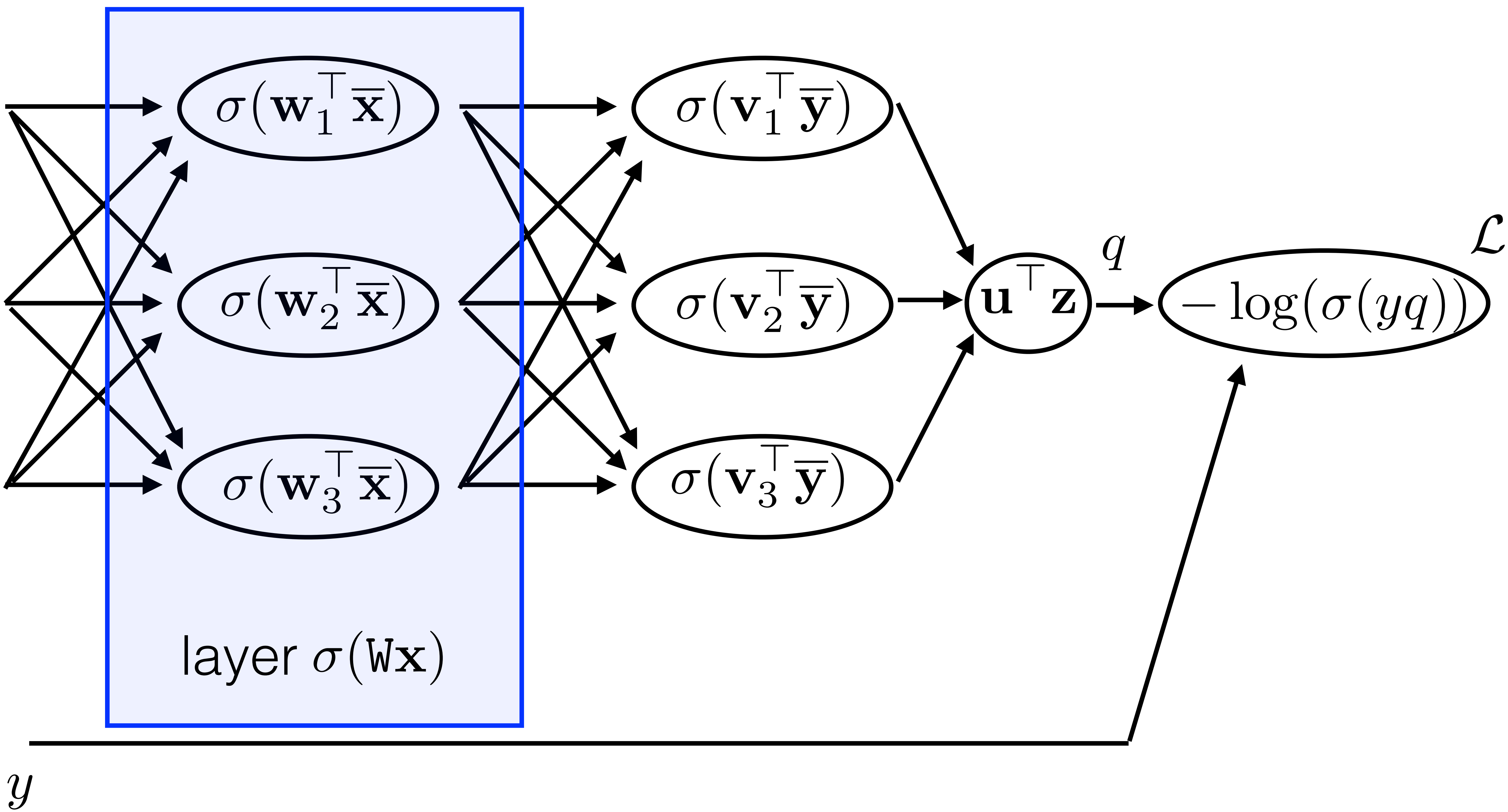
$$\sum_i \frac{\partial \mathcal{L}(f(\mathbf{x}_i, \mathbf{w}), y_i)}{\partial \mathbf{w}}$$

