Computational Graph for Multi-layer Perceptron (MLP)

definitions:

input : $x \in \mathbb{R}^n$; $y \in \mathbb{R}^m$

Our MLP: $f(x\,;\,W)$ weight matrix parameterizing our MLP

loss function: $L(f(x, W), y)$

$\sigma(\cdot)$: activation function, e.g., tanh, sigmoid

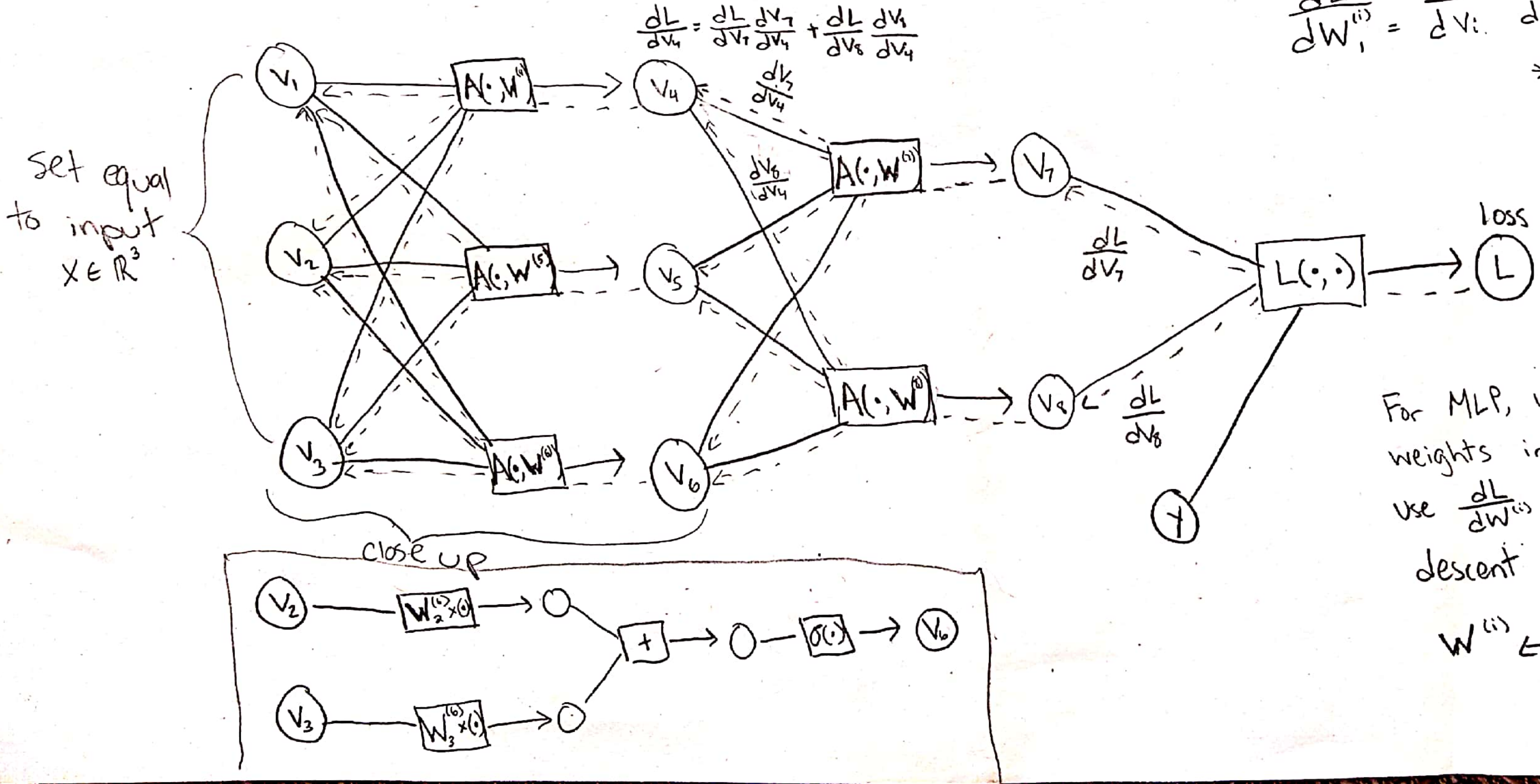
* For simplicity, we consider $A(x, W) = \sigma(x^T W)$ as a primitive

How to calculate $\frac{dL}{dW^{(i)}}$?

$V_i = A(\cdot, W^{(i)}) = \sigma(x^T W^{(i)}) = \sigma(x_1 W_1^{(i)} + \dots + x_n W_n^{(i)})$

$\frac{dL}{dW_1^{(i)}} = \frac{dL}{dV_i} \cdot \frac{dV_i}{dW_1^{(i)}} = \frac{dL}{dV_i}\, \sigma'(x^T W^{(i)})\, x_1^{(i)}$

*here we use $x = \langle V_{pa(i)} \rangle$, the set of nodes coming into V_i



For MLP, we optimize over W, all weights in our MLP. We can then use $\frac{dL}{dW^{(i)}}$ in e.g., gradient descent updates

$W^{(i)} \leftarrow W^{(i)} - \alpha \frac{dL}{dW^{(i)}}$

learning rate