Deep Learning Refresher

Week 2

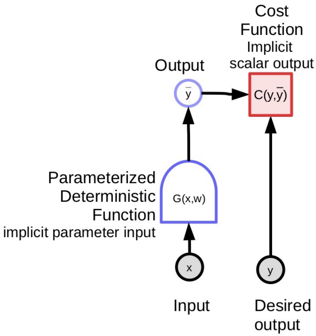
Tirtharaj Dash

Gradient Descent I

- A parameterised model is written as

$$\bar{y} = G(x, w)$$

where, w is a model parameter.



Gradient Descent II

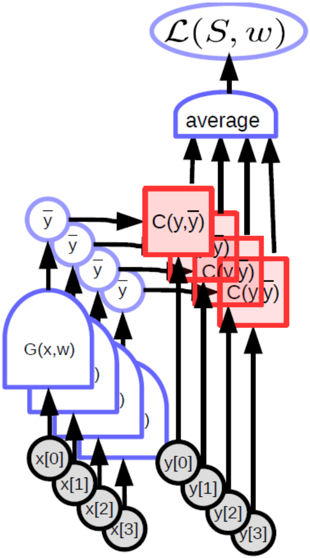
- In the figure, G takes the input argument x, and produces an output $\overline{y}$.
- $C(y, \overline{y})$ represents a scalar cost function. It is also called a loss function denoted as $L(\cdots)$.
- Gradient descent update of parameter w based on a loss function:

$$w \leftarrow w - \eta \frac{\partial L(S, w)}{\partial w}$$

Here, $S = \{(x_i, y_i) : i \in \{1, \ldots, m\}\}$ and

$$L(S, w) = \frac{1}{m} \sum_{i=1}^{m} L(x_i, y_i, w)$$

Gradient Descent III



Gradient Descent IV

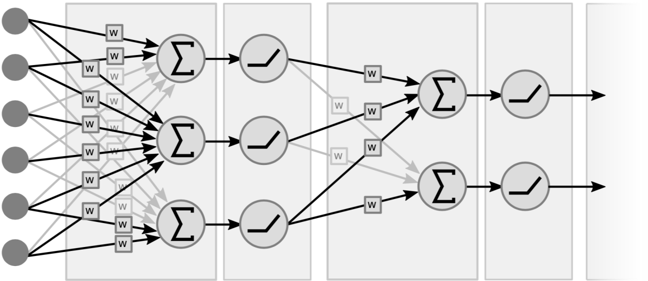
- Stochastic version of the update (also called Stochastic GD):

$$w \leftarrow w - \eta \frac{\partial L(x_k, y_k, w)}{\partial w}$$

Here, k is randomly drawn from $\{1, \ldots, N\}$.

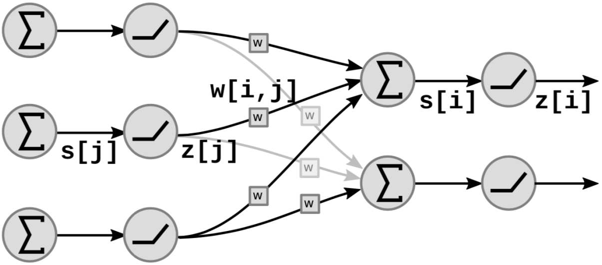
Neural Netw. I

- A 2-layer neural network



Neural Netw. II

- Computation path



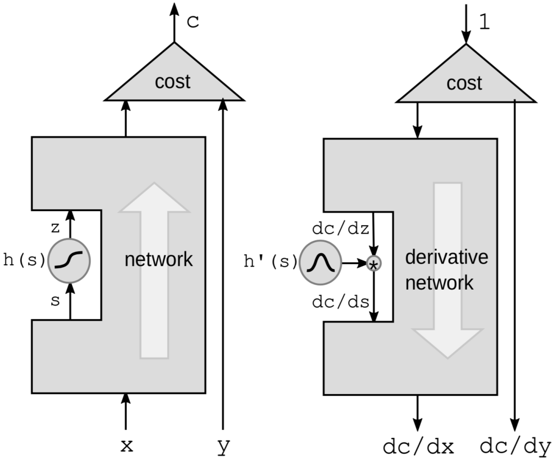
In the graph, $s[i]$ is the weighted sum of unit i:

$$s_i = \sum_{j \in Pred(i)} w_{ij} \cdot z_j$$

- Then, $z_i = f(s_i)$, where f is a non-linear function.

Backpropagation through nonlinear function I

- Let h be a nonlinear function.