2. Update weights:

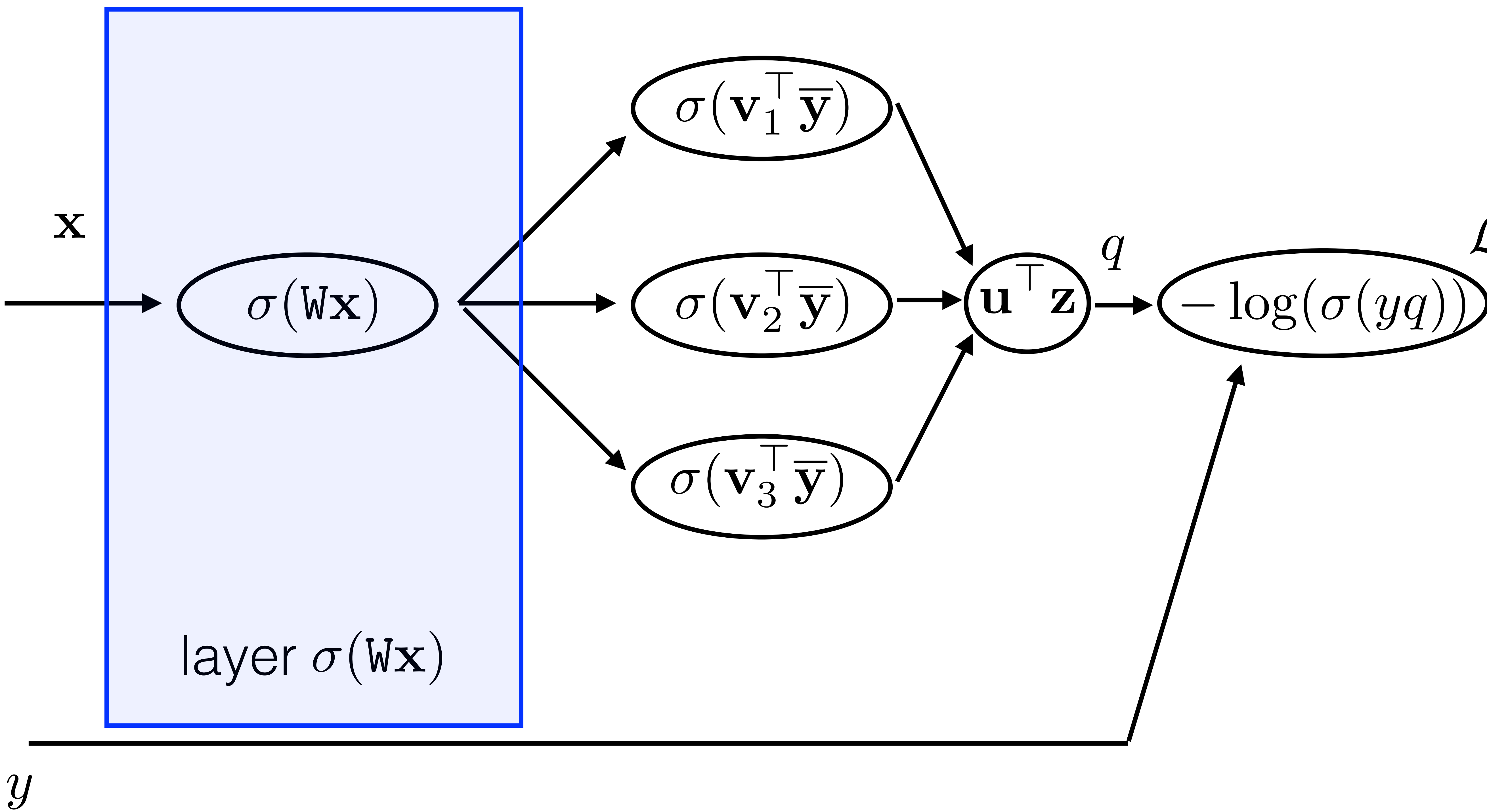
$$\mathbf{w} = \mathbf{w} - \alpha \sum_i \frac{\partial \mathcal{L}(f(\mathbf{x}_i, \mathbf{w}), y_i)}{\partial \mathbf{w}}$$

3. Optionally update learning rate α
4. Repeat until convergence

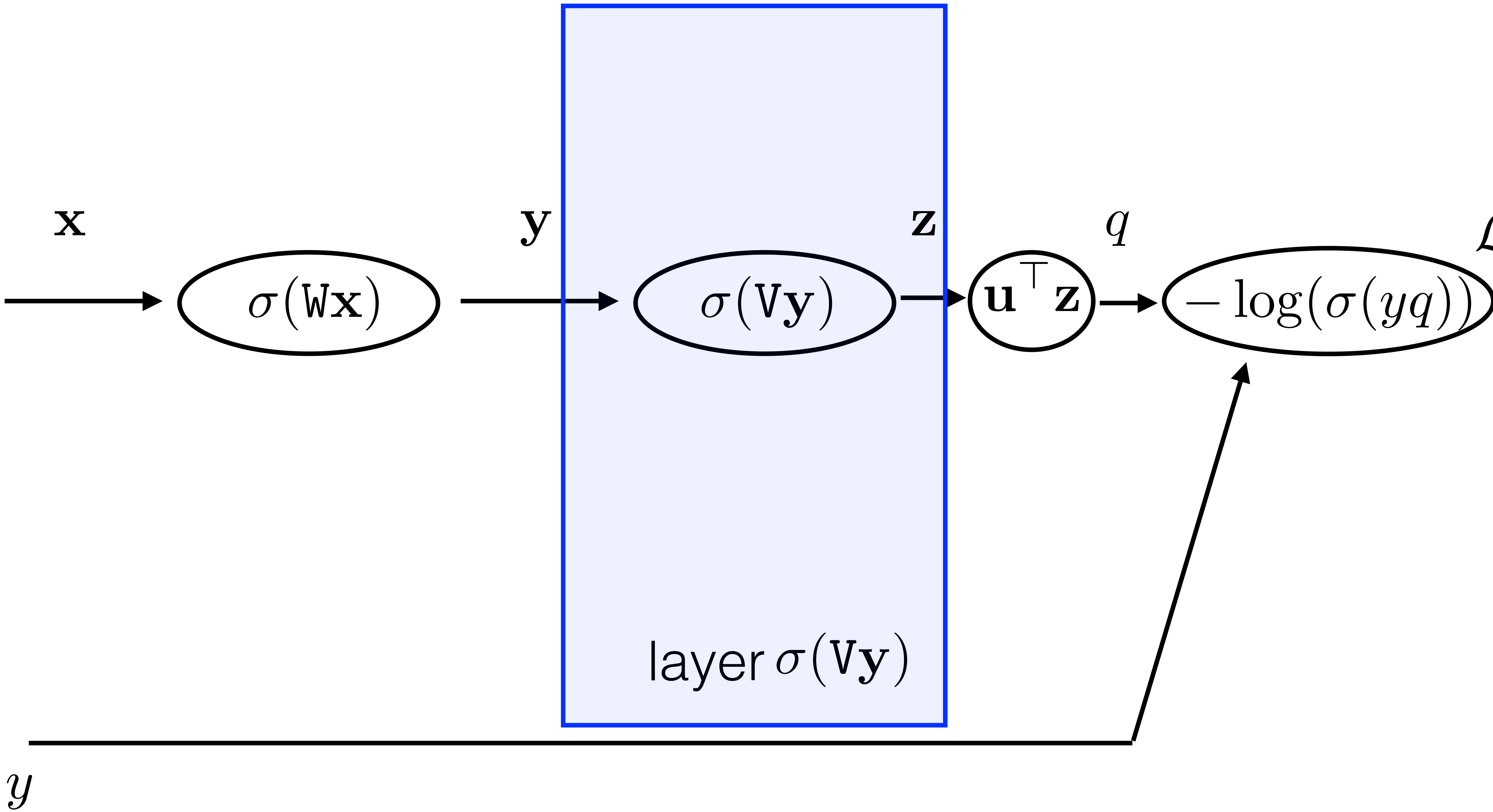
Learning of fully connected neural network



