Video 22

Monday, November 2, 2020 11:48 AM

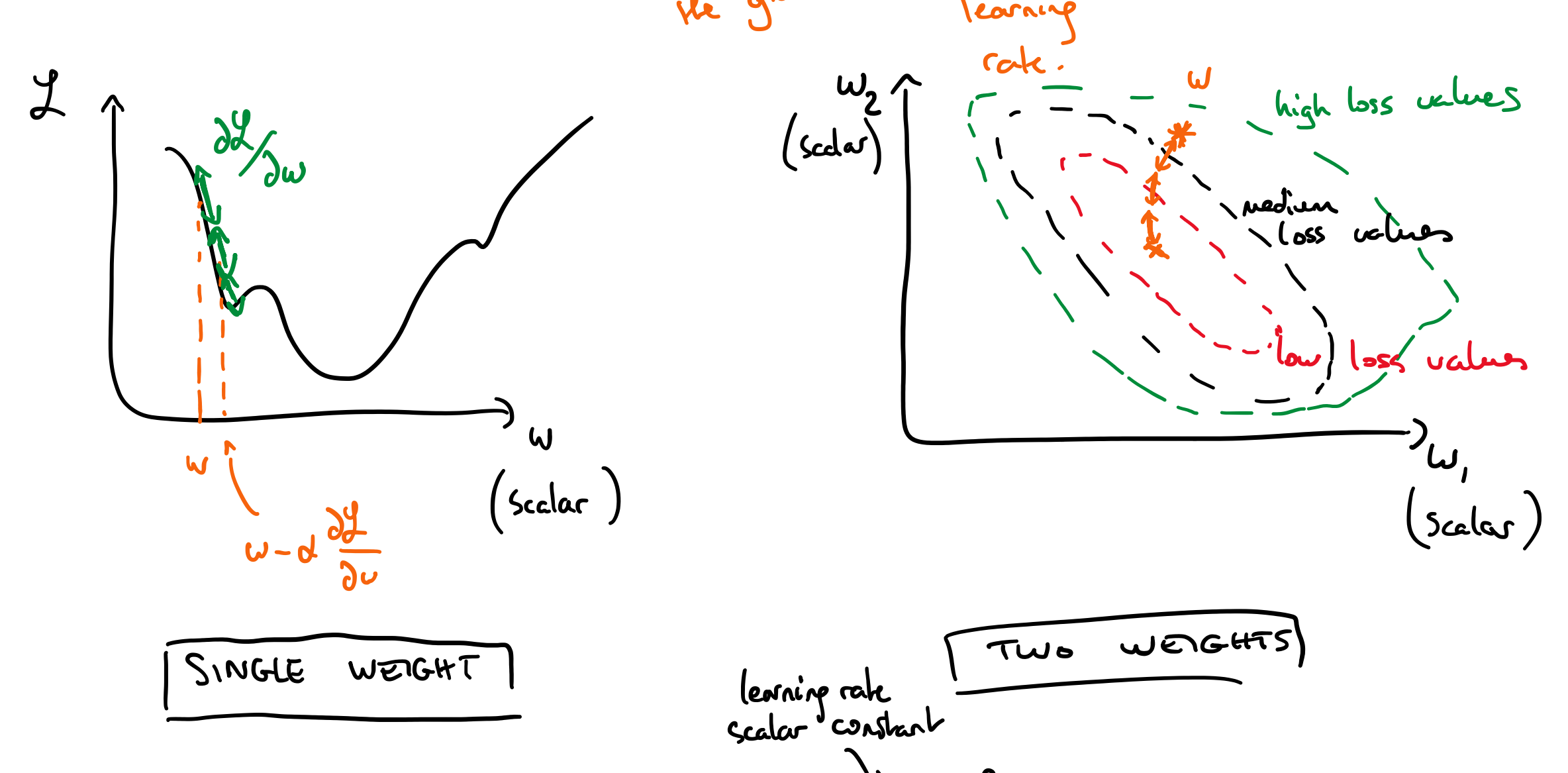
Backpropagation algorithm:

Find the values of a deep neural network's (DNN) weights
given: - a training set
- an objective function / loss / cost.