Backpropagation through nonlinear function II

- s is the sum and z is $h(s)$. The cost C is computed by taking z and y.
- This results in a chain of computation: $(x, y) \to s \to z \to C$.
- So,

$$\frac{\partial C}{\partial s} = \frac{\partial C}{\partial z}\frac{\partial z}{\partial s} = \frac{\partial C}{\partial z}h'(s)$$

Backpropagation through nonlinear function III

- Rewriting:

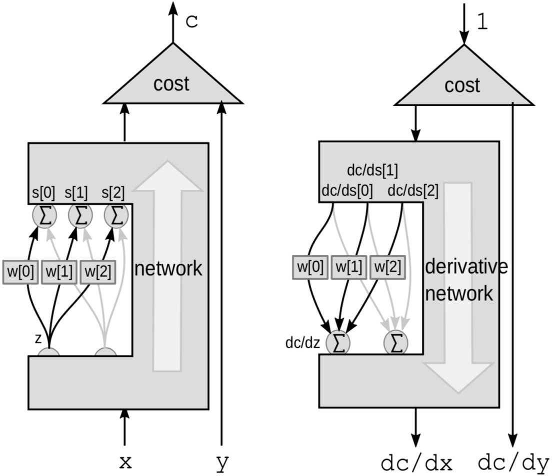
$$\frac{\partial z}{\partial s} = h'(s) \Rightarrow dz = \partial s \cdot h'(s)$$

  means: if we change s by some amount, z is also changed.
- Similarly, the above change also changes C:

$$\partial C = \partial z \frac{\partial C}{\partial z} = \partial s \cdot h'(s) \frac{\partial C}{\partial z}$$

Backpropagation through weighted sum I

- Z influences several branches:



Backpropagation through weighted sum II

- So,

$$\partial s_i = w_i \cdot \partial z; \quad i \in 0, 1, 2$$

- And,

$$\partial C = \sum_{i=0}^{2} \partial s_i \cdot \frac{\partial C}{\partial s_i} = \sum_{i=0}^{2} w_i \cdot \partial z \cdot \frac{\partial C}{\partial s_i}$$

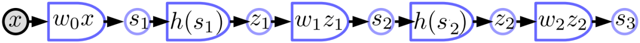
- We can take ∂z out and move it to the l.h.s:

$$\frac{\partial C}{\partial z} = \sum_{i=0}^{2} w_i \cdot \frac{\partial C}{\partial s_i}$$

C varies by the sum of the 3 variations.

Traditional Neural Net I

- Linear block $s_{k+1} = w_k z_k$; nonlinear block: $z_k = h(s_k)$



- This is a very simple neural network with 3 layers. The model parameters w_ks are weight matrices; h is a nonlinear function that operates on the input (s_k) elementwise.
- x (vector) —perform matrix multiplication$(w_0 x) \rightarrow s_1$(vector) (apply elementwise $h(\cdot)$)$\rightarrow z_1$ (vector)
- z_1 then becomes the input vector for the next layer.

(More about PyTorch implementation in the Lab)

Backpropagation through functional module

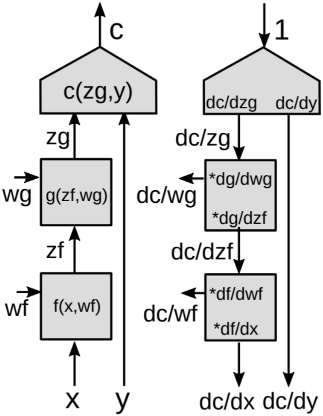
- A more general form of backprop:

  Using chain rule:

  $$\frac{\partial C}{\partial zf} = \frac{\partial C}{\partial zg}\frac{\partial zg}{\partial zf}$$

  $$[1 \times df] = [1 \times dg] \times [dg \times df]$$

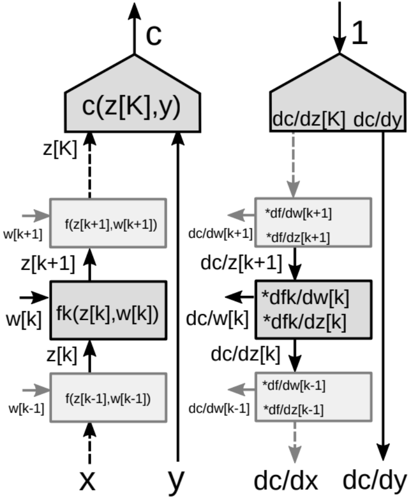
  where, $d_{\_}$ denotes the size. Note that partial derivative of a scalar function w.r.t. a vector is a vector (first term) and partial derivative of a vector function w.r.t. a vector is a matrix (second term)



- Details on Hessian and Jacobian is in the the tutorial slides.