Learning of fully connected neural network

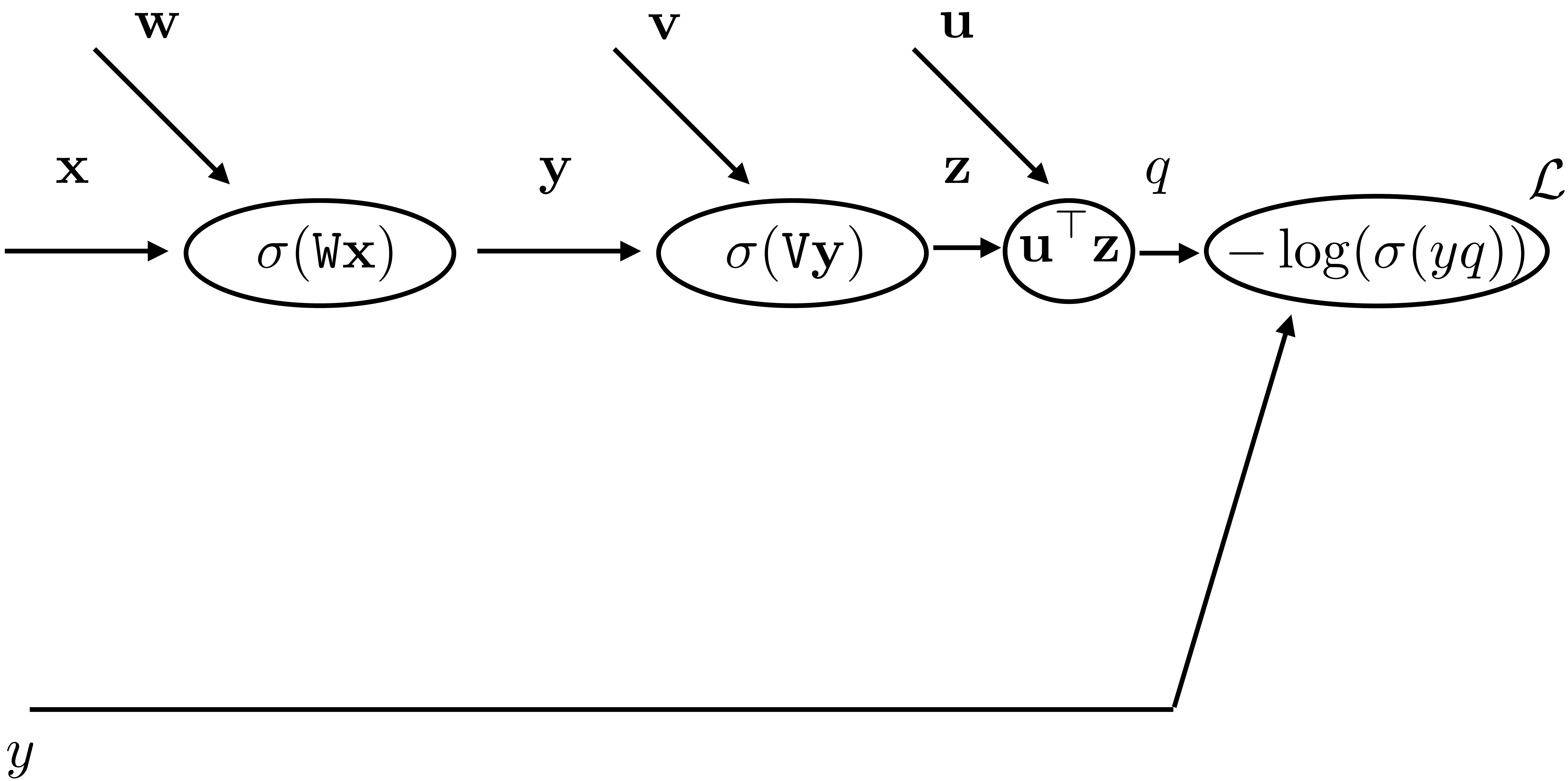


Learning of fully connected neural network



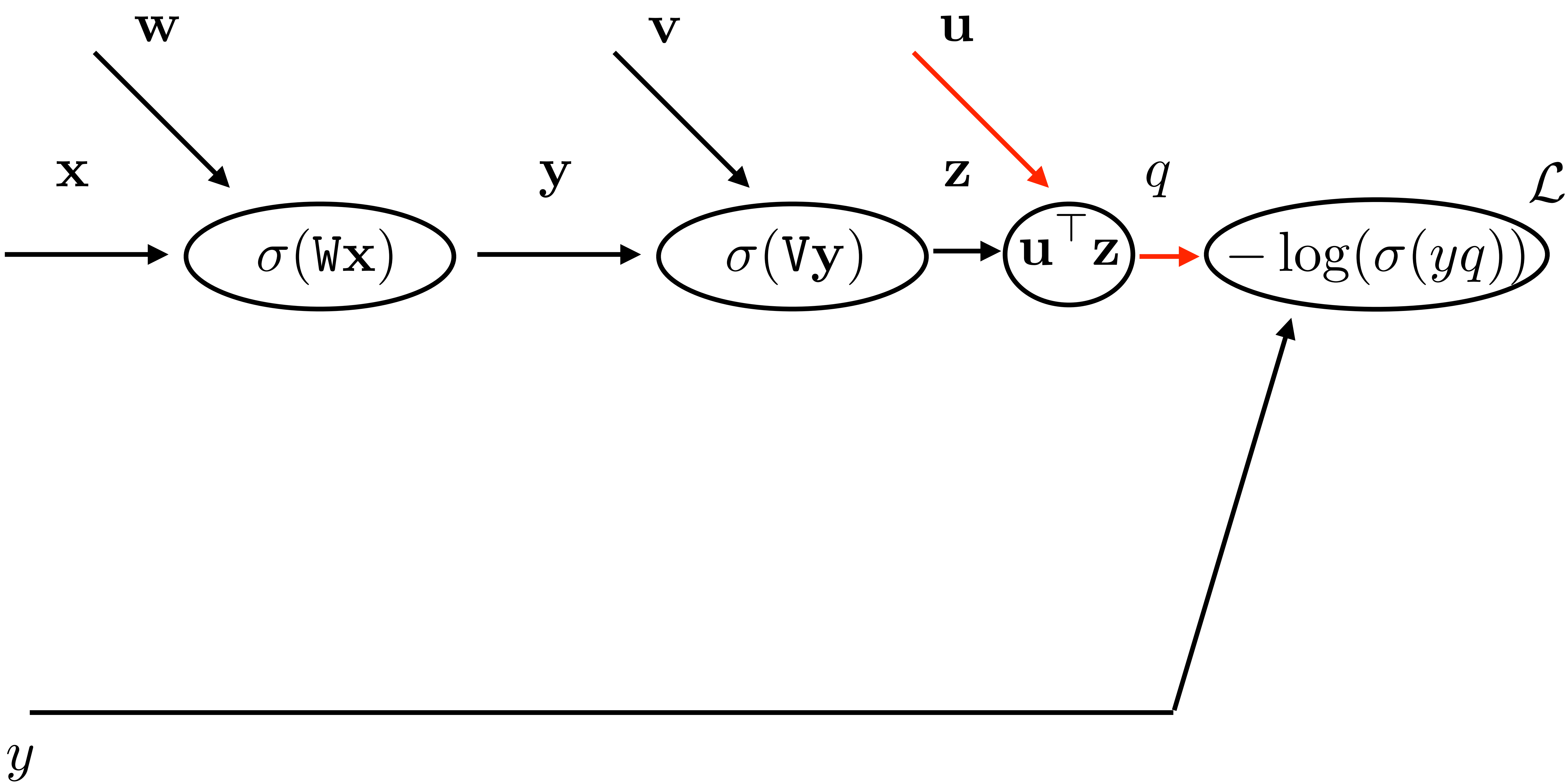
