Training neural nets through Empirical Risk Minimization: Problem Setup

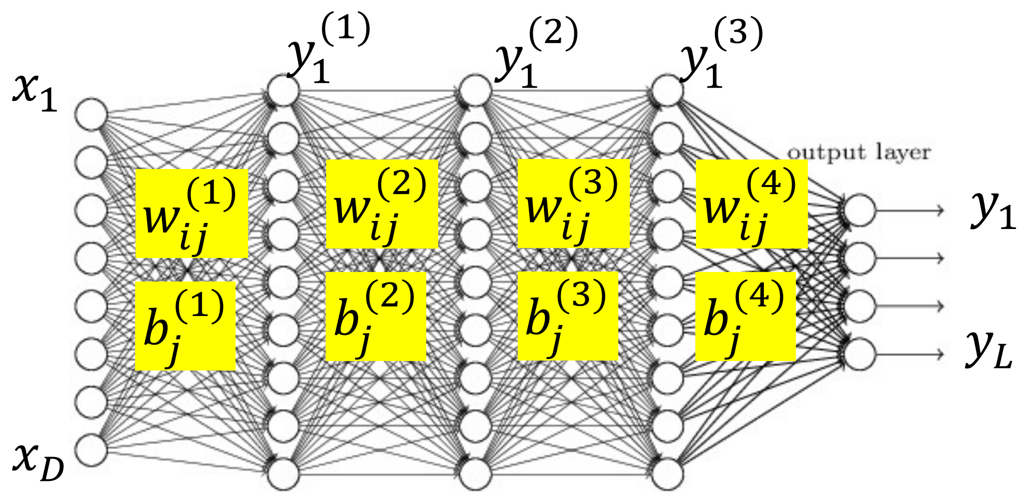
- Given a training set of input-output pairs $(X_1, d_1), (X_2, d_2), \dots, (X_T, d_T)$

- The divergence on the ith instance is $div(Y_i, d_i)$
  - $Y_i = f(X_i; W)$
- The loss (empirical risk)

$$Loss(W) = \frac{1}{T}\sum_i div(Y_i, d_i)$$

- Minimize $Loss$ w.r.t $\left\{w_{ij}^{(k)}, b_j^{(k)}\right\}$ using gradient descent

Notation



- The input layer is the 0th layer
- We will represent the output of the i-th perceptron of the kth layer as $y_i^{(k)}$
  - **Input to network:** $y_i^{(0)} = x_i$
  - **Output of network:** $y_i = y_i^{(N)}$
- We will represent the weight of the connection between the i-th unit of the k-1th layer and the jth unit of the k-th layer as $w_{ij}^{(k)}$
  - The bias to the jth unit of the k-th layer is $b_j^{(k)}$

Recap: Gradient Descent Algorithm

- Initialize:
  - W^0
  - $k = 0$

To minimize any function Loss(W) w.r.t W

- do
  - $W^{k+1} = W^k - \eta^k \nabla Loss(W^k)^T$
  - $k = k + 1$
- while $\left|Loss\left(W^k\right) - Loss\left(W^{k-1}\right)\right| > \varepsilon$

Recap: Gradient Descent Algorithm

- In order to minimize $L(W)$ w.r.t. W
- Initialize:
  - W^0
  - $k = 0$
- do
  - For every component i
    - $W_i^{k+1} = W_i^k - \eta^k \frac{\partial L}{\partial W_i}$

      Explicitly stating it by component
  - $k = k + 1$
- while $\left|L\left(W^k\right) - L\left(W^{k-1}\right)\right| > \varepsilon$

Training Neural Nets through Gradient Descent

Total training Loss:

$$Loss = \frac{1}{T}\sum_{t} Div(\boldsymbol{Y}_t, \boldsymbol{d}_t)$$

- Gradient descent algorithm:

Assuming the bias is also represented as a weight

- Initialize all weights and biases $\left\{w_{ij}^{(k)}\right\}$
  - Using the extended notation: the bias is also a weight
- Do:
  - For every layer k for all i, j, update:
    - $w_{i,j}^{(k)} = w_{i,j}^{(k)} - \eta \frac{dLoss}{dw_{i,j}^{(k)}}$
- Until $Loss$ has converged

The derivative

Total training Loss:

$$Loss = \frac{1}{T}\sum_t Div(\boldsymbol{Y}_t, \boldsymbol{d}_t)$$

Total derivative:

$$\frac{dLoss}{dw_{i,j}^{(k)}} = \frac{1}{T}\sum_t \frac{dDiv(\boldsymbol{Y}_t, \boldsymbol{d}_t)}{dw_{i,j}^{(k)}}$$

- So we must first figure out how to compute the derivative of divergences of individual training inputs

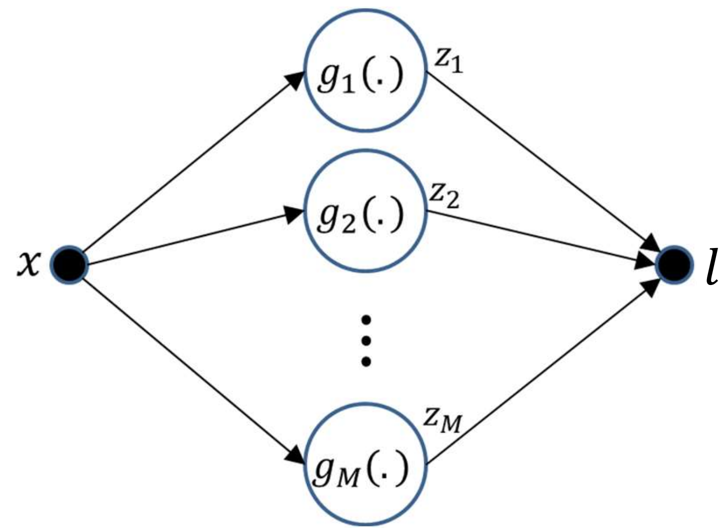
Calculus Refresher: Chain rule summary

For any nested function $l = f(y)$ where $y = g(z)$

$$\frac{dl}{dz} = \frac{dl}{dy}\frac{dy}{dz}$$

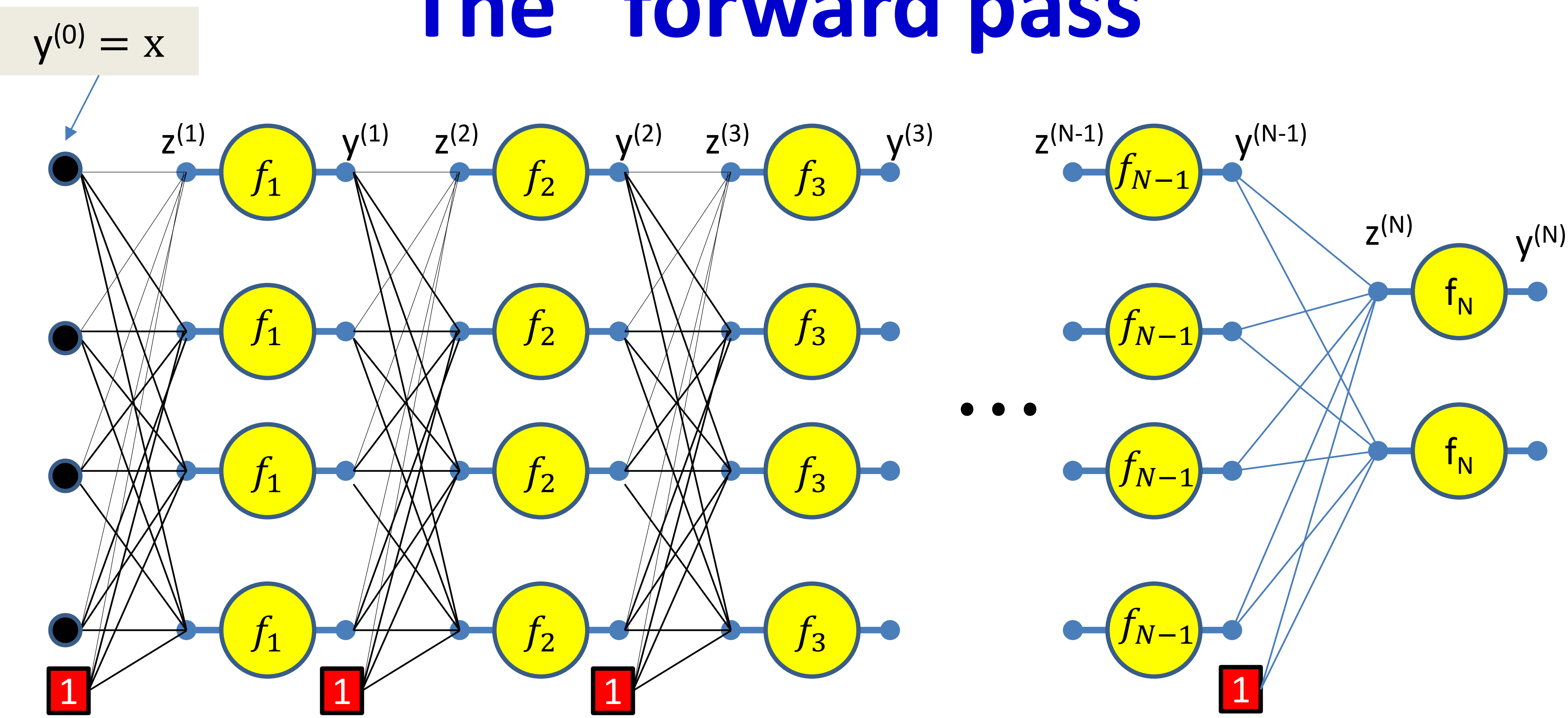
For $l = f(z_1, z_2, \ldots, z_M)$

where $z_i = g_i(x)$



$$\frac{dl}{dx} = \frac{\partial l}{\partial z_1}\frac{dz_1}{dx} + \frac{\partial l}{\partial z_2}\frac{dz_2}{dx} + \cdots + \frac{\partial l}{\partial z_M}\frac{dz_M}{dx}$$

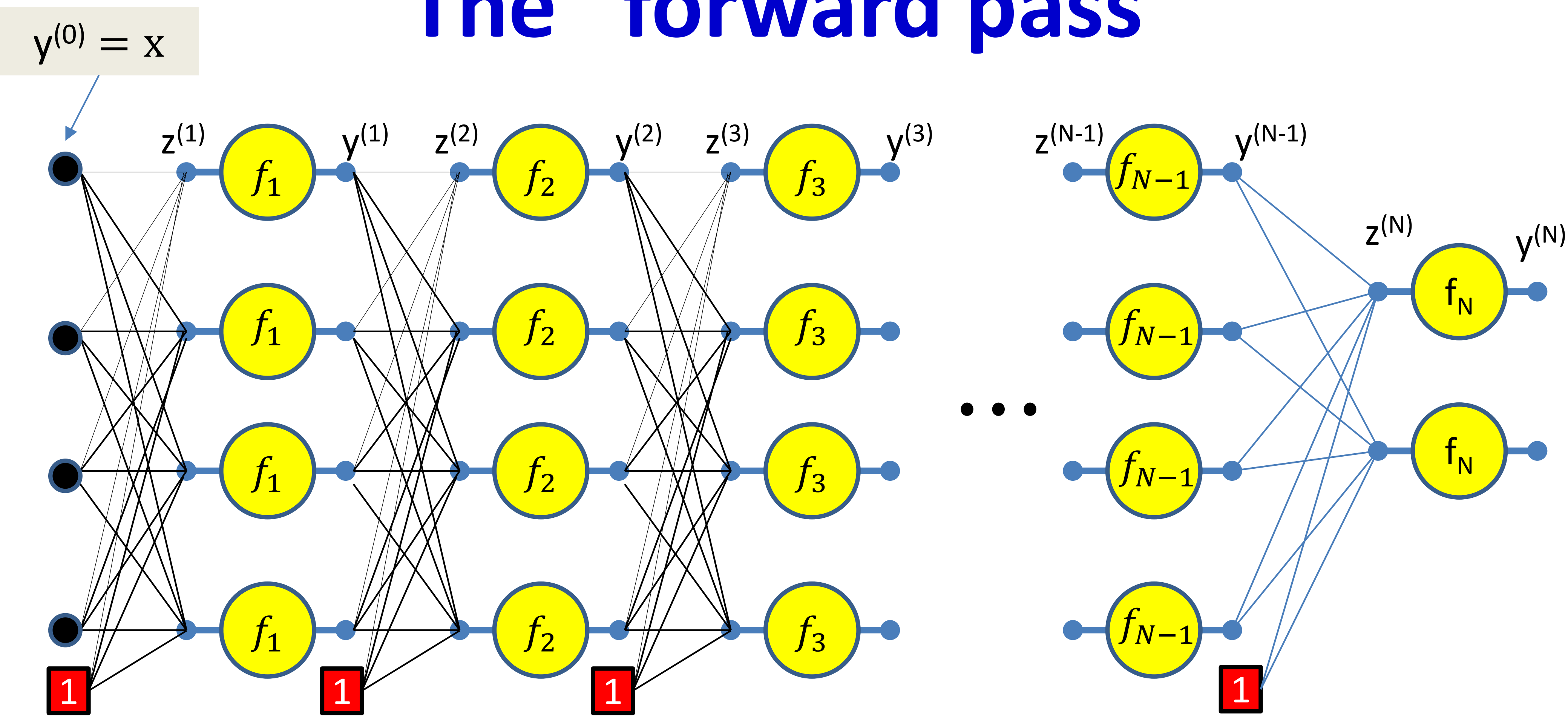
The "forward pass"



We will refer to the process of computing the output from an input as the *forward pass*

We will illustrate the forward pass in the following slides

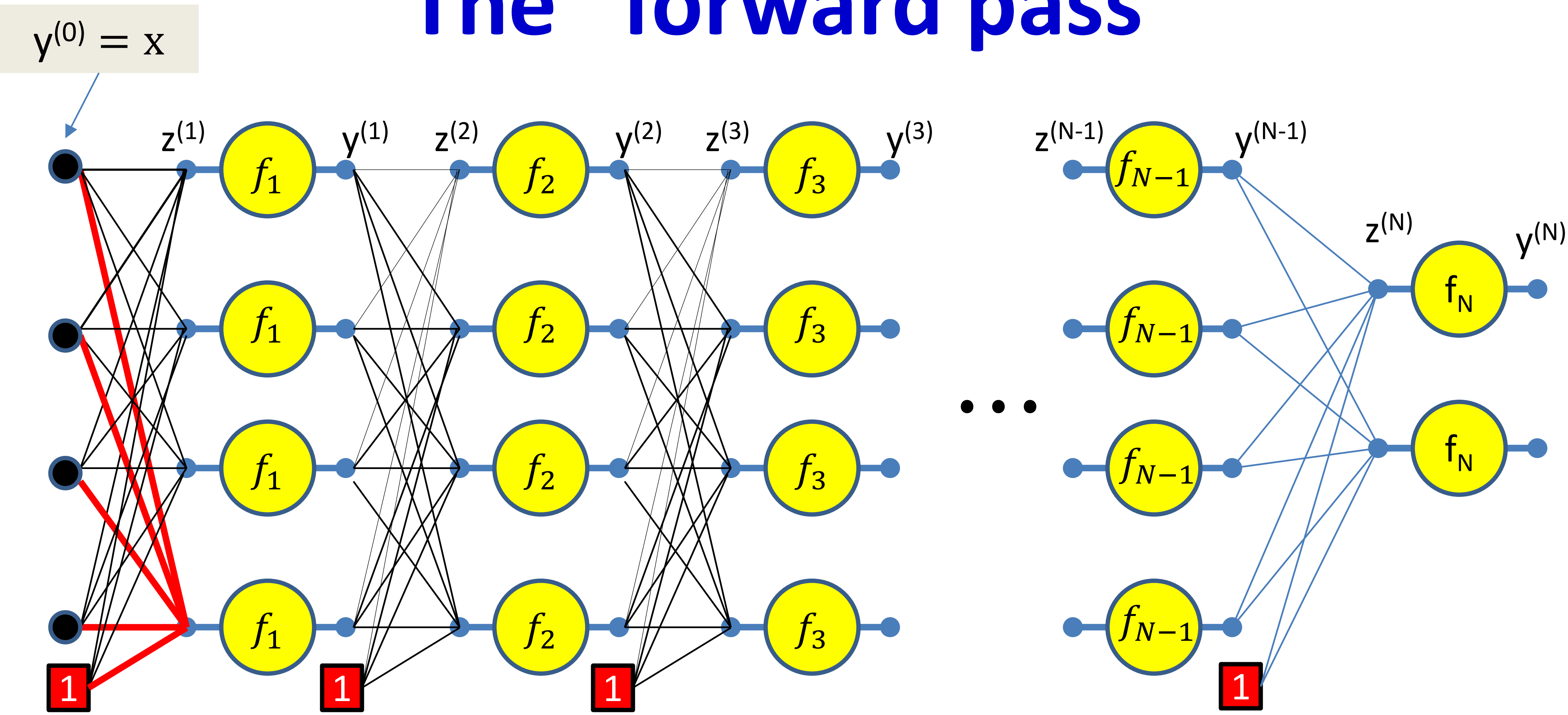
The “forward pass”



Setting $y_i^{(0)} = x_i$ for notational convenience

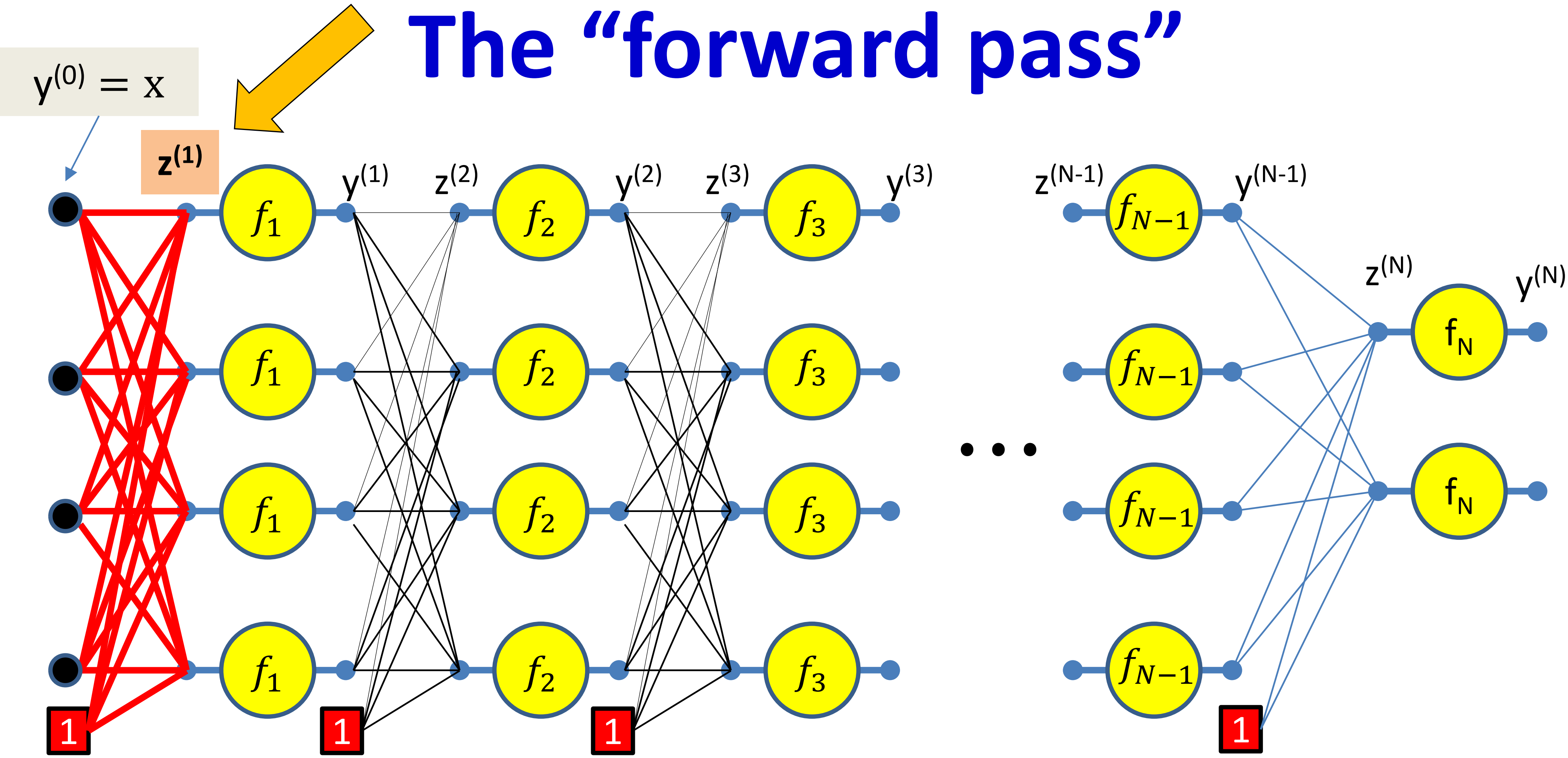
Assuming $w_{0j}^{(k)} = b_j^{(k)}$ and $y_0^{(k)} = 1$ -- assuming the bias is a weight and extending the output of every layer by a constant 1, to account for the biases

The “forward pass”

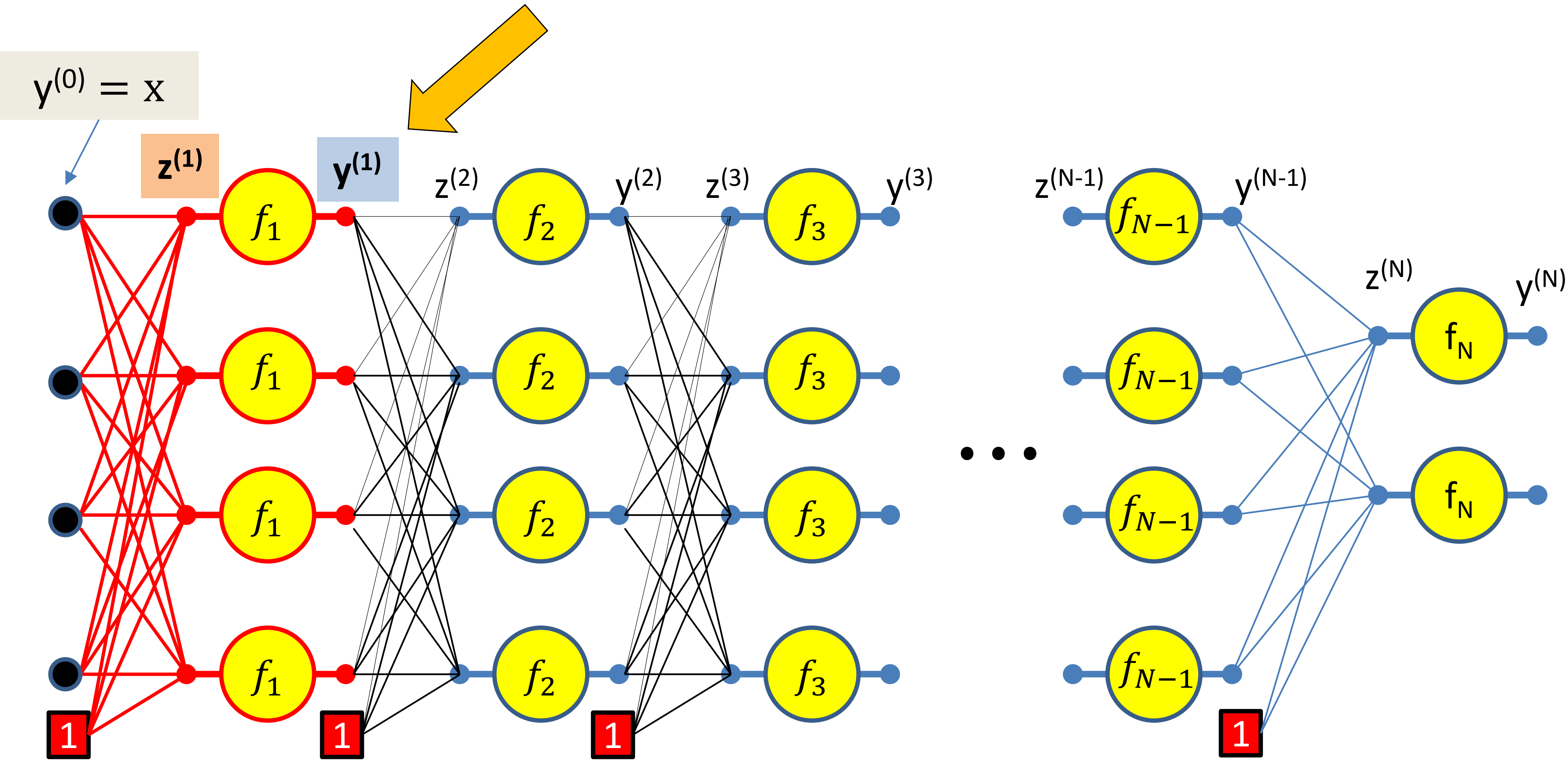


$$z_1^{(1)} = \sum_i w_{i1}^{(1)} y_i^{(0)}$$

The "forward pass"

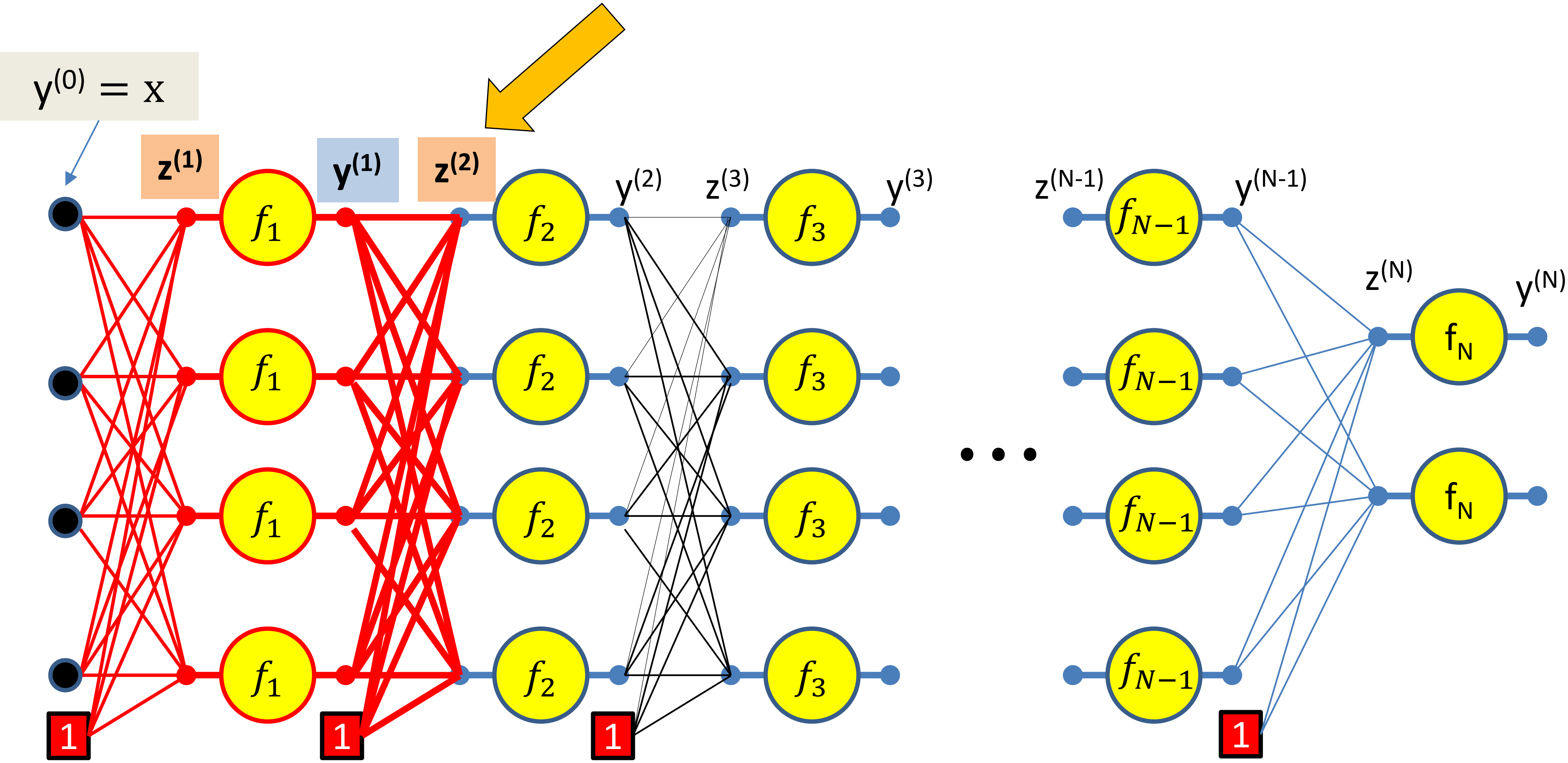


$$z_j^{(1)} = \sum_i w_{ij}^{(1)} y_i^{(0)}$$



$$z_j^{(1)} = \sum_i w_{ij}^{(1)} y_i^{(0)}$$

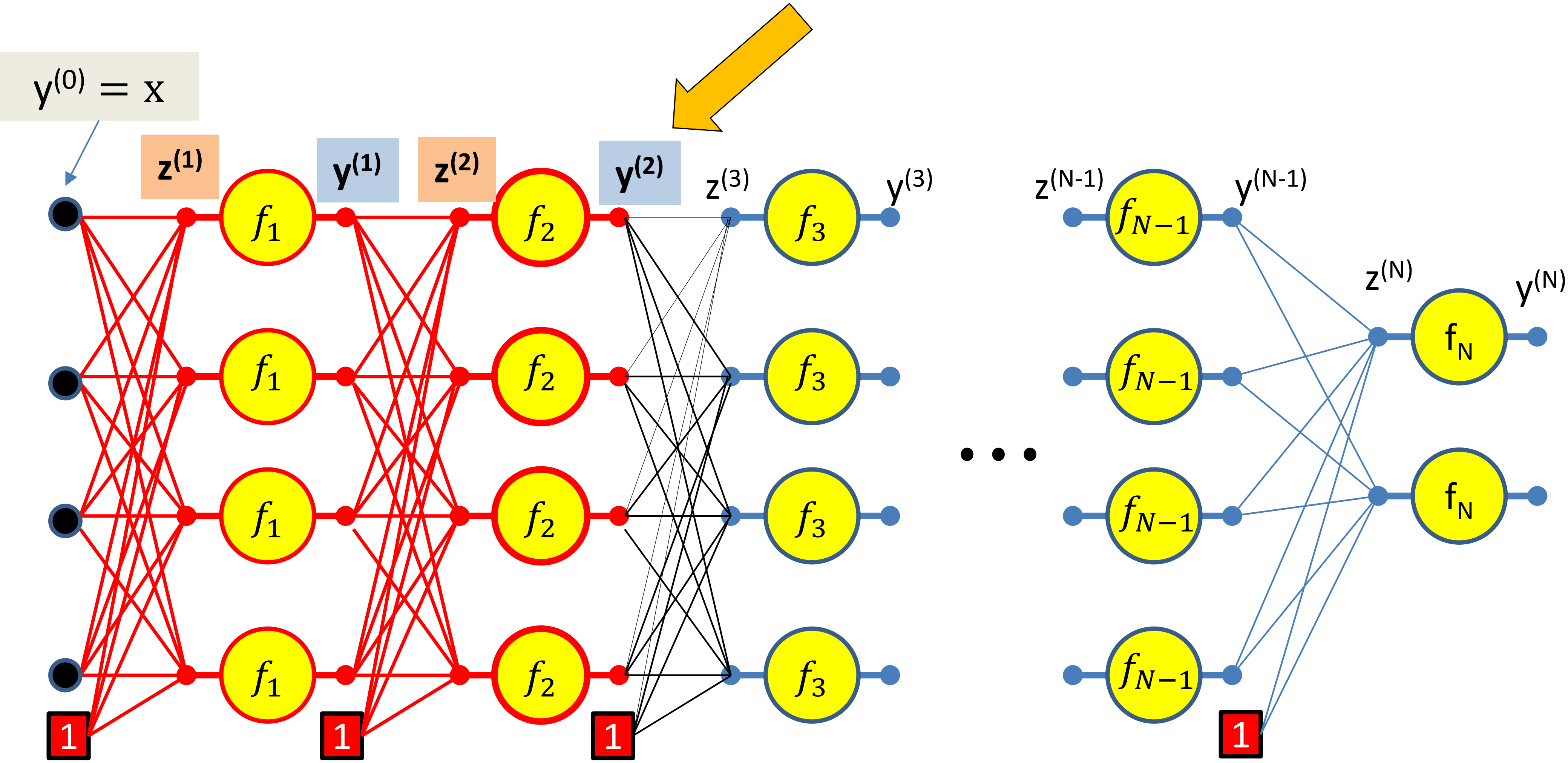
$$\boxed{y_j^{(1)} = f_1\left(z_j^{(1)}\right)}$$



