15/9/23

Deep Learning

- Recap: Back prop ✓
- Optimization for DL model training (Chapter 8 of DL book by Goodfellow et. al.)
  - Gradient descent ✓
  - Stochastic gradient descent ✓
  - Momentum ✓
  - Nesterov momentum ✓

- Recall: $\theta^{(r)} = \theta^{(r-1)} - \eta^{(r)} \cdot \nabla_{\theta^{(r-1)}} R(\theta)$ — ①

$$\Rightarrow \alpha_{mp}^{(r)} = \alpha_{mp}^{(r-1)} - \eta^{(r)} \cdot \frac{\partial R(\theta)}{\partial \alpha_{mp}}$$

$$= \alpha_{mp}^{(r-1)} - \eta^{(r)} \sum_{i=1}^{n} \frac{\partial R^i(\theta)}{\partial \alpha_{mp}}$$

$$= \alpha_{mp}^{(r-1)} - \eta^{(r)} \sum_{i=1}^{n} s_m^i \cdot x_p^i \quad \text{— ②}$$

$|||^{ly}$ $$\beta_{km}^{(r)} = \beta_{km}^{(r-1)} - \eta^{(r)} \sum_{i=1}^{n} \delta_k^i Z_m^i \quad \text{— ③}$$

- Eqn ① is also called deterministic gradient descent since all training samples are used to estimate the gradient
- A drawback however is that it is slow.
- In mini batch or stochastic gradient descent (SGD), a subset of the training samples are used to find the $R(\theta)$ and in turn the gradient

$$R_{SGD}(\theta) = \sum_{i=1}^{m} R^i(\theta) \quad \text{where } m < n$$

- Note: This approach leads to noisy estimates of the gradient

The biggest advantage of SGD is that the complexity does not depend on n.

- SGD Algorithm: • Input: Dataset, m, $\eta^{(1)}, \eta^{(2)}, \dots, \eta^{(k)}$
  - While our stopping condition is not met do
    - Randomly sample 'm' data points
    - Find $R_{SGD}(\theta)$ and the corresponding gradients

- Update model parameters using the gradients

- Iteration: Model sees a set of minibatch samples
- Epoch: Model sees the entire training set.

- Momentum: Idea is to use past gradients in the parameter update rule:

$$v^{(r)} = \alpha v^{(r-1)} - \eta^{(r)} \cdot \nabla_{\theta^{(r-1)}} \cdot R(\theta) \quad - (4)$$

$$\theta^{(r)} = \theta^{(r-1)} + v^{(r)} \quad - (5)$$

- Nesterov Momentum: $v^{(r)} = \alpha v^{(r-1)} - \eta^{(r)} \nabla_{\theta^{(r-1)}} \cdot R(\theta^{(r-1)} + \alpha v^{(r-1)})$