Learning of fully connected neural network

$\mathbf{w} = \mathrm{vec}(\mathtt{W}) \qquad \mathbf{v} = \mathrm{vec}(\mathtt{V})$



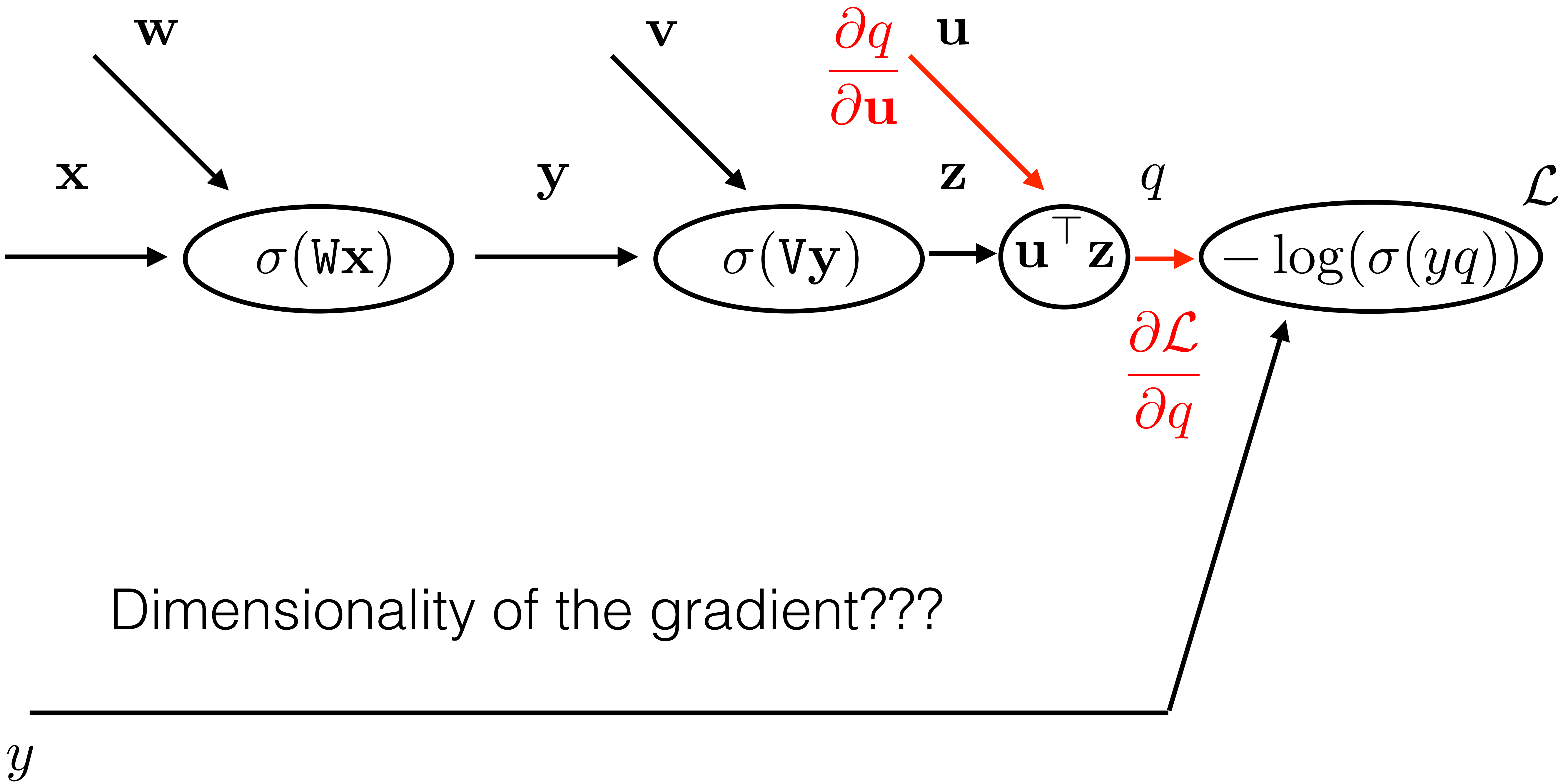
Learning of fully connected neural network