$$z_j^{(1)} = \sum_i w_{ij}^{(1)} y_i^{(0)}$$

$$y_j^{(1)} = f_1\left(z_j^{(1)}\right)$$

$$\boxed{z_j^{(2)} = \sum_i w_{ij}^{(2)} y_i^{(1)}}$$

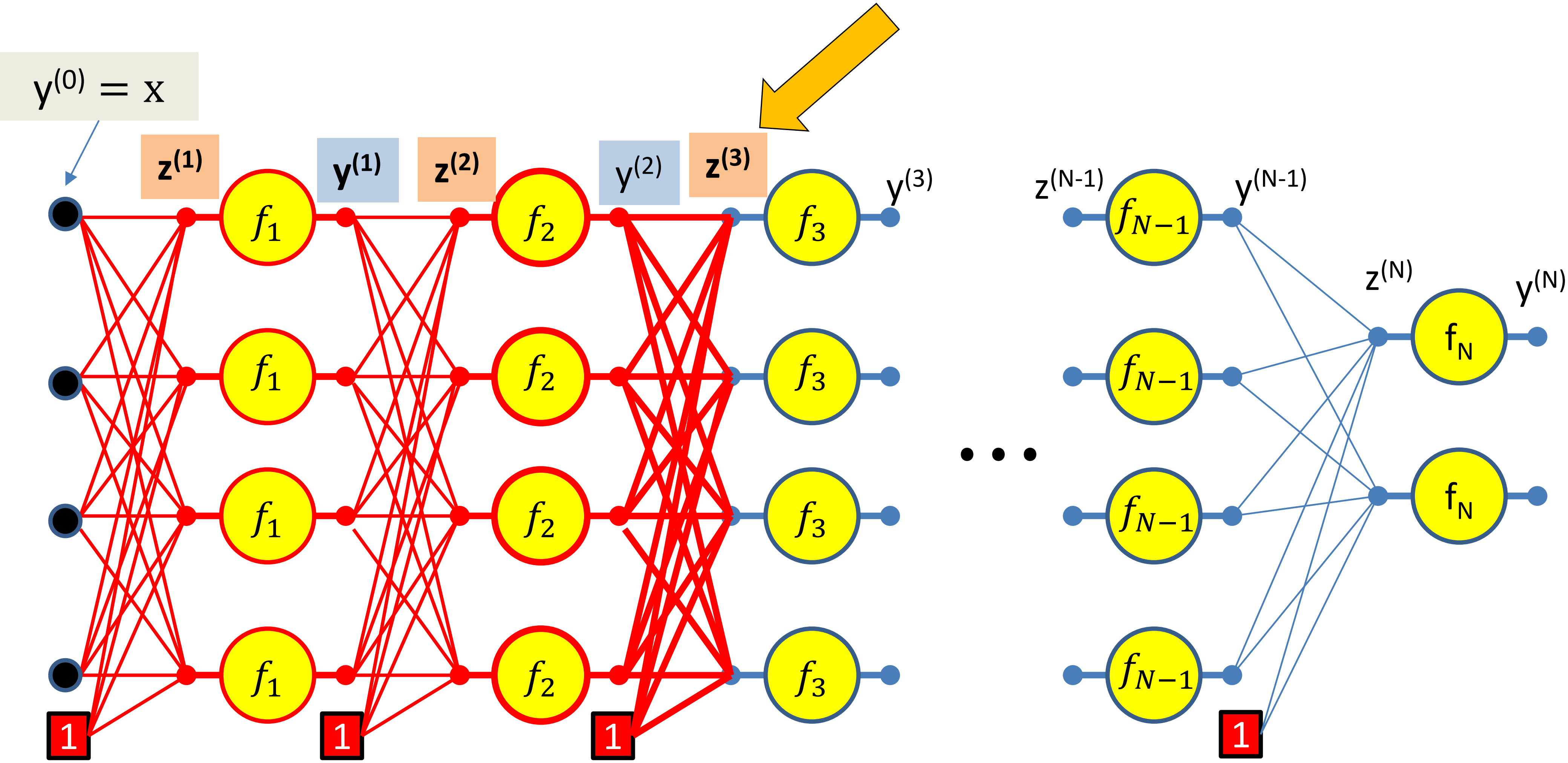


$$z_j^{(1)} = \sum_i w_{ij}^{(1)} y_i^{(0)}$$

$$y_j^{(1)} = f_1\left(z_j^{(1)}\right)$$

$$z_j^{(2)} = \sum_i w_{ij}^{(2)} y_i^{(1)}$$

$$\boxed{y_j^{(2)} = f_2\left(z_j^{(2)}\right)}$$



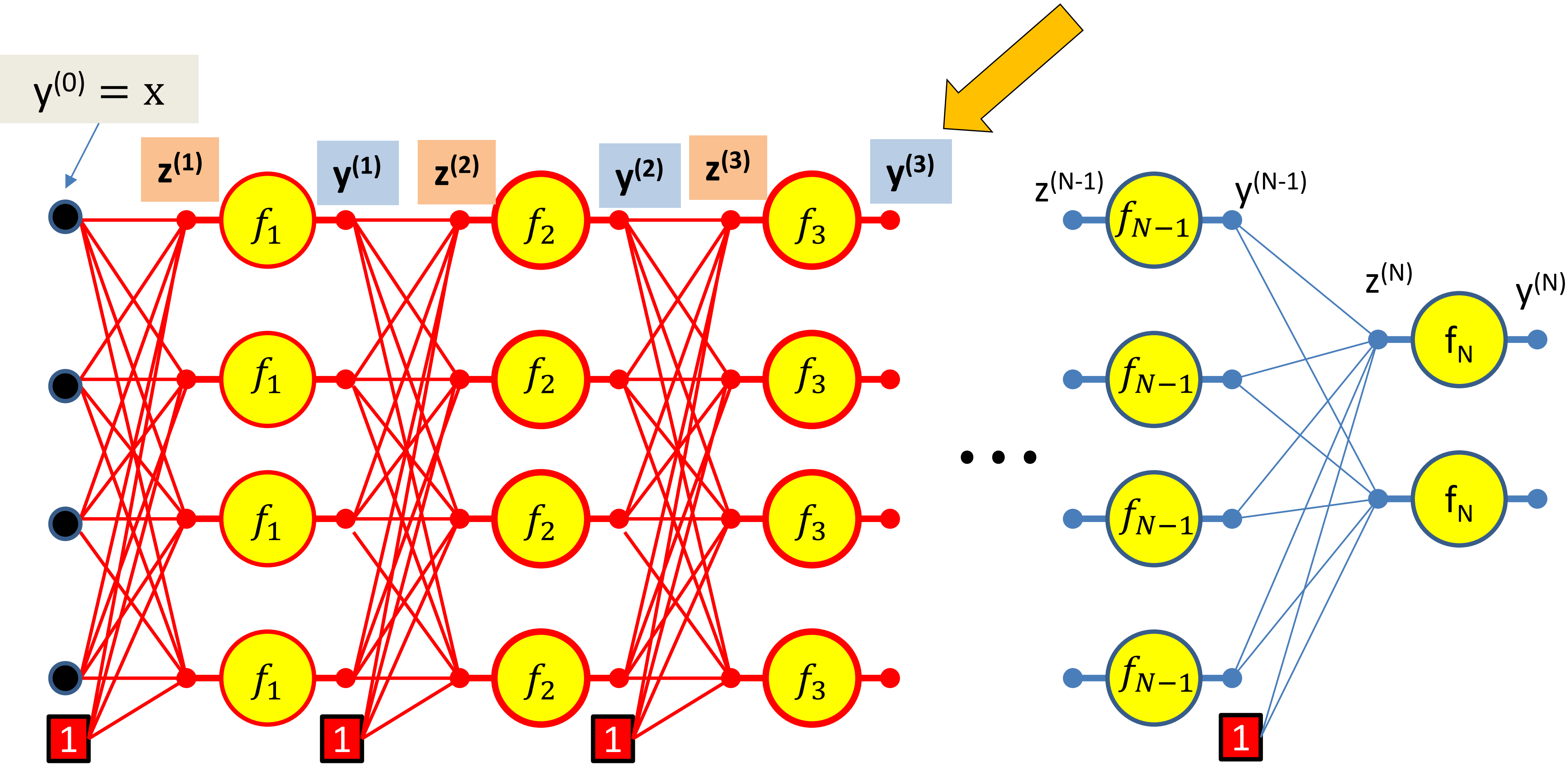
$$z_j^{(1)} = \sum_i w_{ij}^{(1)} y_i^{(0)}$$

$$y_j^{(1)} = f_1\left(z_j^{(1)}\right)$$

$$z_j^{(2)} = \sum_i w_{ij}^{(2)} y_i^{(1)}$$

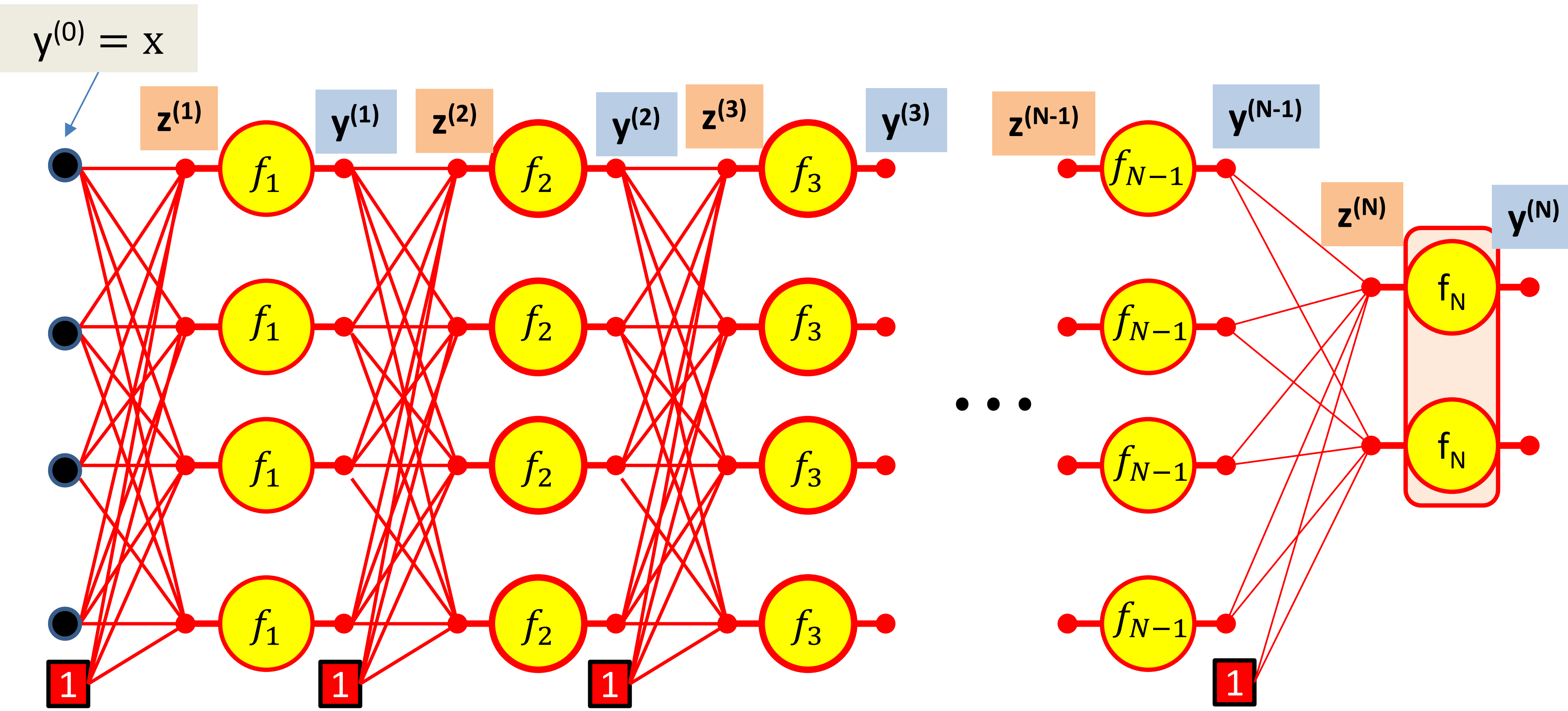
$$y_j^{(2)} = f_2\left(z_j^{(2)}\right)$$

$$\boxed{z_j^{(3)} = \sum_i w_{ij}^{(3)} y_i^{(2)}}$$



$$z_j^{(1)} = \sum_i w_{ij}^{(1)} y_i^{(0)}$$ $$y_j^{(1)} = f_1\left(z_j^{(1)}\right)$$ $$z_j^{(2)} = \sum_i w_{ij}^{(2)} y_i^{(1)}$$ $$y_j^{(2)} = f_2\left(z_j^{(2)}\right)$$

$$z_j^{(3)} = \sum_i w_{ij}^{(3)} y_i^{(2)}$$ $$\boxed{y_j^{(3)} = f_3\left(z_j^{(3)}\right)}$$ $\cdots$

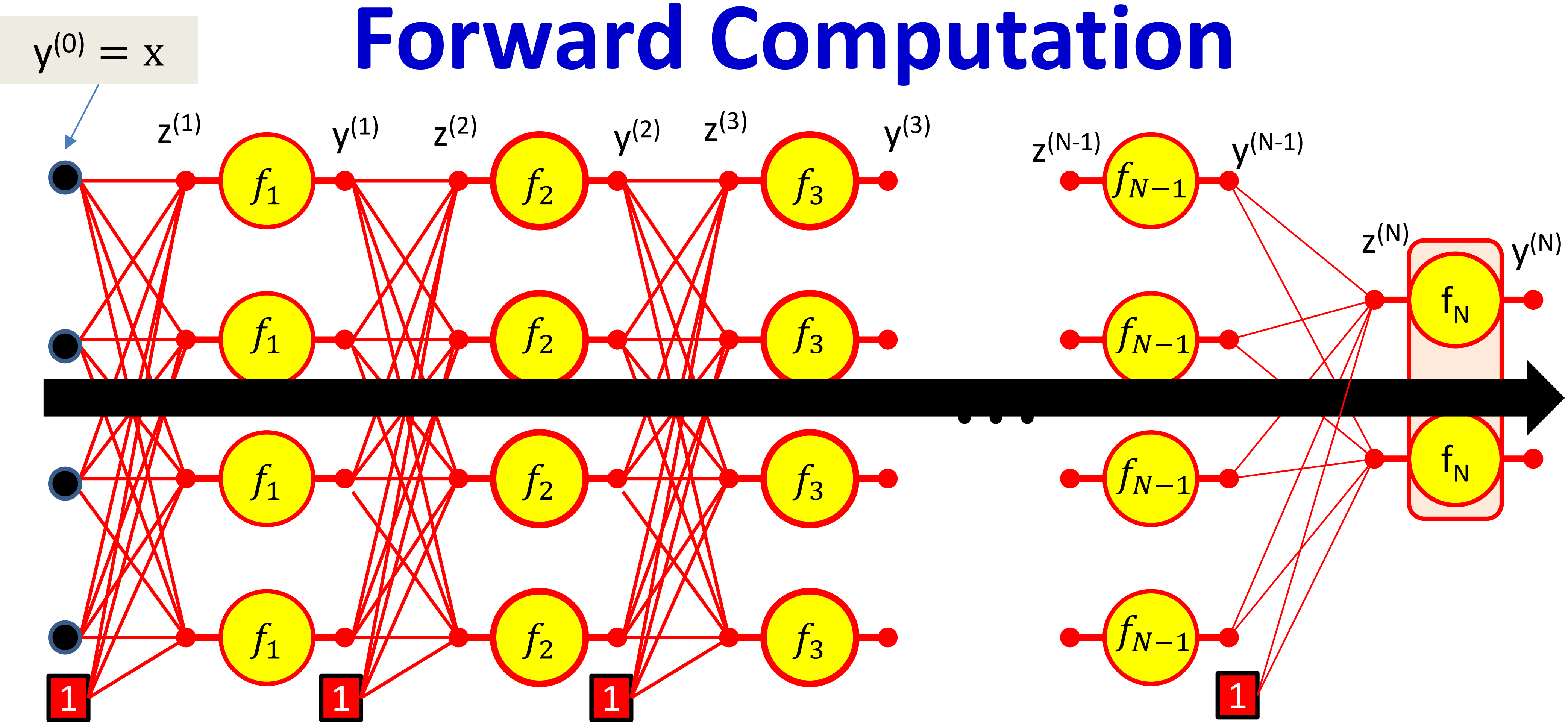


$$y_j^{(N-1)} = f_{N-1}\left(z_j^{(N-1)}\right)$$

$$z_j^{(N)} = \sum_i w_{ij}^{(N)} y_i^{(N-1)}$$

$$\boldsymbol{y}^{(N)} = f_N\left(\boldsymbol{z}^{(N)}\right)$$

Forward Computation



ITERATE FOR k = 1:N

for j = 1:layer-width

$$y_i^{(0)} = x_i$$

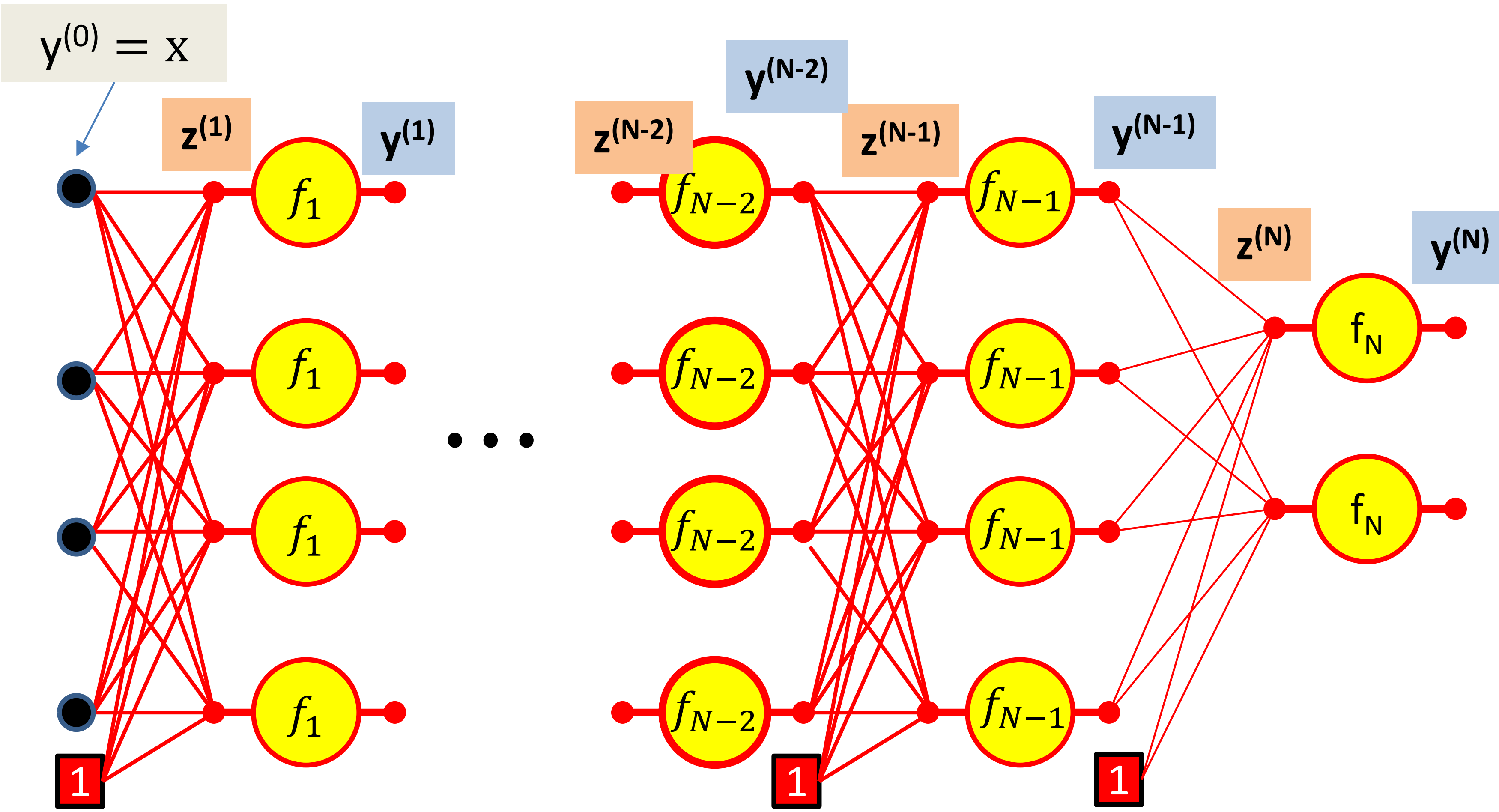
$$z_j^{(k)} = \sum_i w_{ij}^{(k)} y_i^{(k-1)}$$

$$y_j^{(k)} = f_k\left(z_j^{(k)}\right)$$

Forward "Pass"

- Input: D dimensional vector $\mathbf{x} = [x_j,\ j = 1 \dots D]$
- Set:
  - $D_0 = D,$ is the width of the 0th (input) layer
  - $y_j^{(0)} = x_j,\ j = 1 \dots D;$ $y_0^{(k=1\dots N)} = x_0 = 1$
- For layer $k = 1 \dots N$
  - For $j = 1 \dots D_k$ D_k is the size of the kth layer
    - $z_j^{(k)} = \sum_{i=0}^{D_{k-1}} w_{i,j}^{(k)} y_i^{(k-1)}$
    - $y_j^{(k)} = f_k\left(z_j^{(k)}\right)$
- Output:
  - $Y = y_j^{(N)}, j = 1..D_N$

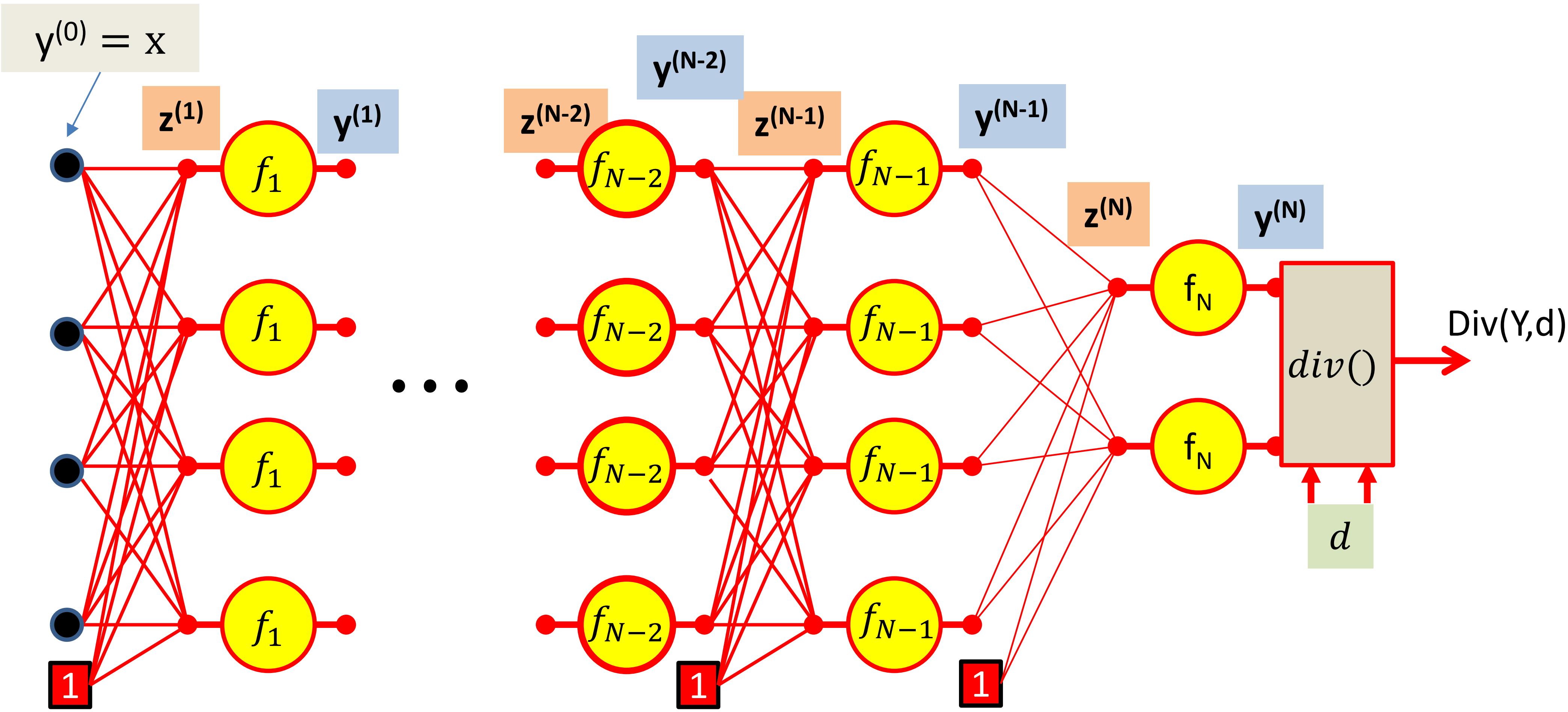
Computing derivatives



We have computed all these intermediate values in the forward computation

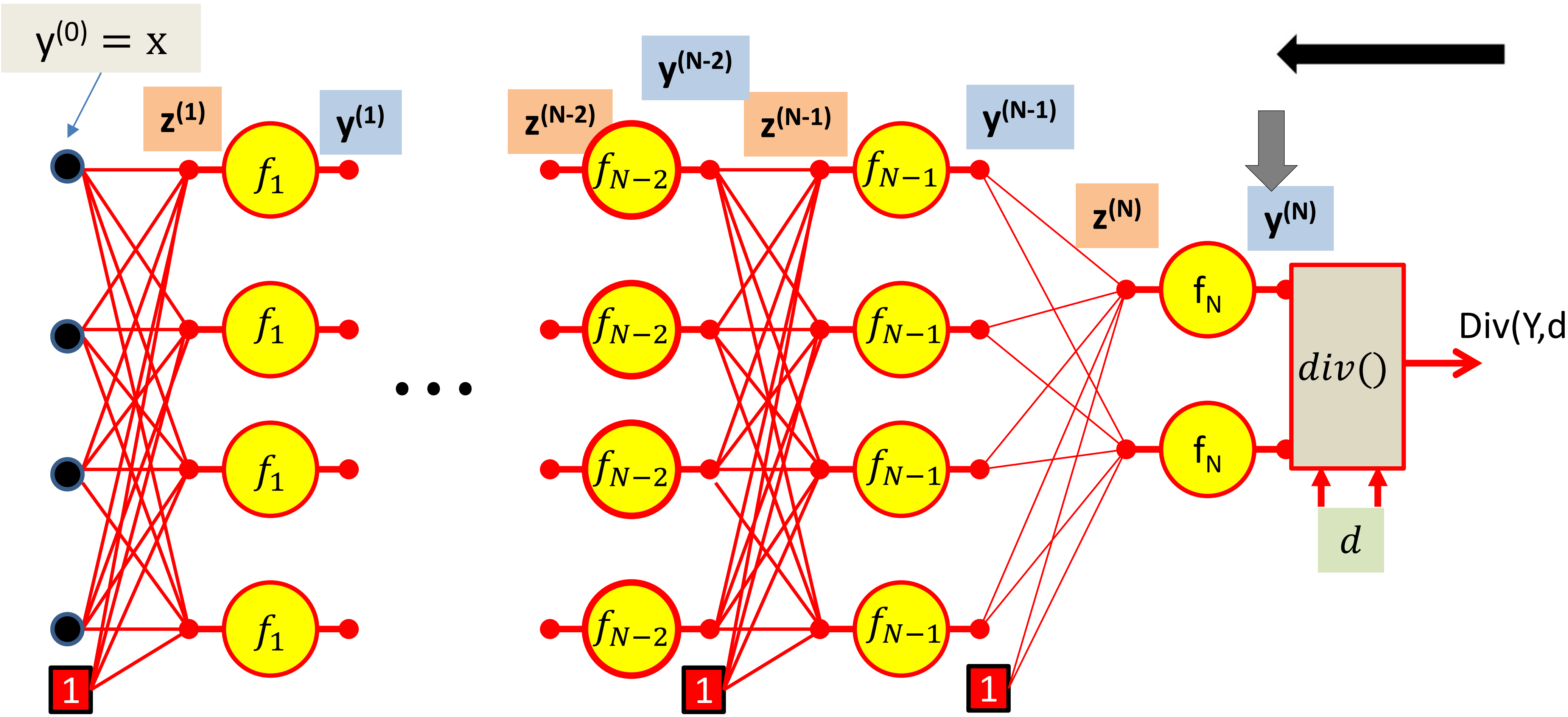
We must remember them – we will need them to compute the derivatives