Gradient descent updates the parameters in the direction of steepest descent (opposite to the gradient)

$$w \leftarrow w - \alpha \frac{\partial \mathcal{L}}{\partial w}$$

weight (w); opposite direction to the gradient (−); learning rate (α); partial derivative of the loss wrt to weight w ($\frac{\partial \mathcal{L}}{\partial w}$)



Vector notation

$$\vec{w} \leftarrow \vec{w} - \alpha \frac{\partial \mathcal{L}}{\partial \vec{w}}$$

$\vec{w}$: d-dimensional vector; α: learning rate scalar constant; $\frac{\partial \mathcal{L}}{\partial \vec{w}}$: d-dimensional vector

$$\frac{\partial \mathcal{L}}{\partial \vec{w}} = \begin{bmatrix} \partial\mathcal{L}/\partial w_1 \\ \partial\mathcal{L}/\partial w_2 \\ \vdots \\ \partial\mathcal{L}/\partial w_d \end{bmatrix}$$

assuming the model has d weights.

Chain rule

If we have $f(x)$ and $x(t)$ which are both univariate functions,

$$\frac{d}{dt}\left(f(x(t))\right) = \frac{df}{dx}\frac{dx}{dt}$$

Motivating example:

For simplicity (of exposition):

- single training example
- single input feature (input is a scalar)
- activation function: logistic function
- loss: least squares

Model:

$$z = wx + b$$

(z scalar, w scalar, input x is scalar, b scalar)

$$y = \sigma(z) = \frac{1}{1+e^{-z}}$$

$$\mathcal{L} = \frac{1}{2}(y-t)^2$$

Add a term to our loss called a regularizer
(a way to get a "simpler" explanation for the data)

Regularizer: $\frac{\lambda}{2} w^2$ (encourage weight to be close to 0)

λ: hyperparameter (the larger λ is, the more we give importance to the regularizer during optimization)

$$\mathcal{L} = \frac{1}{2}(y-t)^2 + \frac{\lambda}{2} w^2$$

With $R = \frac{1}{2}w^2$, we write:

$$\mathcal{L}_{reg} = \frac{1}{2}(y-t)^2 + \lambda R$$

Goal: to train this model, we need to compute:

$$\frac{\partial \mathcal{L}_{reg}}{\partial w} \qquad \frac{\partial \mathcal{L}_{reg}}{\partial b}$$

Naive approach based on calculus:

$$\frac{\partial \mathcal{L}_{reg}}{\partial w} = \frac{\partial}{\partial w}\left(\frac{1}{2}\left(\sigma(wx+b)-t\right)^2 + \frac{\lambda}{2}w^2\right) \quad \text{by definition}$$

$$= \frac{1}{2}\frac{\partial}{\partial w}\left(\sigma(wx+b)-t\right)^2 + \frac{\lambda}{2}\frac{\partial w^2}{\partial w}$$

(apply the chain rule; $\frac{\partial w^2}{\partial w} = 2w$)

$(u^2)' = 2u'u$ where u is a function

$$= \frac{1}{2}\cdot 2\left(\sigma(wx+b)-t\right)\frac{\partial}{\partial w}\left(\sigma(wx+b)-t\right) + \lambda w$$

(t: constant, does not depend on w; apply the chain rule)

$$= \left(\sigma(wx+b)-t\right)\sigma'(wx+b)\frac{\partial}{\partial w}(wx+b) + \lambda w$$

(b: constant, does not depend on w)

$$\boxed{\frac{\partial \mathcal{L}_{reg}}{\partial w} = \left(\sigma(wx+b)-t\right)\sigma'(wx+b)\,x + \lambda w}$$

$$\frac{\partial \mathcal{L}_{reg}}{\partial b} = \frac{\partial}{\partial b}\left(\frac{1}{2}\left(\sigma(wx+b)-t\right)^2 + \frac{\lambda}{2}w^2\right)$$

(independent of b ⇒ derivative $\frac{\partial}{\partial b} = 0$)

$$= \frac{1}{2}\cdot 2\left(\sigma(wx+b)-t\right)\frac{\partial}{\partial b}\left(\sigma(wx+b)-t\right) + 0$$