Derivative wrt $\mathbf{u}$: $\frac{\partial \mathcal{L}}{\partial \mathbf{u}} = ?$



Learning of fully connected neural network

Derivative wrt $\mathbf{u}$: $\frac{\partial \mathcal{L}}{\partial \mathbf{u}} = \frac{\partial \mathcal{L}}{\partial q} \frac{\partial q}{\partial \mathbf{u}}$

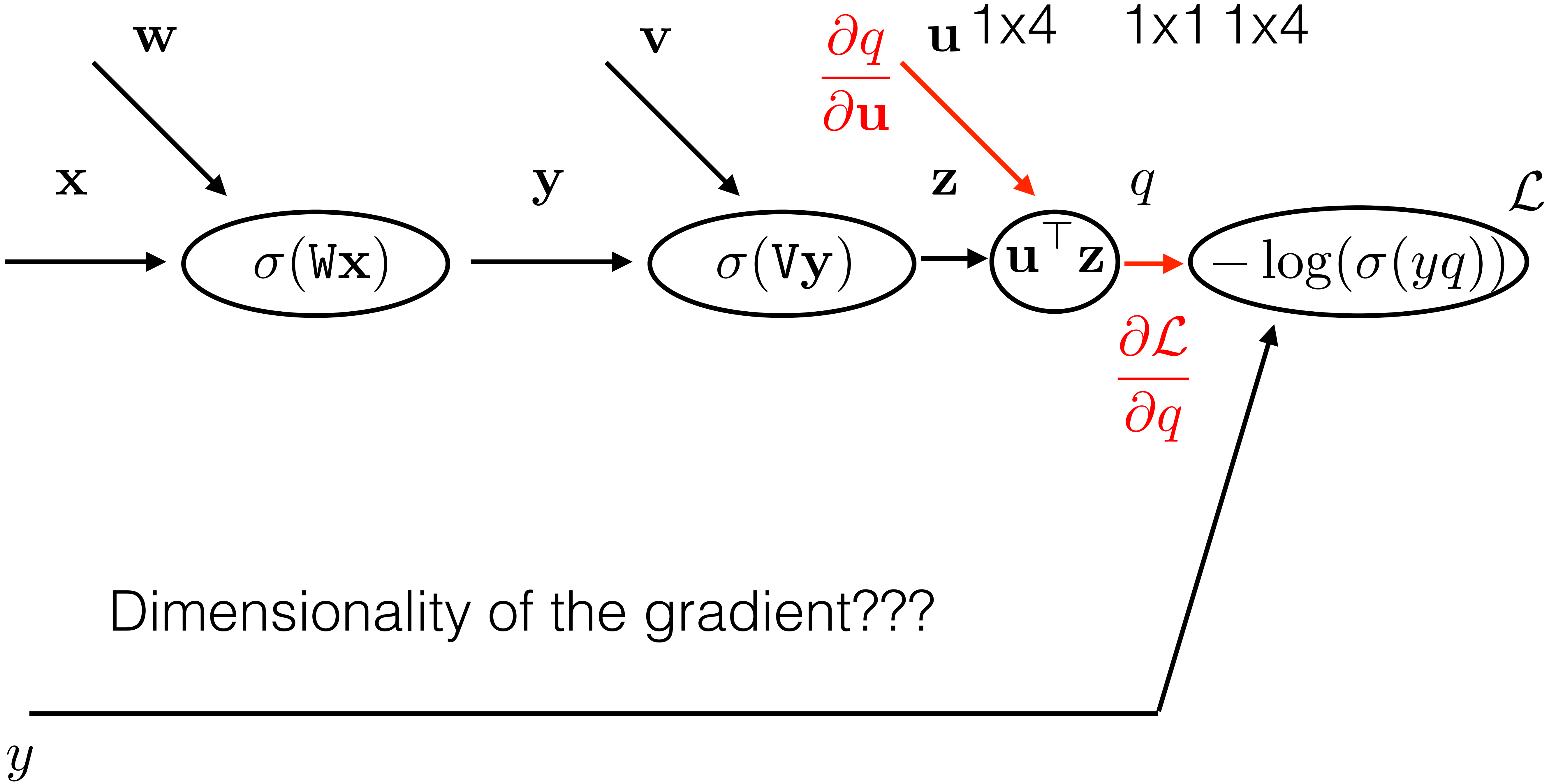


Dimensionality of the gradient???

Learning of fully connected neural network

Derivative wrt $\mathbf{u}$: $\frac{\partial \mathcal{L}}{\partial \mathbf{u}} = \frac{\partial \mathcal{L}}{\partial q} \frac{\partial q}{\partial \mathbf{u}}$