Computing derivatives



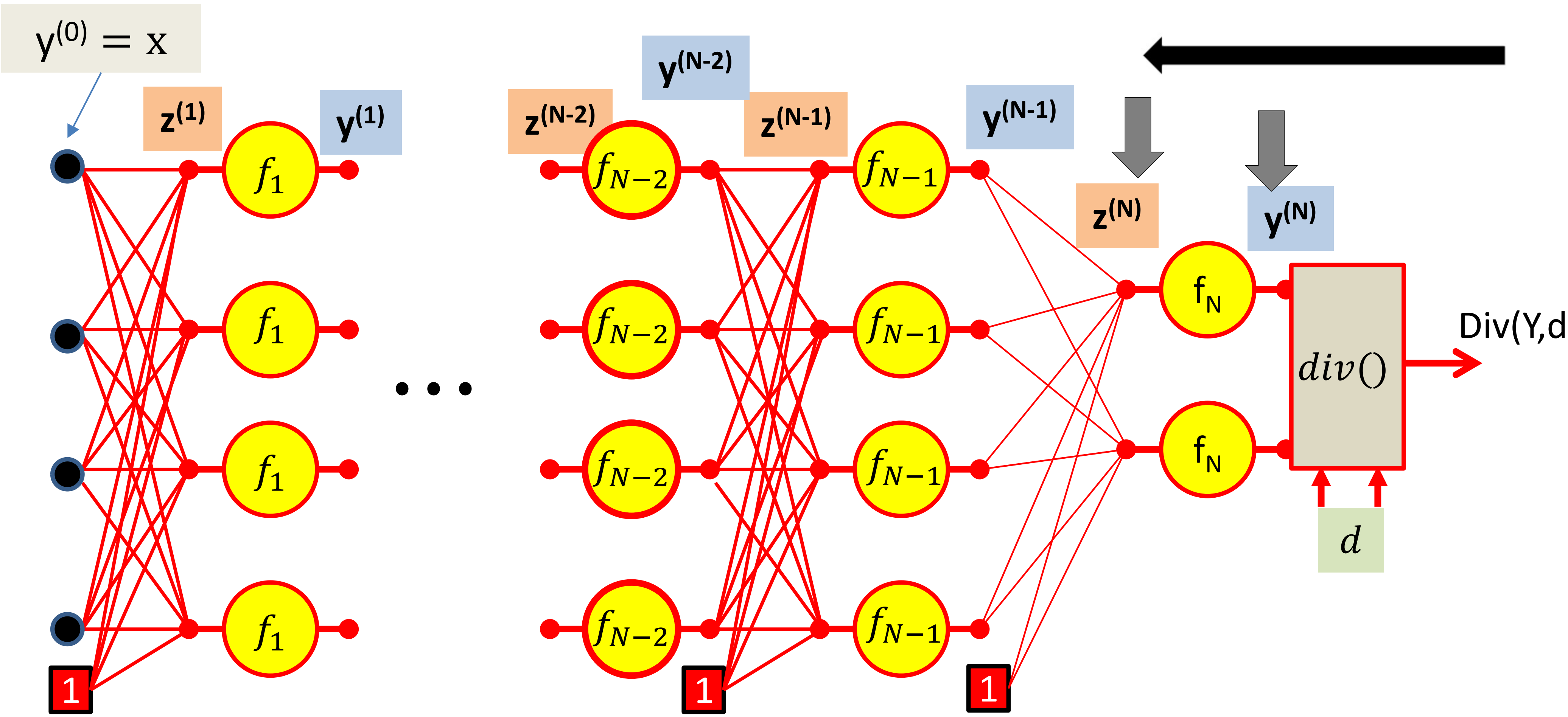
First, we compute the divergence between the output of the net y = $y^{(N)}$ and the desired output d

Computing derivatives



We then compute $\nabla_{Y^{(N)}} div(.)$ the derivative of the divergence w.r.t. the final output of the network $y^{(N)}$

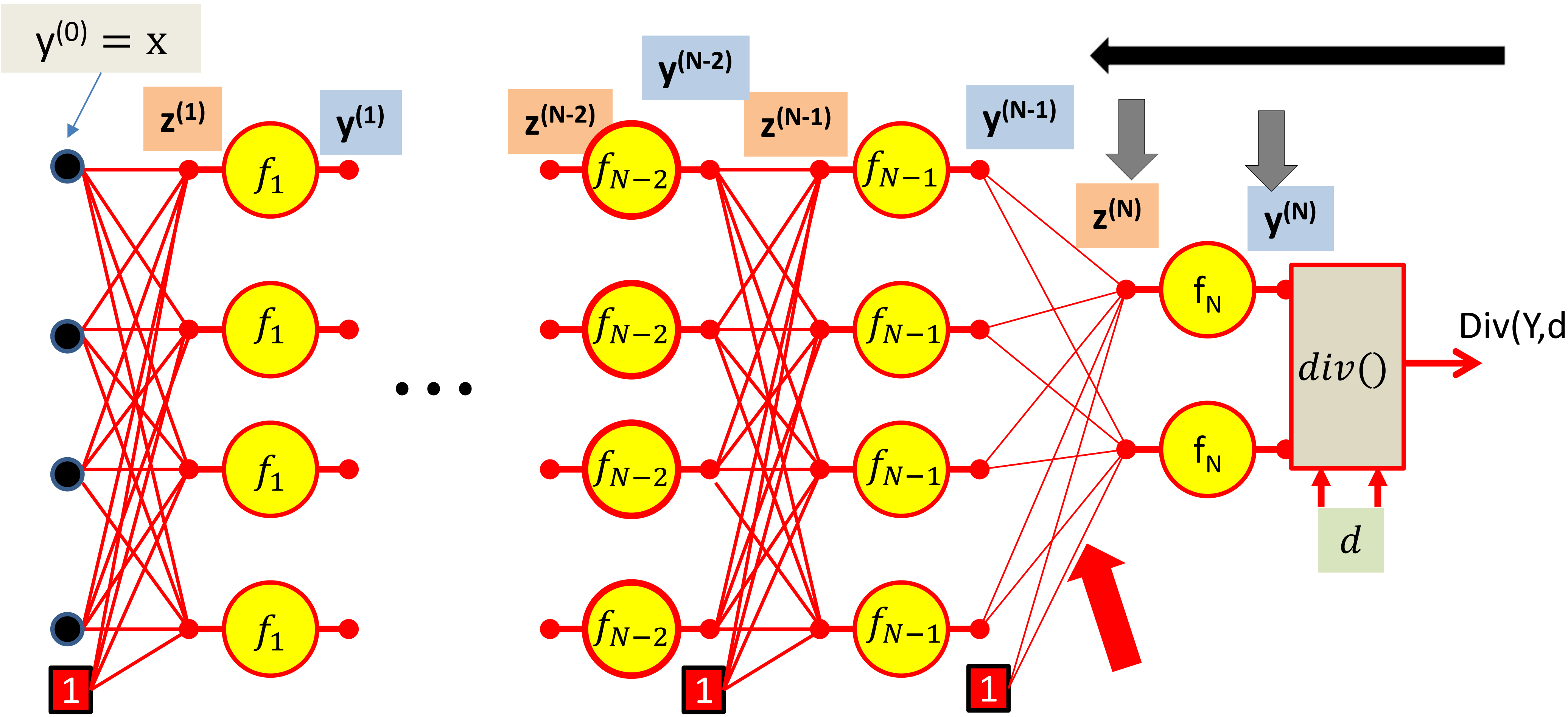
Computing derivatives



We then compute $\nabla_{Y^{(N)}} div(.)$ the derivative of the divergence w.r.t. the final output of the network $y^{(N)}$

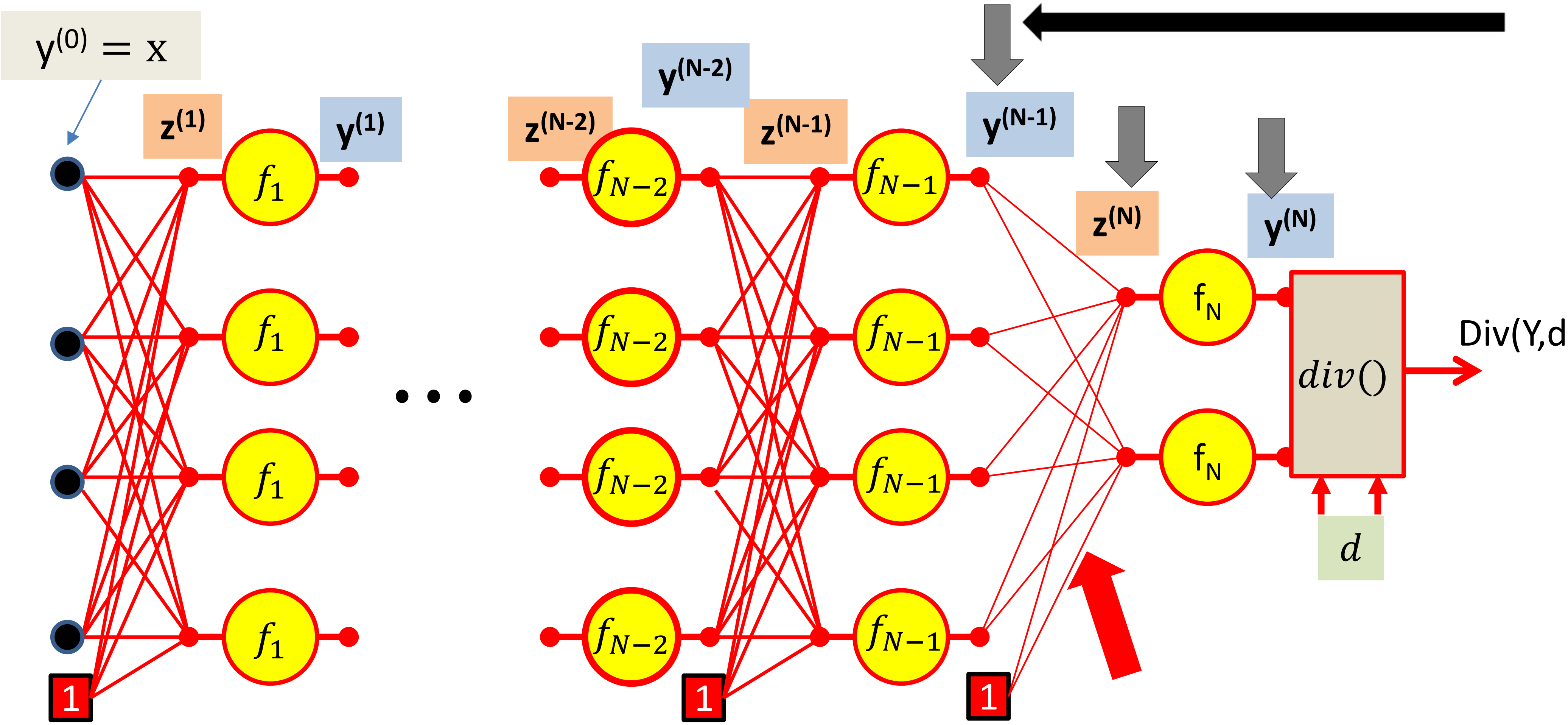
We then compute $\nabla_{z^{(N)}} div(.)$ the derivative of the divergence w.r.t. the *pre-activation* affine combination $z^{(N)}$ using the chain rule

Computing derivatives



Continuing on, we will compute $\nabla_{W^{(N)}} div(.)$ the derivative of the divergence with respect to the weights of the connections to the output layer

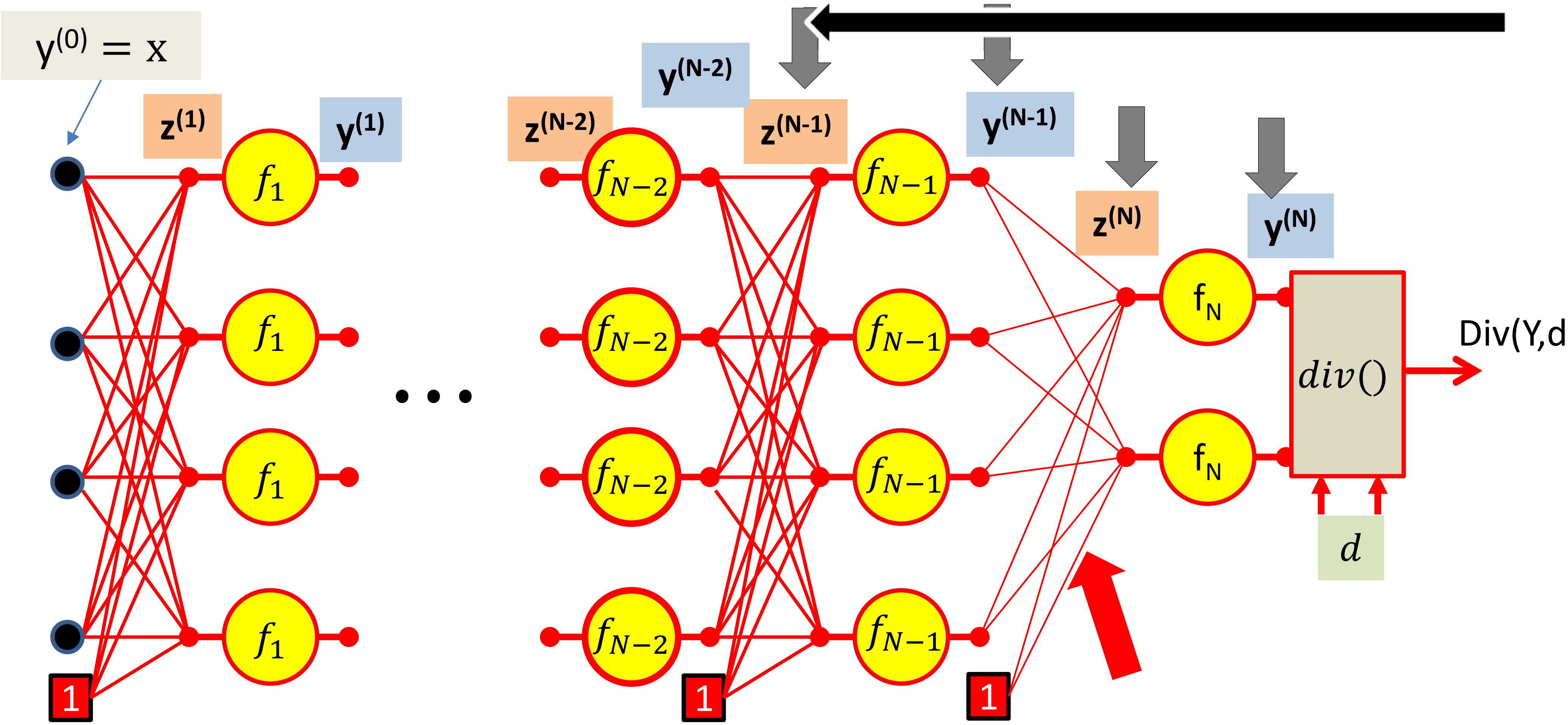
Computing derivatives



Continuing on, we will compute $\nabla_{W^{(N)}} div(.)$ the derivative of the divergence with respect to the weights of the connections to the output layer

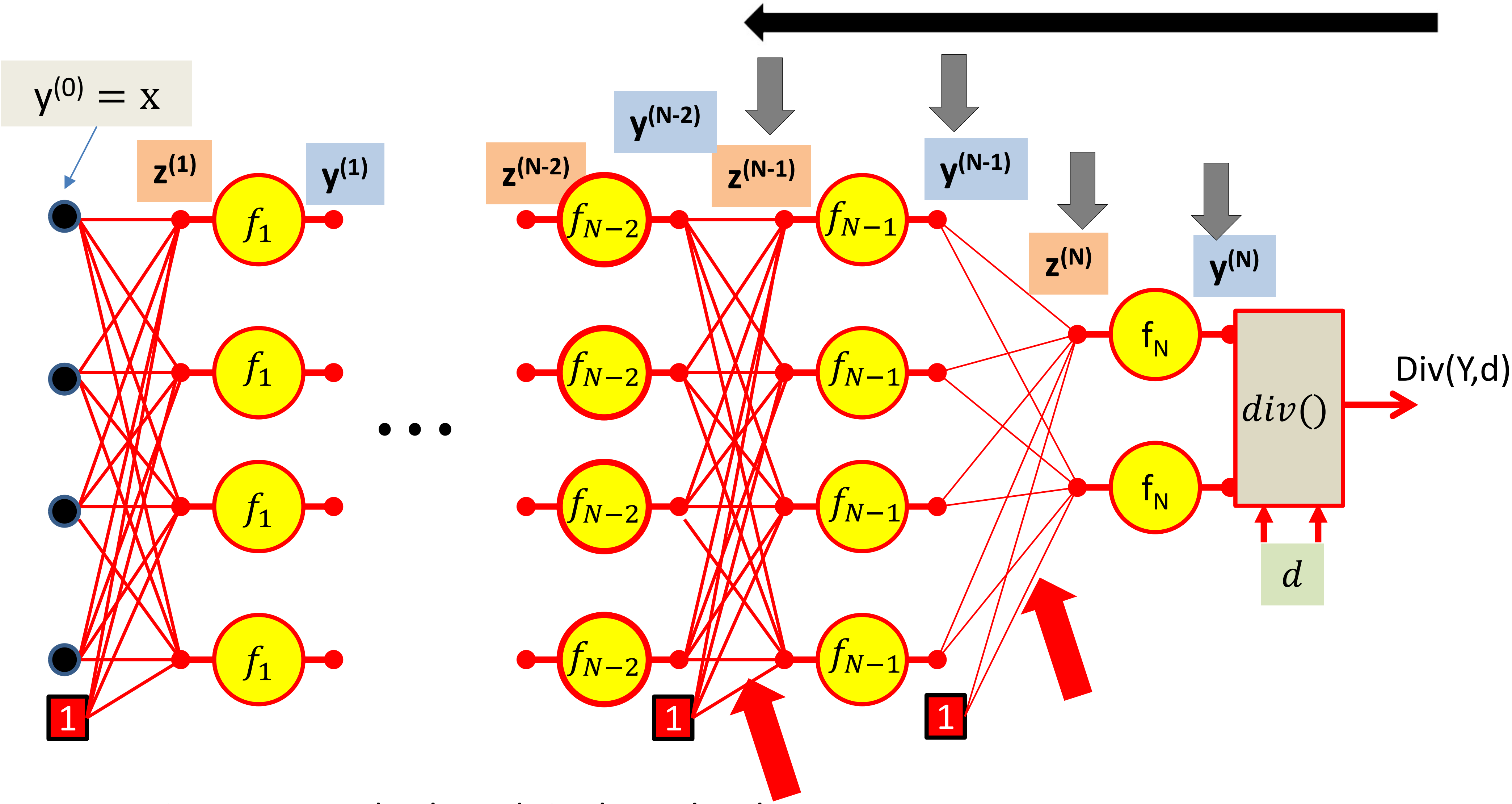
Then continue with the chain rule to compute $\nabla_{Y^{(N-1)}} div(.)$ the derivative of the divergence w.r.t. the output of the N-1th layer

Computing derivatives



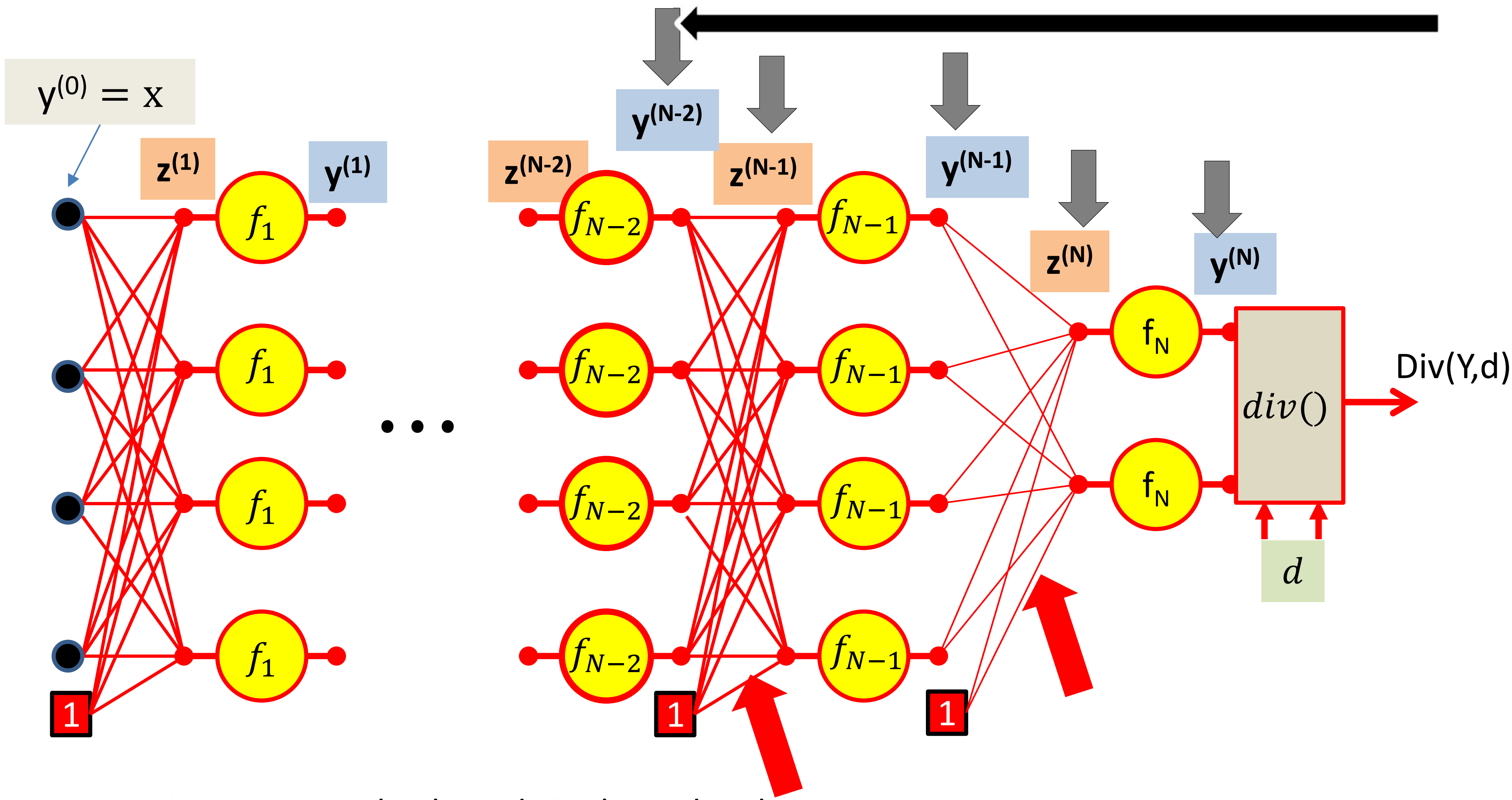
We continue our way backwards in the order shown

$$\nabla_{z^{(N-1)}} div(.)$$



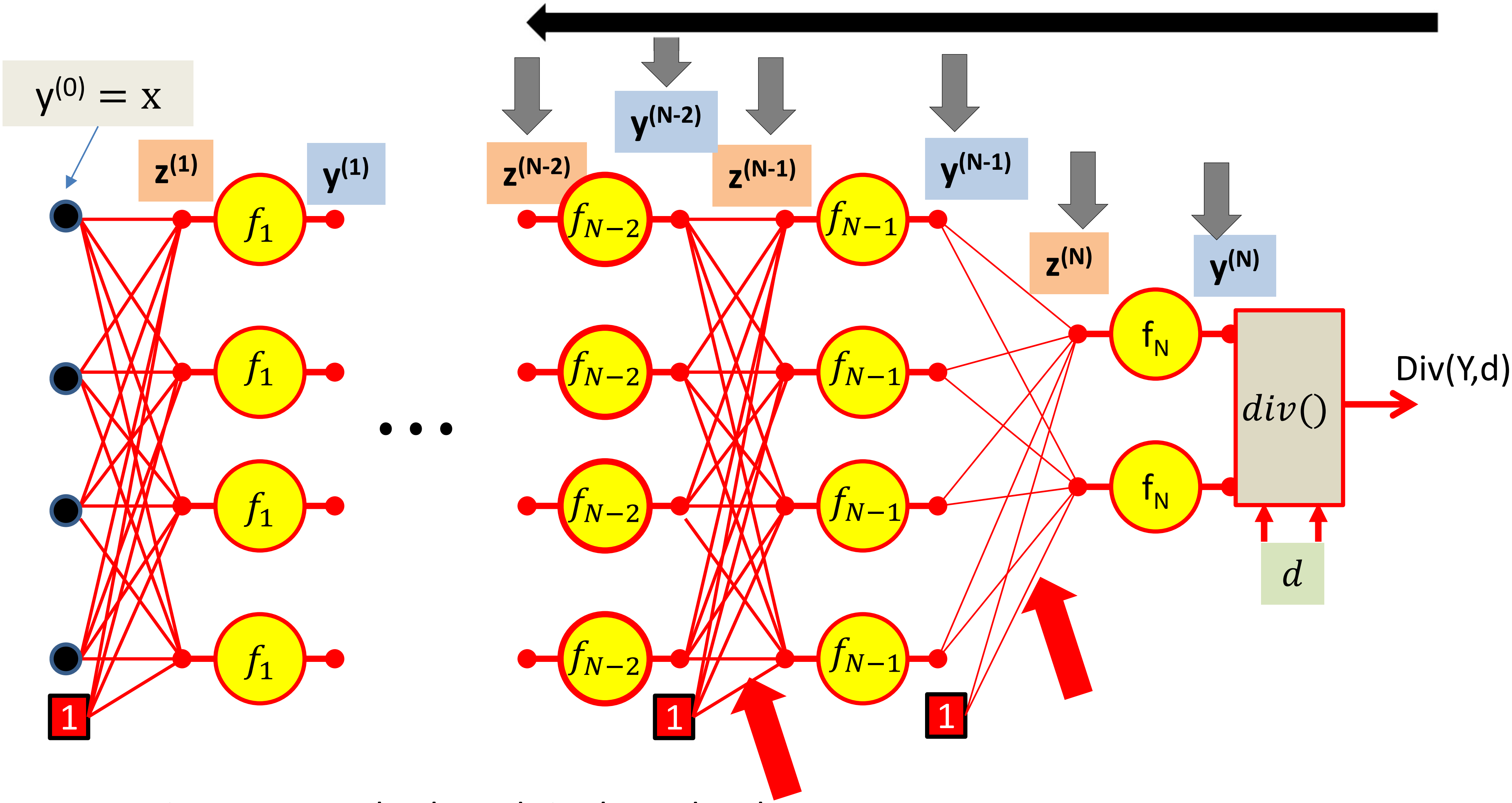
We continue our way backwards in the order shown

$$\nabla_{W^{(N-1)}} div(.)$$



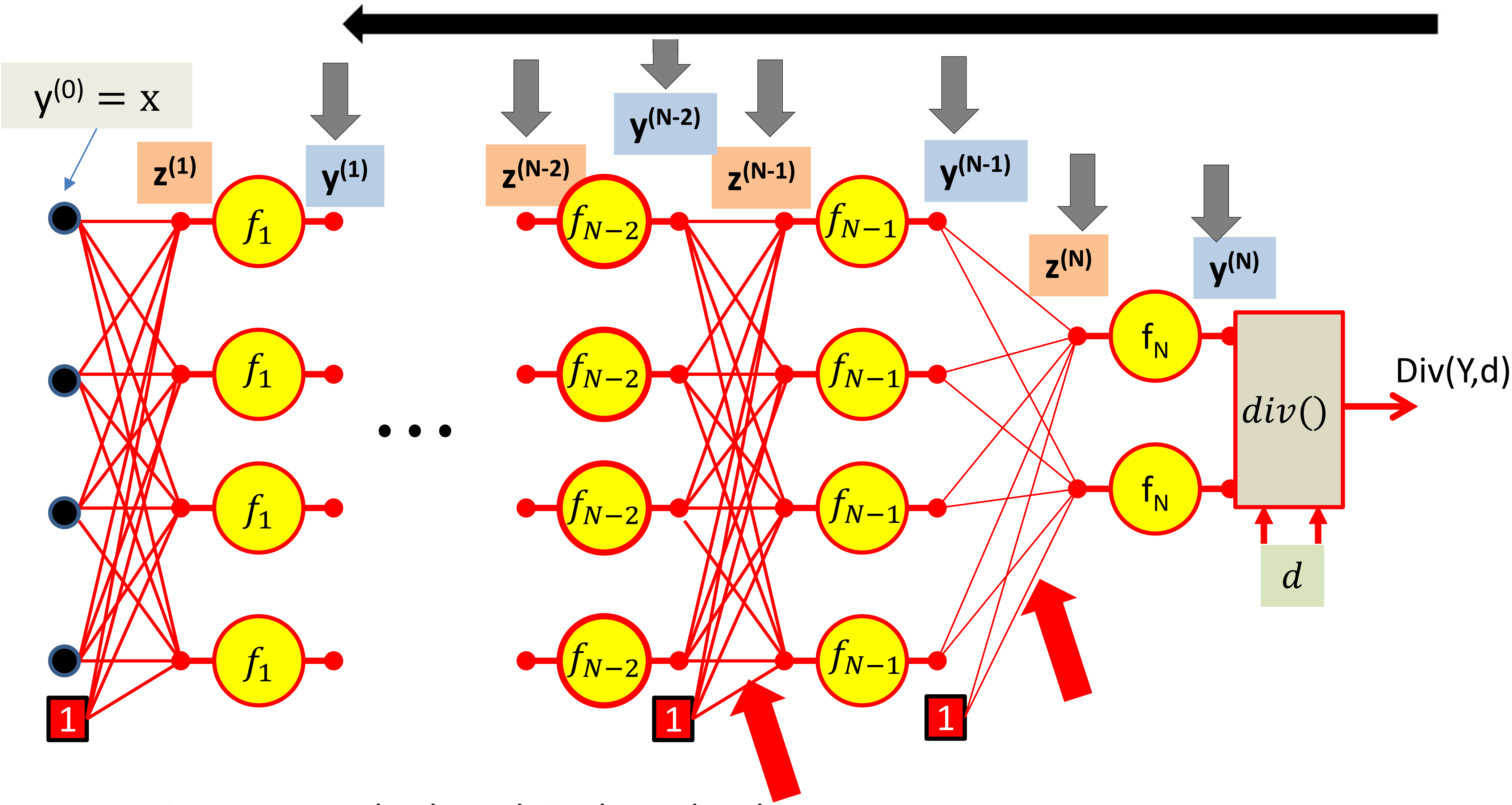
We continue our way backwards in the order shown