Backpropagation through multistage graph I

- Neural network with many functional modules:



Backpropagation through multistage graph II

- Uses chain rule of vector functions: Gradient of a vector function of size m w.r.t. a vector n is a Jacobian matrix of dimension $m \times n$

$$\left(\frac{\partial \boldsymbol{f}}{\partial x}\right)_{ij} = \frac{\partial f_i}{\partial x_j}$$

- From the block diagram, we can see the path and apply the chain rule as:

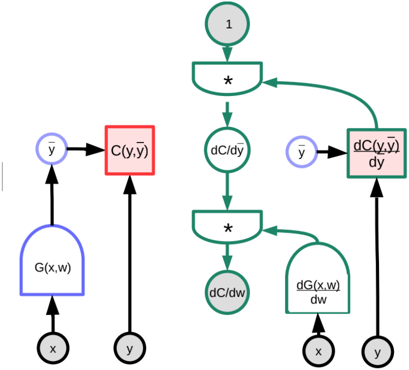
$$\frac{\partial C}{\partial z_k} = \frac{\partial C}{\partial z_{k+1}} \frac{\partial z_{k+1}}{\partial z_k} = \frac{\partial C}{\partial z_{k+1}} \frac{\partial f_k(z_k, w_k)}{\partial z_k}$$

$$\frac{\partial C}{\partial w_k} = \frac{\partial C}{\partial z_{k+1}} \frac{\partial z_{k+1}}{\partial w_k} = \frac{\partial C}{\partial z_{k+1}} \frac{\partial f_k(z_k, w_k)}{\partial w_k}$$

- Two Jacobian matrices for the module:
  - One w.r.t. $z[k]$
  - One w.r.t. $w[k]$

Example I

- Consider an example graph below (the right graph is the corresponding gradient graph):



Example II

- The gradients:

$$\frac{\partial C(y,\bar{y})}{\partial w} = \frac{\partial C(y,\bar{y})}{\partial \bar{y}}\frac{\partial \bar{y}}{\partial w} = \frac{\partial C(y,\bar{y})}{\partial \bar{y}}\frac{\partial G(x,w)}{\partial w}$$

- Dimensions: $y, \bar{y} : [M \times 1]$ i.e. M-dimensional output; $w : [N \times 1]$ i.e. N-dimensional weight vector. The model function $G(x, w)$ returns M-dimensional output. So, dimension of above gradient is

$$[1 \times N] = [1 \times M] \times [M \times N]$$

Basic Modules I

- Linear $Y = W \cdot X$:

$$\frac{\partial C}{\partial X} = W^{\mathsf{T}} \cdot \frac{\partial C}{\partial Y}$$

$$\frac{\partial C}{\partial W} = \frac{\partial C}{\partial Y} \cdot X^{\mathsf{T}}$$

- ReLU: $y = \max(0, x)$

$$\frac{\partial C}{\partial X} = \begin{cases} 0 & x < 0 \\ \frac{\partial C}{\partial Y} & \text{otherwise} \end{cases}$$

Basic Modules II

- Duplicate: $Y_1 = X$, $Y_2 = X$: X is propagates via two paths. While backpropagating, the gradients are summed:

$$\frac{\partial C}{\partial X} = \frac{\partial C}{\partial Y_1} + \frac{\partial C}{\partial Y_2}$$

- Add: $Y = X_1 + X_2$

$$\frac{\partial C}{\partial X_1} = \frac{\partial C}{\partial Y} \cdot 1$$

$$\frac{\partial C}{\partial X_2} = \frac{\partial C}{\partial Y} \cdot 1$$

Basic Modules III

- Max: $Y = \max(X_1, X_2) = \begin{cases} X_1 & X_1 \geq X_2 \\ X_2 & \text{otherwise} \end{cases}$

$$\frac{\partial Y}{\partial X_1} = \begin{cases} 1 & X_1 \geq X_2 \\ 0 & \text{otherwise} \end{cases}$$

and

$$\frac{\partial C}{\partial X_1} = \begin{cases} \frac{\partial C}{\partial Y} \cdot 1 & X_1 \geq X_2 \\ 0 & \text{otherwise} \end{cases}$$

Homework: Backprop in practice

Study the following:

- Use ReLU non-linearities (reason has been told in week1)
- Use cross-entropy loss for classification
- Use Stochastic Gradient Descent on minibatches
- Shuffle the training samples
- Normalize the input variables (zero mean, unit variance)
- Schedule to decrease the learning rate
- Use a bit of L_1 or L_2 regularization on the weights (or a combination)
- Use "dropout" for regularization
- Read relevant papers on various weight initialisation methods

Lab: Practice with MLP