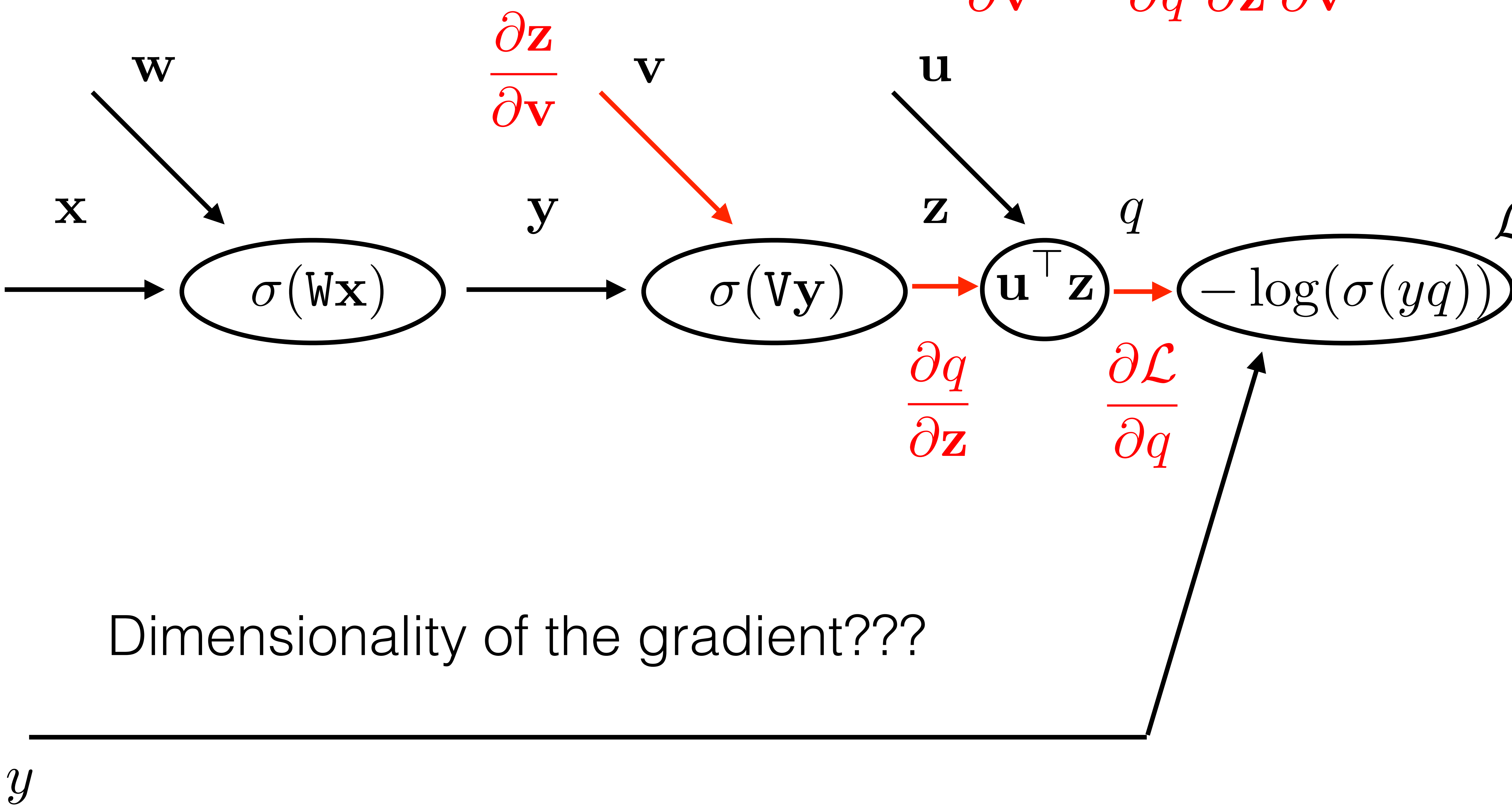
1x4 1x1 1x4



Dimensionality of the gradient???

Learning of fully connected neural network

Derivative wrt $\mathbf{v}$: $\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = \frac{\partial \mathcal{L}}{\partial q}\frac{\partial q}{\partial \mathbf{z}}\frac{\partial \mathbf{z}}{\partial \mathbf{v}}$

