Memory Cost of Storing J^i I

- The Gauss-Newton matrix is

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

- Its size is

$$n \times n,$$

where n is the total number of variables

Memory Cost of Storing J^i II

- But storing J^i needs
$$n_{L+1} \times n \times l,$$
where
$$n_{L+1} : \text{\# nodes in the output layer (\# classes)}$$
$$l : \text{number of data}$$
- If
$$n < n_{L+1} \times l,$$
then storing $J^i, \forall i$ needs more spaces than G

Memory Cost of Storing J^i III

- Then the Hessian-free method cannot work
- A related question is why earlier in calculating the gradient we did not get J^i, store it, and then calculate

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

- Instead we use backpropagation without explicitly storing $J^i, \forall i$

Memory Cost of Storing J^i IV

- For gradient, J^i is used only once
- However, in each Newton iteration we need J^i several times
- J^i is used in every matrix-vector product, so maybe there is a need to store it (or store some information about it)
- Some techniques can be used to alleviate the memory problem of storing $J^i, \forall i$
  - Subsampled Hessian Newton method. This technique reduces the memory consumption of storing J^i

Memory Cost of Storing J^i V

- Forward and reverse modes of automatic differentiation. This technique leads to the calculation of matrix-vector products without storing J^i

Subsampled Hessian Newton Method I

- We know the gradient needs a sum over all data

$$\nabla f(\boldsymbol{\theta}) = \frac{1}{C}\boldsymbol{\theta} + \frac{1}{l}\sum_{i=1}^{l} \nabla_{\boldsymbol{\theta}} \xi_i$$

- In stochastic gradient, we do mini-batch
- Like mini-batch, in Newton we can use a subset of data for

matrix-vector products

and

function/gradient evaluation

Subsampled Hessian Newton Method II

- This is possible: subsampled Newton method (Byrd et al., 2011; Martens, 2010; Wang et al., 2015)
- Assume the large number of data points are from the same distribution
- We can select a subset $S \subset \{1, \ldots, l\}$ and have

$$G^S = \frac{1}{C}\mathcal{I} + \frac{1}{|S|}\sum_{i \in S} (J^i)^T B^i J^i \approx G.$$

Subsampled Hessian Newton Method III

- Then the matrix-vector product becomes

$$G^S \mathbf{v} = \frac{1}{C}\mathbf{v} + \frac{1}{|S|}\sum_{i \in S}\left((J^i)^T\left(B^i(J^i\mathbf{v})\right)\right) \tag{1}$$

- The cost of storing J^i is reduced from

$$n_{L+1} \times n \times l$$

to

$$n_{L+1} \times n \times |S|$$

Subsampled Hessian Newton Method IV

- Typically a choice may be

$$|S| = (0.05 \text{ or } 0.01) \times l$$

- The selection of the size $|S|$ is still an issue worth investigation
- At this moment we consider

  subset for matrix-vector products

  and

  full set for function/gradient evaluation
- Reason: no matter what a subset S is chosen,

Subsampled Hessian Newton Method V

$$G^S \text{ is still positive definite}$$

- Then

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

  leads to

$$\nabla f(\boldsymbol{\theta})^T \boldsymbol{d} = -\nabla f(\boldsymbol{\theta})^T (G^S)^{-1} \nabla f(\boldsymbol{\theta}) < 0$$

- If we use a subset for the gradient, then the above inequality may not hold
- Then the situation becomes more complicated

Subsampled Hessian Newton Method VI

- Note that if using the full set for function/gradient evaluations, we have theoretical asymptotic convergence to a stationary point (Byrd et al., 2011)