Backpropagation of Deep Neural Network

Alex Tse
University College London

Training a DNN

Goal

To optimise the model parameters of DNN using gradient based methods, the crucial step is to *compute the gradients efficiently*.

Gradient calculation

Let $h^{(L)}$ denote the output layer of DNN with model parameters $\theta := (W^{(l)}, b^{(l)})_l$. Recall that the loss function $L(\theta|\mathcal{D})$ to be minimized is in the additive form that $L(\theta|\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} Q_\theta(x_i, y_i)$ and $Q_\theta(x, y) = Q(h^{(L)}(x) - y)$.

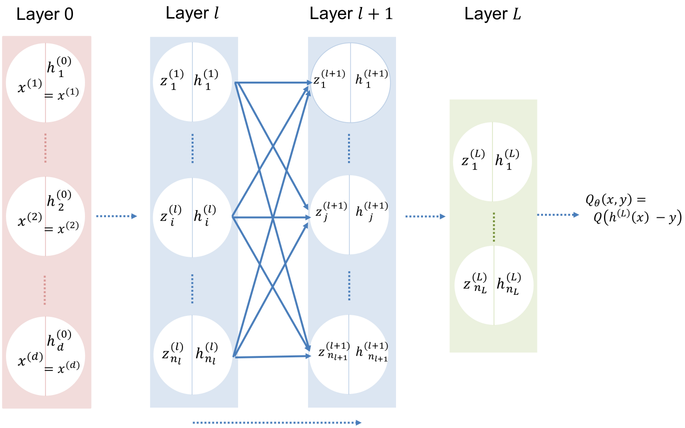
Problem: Compute $\nabla Q_\theta(x, y)$, i.e. $\partial_{W_{i,j}^{(l)}} Q_\theta(x, y)$ and $\partial_{b_i^{(l)}} Q_\theta(x, y)$.

Solution: Recursion and Chain Rule.

Backpropagation of DNN

- Forward Phase: Neural network evaluation.
- Backward Phase: Inductive gradient computation.

Forward Phase



For each $l \in \{0, 1, \cdots, L-1\}$, compute

$$z^{(l+1)}(x) = h^{(l)}(x) W^{(l)} + b^{(l)},$$
$$h^{(l+1)}(x) = \sigma_{l+1}(z^{(l+1)}(x)).$$

Backward Phase

Goal: To compute $\nabla Q_\theta(x, y)$, i.e.

$$\partial_{W_{i,j}^{(l)}} Q_\theta(x, y) \text{ and } \partial_{b_i^{(l)}} Q_\theta(x, y),$$

where $Q_\theta(x, y) = Q(h^{(L)}(x) - y)$.[a]
Solution: Recursion and Chain Rule.

$$\boxed{Q_\theta(x, y) = q(z^{(l+1)}, W^{(l+1)}, b^{(l+1)}, \cdots, W^{(L-1)}, b^{(L-1)}, y)}$$

which depends on $W^{(l)}$ and $b^{(l)}$ only via $z^{(l+1)} = z^{(l+1)}(x, W^{(0)}, b^{(0)}, \cdots, W^{(l)}, b^{(l)})$.

[a] For more details refer to [1, 2].

The important intermediate variable is

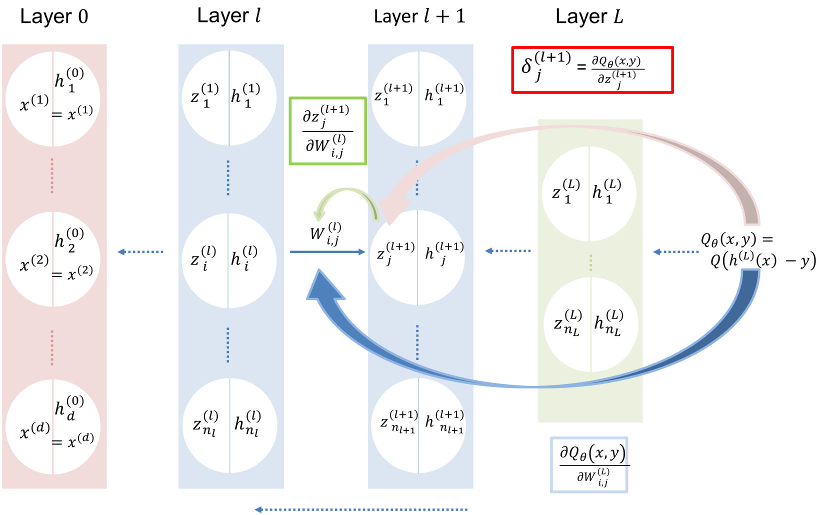
$$\delta_i^{(l)} := \partial_{z_i^{(l)}} Q_\theta(x, y),$$