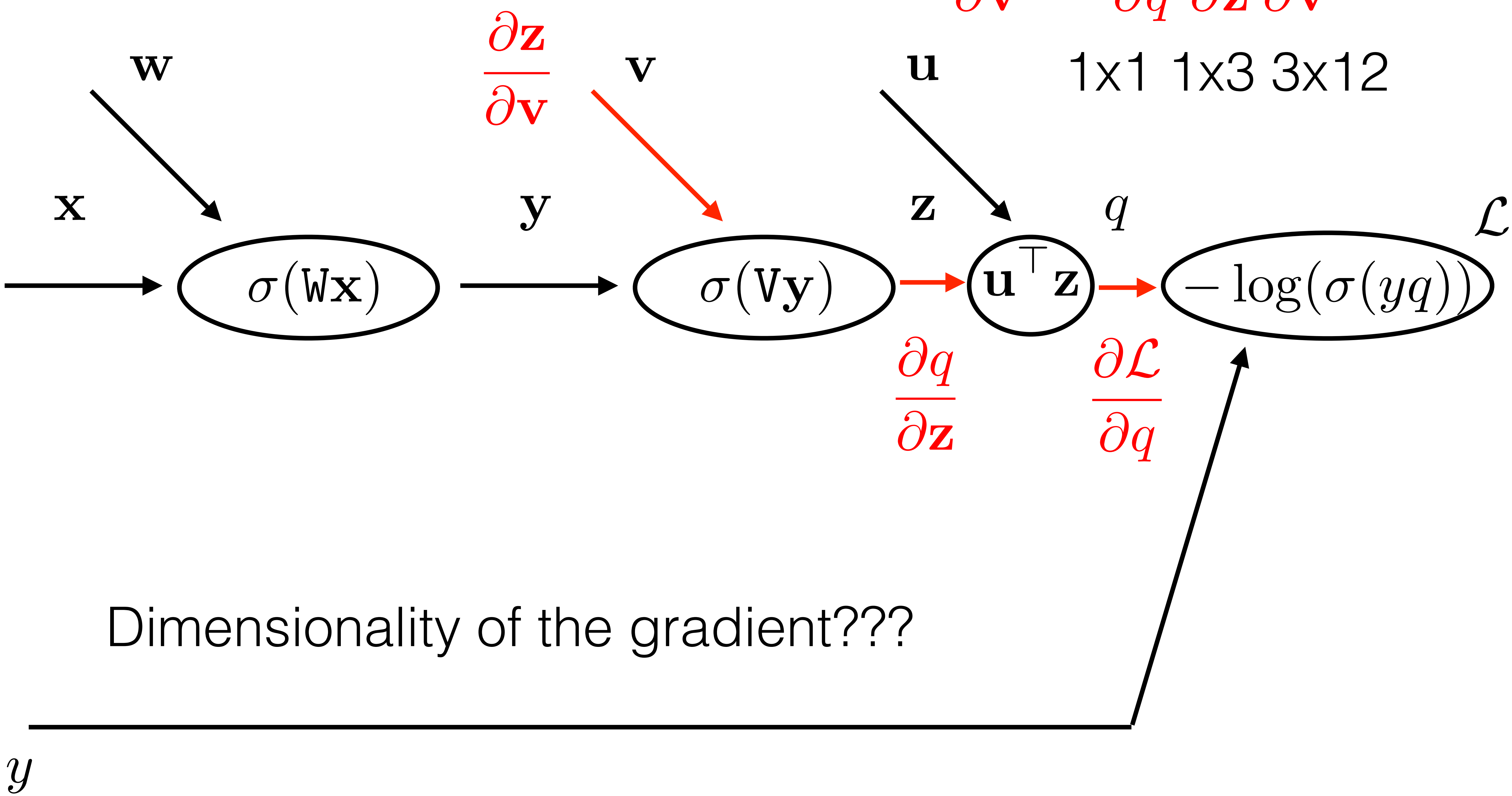


Dimensionality of the gradient???

Learning of fully connected neural network

Derivative wrt $\mathbf{v}$: $\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = \frac{\partial \mathcal{L}}{\partial q} \frac{\partial q}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{v}}$

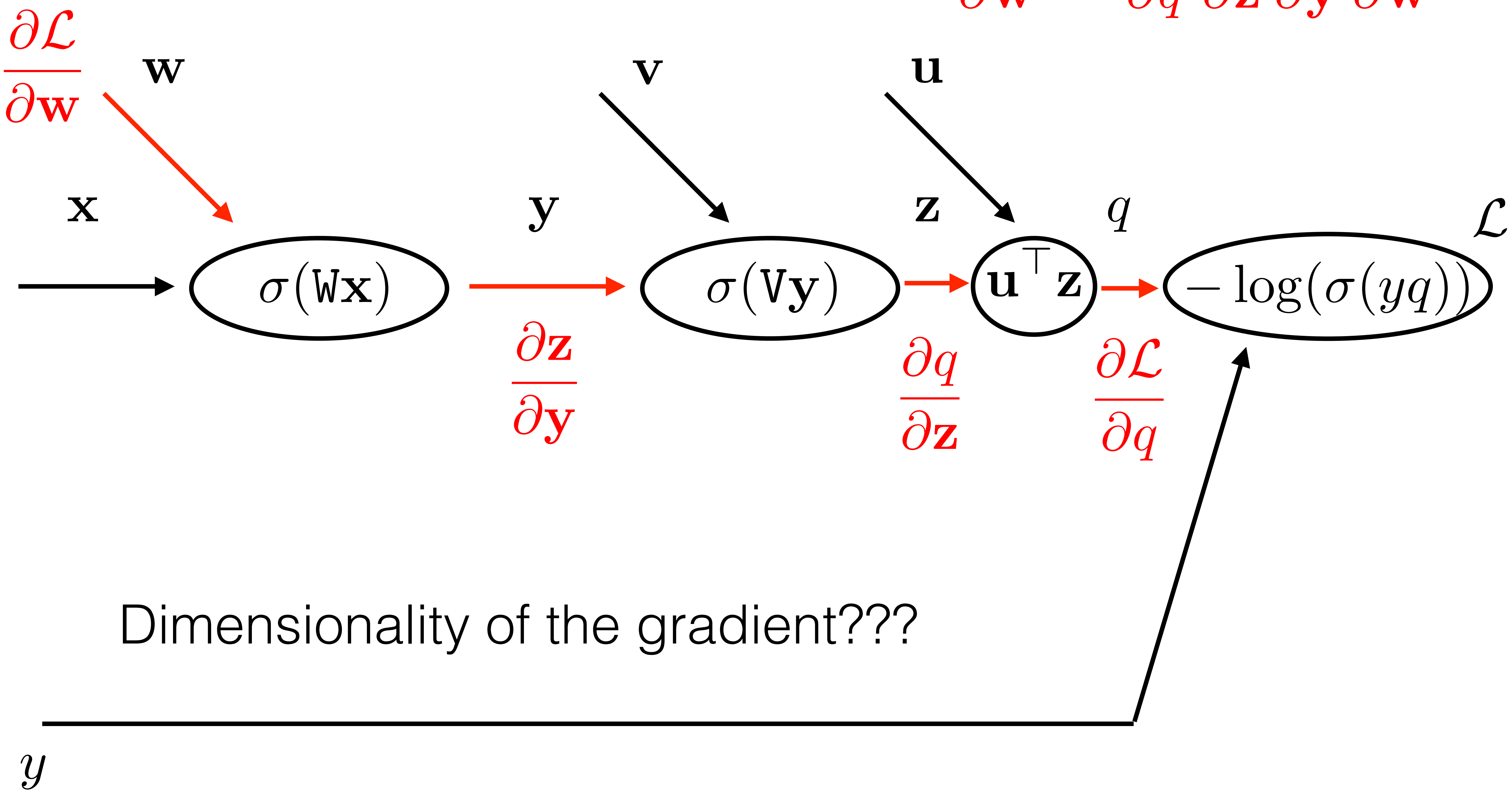
1x1 1x3 3x12



Dimensionality of the gradient???