$$\nabla_{Y^{(N-2)}} div(.)$$



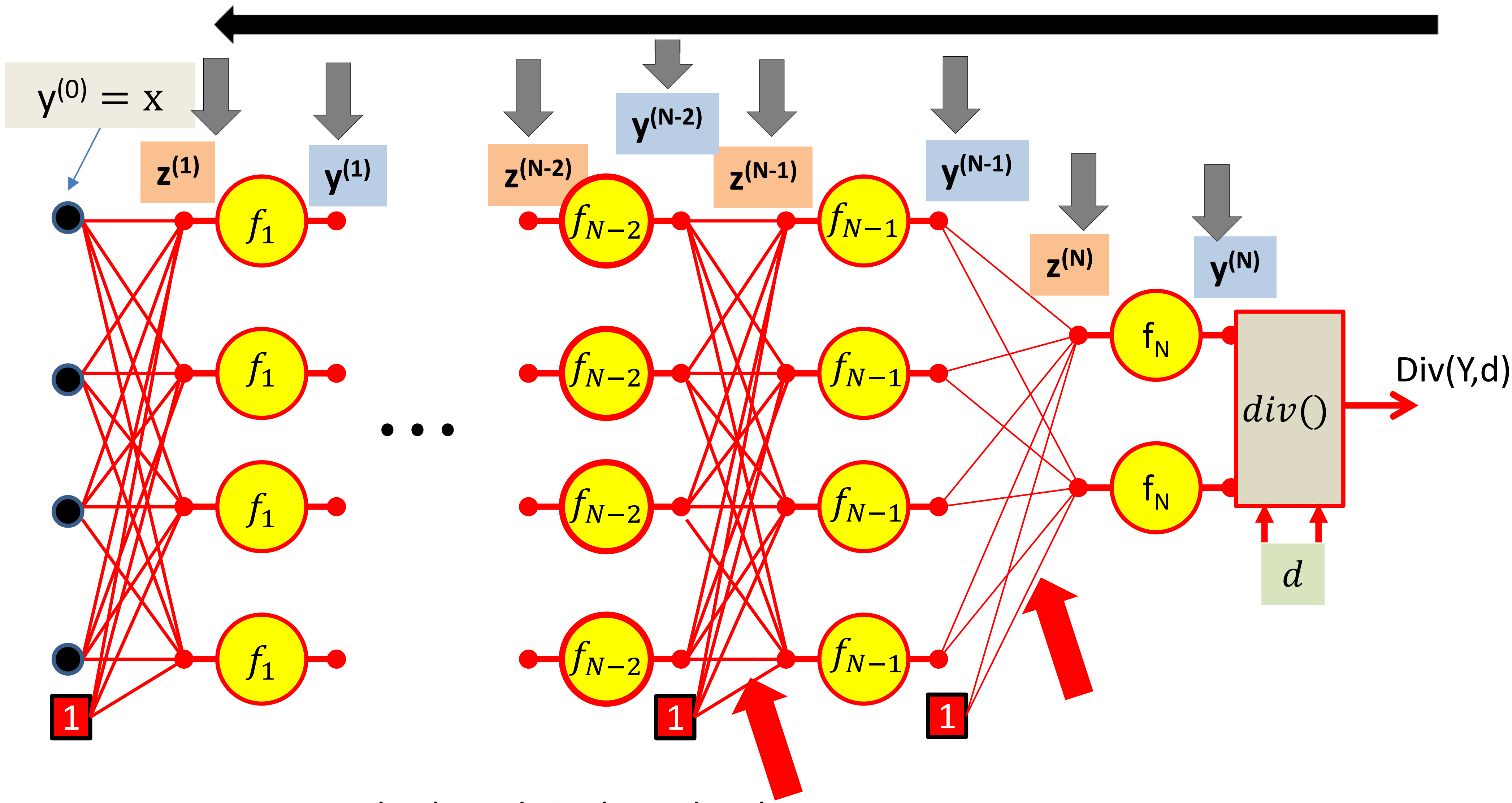
We continue our way backwards in the order shown

$$\nabla_{z^{(N-2)}} div(.)$$



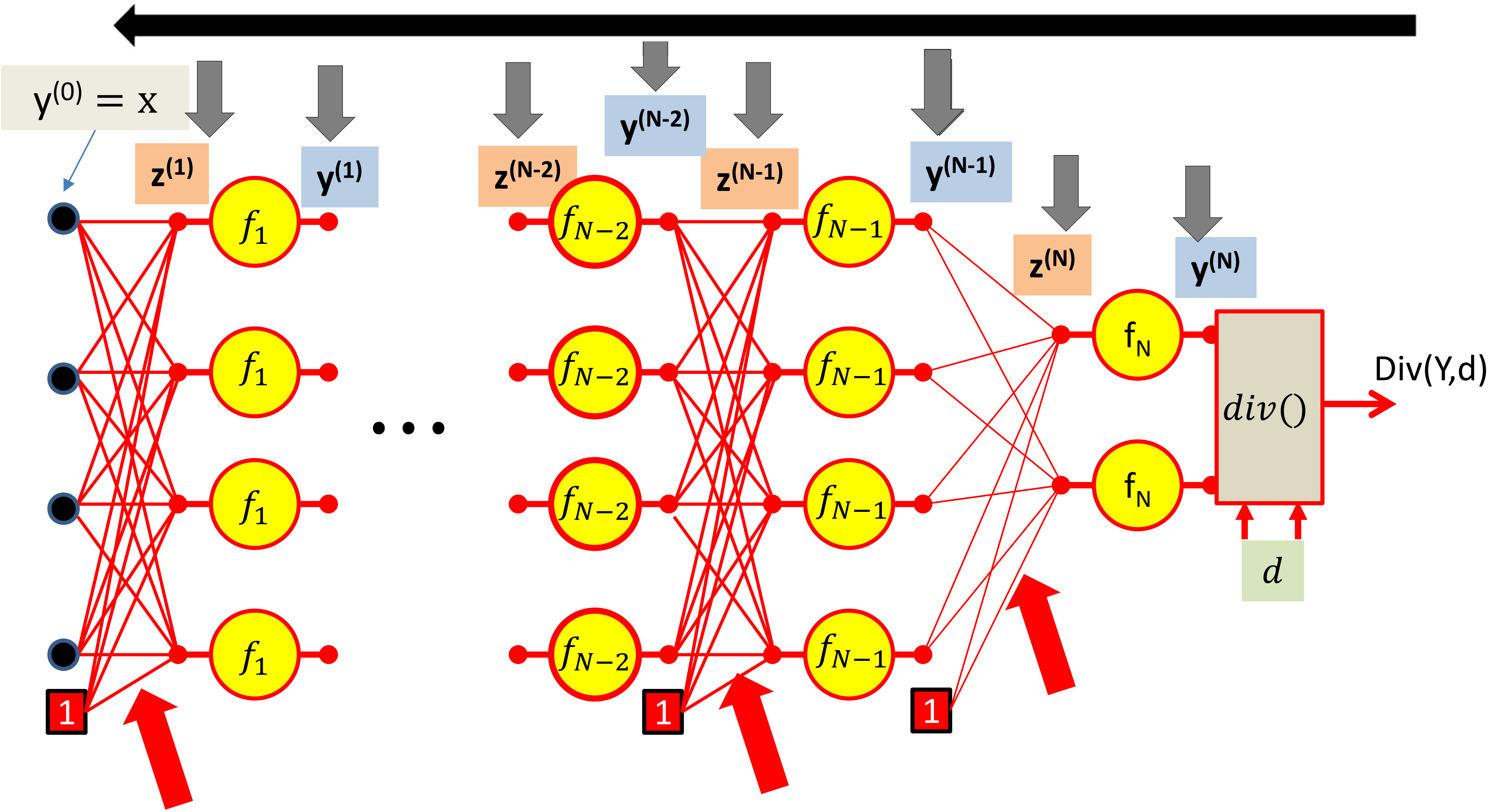
We continue our way backwards in the order shown

$$\nabla_{Y^{(1)}} div(.)$$



We continue our way backwards in the order shown

$$\nabla_{z^{(1)}} div(.)$$



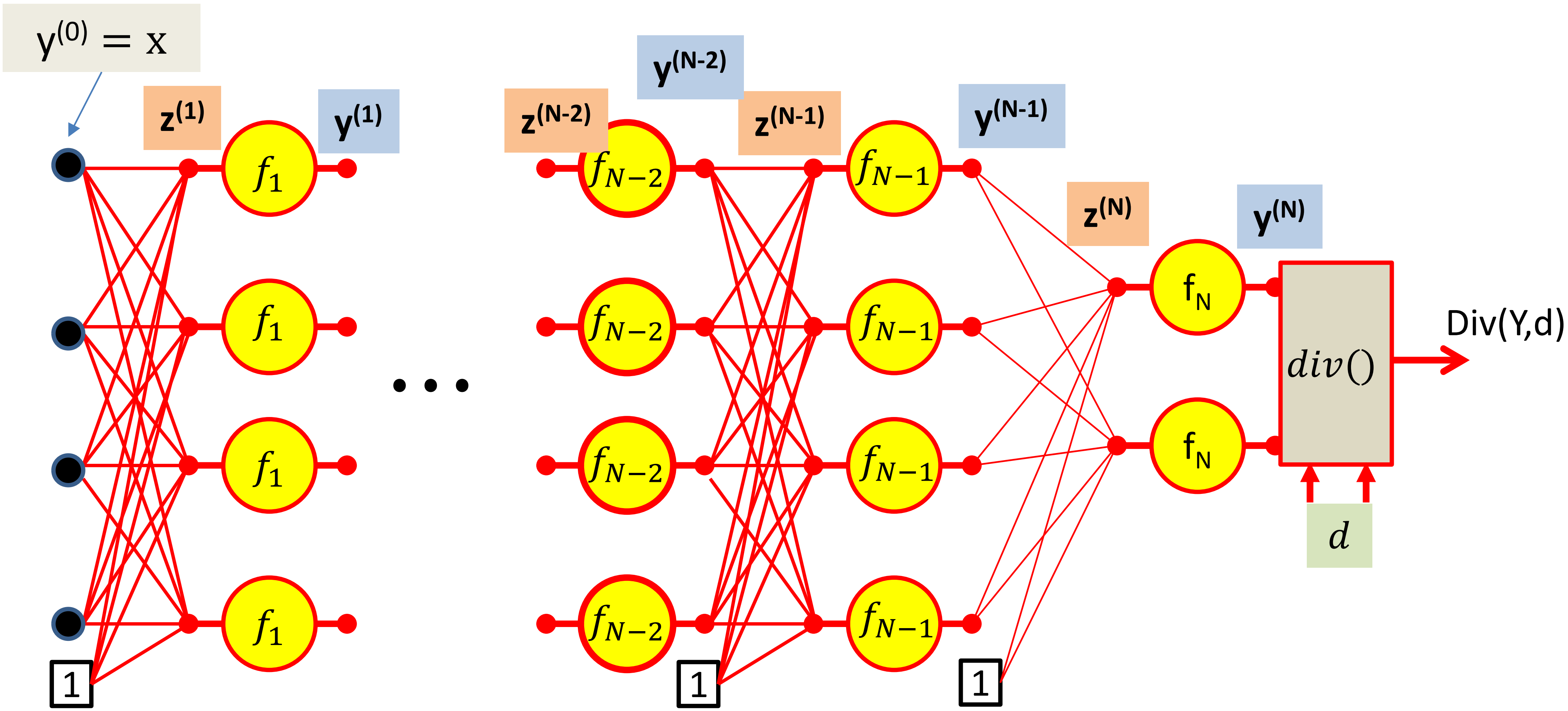
We continue our way backwards in the order shown

$$\nabla_{W^{(1)}} div(.)$$

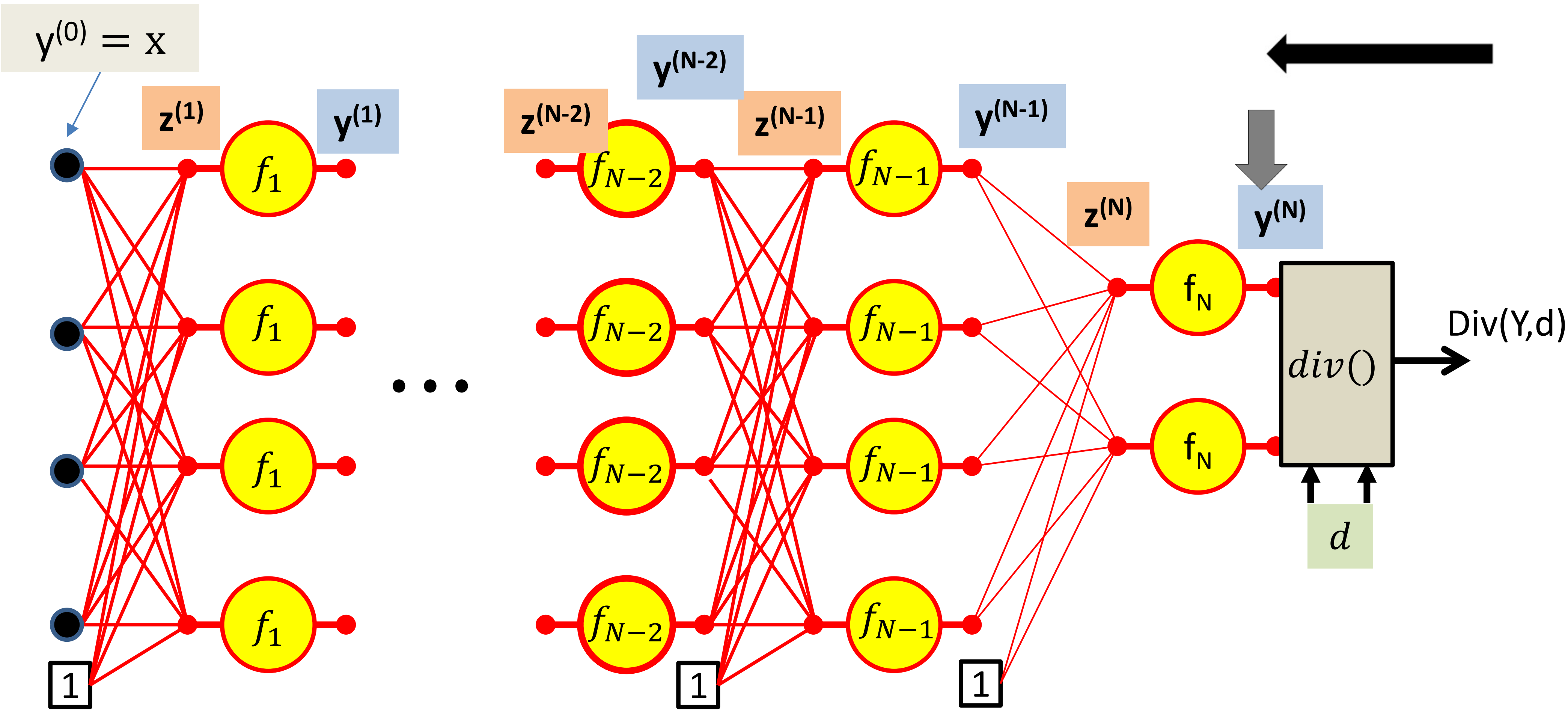
Backward Gradient Computation

- Let's actually see the math..

Computing derivatives



Computing derivatives



The derivative w.r.t the actual output of the final layer of the network is simply the derivative w.r.t to the output of the network

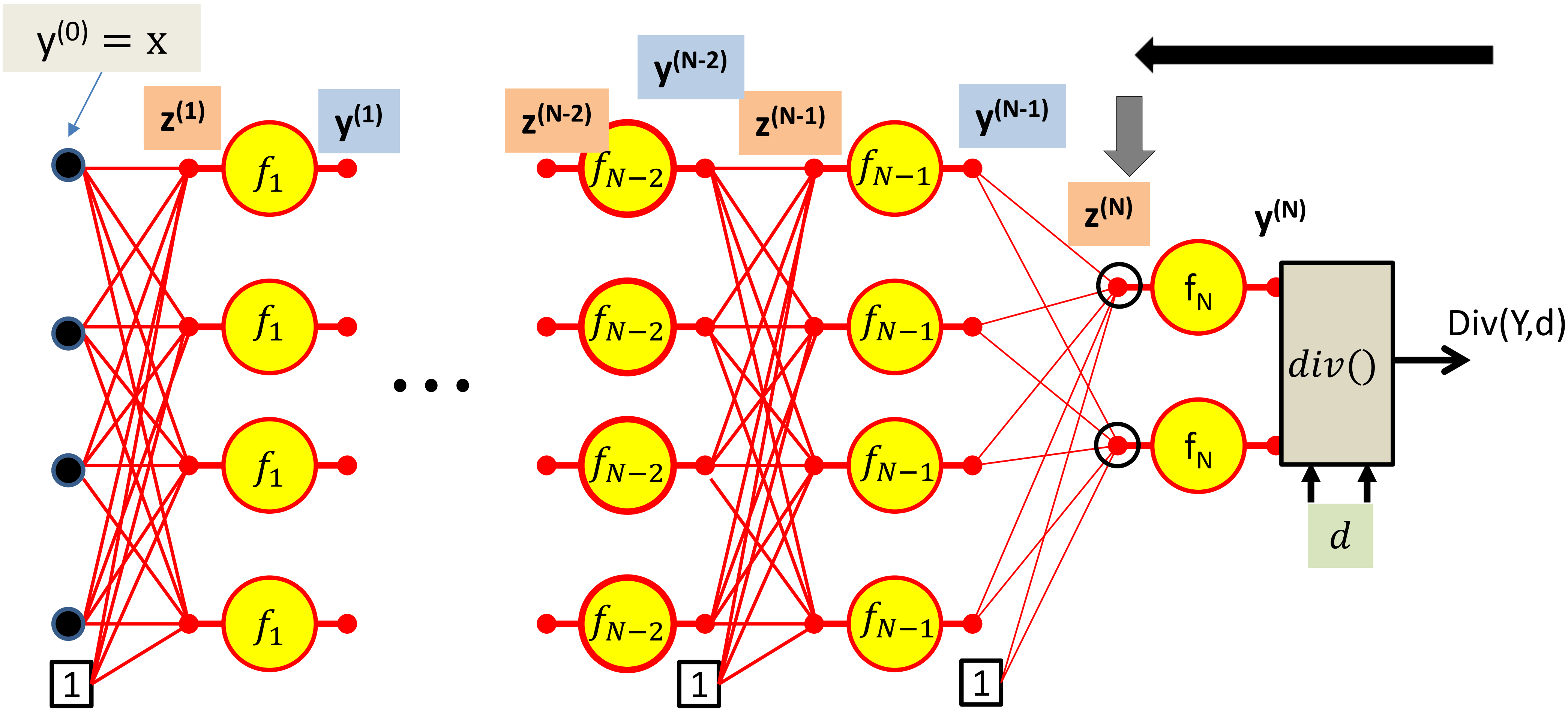
$$\frac{\partial Div(Y,d)}{\partial y_i^{(N)}} = \frac{\partial Div(Y,d)}{\partial y_i}$$

Calculus Refresher: Chain rule

For any nested function $l = f(y)$ where $y = g(z)$

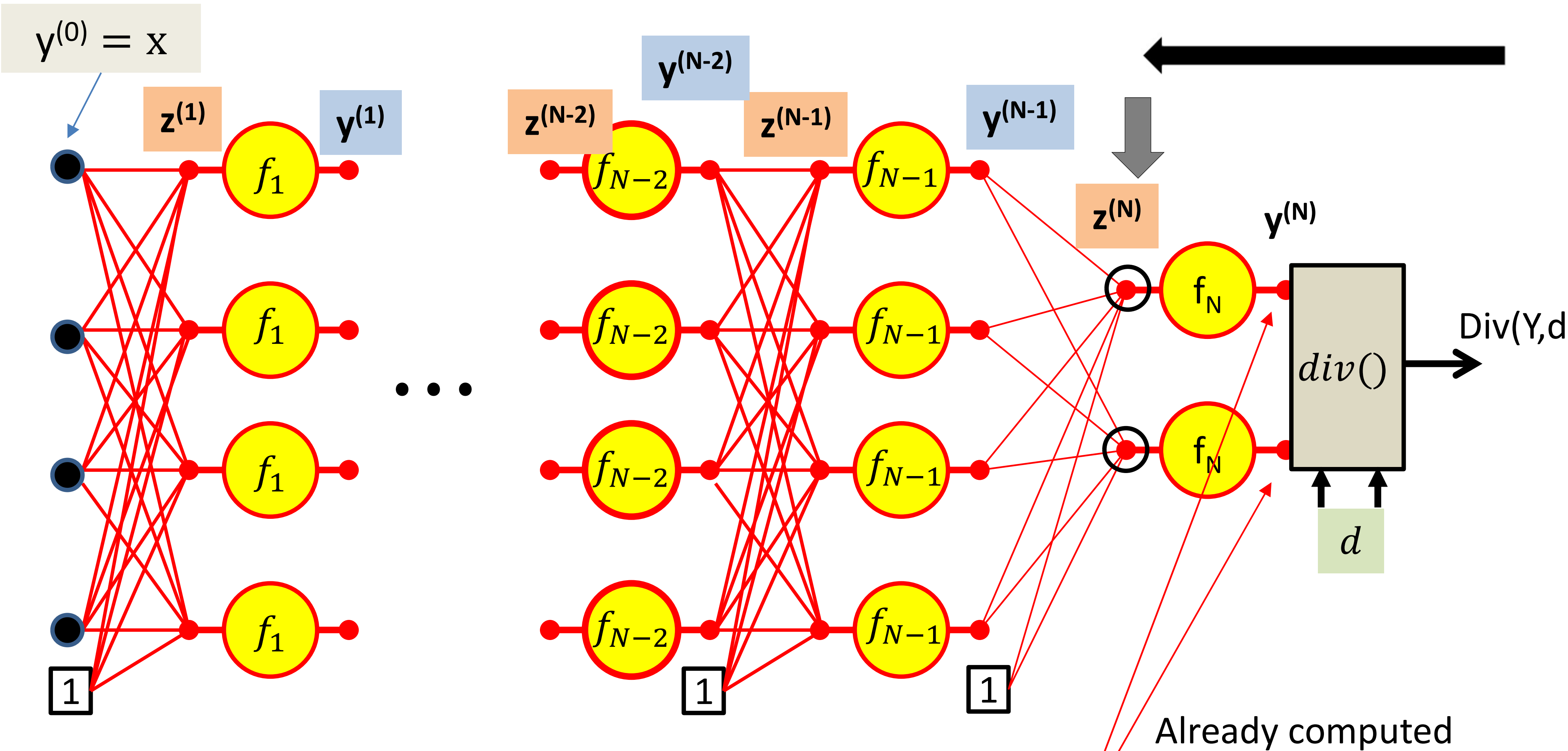
$$\boxed{\frac{dl}{dz} = \frac{dl}{dy}\frac{dy}{dz}}$$

Computing derivatives



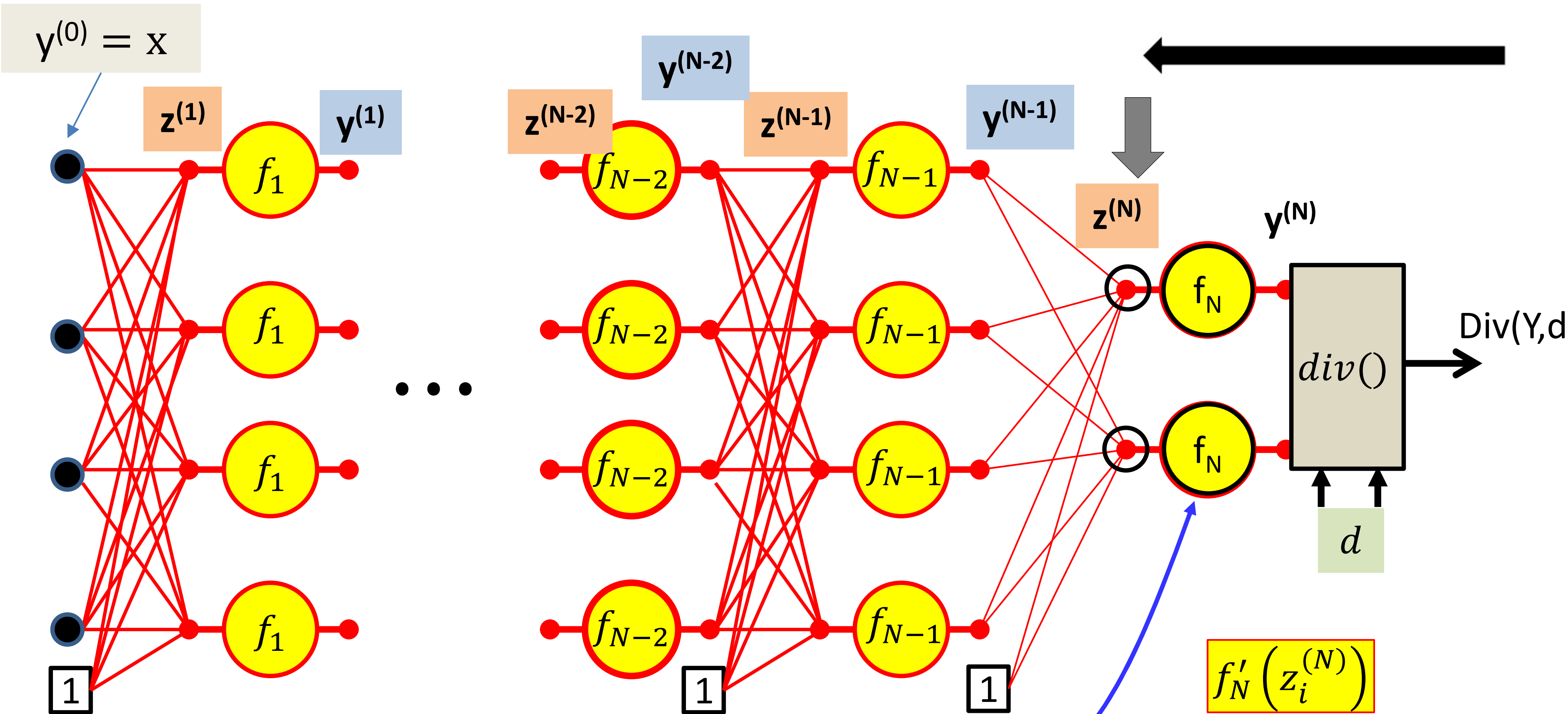
$$\frac{\partial Div}{\partial z_i^{(N)}} = \frac{\partial y_i^{(N)}}{\partial z_i^{(N)}} \frac{\partial Div}{\partial y_i^{(N)}}$$

Computing derivatives



$$\frac{\partial Div}{\partial z_i^{(N)}} = \frac{\partial y_i^{(N)}}{\partial z_i^{(N)}} \frac{\partial Div}{\partial y_i^{(N)}}$$

Computing derivatives

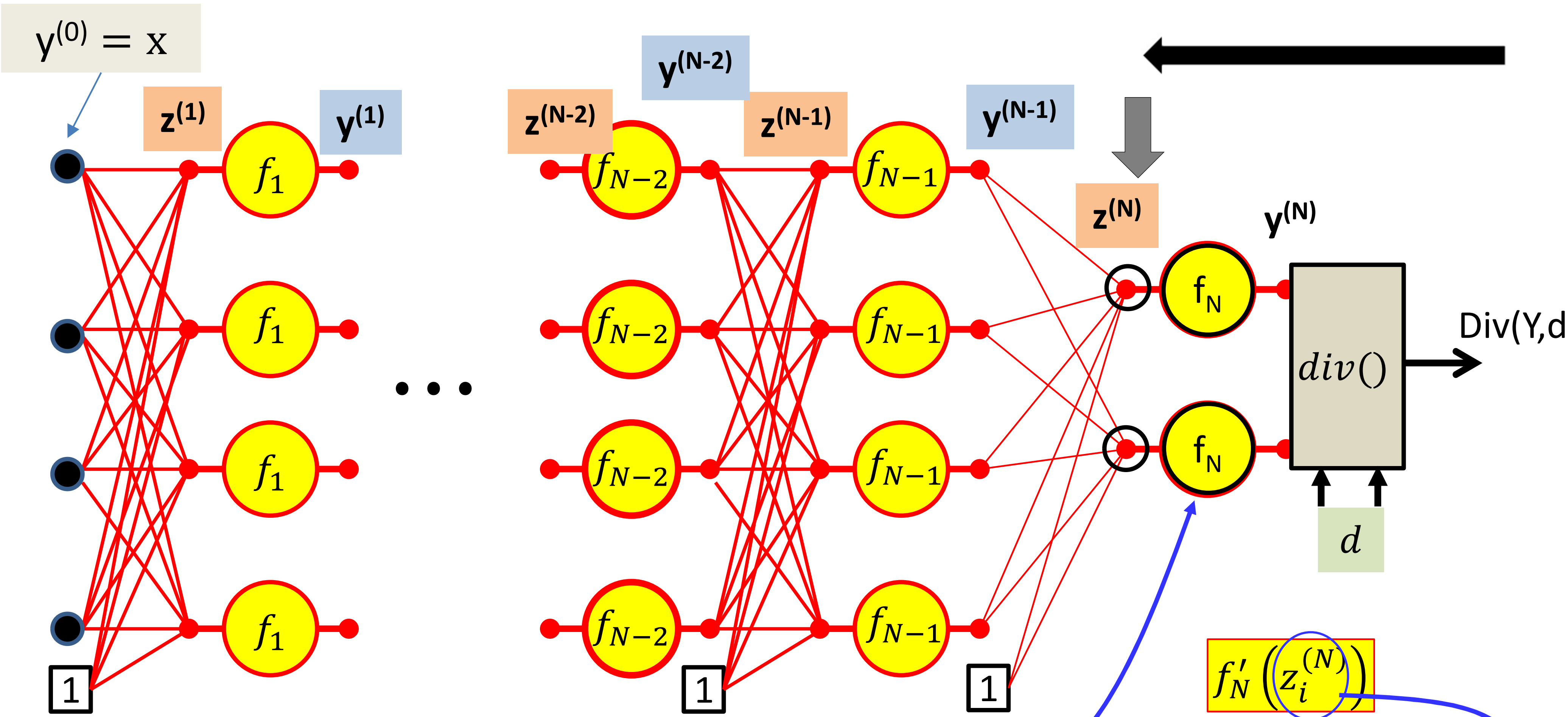


$$f_N'\left(z_i^{(N)}\right)$$

Derivative of activation function

$$\frac{\partial Div}{\partial z_i^{(N)}} = \frac{\partial y_i^{(N)}}{\partial z_i^{(N)}} \frac{\partial Div}{\partial y_i^{(N)}}$$