(t independent of b; $\frac{\partial (wx+b)}{\partial b} = 1$ & applied the chain rule.)

$$\boxed{\frac{\partial \mathcal{L}_{reg}}{\partial b} = \left(\sigma(wx+b)-t\right)\sigma'(wx+b)}$$

Several drawbacks

1. These calculations are cumbersome.
2. Calculations involved a lot of redundant work.
3. Final expressions of $\frac{\partial \mathcal{L}_{reg}}{\partial w}$ and $\frac{\partial \mathcal{L}_{reg}}{\partial b}$ involved repeated terms.

Multivariable chain rule

$$\frac{d}{dt} f(x(t), y(t)) = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$

(f depends on x, which depends on t; f depends on y, which depends on t)

Notation

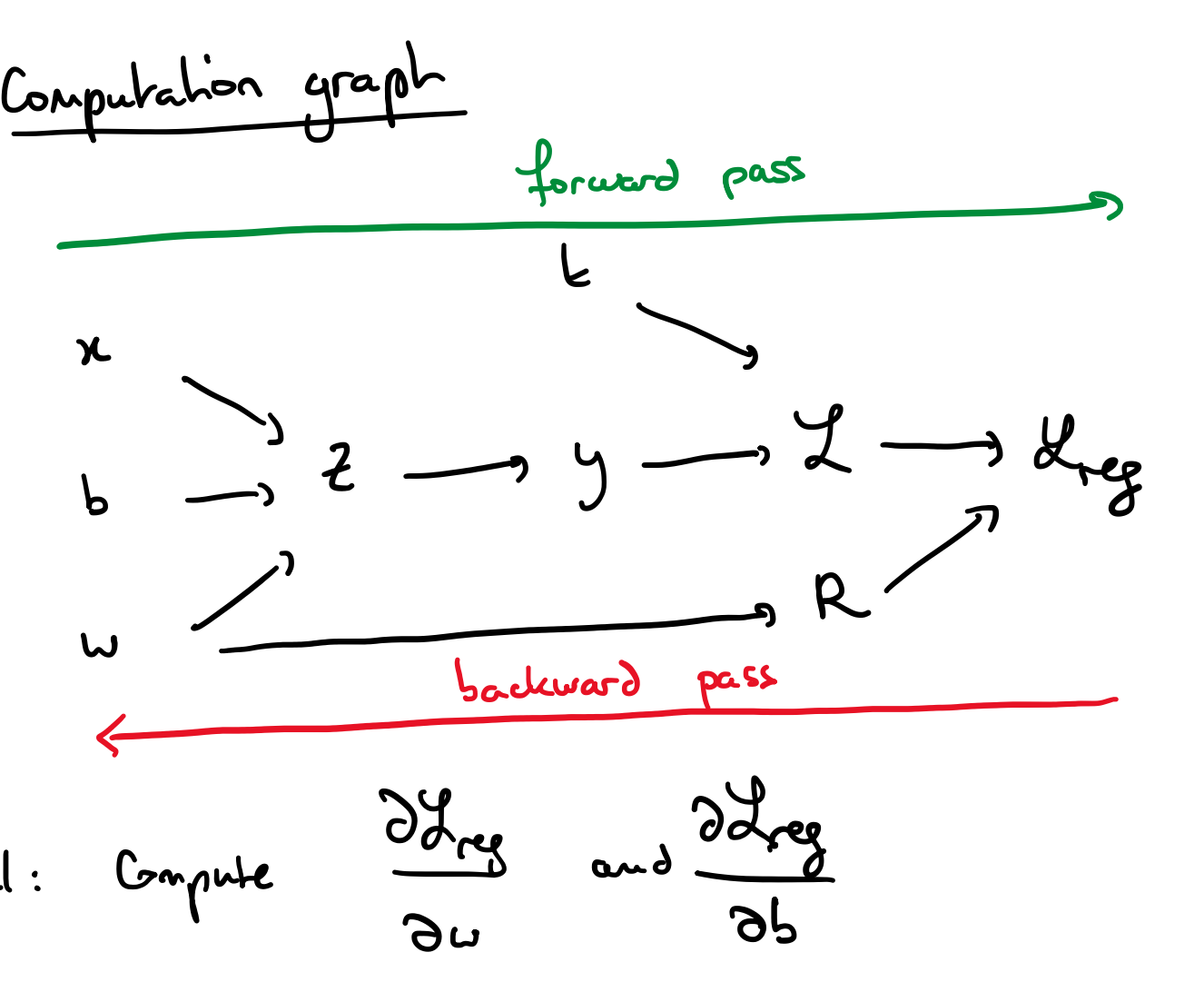
$$\boxed{\bar{v} \triangleq \frac{\partial \mathcal{L}}{\partial v}}$$