as $\partial_{W^{(l)}} Q_\theta(x)$ and $\partial_{b^{(l)}} Q_\theta(x)$ follows

$$\begin{aligned}
\partial_{W_{i,j}^{(l)}} Q_\theta(x) &= \partial_{W_{i,j}^{(l)}} z_j^{(l+1)} \cdot \partial_{z_j^{(l+1)}} Q_\theta(x) = \left(\partial_{W_{i,j}^{(l)}} z_j^{(l+1)}\right) \delta_j^{(l+1)}. \\
\partial_{b_i^{(l)}} Q_\theta(x) &= \partial_{b_i^{(l)}} z_j^{(l+1)} \cdot \partial_{z_j^{(l+1)}} Q_\theta(x) = \left(\partial_{b_i^{(l)}} z_j^{(l+1)}\right) \delta_j^{(l+1)}.
\end{aligned}$$

The problem is reduced to compute $\partial_{W_{i,j}^{(l)}} z_j^{(l+1)}$, $\partial_{b_i^{(l)}} z_j^{(l+1)}$ and $\delta_i^{(l)}$.

$$\delta_i^{(l)} = \sum_{j=1}^{n_{l+1}} \delta_j^{(l+1)} \partial_{z_i^{(l)}} z_j^{(l+1)}.$$



Layer 0
Layer l
Layer $l+1$
Layer L
$\delta_j^{(l+1)} = \frac{\partial Q_\theta(x,y)}{\partial z_j^{(l+1)}}$
$x^{(1)}$ $h_1^{(0)} = x^{(1)}$
$x^{(2)}$ $h_2^{(0)} = x^{(2)}$
$x^{(d)}$ $h_d^{(0)} = x^{(d)}$
$z_1^{(l)}$ $h_1^{(l)}$
$z_i^{(l)}$ $h_i^{(l)}$
$z_{n_l}^{(l)}$ $h_{n_l}^{(l)}$
$\frac{\partial z_j^{(l+1)}}{\partial W_{i,j}^{(l)}}$
$W_{i,j}^{(l)}$
$z_1^{(l+1)}$ $h_1^{(l+1)}$
$z_j^{(l+1)}$ $h_j^{(l+1)}$
$z_{n_{l+1}}^{(l+1)}$ $h_{n_{l+1}}^{(l+1)}$
$z_1^{(L)}$ $h_1^{(L)}$
$z_{n_L}^{(L)}$ $h_{n_L}^{(L)}$
$Q_\theta(x,y) = Q(h^{(L)}(x) - y)$
$\frac{\partial Q_\theta(x,y)}{\partial W_{i,j}^{(L)}}$