Computing derivatives



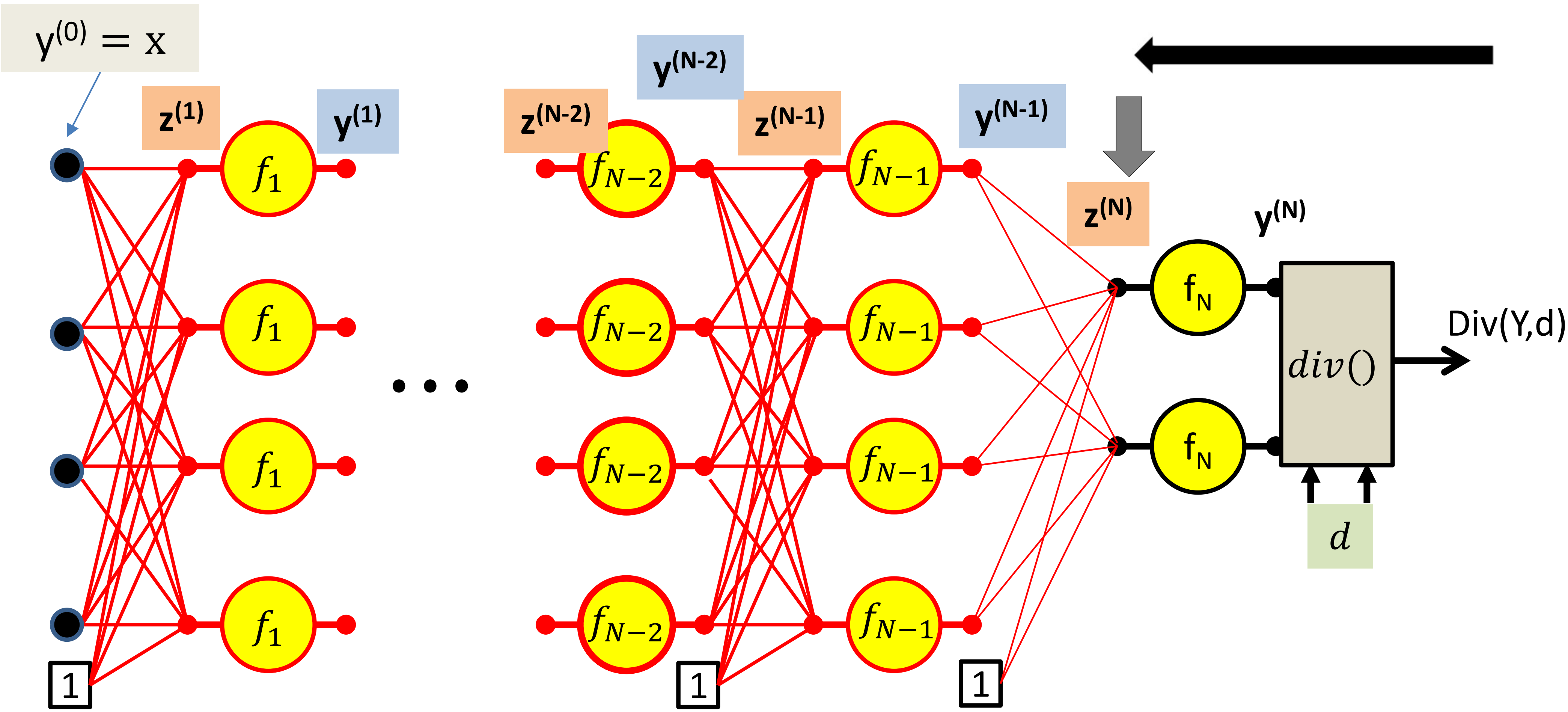
$f_N'\left(z_i^{(N)}\right)$

Derivative of activation function

$$\frac{\partial Div}{\partial z_i^{(N)}} = \frac{\partial y_i^{(N)}}{\partial z_i^{(N)}} \frac{\partial Div}{\partial y_i^{(N)}}$$

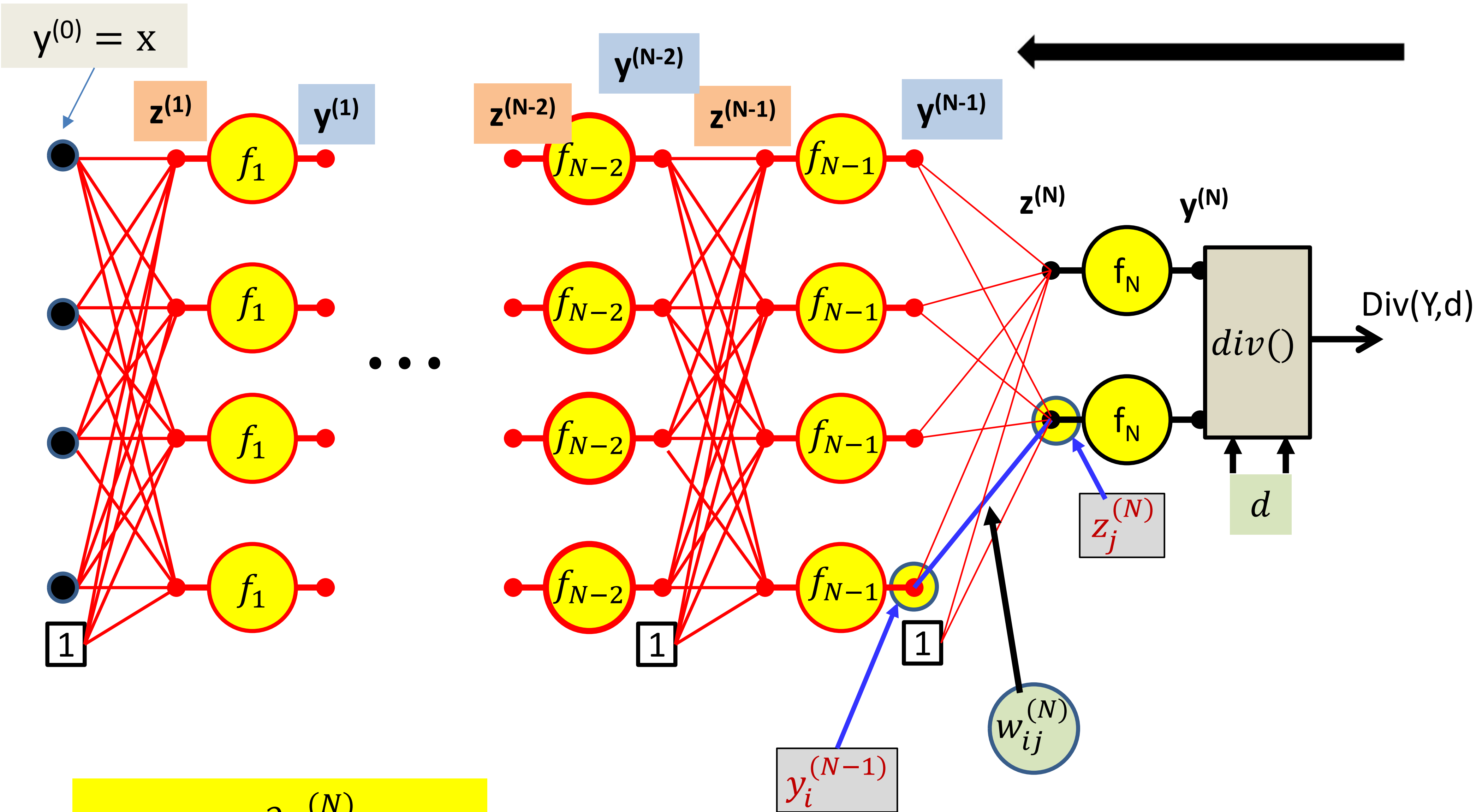
Computed in forward pass

Computing derivatives



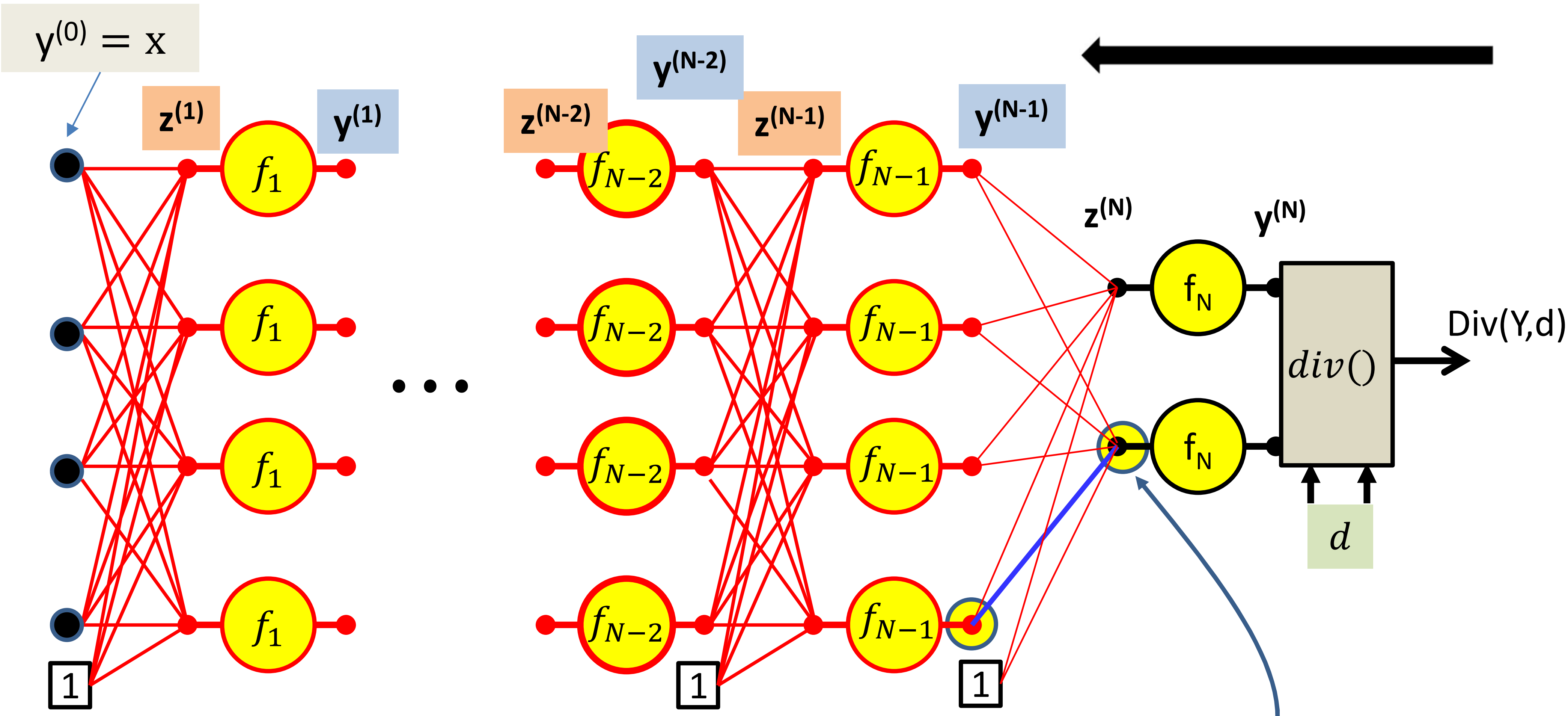
$$\frac{\partial Div}{\partial z_i^{(N)}} = f_N'\left(z_i^{(N)}\right)\frac{\partial Div}{\partial y_i^{(N)}}$$

Computing derivatives



$$\frac{\partial Div}{\partial w_{ij}^{(N)}} = \frac{\partial z_j^{(N)}}{\partial w_{ij}^{(N)}} \frac{\partial Div}{\partial z_j^{(N)}}$$

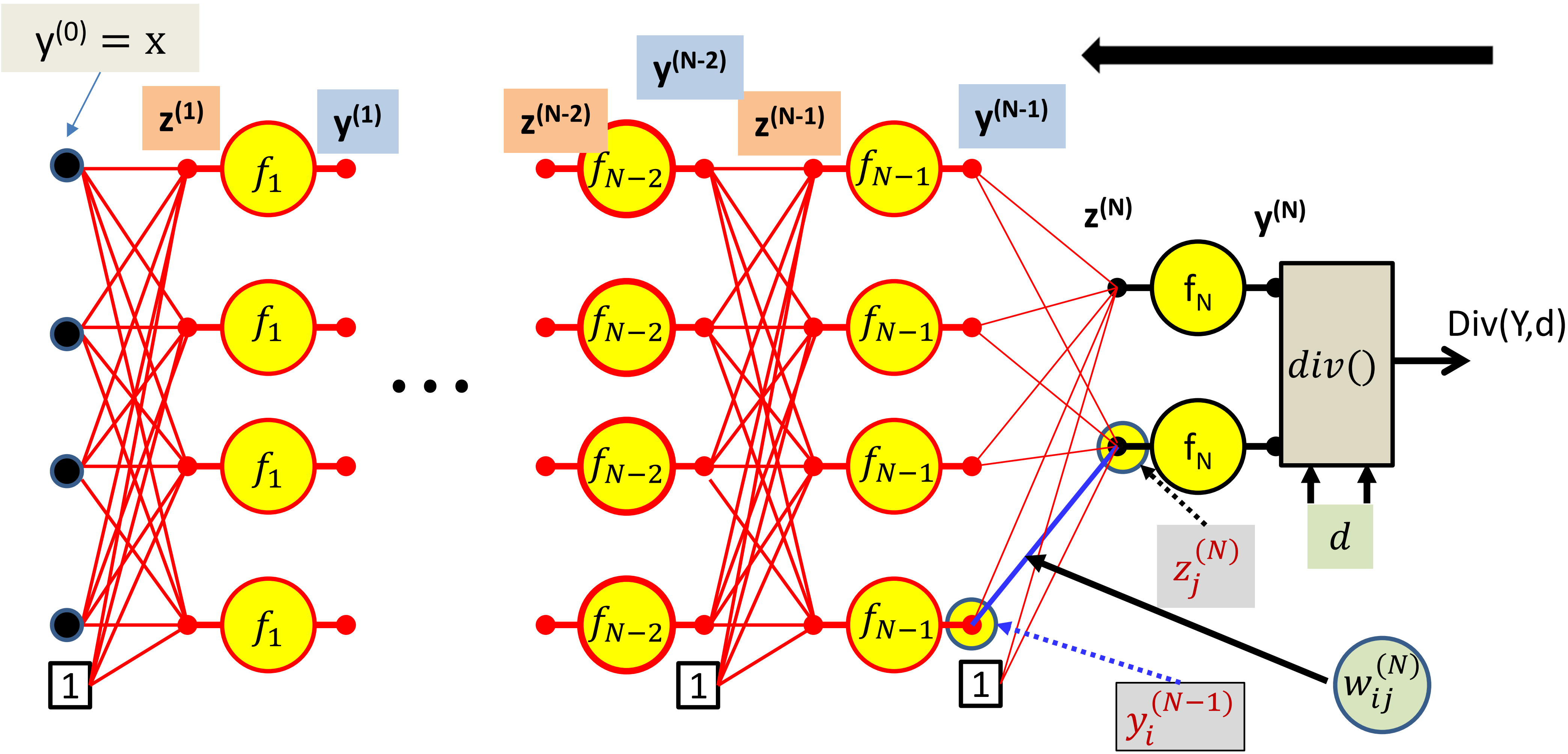
Computing derivatives



$$\frac{\partial Div}{\partial w_{ij}^{(N)}} = \frac{\partial z_j^{(N)}}{\partial w_{ij}^{(N)}} \frac{\partial Div}{\partial z_j^{(N)}}$$

Just computed

Computing derivatives



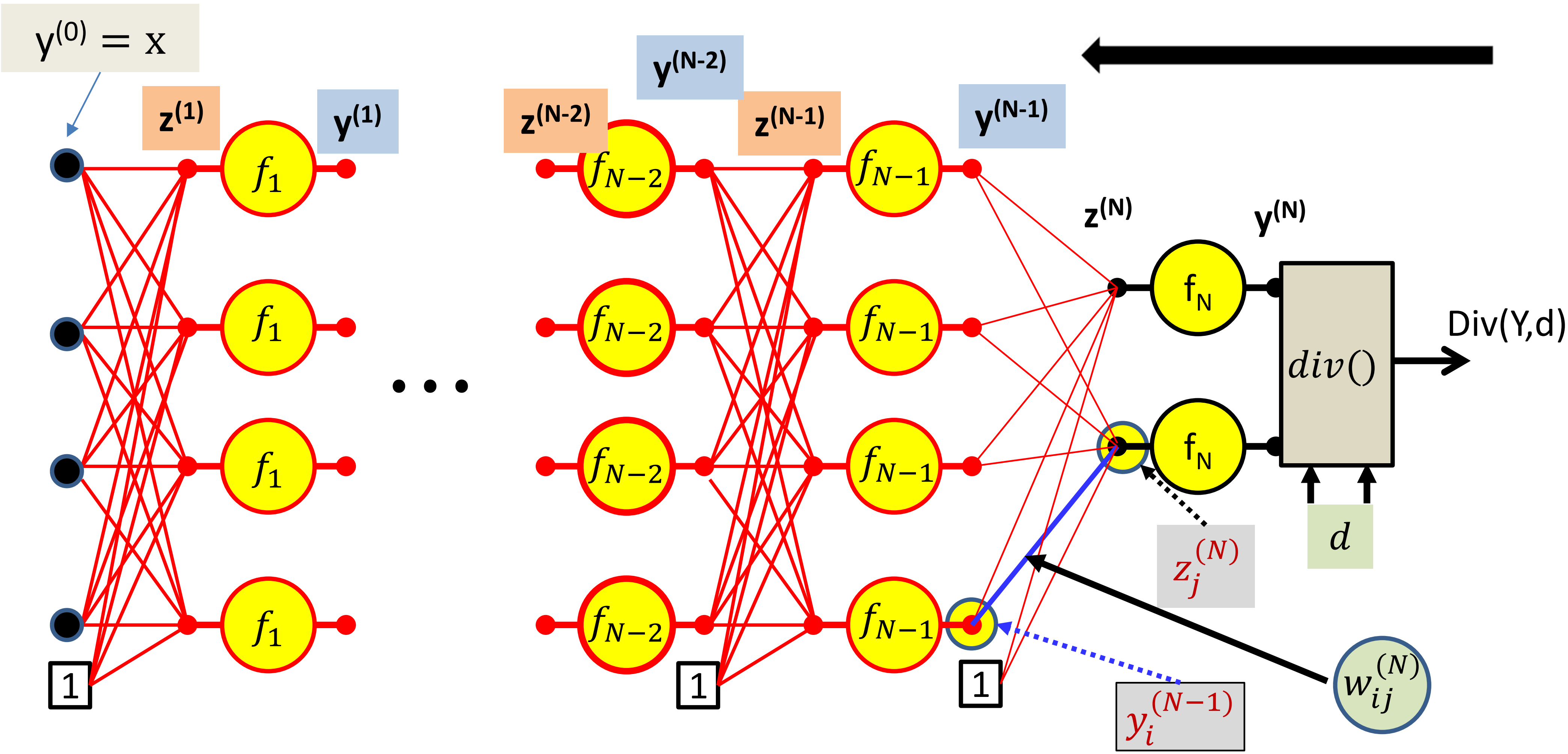
$$\frac{\partial Div}{\partial w_{ij}^{(N)}} = \frac{\partial z_j^{(N)}}{\partial w_{ij}^{(N)}} \frac{\partial Div}{\partial z_j^{(N)}}$$

$y_i^{(N-1)}$

Because

$$z_j^{(N)} = w_{ij}^{(N)} y_i^{(N-1)} + \text{other terms}$$

Computing derivatives



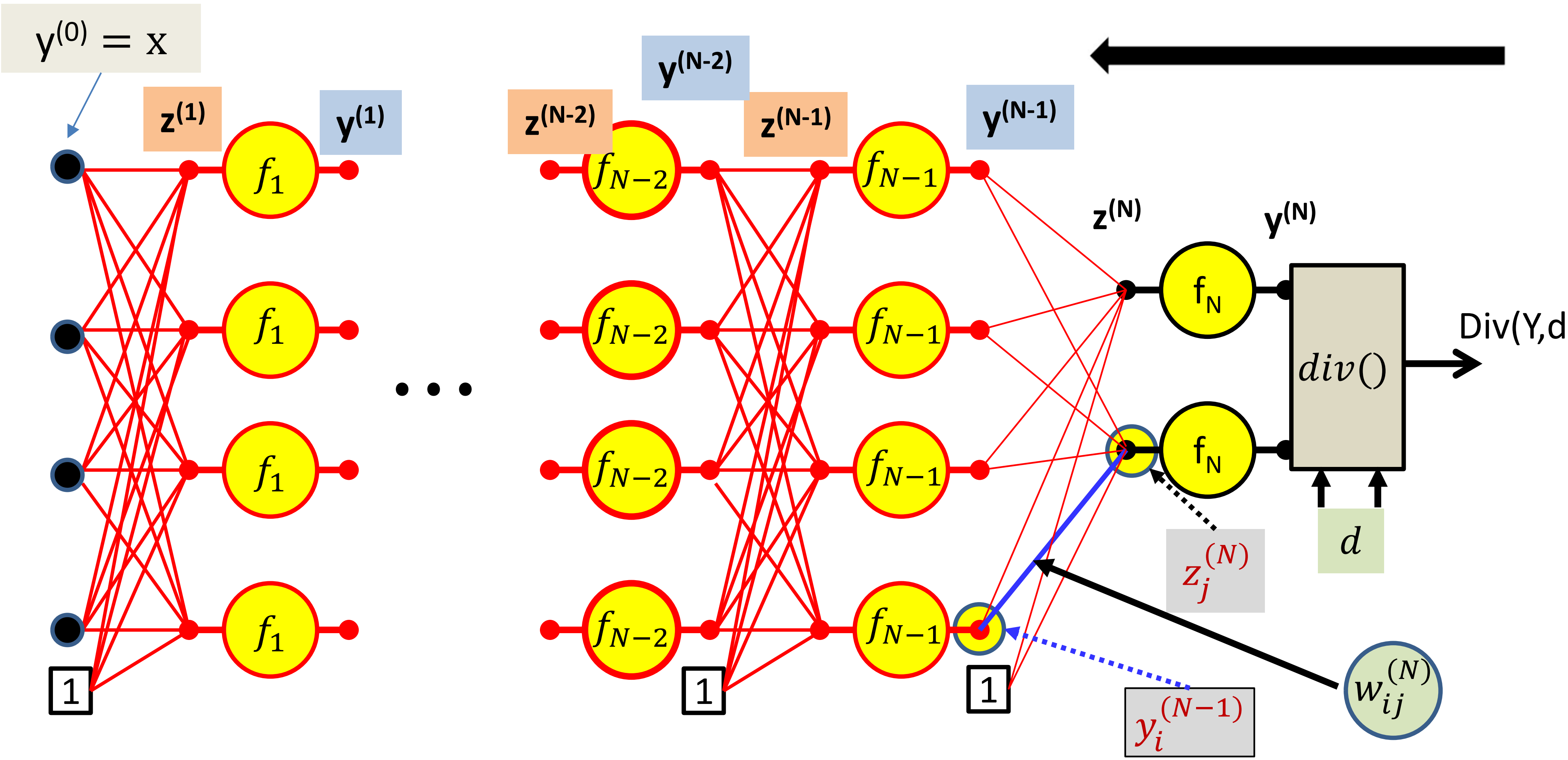
$$\frac{\partial Div}{\partial w_{ij}^{(N)}} = \frac{\partial z_j^{(N)}}{\partial w_{ij}^{(N)}} \frac{\partial Div}{\partial z_j^{(N)}}$$

$y_i^{(N-1)}$

Computed in forward pass

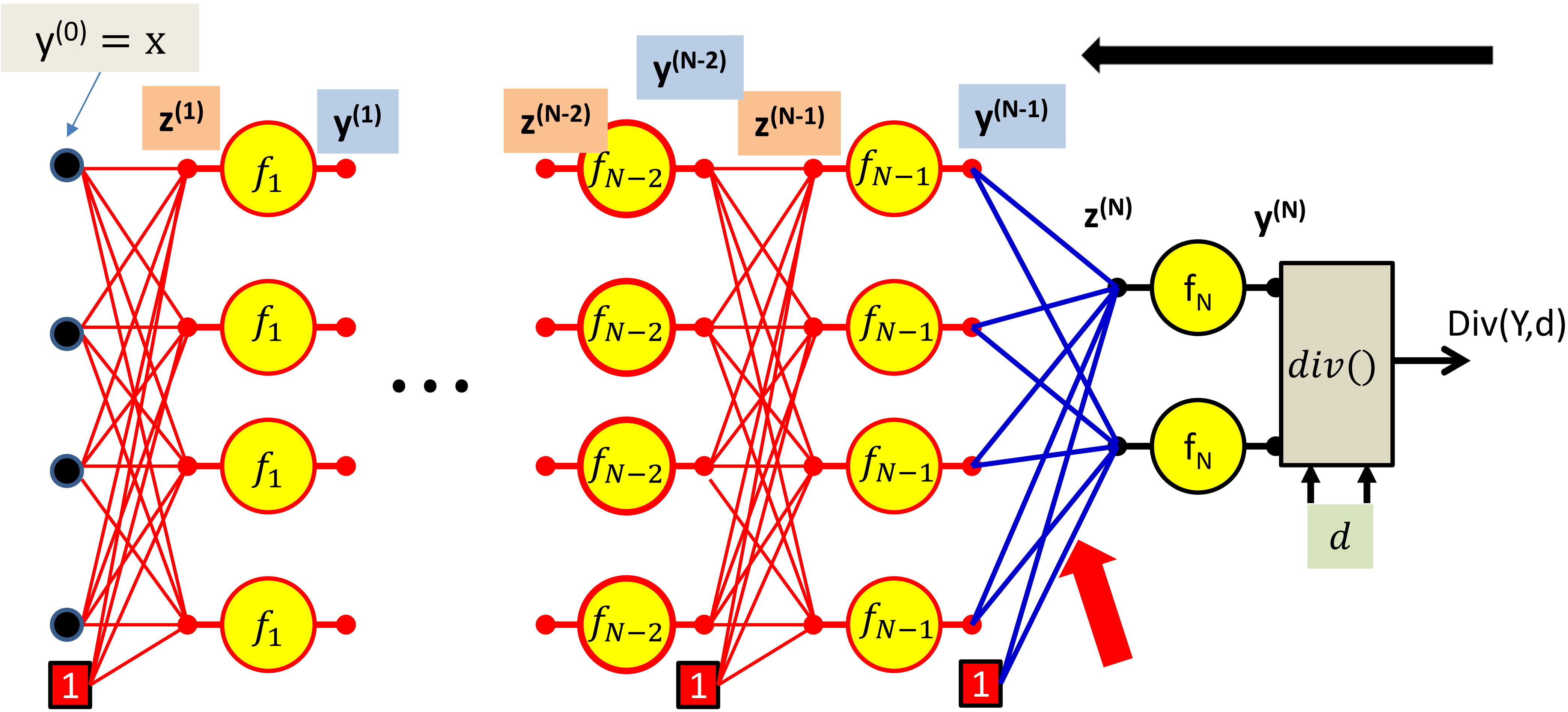
Because
$$z_j^{(N)} = w_{ij}^{(N)} y_i^{(N-1)} + \text{other terms}$$

Computing derivatives



$$\frac{\partial Div}{\partial w_{ij}^{(N)}} = y_i^{(N-1)} \frac{\partial Div}{\partial z_j^{(N)}}$$

Computing derivatives



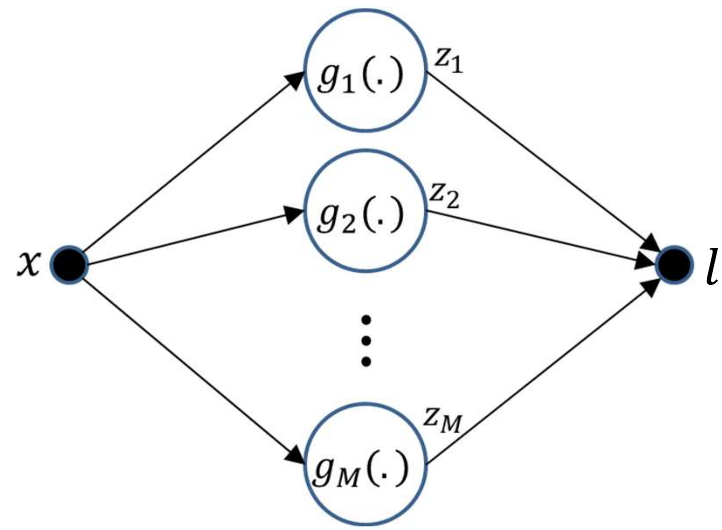
$$\boxed{\frac{\partial Div}{\partial w_{ij}^{(N)}} = y_i^{(N-1)} \frac{\partial Div}{\partial z_j^{(N)}}}$$

