Deep Learning basics

Backpropagation

Learning goals

- Understand the concept of regularization
- Understand L2 regularization in more detail

- Forward propagation: Input information $\vec{x}$ propagates through network to produce output $\hat{y}$ (and cost $J(\vec{\theta})$ in training)
- Back-propagation:
  - compute gradient w.r.t. model parameters
  - Cost gradient propagates backwards through the network
- Back-propagation is part of learning procedure (e.g. stochastic gradient descent), not learning procedure in itself.

CHAIN RULE OF CALCULUS: REAL FUNCTIONS

- Let

$$x, y, z \in \mathbb{R}$$

$$f, g : \mathbb{R} \to \mathbb{R}$$

$$y = g(x)$$

$$z = f(g(x)) = f(y)$$

- Then

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

CHAIN RULE OF CALCULUS: MULTIVARIATE FUNCTIONS

- Let

$$\vec{x} \in \mathbb{R}^m, \vec{y} \in \mathbb{R}^n, z \in \mathbb{R}$$

$$f : \mathbb{R}^n \to \mathbb{R}$$

$$g : \mathbb{R}^m \to \mathbb{R}^n$$

$$\vec{y} = g(\vec{x})$$

$$z = f(g(\vec{x})) = f(\vec{y})$$

- Then

$$\frac{\partial z}{\partial x_i} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x_i} + \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x_i} + \ldots + \frac{\partial z}{\partial y_n}\frac{\partial y_n}{\partial x_i} = \sum_{j=1}^{n} \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}$$

- In order to write this in vector notation, we need to define the Jacobian matrix.

JACOBIAN

- The Jacobian matrix is the matrix of all first-order partial derivatives of a vector-valued function.

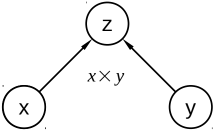
$$\vec{J} = \frac{\partial \vec{g}(\vec{x})}{\partial \vec{x}} = \begin{bmatrix} \frac{\partial g(\vec{x})_1}{\partial x_1} & \cdots & \frac{\partial g(\vec{x})_1}{\partial x_m} \\ \frac{\partial g(\vec{x})_2}{\partial x_1} & & \frac{\partial g(\vec{x})_2}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial g(\vec{x})_n}{\partial x_1} & \cdots & \frac{\partial g(\vec{x})_n}{\partial x_m} \end{bmatrix}$$

- How to write in terms of gradients?
- We can write the chain rule as:

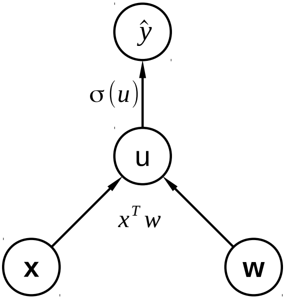
$$\nabla_{\vec{x}} z =$$

VIEWING THE NETWORK AS A GRAPH

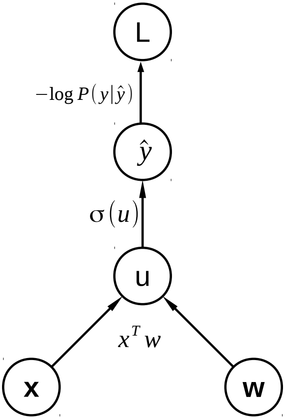
- Nodes are function outputs (can be scalar or vector valued)
- Arrows are inputs
- Example: Scalar multiplication $z = xy$.



WHICH FUNCTION?

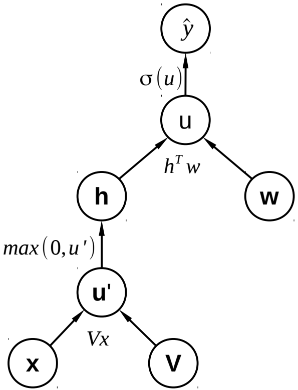


GRAPH WITH COST



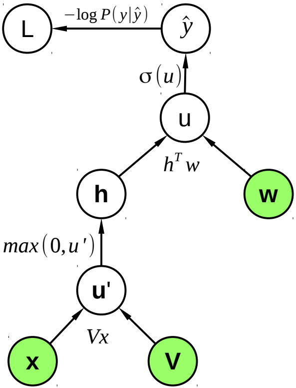
WHICH FUNCTION?

- Parameter vectors can be converted to matrix as needed.



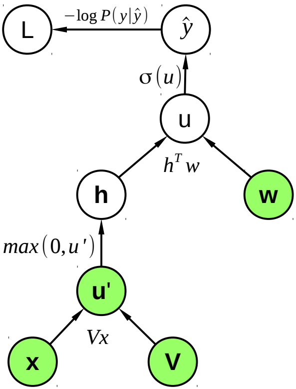
FORWARD PASS

- Green: known or computed.



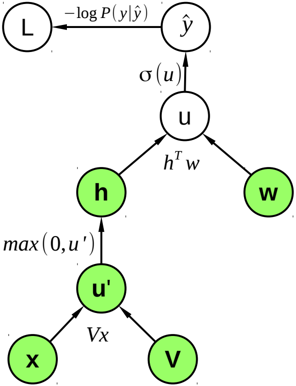
FORWARD PASS

- Green: known or computed.



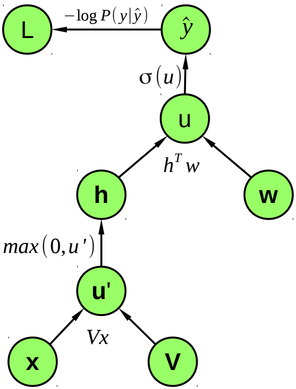
FORWARD PASS

- Green: known or computed.



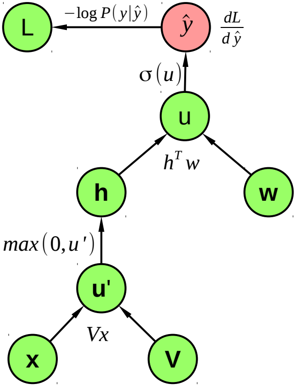
FORWARD PASS

- End of forward pass (some steps skipped).



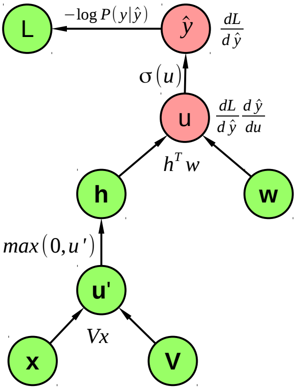
BACKWARD PASS

- Red: gradient of cost computed for node.



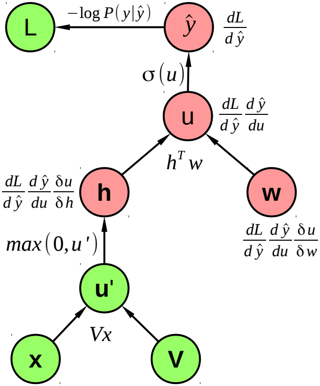
BACKWARD PASS

- Red: gradient of cost computed for node.



BACKWARD PASS

- Red: gradient of cost computed for node.



BACKWARD PASS

- We have the gradients for all parameters, let's use them for SGD.