- We will check techniques to address the difficulty of storing or inverting the Hessian
- But before that let's derive the mathematical form

Hessian Matrix I

- For CNN, the gradient of $f(\boldsymbol{\theta})$ is

$$\nabla f(\boldsymbol{\theta}) = \frac{1}{C}\boldsymbol{\theta} + \frac{1}{l}\sum_{i=1}^{l}(J^i)^T \nabla_{\boldsymbol{z}^{L+1,i}} \xi(\boldsymbol{z}^{L+1,i}; \boldsymbol{y}^i, Z^{1,i}), \tag{1}$$

where

$$J^i = \begin{bmatrix} \frac{\partial z_1^{L+1,i}}{\partial \theta_1} & \dots & \frac{\partial z_1^{L+1,i}}{\partial \theta_n} \\ \vdots & \vdots & \vdots \\ \frac{\partial z_{n_{L+1}}^{L+1,i}}{\partial \theta_1} & \dots & \frac{\partial z_{n_{L+1}}^{L+1,i}}{\partial \theta_n} \end{bmatrix}_{n_{L+1} \times n}, \quad i = 1, \dots, l, \tag{2}$$

Hessian Matrix II

is the Jacobian of $\boldsymbol{z}^{L+1,i}(\boldsymbol{\theta})$.

- The Hessian matrix of $f(\boldsymbol{\theta})$ is

$$
\begin{aligned}
\nabla^2 f(\boldsymbol{\theta}) = {} & \frac{1}{C}\mathcal{I} + \frac{1}{l}\sum_{i=1}^{l}(J^i)^T B^i J^i \\
& + \frac{1}{l}\sum_{i=1}^{l}\sum_{j=1}^{n_L}\frac{\partial \xi(\boldsymbol{z}^{L+1,i};\boldsymbol{y}^i,Z^{1,i})}{\partial z_j^{L+1,i}}
\begin{bmatrix}
\frac{\partial^2 z_j^{L+1,i}}{\partial\theta_1\partial\theta_1} & \cdots & \frac{\partial^2 z_j^{L+1,i}}{\partial\theta_1\partial\theta_n} \\
\vdots & \ddots & \vdots \\
\frac{\partial^2 z_j^{L+1,i}}{\partial\theta_n\partial\theta_1} & \cdots & \frac{\partial^2 z_j^{L+1,i}}{\partial\theta_n\partial\theta_n}
\end{bmatrix},
\end{aligned}
$$

Hessian Matrix III

where $\mathcal{I}$ is the identity matrix and B^i is the Hessian of $\xi(\cdot)$ with respect to $\boldsymbol{z}^{L+1,i}$:

$$B^i = \nabla^2_{\boldsymbol{z}^{L+1,i},\boldsymbol{z}^{L+1,i}} \xi(\boldsymbol{z}^{L+1,i}; \boldsymbol{y}^i, Z^{1,i})$$

- More precisely,

$$B^i_{ts} = \frac{\partial^2 \xi(\boldsymbol{z}^{L+1,i}; \boldsymbol{y}^i, Z^{1,i})}{\partial z_t^{L+1,i} \partial z_s^{L+1,i}}, \forall t, s = 1, \ldots, n_{L+1}. \quad (3)$$

- Usually B^i is very simple.

Hessian Matrix IV

- For example, if the squared loss is used,

$$\xi(\boldsymbol{z}^{L+1,i};\boldsymbol{y}^i) = ||\boldsymbol{z}^{L+1,i} - \boldsymbol{y}^i||^2.$$

  then

$$B^i = \begin{bmatrix} 2 & & \\ & \ddots & \\ & & 2 \end{bmatrix}$$

- Usually we consider a convex loss function

$$\xi(\boldsymbol{z}^{L+1,i};\boldsymbol{y}^i)$$

  with respect to $\boldsymbol{z}^{L+1,i}$

Hessian Matrix V

- Thus B^i is positive semi-definite
- The last term of $\nabla^2 f(\boldsymbol{\theta})$ may not be positive semi-definite
- Note that for a twice differentiable function $f(\boldsymbol{\theta})$

$$f(\boldsymbol{\theta}) \text{ is convex}$$

if and only if

$$\nabla^2 f(\boldsymbol{\theta}) \text{ is positive semi-definite}$$

Jacobian Matrix

- The Jacobian matrix of $\boldsymbol{z}^{L+1,i}(\boldsymbol{\theta}) \in R^{n_{L+1}}$ is

$$J^i = \begin{bmatrix} \frac{\partial z_1^{L+1,i}}{\partial \theta_1} & \cdots & \frac{\partial z_1^{L+1,i}}{\partial \theta_n} \\ \vdots & \vdots & \vdots \\ \frac{\partial z_{n_L}^{L+1,i}}{\partial \theta_1} & \cdots & \frac{\partial z_{n_L}^{L+1,i}}{\partial \theta_n} \end{bmatrix} \in R^{n_{L+1} \times n}, \quad i = 1, \ldots l.$$

- n_{L+1}: # of neurons in the output layer
- n: number of total variables
- $n_{L+1} \times n$ can be large

Gauss-Newton Matrix I

- The Hessian matrix $\nabla^2 f(\boldsymbol{\theta})$ is now not positive definite.
- We may need a positive definite approximation
- Many existing Newton methods for NN has considered the Gauss-Newton matrix (Schraudolph, 2002)

$$G = \frac{1}{C}\mathcal{I} + \frac{1}{l}\sum_{i=1}^{l}(J^i)^T B^i J^i$$

by removing the last term in $\nabla^2 f(\boldsymbol{\theta})$

Gauss-Newton Matrix II

- The Gauss-Newton matrix is positive definite if B^i is positive semi-definite
- This can be achieved if we use a convex loss function in terms of $\boldsymbol{z}^{L+1,i}(\boldsymbol{\theta})$
- We then solve

$$G\boldsymbol{d} = -\nabla f(\boldsymbol{\theta})$$

References I

N. N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002.