For the bias term $y_0^{(N-1)} = 1$

Calculus Refresher: Chain rule

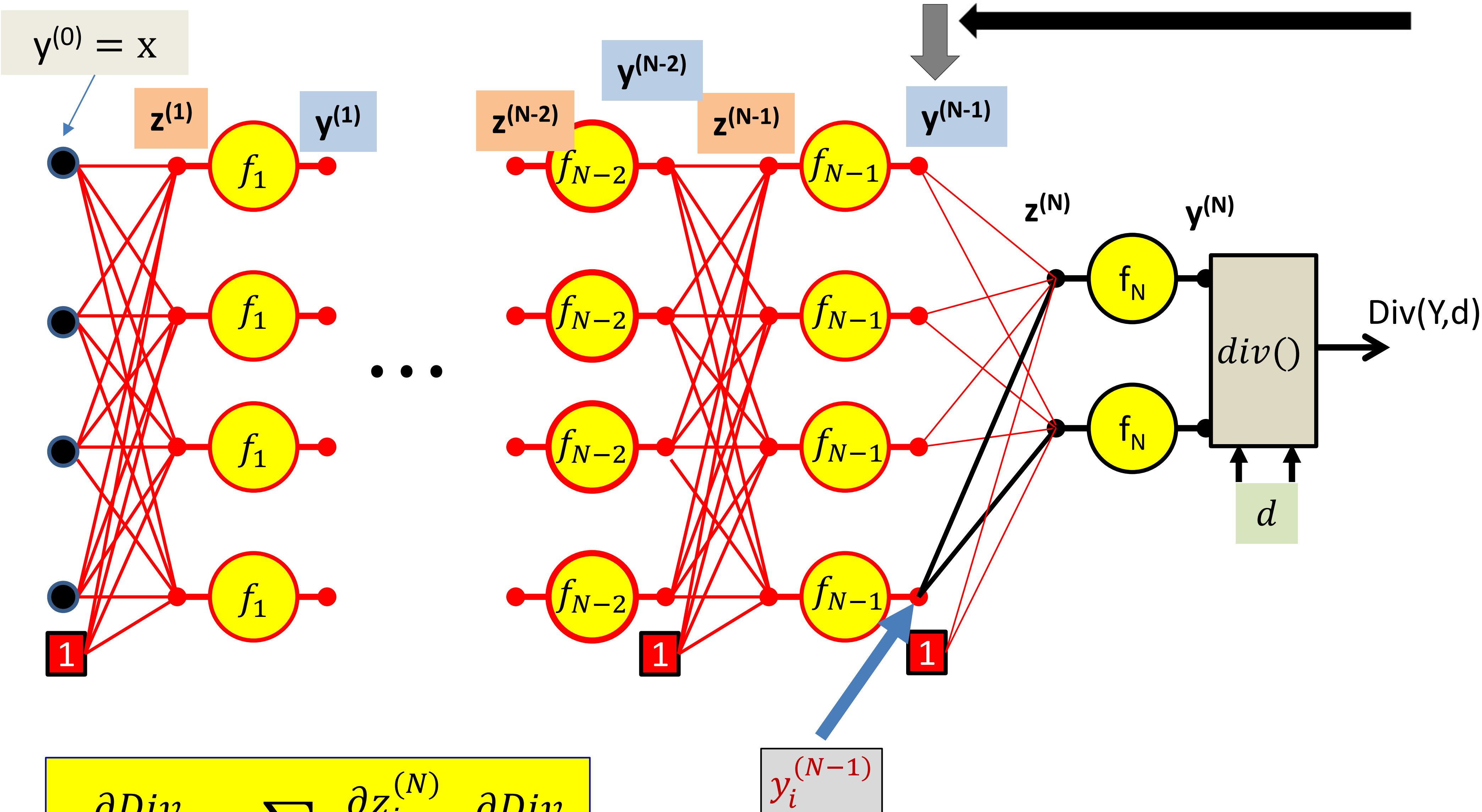
For $l = f(z_1, z_2, \dots, z_M)$

where $z_i = g_i(x)$



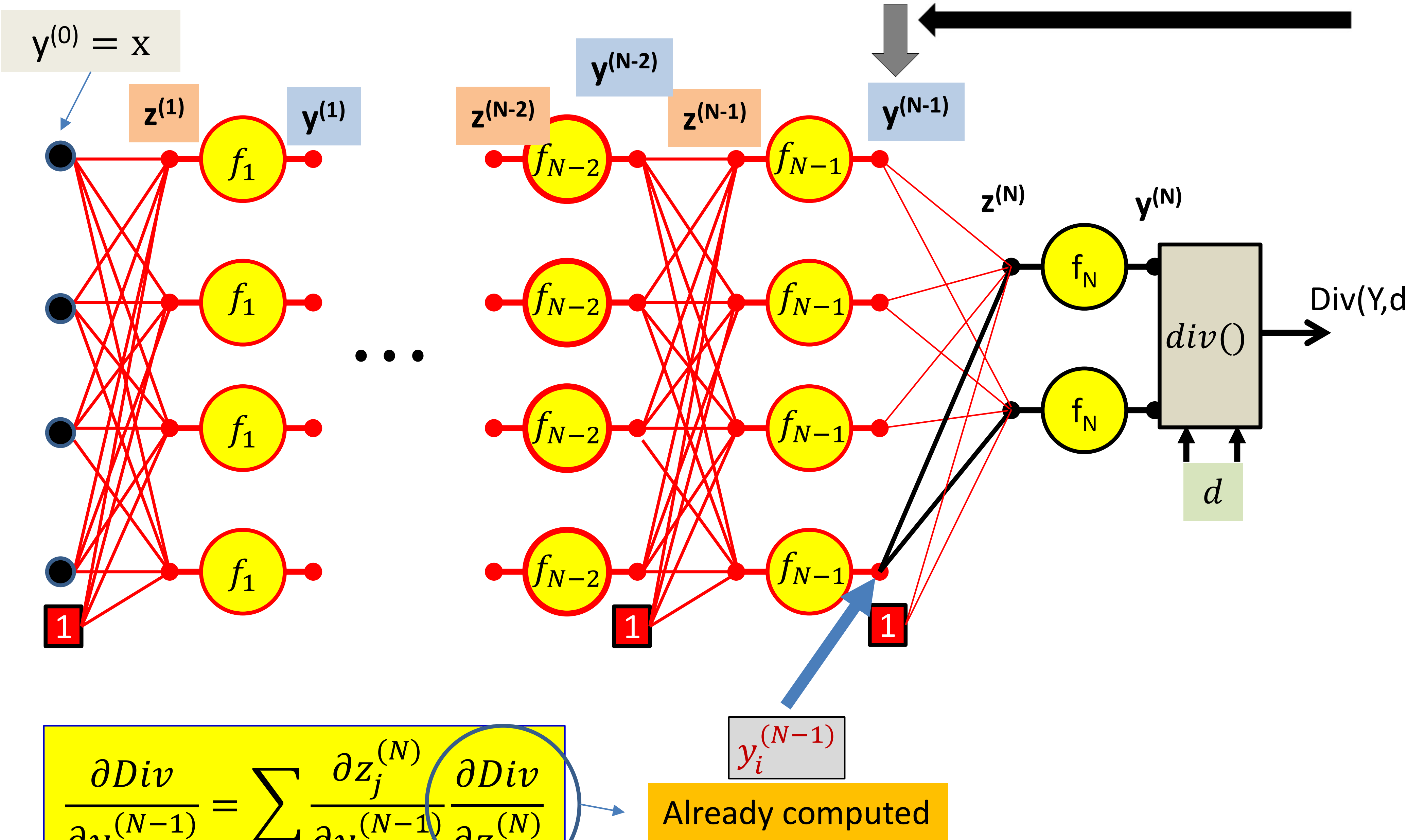
$$\boxed{\frac{dl}{dx} = \frac{\partial l}{\partial z_1}\frac{dz_1}{dx} + \frac{\partial l}{\partial z_2}\frac{dz_2}{dx} + \cdots + \frac{\partial l}{\partial z_M}\frac{dz_M}{dx}}$$

Computing derivatives



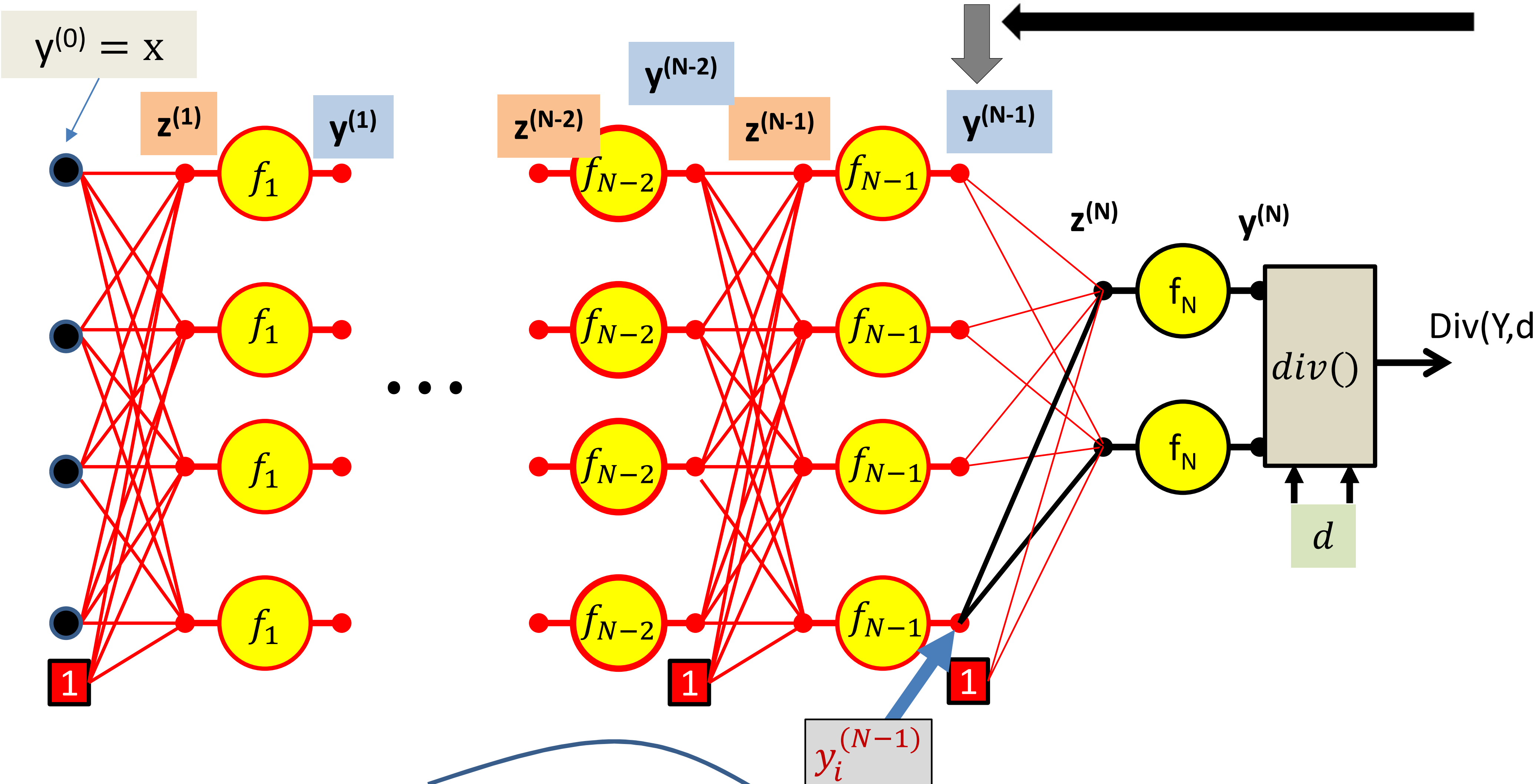
$$\frac{\partial Div}{\partial y_i^{(N-1)}} = \sum_j \frac{\partial z_j^{(N)}}{\partial y_i^{(N-1)}} \frac{\partial Div}{\partial z_j^{(N)}}$$

Computing derivatives



$$\frac{\partial Div}{\partial y_i^{(N-1)}} = \sum_j \frac{\partial z_j^{(N)}}{\partial y_i^{(N-1)}} \frac{\partial Div}{\partial z_j^{(N)}}$$

Computing derivatives



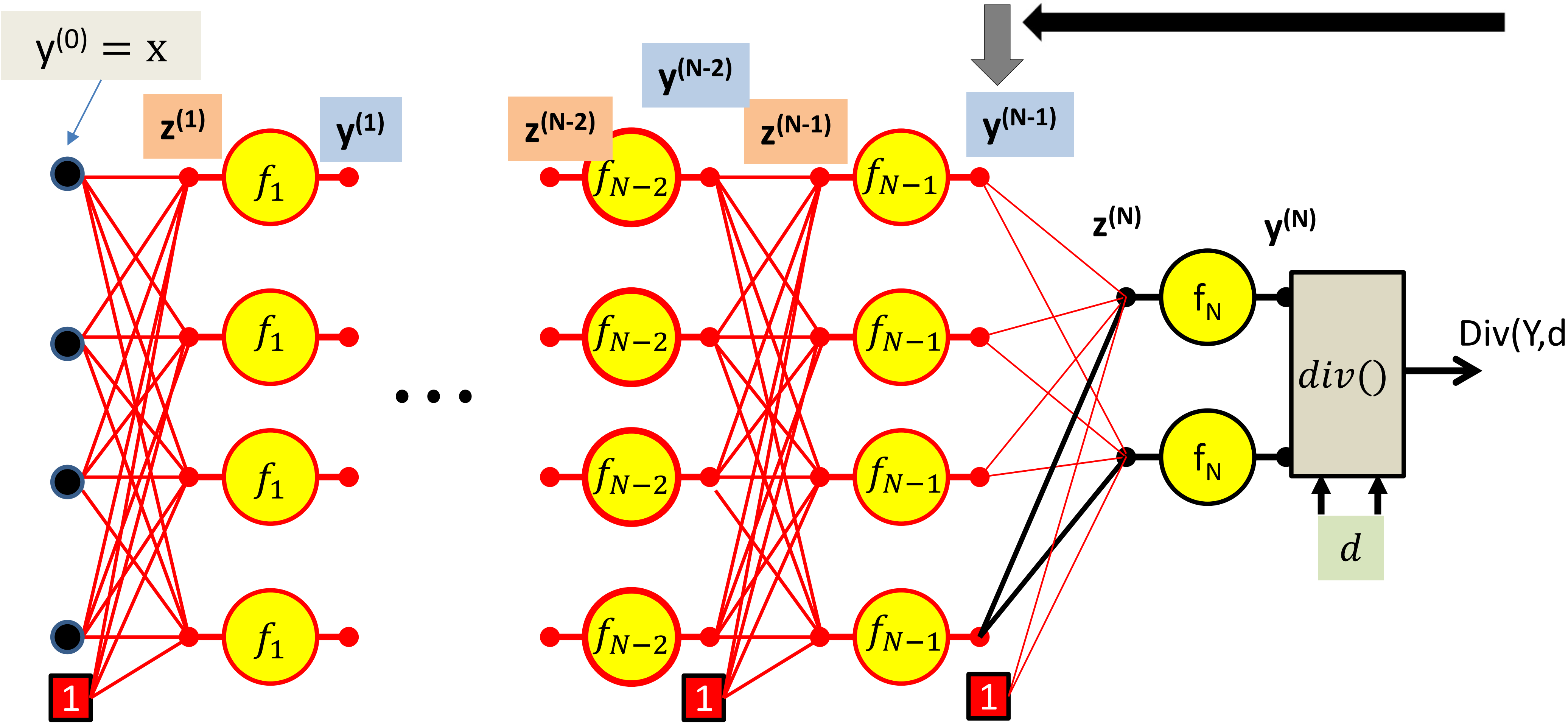
$$\frac{\partial Div}{\partial y_i^{(N-1)}} = \sum_j \frac{\partial z_j^{(N)}}{\partial y_i^{(N-1)}} \frac{\partial Div}{\partial z_j^{(N)}}$$

$w_{ij}^{(N)}$

Because

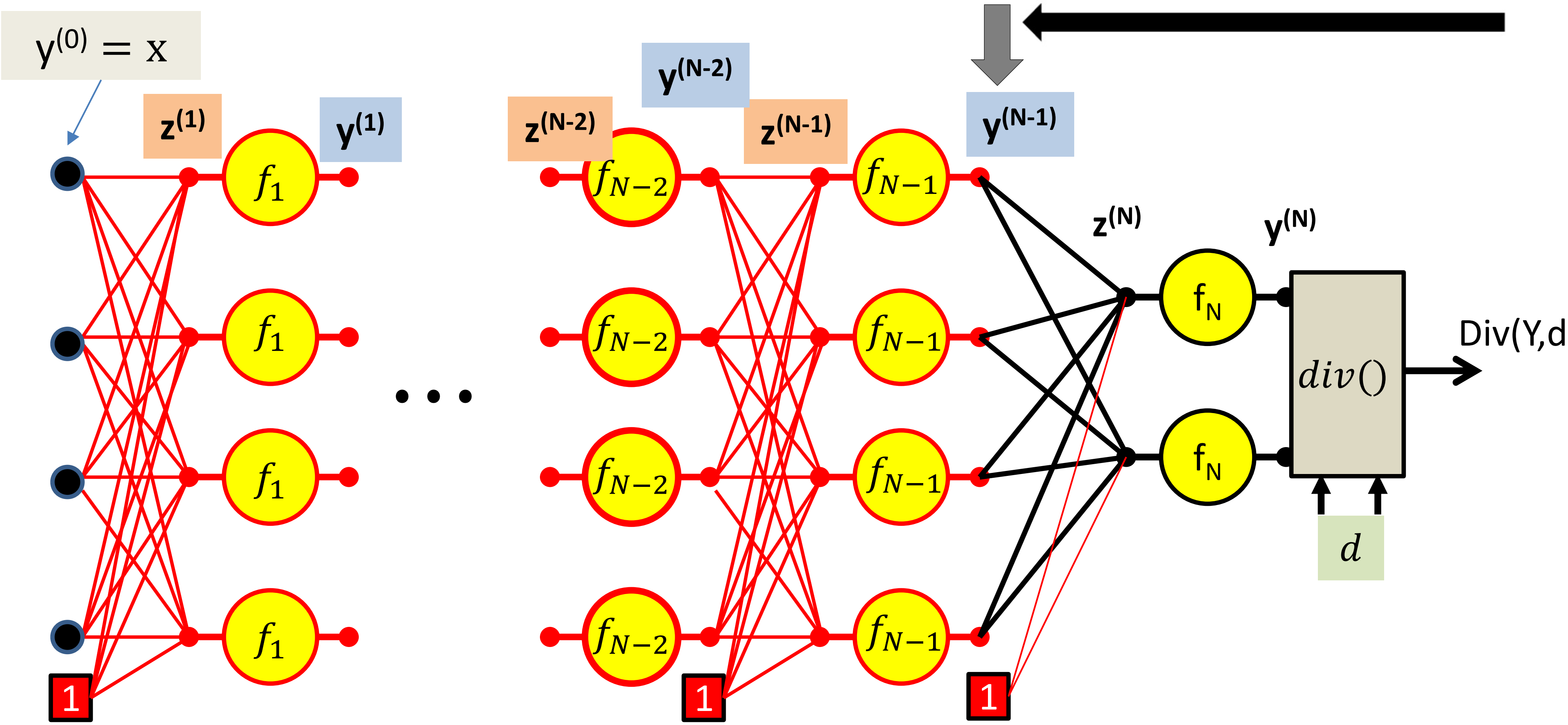
$$z_j^{(N)} = w_{ij}^{(N)} y_i^{(N-1)} + \text{other terms}$$

Computing derivatives



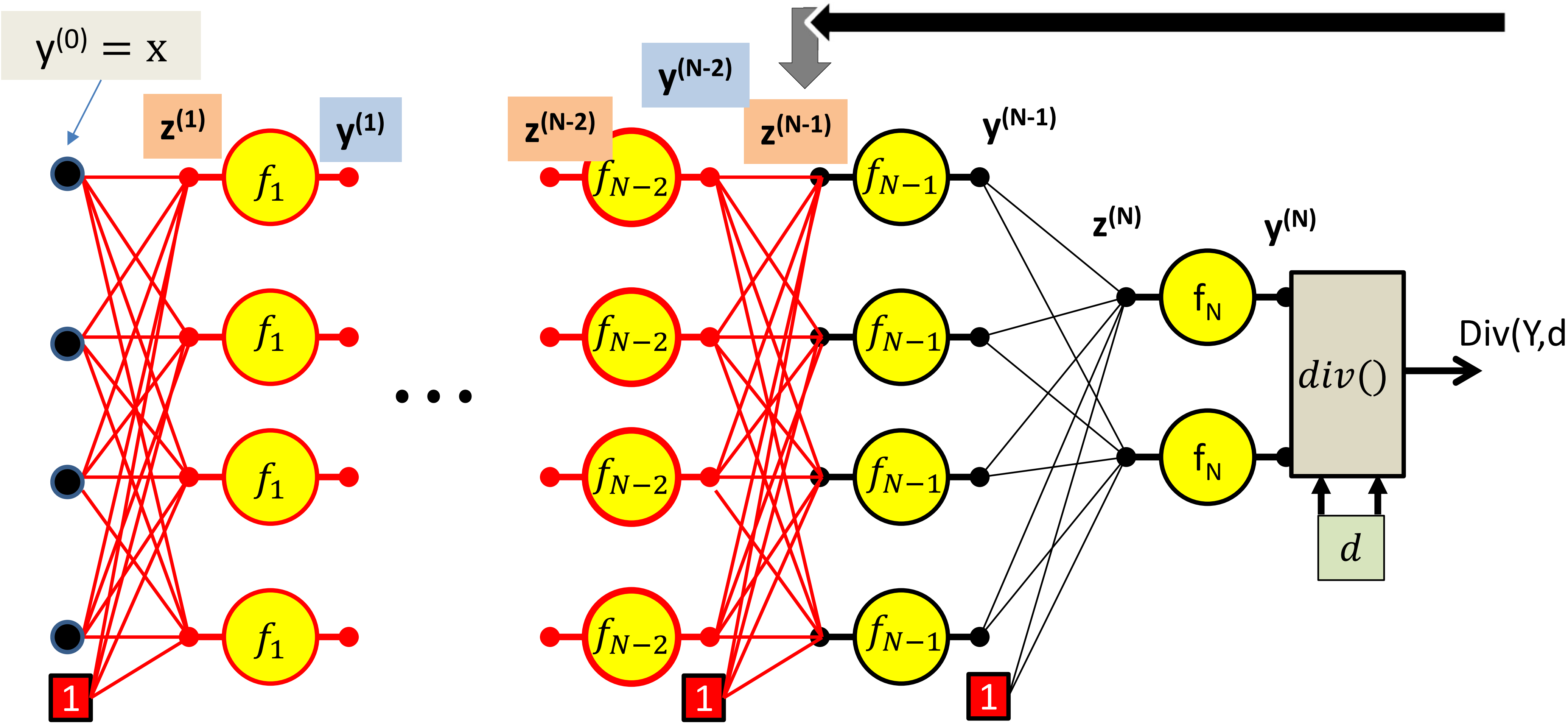
$$\frac{\partial Div}{\partial y_i^{(N-1)}} = \sum_j w_{ij}^{(N)} \frac{\partial Div}{\partial z_j^{(N)}}$$

Computing derivatives



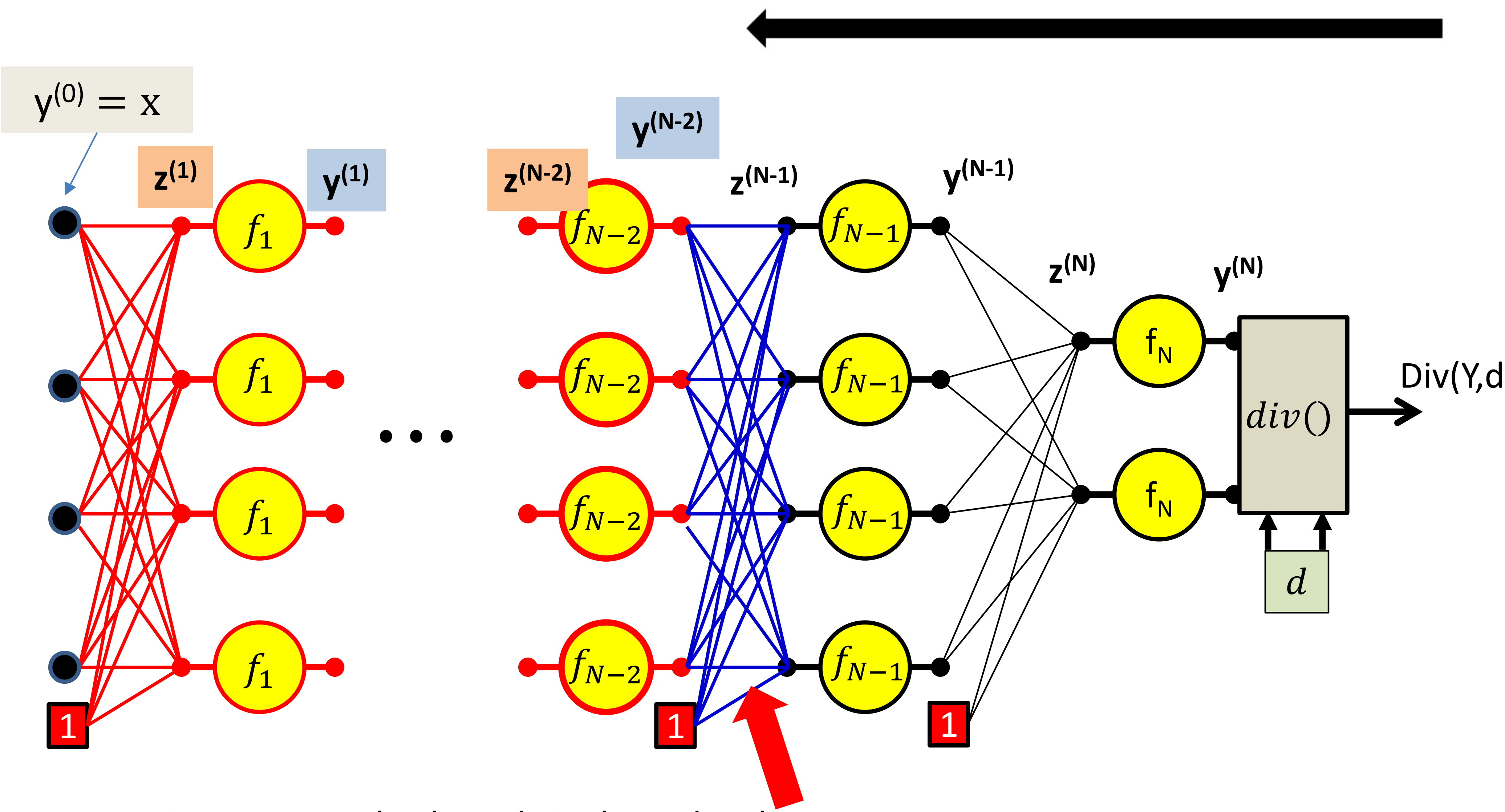
$$\frac{\partial Div}{\partial y_i^{(N-1)}} = \sum_j w_{ij}^{(N)} \frac{\partial Div}{\partial z_j^{(N)}}$$

Computing derivatives



We continue our way backwards in the order shown

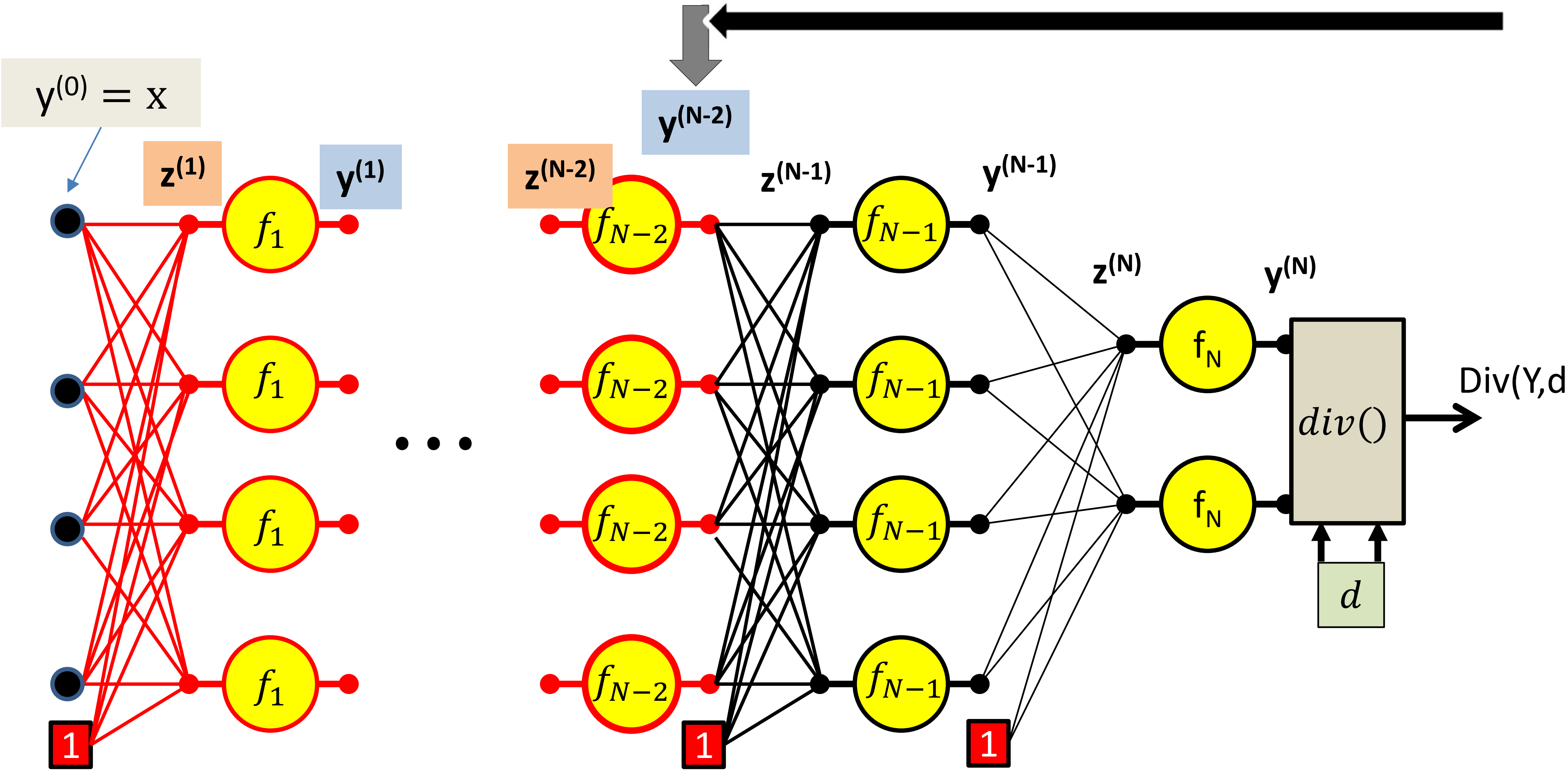
$$\frac{\partial Div}{\partial z_i^{(N-1)}} = f'_{N-1}\left(z_i^{(N-1)}\right)\frac{\partial Div}{\partial y_i^{(N-1)}}$$