($\mathcal{L}$: any loss we would like to minimize (can be a regularized loss))

We can rewrite the chain rule: $\bar{t} = \bar{x}\frac{dx}{dt} + \bar{y}\frac{dy}{dt}$ (if $f = \mathcal{L}$)

Backpropagation:

Computation graph



$$z = wx + b$$
$$y = \sigma(z)$$
$$\mathcal{L} = \frac{1}{2}(y-t)^2$$
$$R = \frac{1}{2}w^2$$
$$\boxed{\mathcal{L}_{reg} = \mathcal{L} + \lambda R}$$

Goal: Compute $\frac{\partial \mathcal{L}_{reg}}{\partial w}$ and $\frac{\partial \mathcal{L}_{reg}}{\partial b}$

equivalent $\bar{w}$ and $\bar{b}$ since $\bar{w} = \frac{\partial \mathcal{L}_{reg}}{\partial w}$, $\bar{b} = \frac{\partial \mathcal{L}_{reg}}{\partial b}$

Backpropagation algorithm

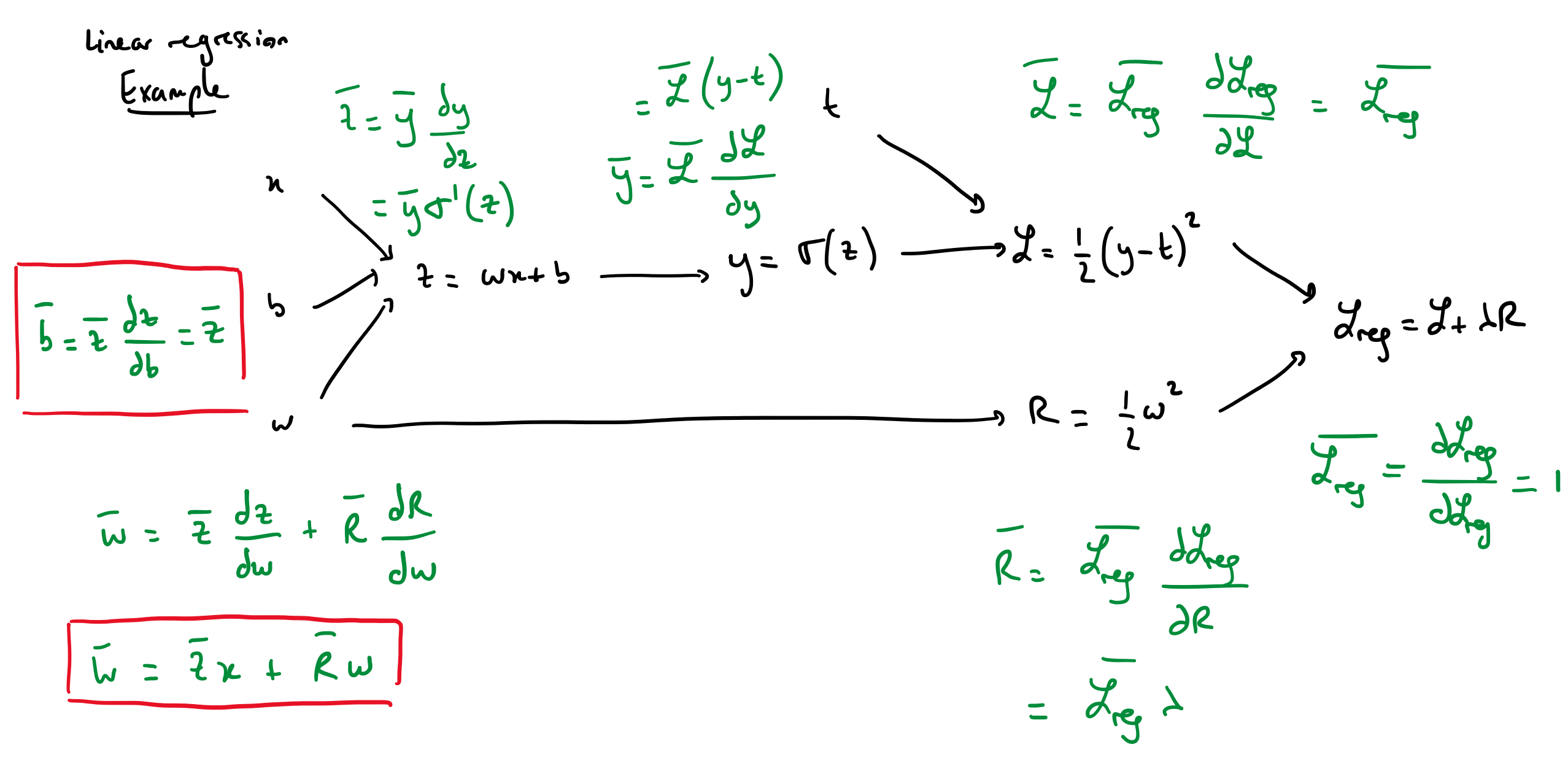
Forward pass:
For $i \in 1..N$
  Compute v_i as a function of $Pa(v_i)$ (parents of v_i)