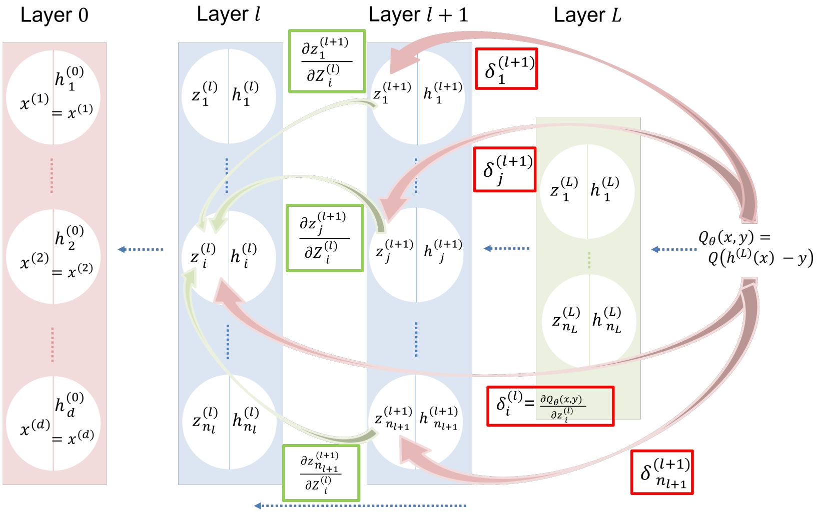


Figure: Illustration of recursive computation for $\delta_i^{(l)} = \partial_{z_i^{(l)}} Q_\theta(x, y)$

Derivatives between Layers

$$\partial_{W_{i,j}^{(l)}} z_j^{(l+1)} = \partial_{W_{ij}^{(l)}} (\sum_k W_{k,j}^{(l)} h_k^{(l)} + b_j^{(l)}) = h_i^{(l)}$$

$$\partial_{b_i^{(l)}} z_j^{(l+1)} = \partial_{b_i^{(l)}} (\sum_k W_{k,j}^{(l)} h_k^{(l)} + b_j^{(l)}) = 1$$

$$\partial_{z_i^{(l)}} z_j^{(l+1)} = \partial_{z_i^{(l)}} (\sum_k W_{k,j}^{(l)} h_k^{(l)} + b_j^{(l)}) = \partial_{z_i^{(l)}} (W_{i,j}^{(l)} h_i^{(l)}) = W_{i,j}^{(l)} \partial_{z_i^{(l)}} (\sigma_l(z_i^{(l)})) = W_{i,j}^{(l)} \sigma_l'(z_i^{(l)}).$$

Recursively Compute $\delta_i^{(l)} = \partial_{z_i^{(l)}} Q_\theta(x, y)$.

$$\delta_i^{(l)} = \sum_j \partial_{z_i^{(l)}} z_j^{(l+1)} \delta_j^{(l+1)} = \sum_j \sigma_l'(z_i^{(l)}) W_{i,j}^{(l)} \delta_j^{(l+1)} = \sigma_l'(z_i^{(l)}) (W^{(l)} \delta^{(l+1)})_i.$$

Backward Propagation Algorithm

Input: θ and (x, y); *Output*: $\nabla_\theta Q_\theta(x, y)$.

1. **Forward Phase**: for each $l \in \{1, \cdots, L-1\}$, compute

$$z^{(l+1)}(x) = h^{(l)}(x)W^{(l)} + b^{(l)},$$
$$h^{(l+1)}(x) = \sigma_{l+1}(z^{(l+1)}(x)).$$

2. **Backward Phase**:
For $l = L$, $\delta^{(L)} = \partial_{z^{(L)}} Q_\theta(x, y) = \partial_{z^{(L)}} Q(h^{(L)}(x) - y) = Q'(h^{(L)}(x) - y)\sigma'_L(z^{(L)}(x))$;
For $l = L - 1 : -1 : 1$,

$$\delta^{(l)} = \sigma'_l(z^{(l)}(x)) \odot (W^{(l)}\delta^{(l+1)});$$
$$\partial_{W^{(l)}_{i,j}} Q_\theta(x, y) = h^{(l)}_i \delta^{(l+1)}_j; \qquad \partial_{b^{(l)}_j} Q_\theta(x, y) = \delta^{(l+1)}_j.$$

Thanks for your attention!

References I

- Hao Ni, Xin Dong, Jinsong Zheng, and Guangxi Yu.
  An Introduction to Machine Learning in Quantitative Finance (Chinese version).
  Tsinghua University Press, 2021.
- Hao Ni, Xin Dong, Jinsong Zheng, and Guangxi Yu.
  An Introduction to Machine Learning in Quantitative Finance (English version).
  World Scientific, 2021.