We continue our way backwards in the order shown

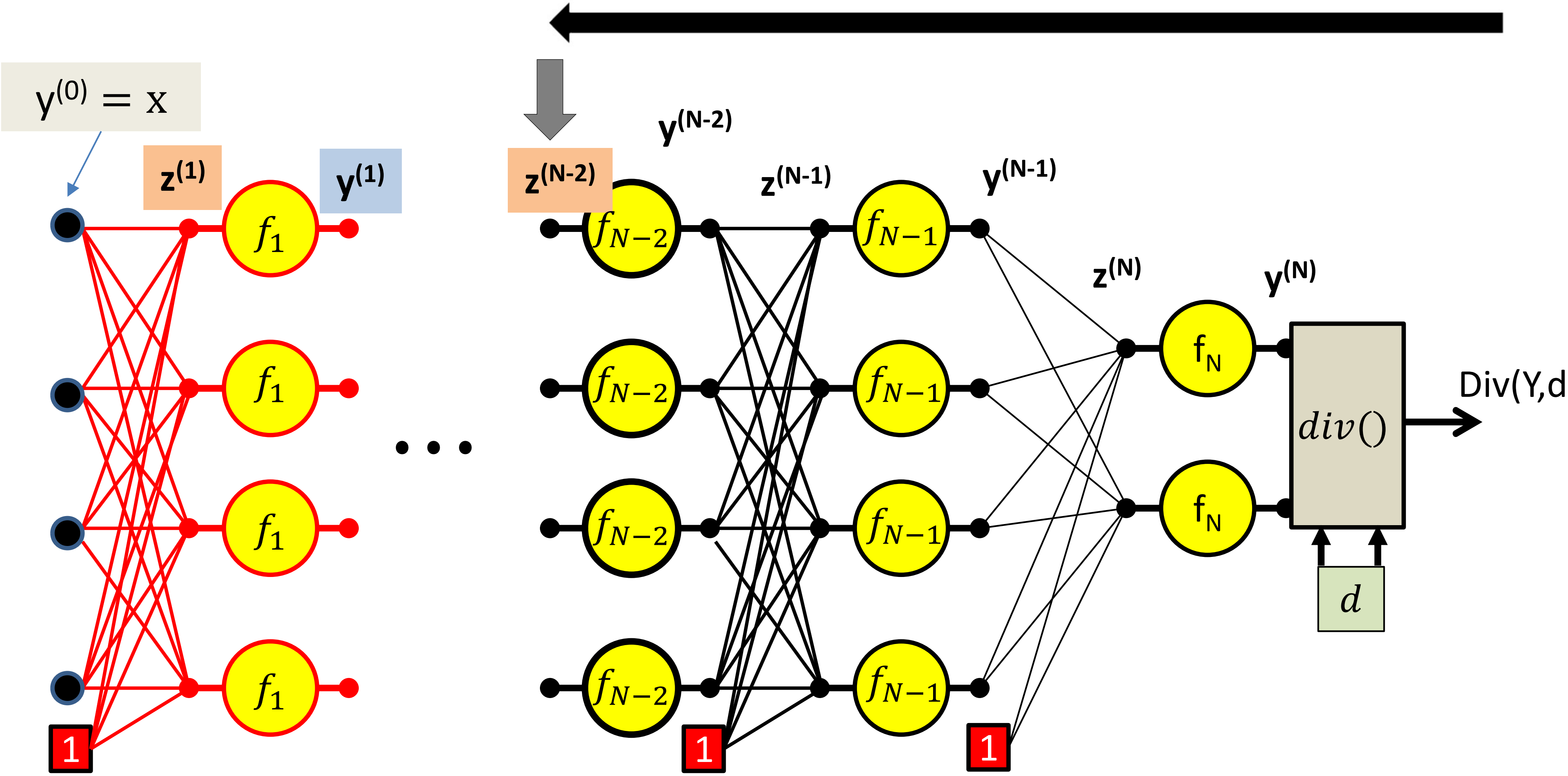
$$\frac{\partial Div}{\partial w_{ij}^{(N-1)}} = y_i^{(N-2)} \frac{\partial Div}{\partial z_j^{(N-1)}}$$

For the bias term $y_0^{(N-2)} = 1$



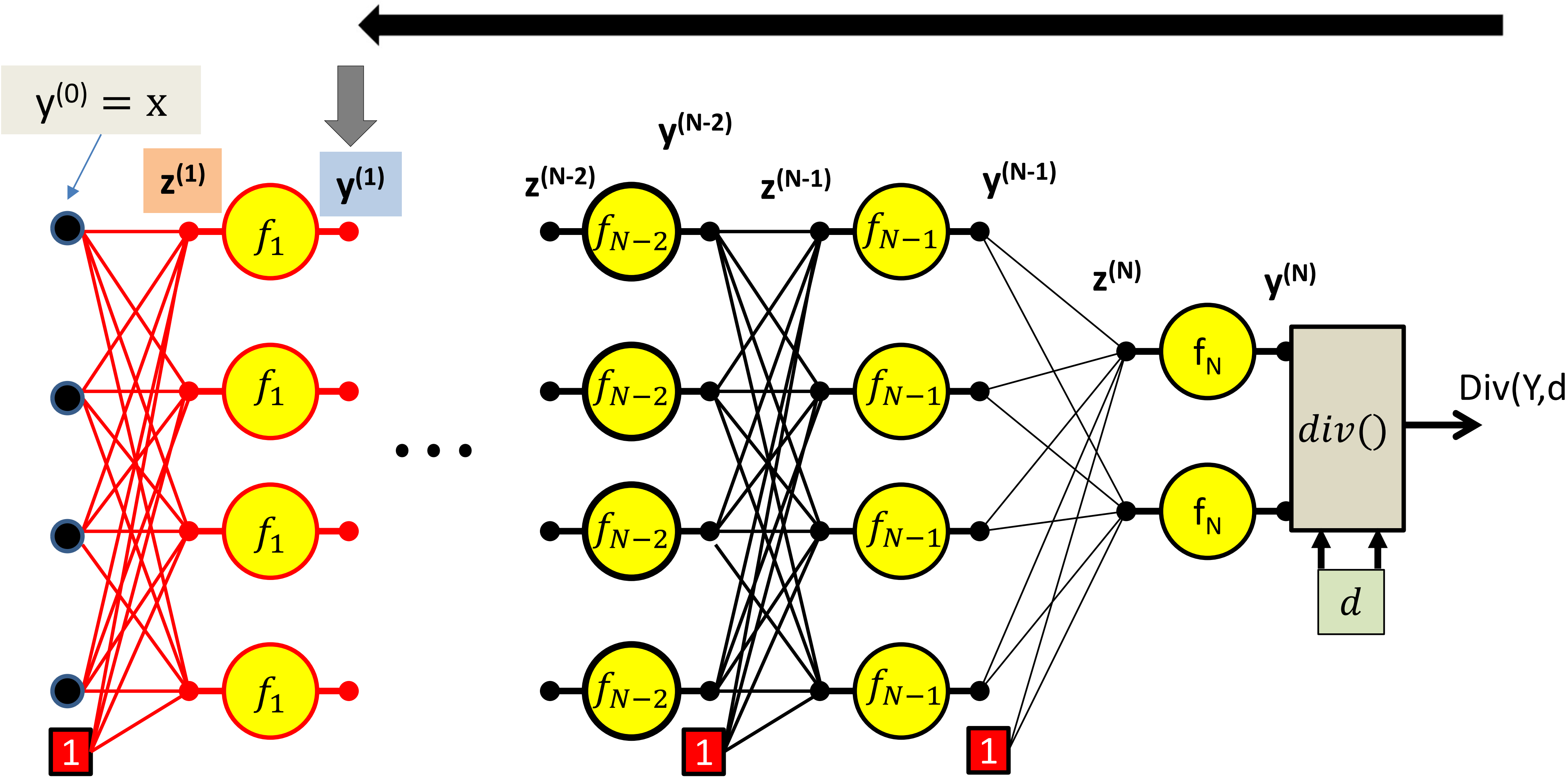
We continue our way backwards in the order shown

$$\frac{\partial Div}{\partial y_i^{(N-2)}} = \sum_j w_{ij}^{(N-1)} \frac{\partial Div}{\partial z_j^{(N-1)}}$$



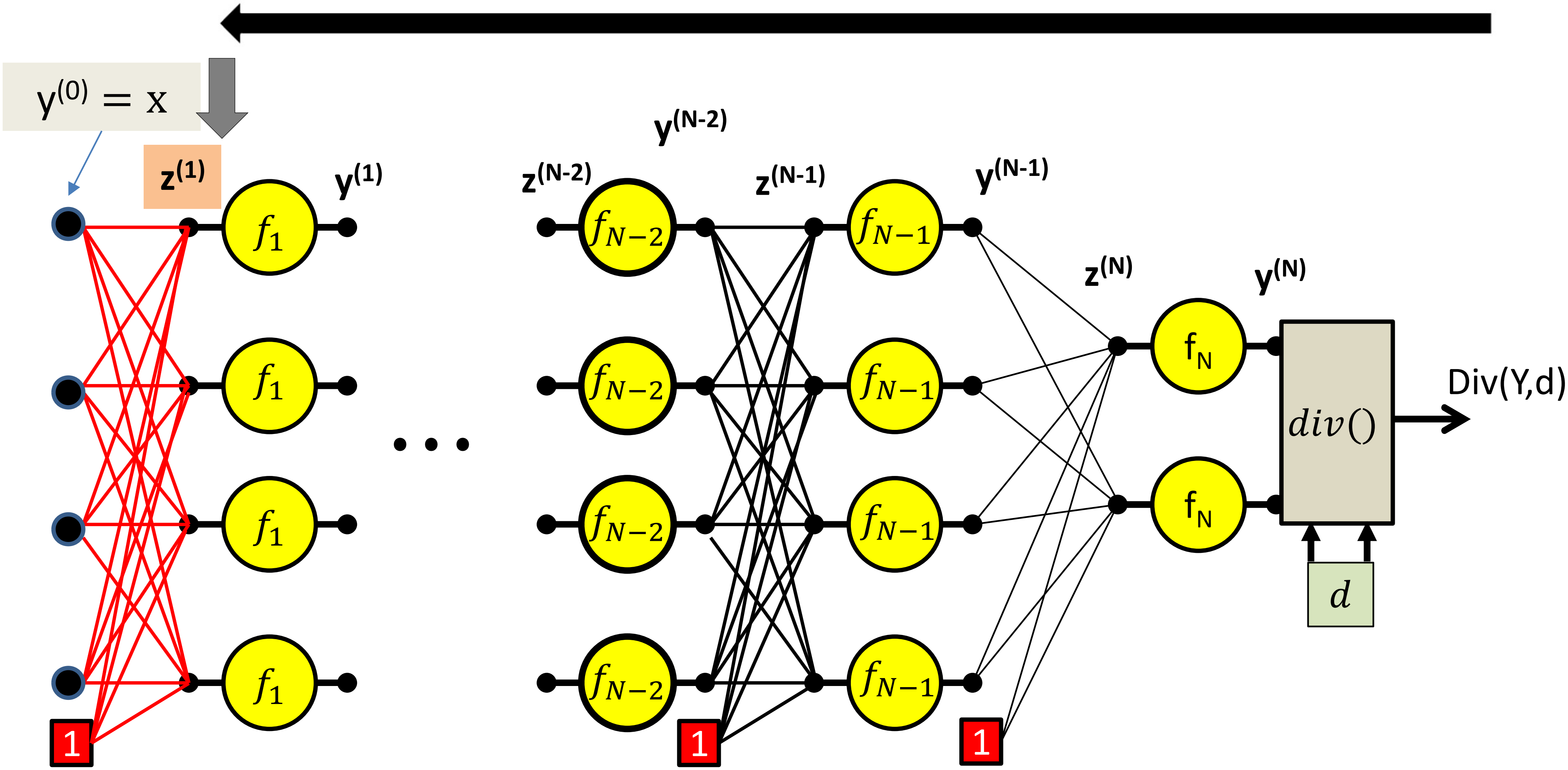
We continue our way backwards in the order shown

$$\frac{\partial Div}{\partial z_i^{(N-2)}} = f'_{N-2}\left(z_i^{(N-2)}\right)\frac{\partial Div}{\partial y_i^{(N-2)}}$$



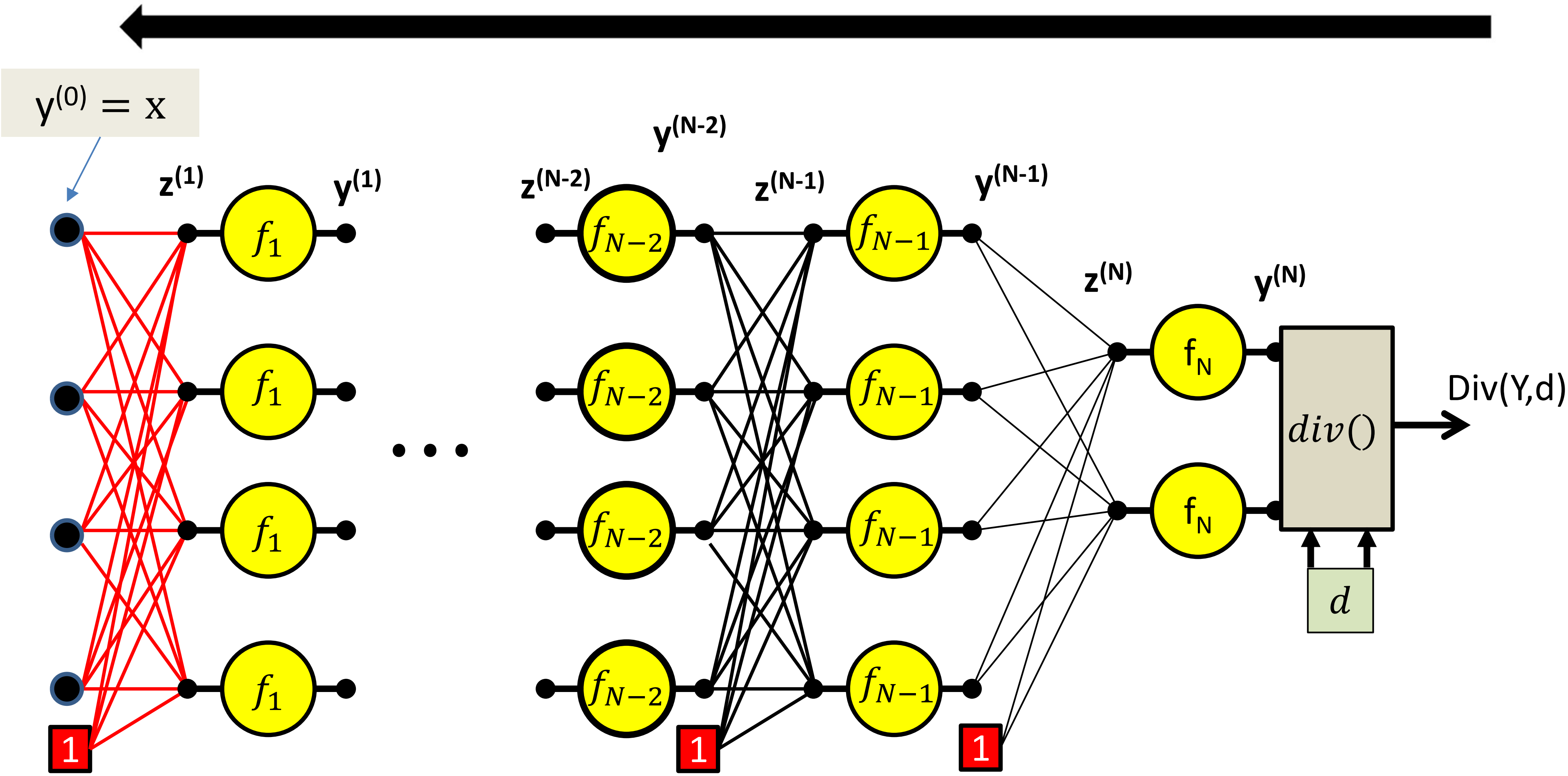
We continue our way backwards in the order shown

$$\frac{\partial Div}{\partial y_1^{(1)}} = \sum_j w_{ij}^{(2)} \frac{\partial Div}{\partial z_j^{(2)}}$$



We continue our way backwards in the order shown

$$\frac{\partial Div}{\partial z_i^{(1)}} = f_1'\left(z_i^{(1)}\right)\frac{\partial Div}{\partial y_i^{(1)}}$$



We continue our way backwards in the order shown

$$\frac{\partial Div}{\partial w_{ij}^{(1)}} = y_i^{(0)} \frac{\partial Div}{\partial z_j^{(1)}}$$

Gradients: Backward Computation

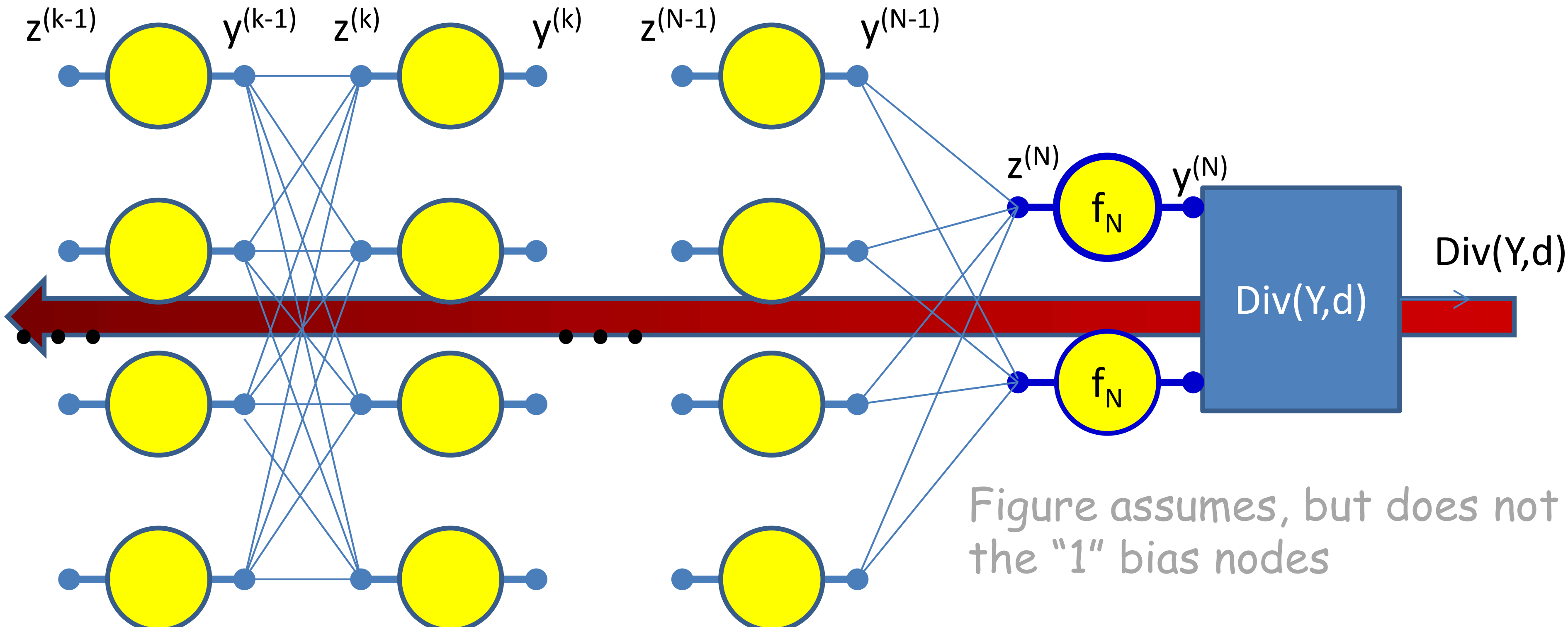


Figure assumes, but does not show the "1" bias nodes

Initialize: Gradient w.r.t network output

$$\frac{\partial Div}{\partial y_i^{(N)}} = \frac{\partial Div(Y,d)}{\partial y_i}$$

$$\frac{\partial Div}{\partial z_i^{(N)}} = f_k'\left(z_i^{(N)}\right)\frac{\partial Div}{\partial y_i^{(N)}}$$

$For\ k = N-1..0$

$\quad For\ i = 1: layer\ width$

$$\frac{\partial Div}{\partial y_i^{(k)}} = \sum_j w_{ij}^{(k+1)} \frac{\partial Div}{\partial z_j^{(k+1)}}$$

$$\boxed{\frac{\partial Div}{\partial z_i^{(k)}} = f_k'\left(z_i^{(k)}\right)\frac{\partial Div}{\partial y_i^{(k)}}}$$

$$\boxed{\forall j\ \frac{\partial Div}{\partial w_{ij}^{(k+1)}} = y_i^{(k)} \frac{\partial Div}{\partial z_j^{(k+1)}}}$$

Backward Pass

- Output layer (N) :
  - For $i = 1 \dots D_N$
    - $\frac{\partial Div}{\partial y_i^{(N)}} = \frac{\partial Div(Y,d)}{\partial y_i}$ [This is the derivative of the divergence]
    - $\frac{\partial Div}{\partial z_i^{(N)}} = \frac{\partial Div}{\partial y_i^{(N)}} f_N'\left(z_i^{(N)}\right)$
    - $\frac{\partial Div}{\partial w_{ij}^{(N)}} = y_i^{(N-1)} \frac{\partial Div}{\partial z_j^{(N)}}$ for $j = 0 \dots D_{N-1}$
- For layer $k = N - 1\ downto\ 1$
  - For $i = 1 \dots D_k$
    - $\frac{\partial Div}{\partial y_i^{(k)}} = \sum_j w_{ij}^{(k+1)} \frac{\partial Div}{\partial z_j^{(k+1)}}$
    - $\frac{\partial Div}{\partial z_i^{(k)}} = \frac{\partial Div}{\partial y_i^{(k)}} f_k'\left(z_i^{(k)}\right)$
    - $\frac{\partial Div}{\partial w_{ij}^{(k)}} = y_i^{(k-1)} \frac{\partial Div}{\partial z_j^{(k)}}$ for $j = 0 \dots D_{k-1}$

Backward Pass

- Output layer (N) :
  - For $i = 1 \ldots D_N$
    - $\frac{\partial Div}{\partial y_i^{(N)}} = \frac{\partial Div(Y,d)}{\partial y_i}$
    - $\frac{\partial Div}{\partial z_i^{(N)}} = \frac{\partial Div}{\partial y_i^{(N)}} f_N'\left(z_i^{(N)}\right)$
    - $\frac{\partial Div}{\partial w_{ij}^{(N)}} = y_i^{(N-1)} \frac{\partial Div}{\partial z_j^{(N)}}$ for $j = 0 \ldots D_{N-1}$

Called "Backpropagation" because the derivative of the loss is propagated "backwards" through the network

- For layer $k = N - 1\ downto\ 1$
  - For $i = 1 \ldots D_k$
    - $\frac{\partial Div}{\partial y_i^{(k)}} = \sum_j w_{ij}^{(k+1)} \frac{\partial Div}{\partial z_j^{(k+1)}}$
    - $\frac{\partial Div}{\partial z_i^{(k)}} = \frac{\partial Div}{\partial y_i^{(k)}} f_k'\left(z_i^{(k)}\right)$
    - $\frac{\partial Div}{\partial w_{ij}^{(k)}} = y_i^{(k-1)} \frac{\partial Div}{\partial z_j^{(k)}}$ for $j = 0 \ldots D_{k-1}$

Very analogous to the forward pass:

Backward weighted combination of next layer

Backward equivalent of activation

Using notation $\dot{y} = \frac{\partial Div(Y,d)}{\partial y}$ etc (overdot represents derivative of Div w.r.t variable)

- Output layer (N) :
  - For $i = 1 \dots D_N$
    - $\dot{y}_i^{(N)} = \frac{\partial Div}{\partial y_i}$
    - $\dot{z}_i^{(N)} = \dot{y}_i^{(N)} f_N'\left(z_i^{(N)}\right)$
    - $\frac{\partial Div}{\partial w_{ji}^{(N)}} = y_j^{(N-1)} \dot{z}_i^{(N)}$ for $j = 0 \dots D_{N-1}$

Called "Backpropagation" because the derivative of the loss is propagated "backwards" through the network

- For layer $k = N - 1\ downto\ 1$
  - For $i = 1 \dots D_k$
    - $\dot{y}_i^{(k)} = \sum_j w_{ij}^{(k+1)} \dot{z}_j^{(k+1)}$
    - $\dot{z}_i^{(k)} = \dot{y}_i^{(k)} f_k'\left(z_i^{(k)}\right)$
    - $\frac{\partial Div}{\partial w_{ji}^{(k)}} = y_j^{(k-1)} \dot{z}_i^{(k)}$ for $j = 0 \dots D_{k-1}$

Very analogous to the forward pass:

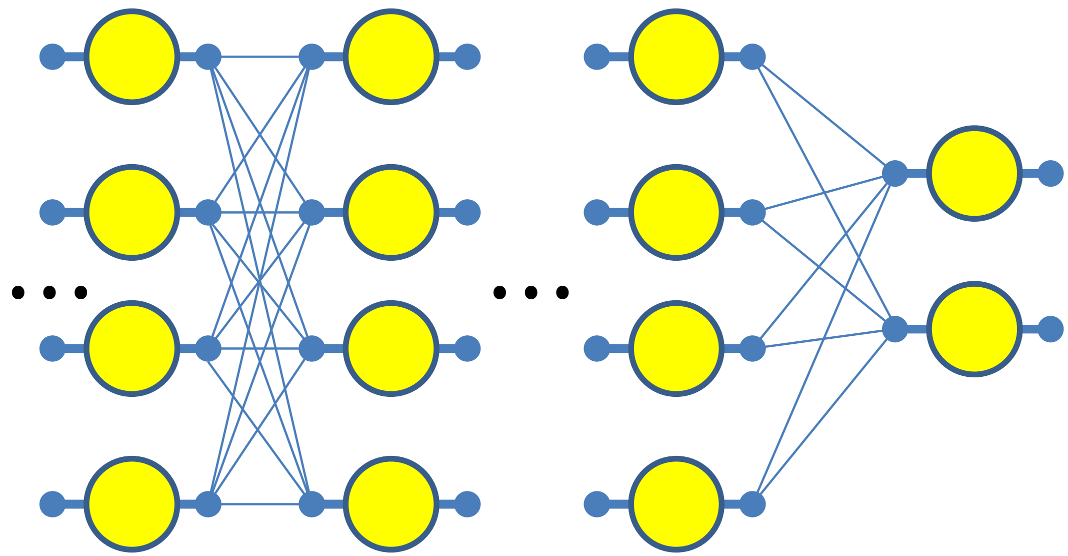
Backward weighted combination of next layer

Backward equivalent of activation

For comparison: the forward pass again

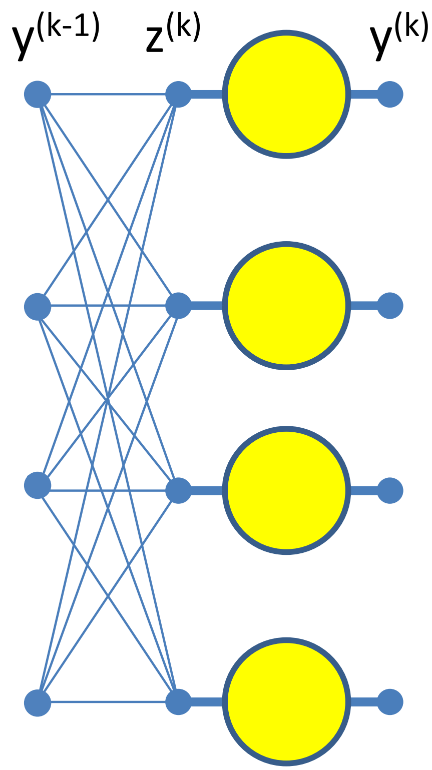
- Input: D dimensional vector $\mathbf{x} = [x_j,\ j = 1 \dots D]$
- Set:
  - $D_0 = D,$ is the width of the 0th (input) layer
  - $y_j^{(0)} = x_j,\ j = 1 \dots D;\qquad y_0^{(k=1 \dots N)} = x_0 = 1$
- For layer $k = 1 \dots N$
  - For $j = 1 \dots D_k$
    - $z_j^{(k)} = \sum_{i=0}^{N_k} w_{i,j}^{(k)} y_i^{(k-1)}$
    - $y_j^{(k)} = f_k\left(z_j^{(k)}\right)$
- Output:
  - $Y = y_j^{(N)}, j = 1..D_N$

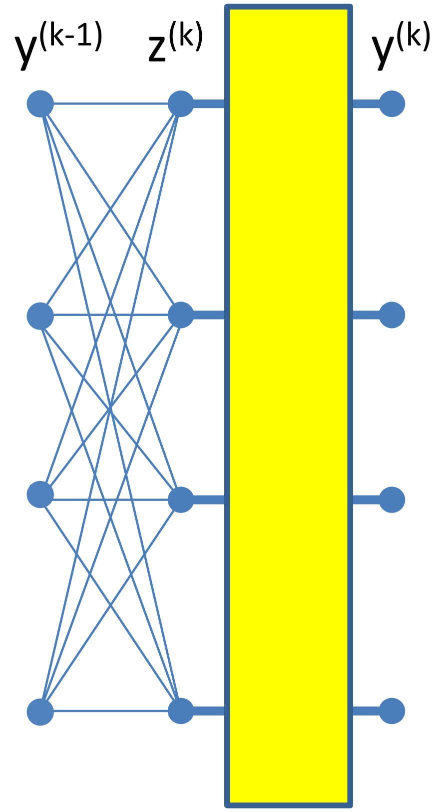
Special cases



- Have assumed so far that
  1. The computation of the output of one neuron does not directly affect computation of other neurons in the same (or previous) layers
  2. Inputs to neurons only combine through weighted addition
  3. Activations are actually differentiable
  - All of these conditions are frequently not applicable
- Will not discuss all of these in class, but explained in slides
  - Will appear in quiz. Please read the slides