Backward pass:
$\bar{v}_N = 1$ ← $\bar{v}_N = \frac{\partial \mathcal{L}_{reg}}{\partial \mathcal{L}_{reg}} = 1$
For i in N−1, ..., 1
  $\bar{v}_i = \sum_{j \in Ch(v_i)} \bar{v}_j \frac{\partial v_j}{\partial v_i}$ } application of the chain rule
  ($Ch(v_i)$: children of v_i)

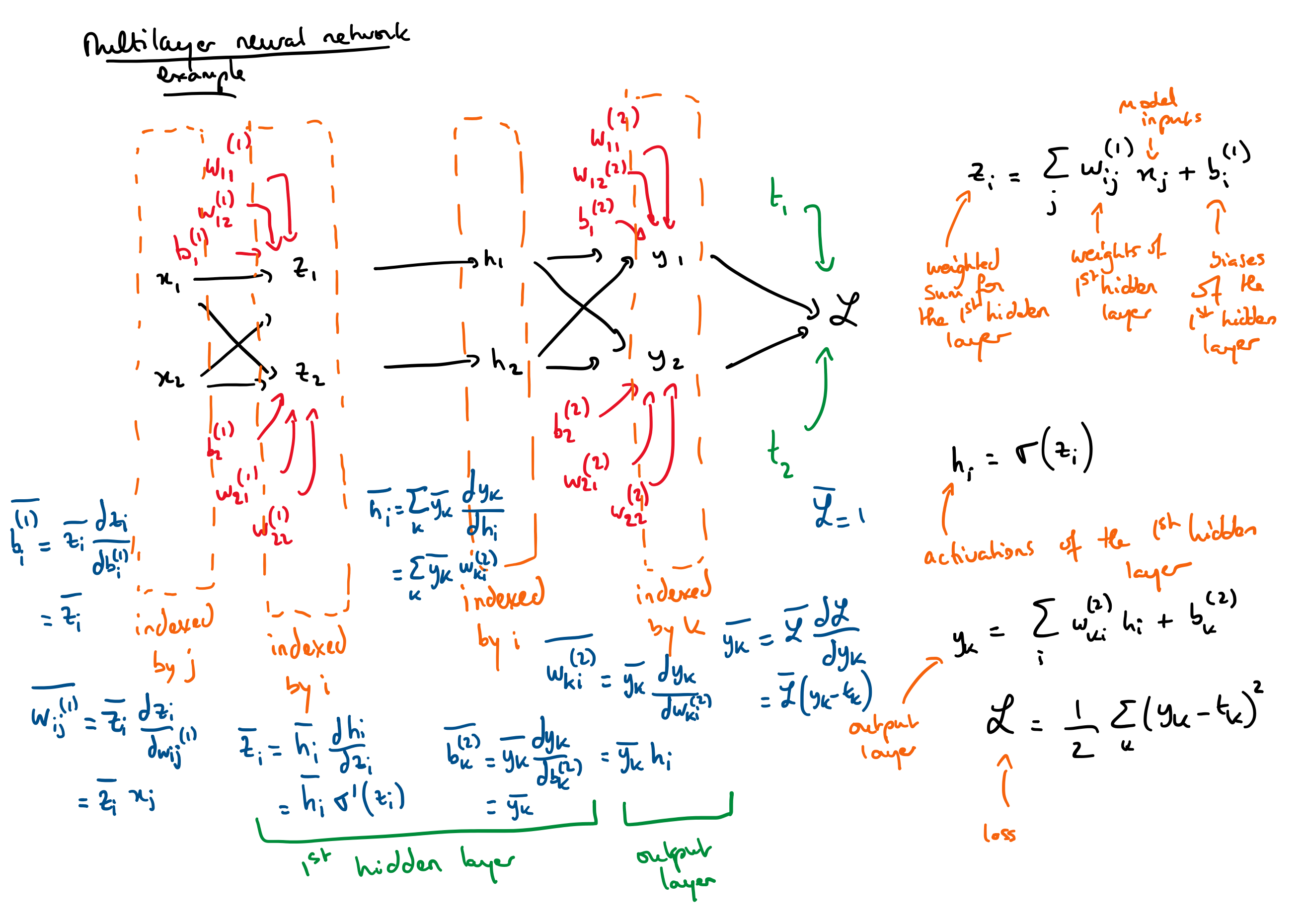
v_i are the nodes in the computation graph
v_N is the quantity we'd like to get derivatives of, it's our loss (here, regularized loss)

Here $v_1, v_2, \dots v_N$ are the nodes of the computational graph in topological order (parents come before children).

Linear regression Example



Multilayer neural network example



$$z_i = \sum_j w_{ij}^{(1)} x_j + b_i^{(1)}$$

(z_i: weighted sum for the 1st hidden layer; $w_{ij}^{(1)}$: weights of 1st hidden layer; x_j: model inputs; $b_i^{(1)}$: biases of the 1st hidden layer)