Learning of fully connected neural network

Derivative wrt $\mathbf{w}$: $\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}}{\partial q}\frac{\partial q}{\partial \mathbf{z}}\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \mathbf{w}}$

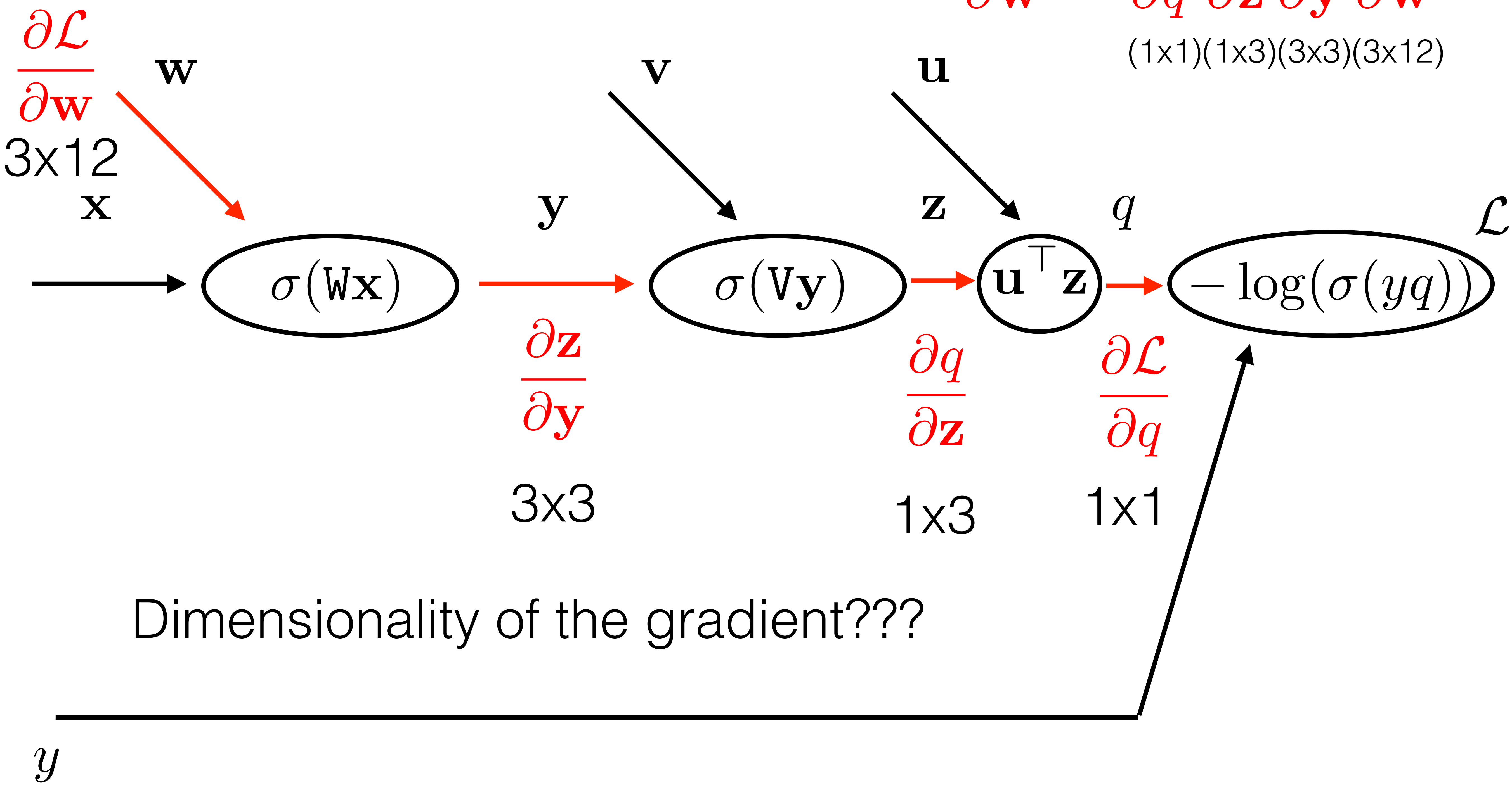


Dimensionality of the gradient???

Learning of fully connected neural network

Derivative wrt $\mathbf{w}$: $$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}}{\partial q}\frac{\partial q}{\partial \mathbf{z}}\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \mathbf{w}}$$

(1x1)(1x3)(3x3)(3x12)



Dimensionality of the gradient???

Learning of fully connected neural network

1. Estimate all required local gradients
2. Update weights:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{u}} = \frac{\partial \mathcal{L}}{\partial q}\frac{\partial q}{\partial \mathbf{u}} \qquad \mathbf{u} = \mathbf{u} - \alpha \left[\frac{\partial \mathcal{L}}{\partial \mathbf{u}}\right]^\top$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = \frac{\partial \mathcal{L}}{\partial q}\frac{\partial q}{\partial \mathbf{z}}\frac{\partial \mathbf{z}}{\partial \mathbf{v}} \qquad \mathbf{v} = \mathbf{v} - \alpha \left[\frac{\partial \mathcal{L}}{\partial \mathbf{v}}\right]^\top$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}}{\partial q}\frac{\partial q}{\partial \mathbf{z}}\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \mathbf{w}} \qquad \mathbf{w} = \mathbf{w} - \alpha \left[\frac{\partial \mathcal{L}}{\partial \mathbf{w}}\right]^\top$$

3. Optionally update learning rate α
4. Repeat until convergence

Neural nets summary

- Neural net is a function created as concatenation of simplier functions (e.g. neurons or layers of neurons)
- Gradient optimization of the neural net is called backpropagation
- Neural net frameworks has many predefined layers
- **Spoiler alert:** It does not work (on images) at all - why?

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left(\sum_i -\log(p(y_i|\mathbf{x}_i, \mathbf{w})) \right) + (-\log p(\mathbf{w}))$$

loss function prior/regulariser

- Class of function represented by a NN is too general.
- Naive regulariser helps a bit, but dimensionality/wildness is huge => curse-of-dimensionality, overfitting,…
- What is number of weights between two 1000-neuron layers?
- **Next lecture:** study animal cortex to find a stronger prior on the class of suitable functions.
- **Spoiler alert 2:**
  reduce very general class of functions ”neuron layer” to very specific sub-class of functions “convolution layer”