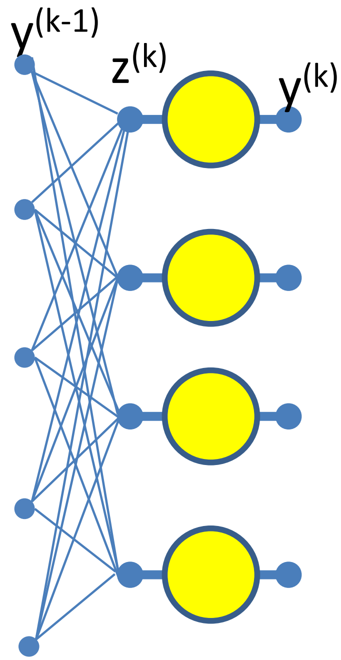
Special Case 1. Vector activations

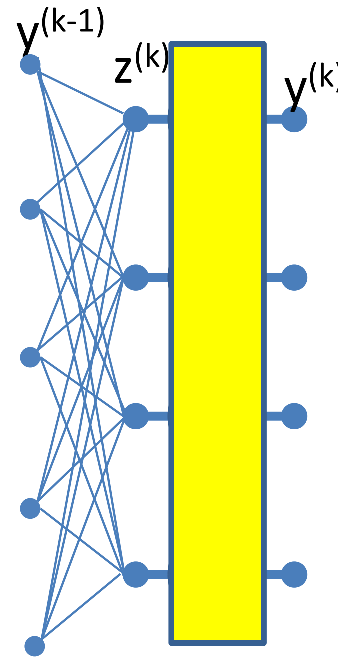




- Vector activations: all outputs are functions of all inputs

Special Case 1. Vector activations





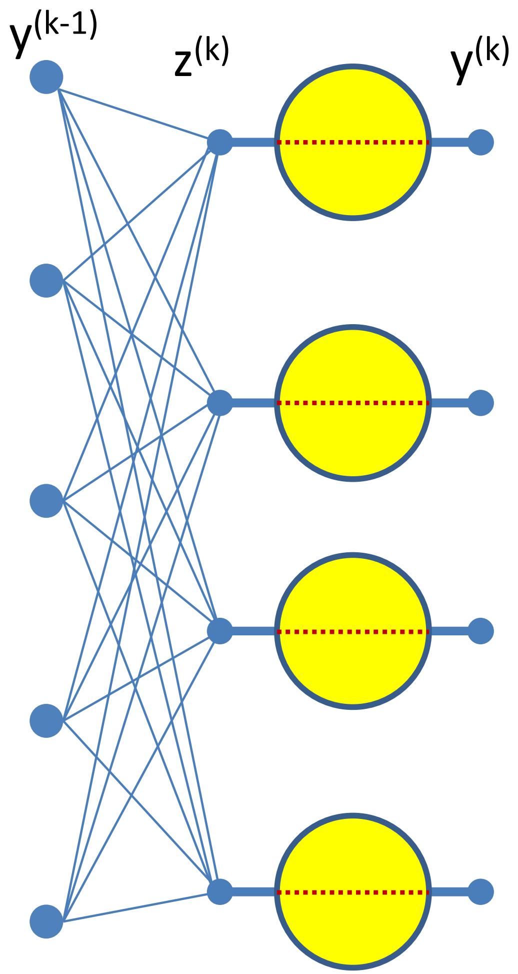
Scalar activation: Modifying a z_i only changes corresponding y_i

$$y_i^{(k)} = f\left(z_i^{(k)}\right)$$

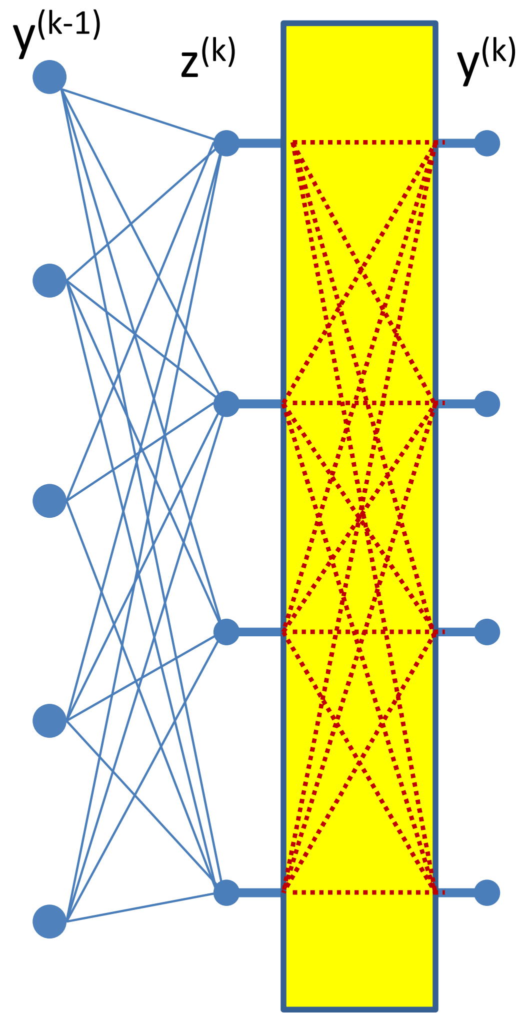
Vector activation: Modifying a z_i potentially changes all, $y_1 \dots y_M$

$$\begin{bmatrix} y_1^{(k)} \\ y_2^{(k)} \\ \vdots \\ y_M^{(k)} \end{bmatrix} = f\left(\begin{bmatrix} z_1^{(k)} \\ z_2^{(k)} \\ \vdots \\ z_D^{(k)} \end{bmatrix}\right)$$

"Influence" diagram

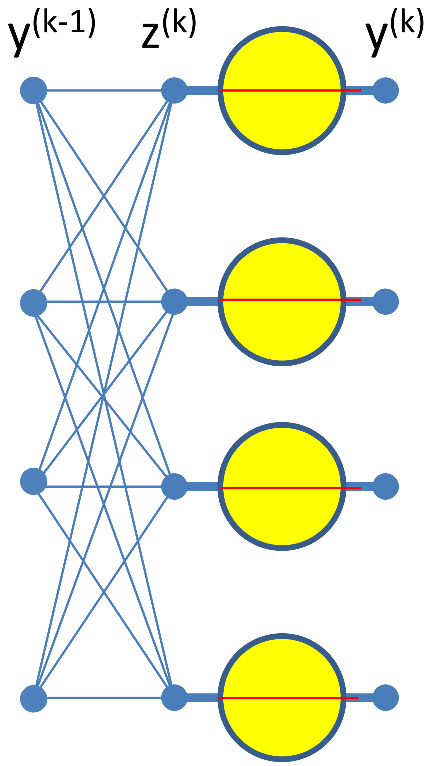


Scalar activation: Each z_i influences *one* y_i



Vector activation: Each z_i influences all, $y_1 \ldots y_M$

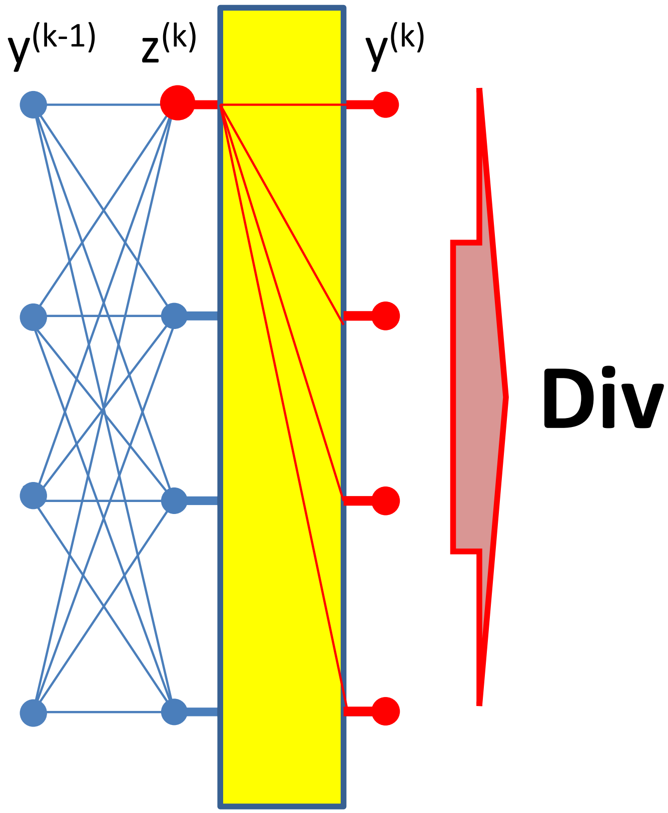
Scalar Activation: Derivative rule



$$\frac{\partial Div}{\partial z_i^{(k)}} = \frac{\partial Div}{\partial y_i^{(k)}} \frac{dy_i^{(k)}}{dz_i^{(k)}}$$

- In the case of *scalar* activation functions, the derivative of the loss w.r.t to the input to the unit is a simple product of derivatives

Derivatives of vector activation



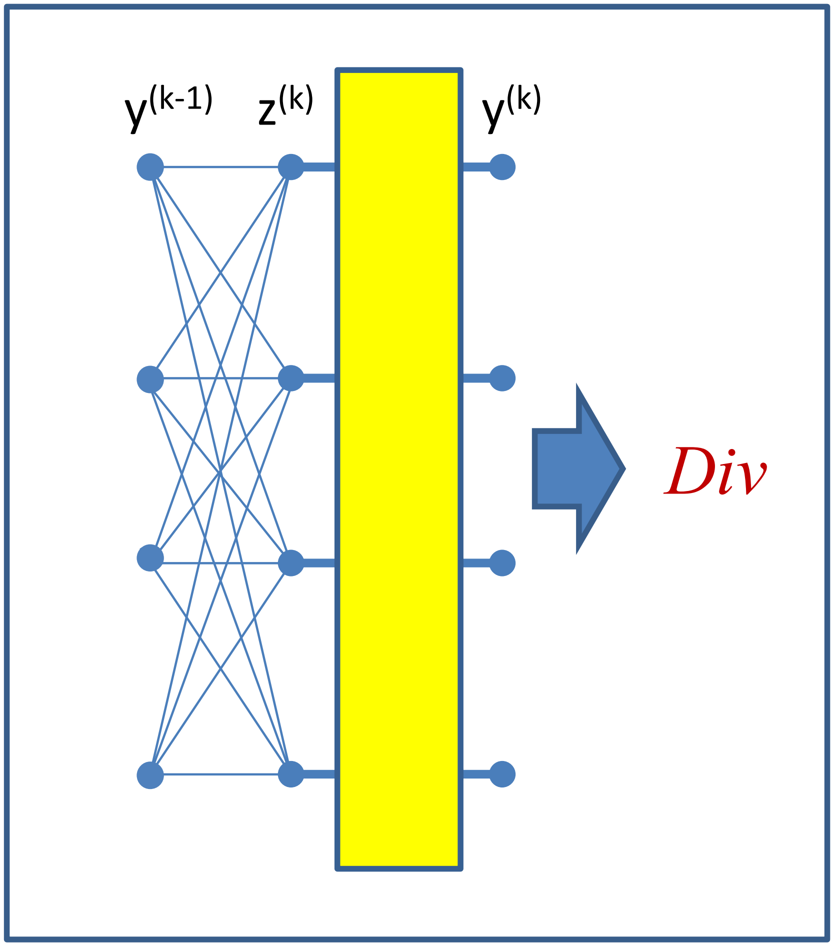
$$\frac{\partial Div}{\partial z_i^{(k)}} = \sum_j \frac{\partial Div}{\partial y_j^{(k)}} \frac{\partial y_j^{(k)}}{\partial z_i^{(k)}}$$

Note: derivatives of scalar activations are just a special case of vector activations:

$$\frac{\partial y_j^{(k)}}{\partial z_i^{(k)}} = 0 \;\; for \;\; i \neq j$$

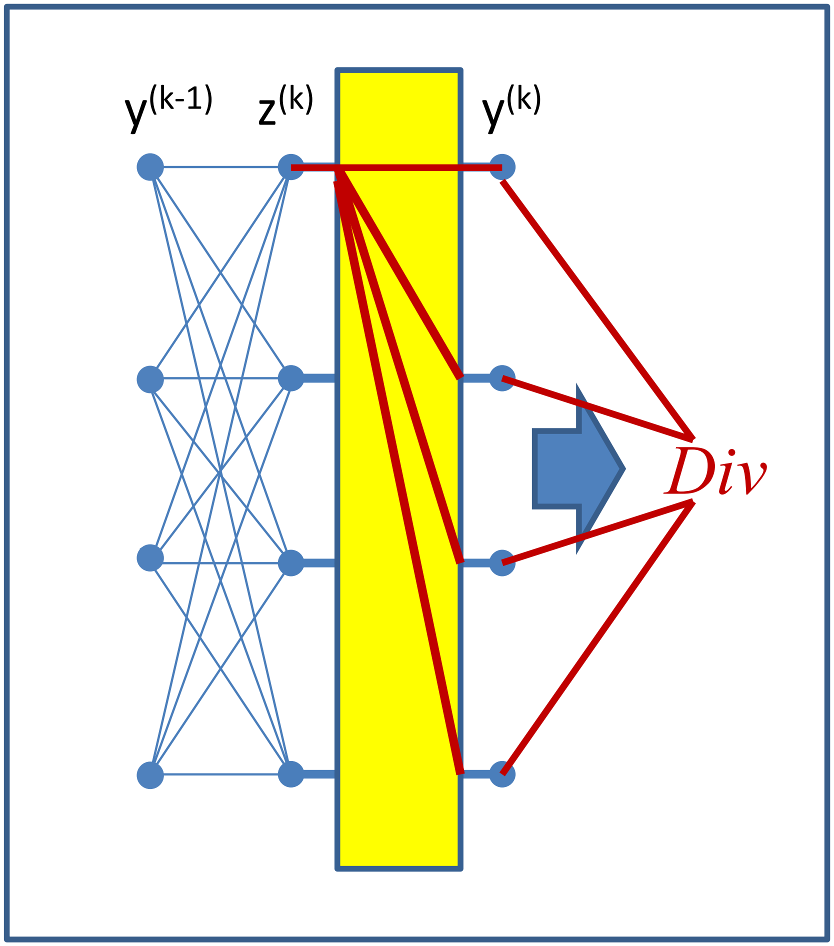
- For *vector* activations the derivative of the loss w.r.t. to any input is a sum of partial derivatives
  - Regardless of the number of outputs $y_j^{(k)}$

Example Vector Activation: Softmax



$$y_i^{(k)} = \frac{exp\left(z_i^{(k)}\right)}{\sum_j exp\left(z_j^{(k)}\right)}$$

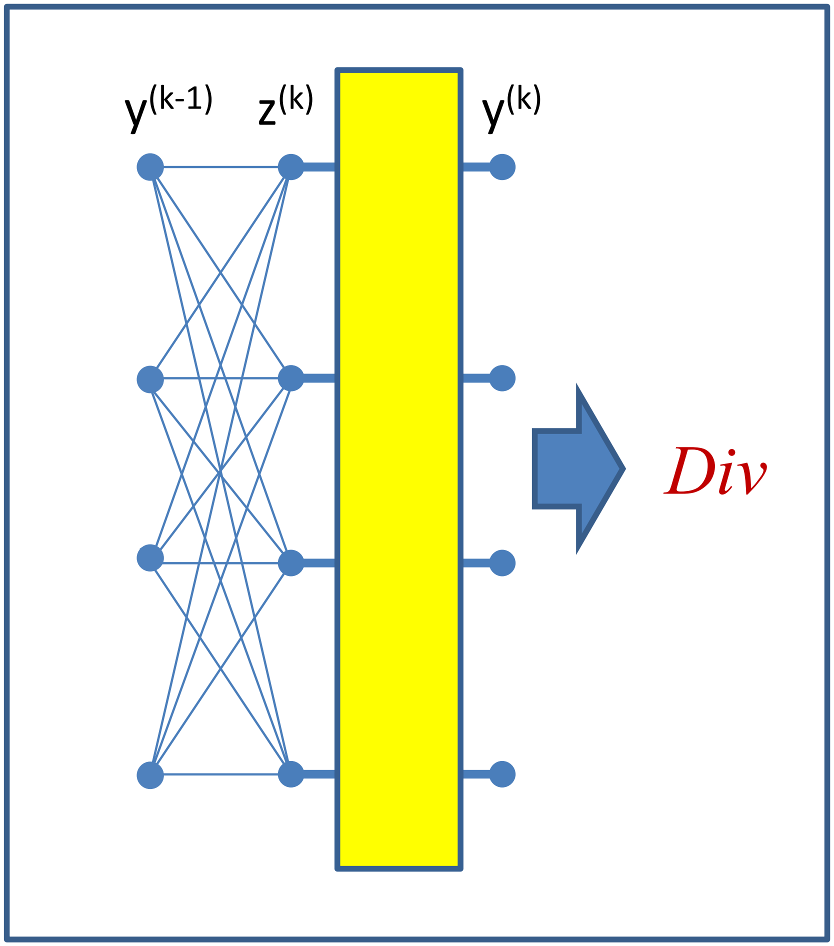
Example Vector Activation: Softmax



$$y_i^{(k)} = \frac{exp\left(z_i^{(k)}\right)}{\sum_j exp\left(z_j^{(k)}\right)}$$

$$\boxed{\frac{\partial Div}{\partial z_i^{(k)}} = \sum_j \frac{\partial Div}{\partial y_j^{(k)}} \frac{\partial y_j^{(k)}}{\partial z_i^{(k)}}}$$

Example Vector Activation: Softmax

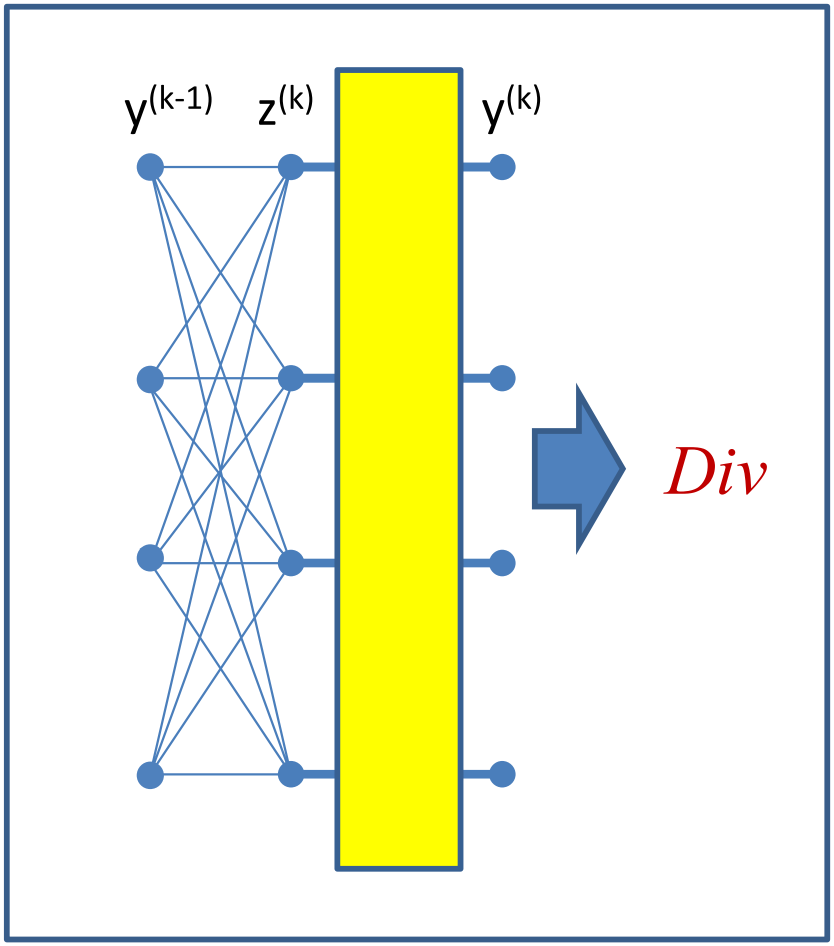


$$y_i^{(k)} = \frac{exp\left(z_i^{(k)}\right)}{\sum_j exp\left(z_j^{(k)}\right)}$$

$$\boxed{\frac{\partial Div}{\partial z_i^{(k)}} = \sum_j \frac{\partial Div}{\partial y_j^{(k)}} \frac{\partial y_j^{(k)}}{\partial z_i^{(k)}}}$$

$$\boxed{\frac{\partial y_j^{(k)}}{\partial z_i^{(k)}} = \begin{cases} y_i^{(k)}\left(1 - y_i^{(k)}\right) & \text{if } i = j \\ -y_i^{(k)} y_j^{(k)} & \text{if } i \neq j \end{cases}}$$

Example Vector Activation: Softmax



$$y_i^{(k)} = \frac{exp\left(z_i^{(k)}\right)}{\sum_j exp\left(z_j^{(k)}\right)}$$

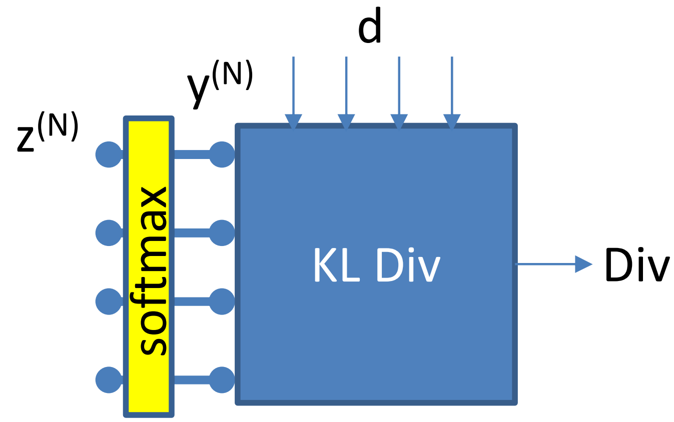
$$\boxed{\frac{\partial Div}{\partial z_i^{(k)}} = \sum_j \frac{\partial Div}{\partial y_j^{(k)}} \frac{\partial y_j^{(k)}}{\partial z_i^{(k)}}}$$

$$\boxed{\frac{\partial y_j^{(k)}}{\partial z_i^{(k)}} = \begin{cases} y_i^{(k)}\left(1 - y_i^{(k)}\right) & \text{if } i = j \\ -y_i^{(k)} y_j^{(k)} & \text{if } i \neq j \end{cases}}$$

$$\boxed{\frac{\partial Div}{\partial z_i^{(k)}} = \sum_j \frac{\partial Div}{\partial y_j^{(k)}} y_j^{(k)} \left(\delta_{ij} - y_i^{(k)}\right)}$$

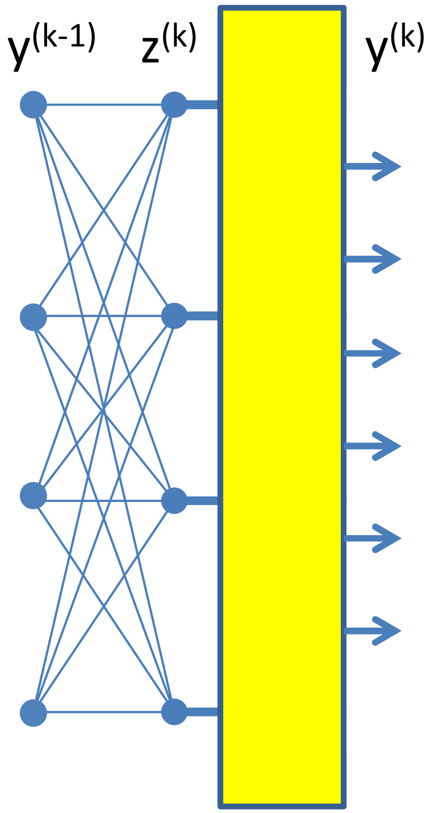
- For future reference
- δ_{ij} is the Kronecker delta: $\delta_{ij} = 1 \text{ if } i = j, \quad 0 \text{ if } i \neq j$

Backward Pass for *softmax output layer*



- Output layer (N) :
  - For $i = 1 \dots D_N$
    - $\frac{\partial Div}{\partial y_i^{(N)}} = \frac{\partial Div(Y,d)}{\partial y_i}$
    - $\frac{\partial Div}{\partial z_i^{(N)}} = \sum_j \frac{\partial Div(Y,d)}{\partial y_j^{(N)}} y_i^{(N)} \left(\delta_{ij} - y_j^{(N)}\right)$
    - $\frac{\partial D}{\partial w_{ij}^{(N)}} = y_i^{(N-1)} \frac{\partial Div}{\partial z_j^{(N)}}$ for $j = 0 \dots D_{N-1}$
- For layer $k = N - 1\ downto\ 1$
  - For $i = 1 \dots D_k$
    - $\frac{\partial Div}{\partial y_i^{(k)}} = \sum_j w_{ij}^{(k+1)} \frac{\partial Div}{\partial z_j^{(k+1)}}$
    - $\frac{\partial Div}{\partial z_i^{(k)}} = \frac{\partial Div}{\partial y_i^{(k)}} f_k'\left(z_i^{(k)}\right)$
    - $\frac{\partial Div}{\partial w_{ij}^{(k)}} = y_i^{(k-1)} \frac{\partial Div}{\partial z_j^{(k)}}$ for $j = 0 \dots D_{k-1}$

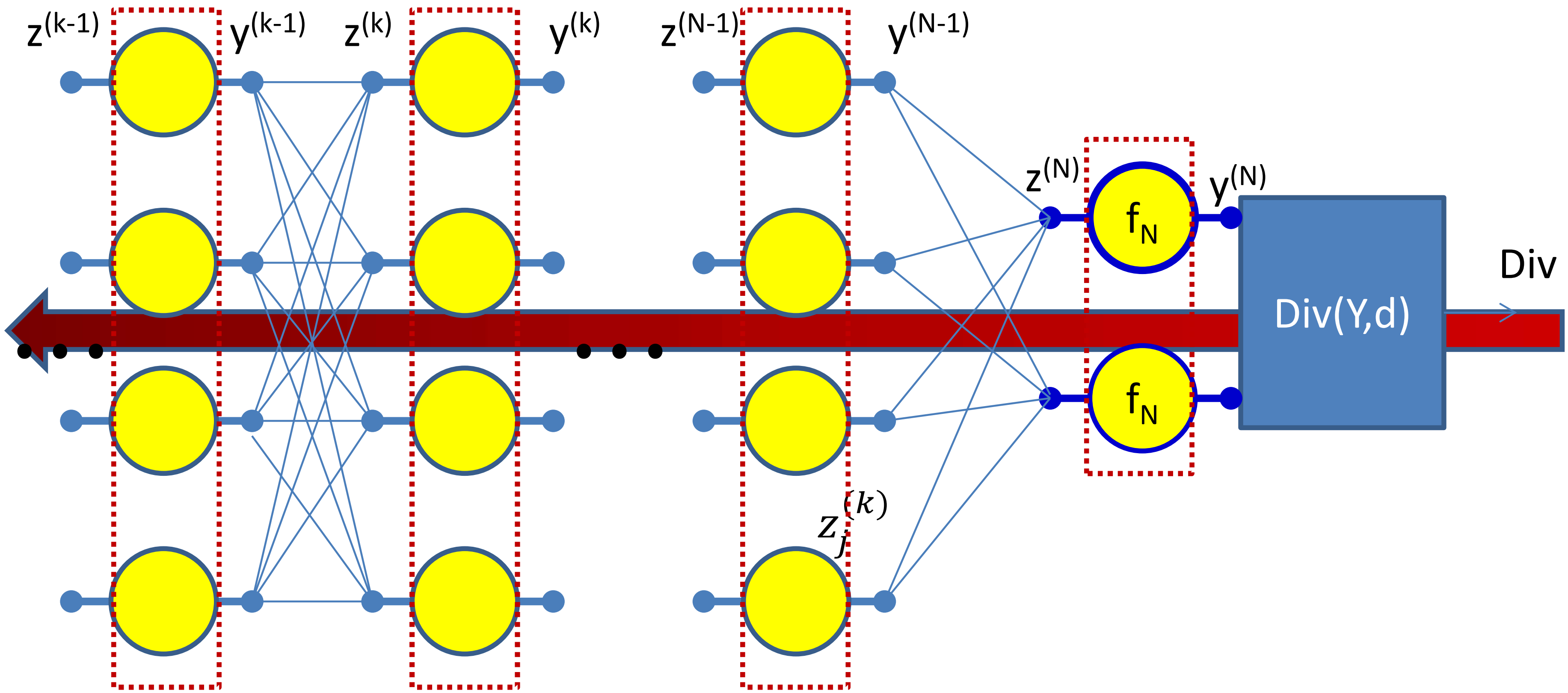
Vector Activations



$$\begin{bmatrix} y_1^{(k)} \\ y_2^{(k)} \\ \vdots \\ y_M^{(k)} \end{bmatrix} = f\left(\begin{bmatrix} z_1^{(k)} \\ z_2^{(k)} \\ \vdots \\ z_D^{(k)} \end{bmatrix}\right)$$

- In reality the vector combinations can be anything
  - E.g. linear combinations, polynomials, logistic (softmax), etc.

Gradients: Backward Computation



For $k = N \ldots 1$

For $i = 1$:*layer width*

If layer has vector activation

$$\frac{\partial Div}{\partial z_i^{(k)}} = \sum_j \frac{\partial Div}{\partial y_j^{(k)}} \frac{\partial y_j^{(k)}}{\partial z_i^{(k)}}$$

$\Leftrightarrow$

Else if activation is scalar

$$\frac{\partial Div}{\partial z_i^{(k)}} = \frac{\partial Div}{\partial y_i^{(k)}} \frac{\partial y_i^{(k)}}{\partial z_i^{(k)}}$$

$$\frac{\partial Div}{\partial y_i^{(k-1)}} = \sum_j w_{ij}^{(k)} \frac{\partial Div}{\partial z_j^{(k)}}$$

$$\frac{\partial Div}{\partial w_{ij}^{(k)}} = y_i^{(k-1)} \frac{\partial Div}{\partial z_j^{(k)}}$$