$$h_i = \sigma(z_i)$$

(activations of the 1st hidden layer)

$$y_k = \sum_i w_{ki}^{(2)} h_i + b_k^{(2)}$$

(output layer)

$$\mathcal{L} = \frac{1}{2}\sum_k (y_k - t_k)^2$$

(loss)

Forward pass:

$$\vec{z} = W^{(1)}\vec{x} + \vec{b}^{(1)}$$
$$\vec{h} = \sigma(\vec{z})$$
$$\vec{y} = W^{(2)}\vec{h} + \vec{b}^{(2)}$$
$$\mathcal{L} = \frac{1}{2}\|y - t\|^2$$

Backward pass:

$$\bar{\mathcal{L}} = 1$$
$$\bar{\vec{y}} = \bar{\mathcal{L}}(\vec{y} - \vec{t})$$
$$\overline{W^{(2)}} = \bar{\vec{y}}\,\vec{h}^T$$
$$\overline{\vec{b}^{(2)}} = \bar{\vec{y}}$$
$$\bar{\vec{h}} = W^{(2)T}\bar{\vec{y}}$$
$$\bar{\vec{z}} = \bar{\vec{h}} \cdot \sigma'(\vec{z})$$
$$\overline{W^{(1)}} = \bar{\vec{z}}\,\vec{x}^T$$
$$\overline{\vec{b}^{(1)}} = \bar{\vec{z}}$$