Hessian-free Newton Method I

- Recall that at each Newton iteration we must solve a linear system

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

  and G is huge
- G's size is

$$n \times n,$$

  where n is the total number of variables
- It is not possible to store G

Hessian-free Newton Method II

- Thus methods such as Gaussian elimination are not possible
- If G has certain structures, it's possible to use iterative methods to solve the linear system by a sequence of matrix-vector products

$$G\boldsymbol{v}^1, G\boldsymbol{v}^2, \ldots$$

without storing G

- This is called Hessian-free in optimization

Hessian-free Newton Method III

- For example, conjugate gradient (CG) method can be used to solve

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

  by a sequence of matrix-vector products (Hestenes and Stiefel, 1952)
- We don't discuss details of CG here though the procedure will be shown in a later slide
- You can check Golub and Van Loan (2012) for a good introduction

Hessian-free Newton Method IV

- Each CG step involves one matrix-vector product
- For many machine learning methods, G has certain structures so that calculating

$$G\boldsymbol{d}$$

  is practically feasible
- The cost of Hessian-free Newton is

$$\begin{aligned}&(\#\text{matrix-vector products} + \\ &\ \text{function/gradient evaluation}) \times \#\text{iterations}\end{aligned}$$

Hessian-free Newton Method V

- Usually the number of iterations is small
- In theory, the number of CG steps (matrix-vector products) is $\leq$ the number of variables
- For our problem we will see that each matrix-vector product can be as expensive as one function/gradient evaluation
- Thus, matrix-vector products can be the bottleneck

Conjugate Gradient Method I

- We would like to solve

$$Ax = b,$$

  where A is symmetric positive definite
- The procedure

$$
\begin{aligned}
& k = 0;\ x = 0;\ r = b;\ \rho_0 = \|r\|_2^2 \\
& \text{while } \sqrt{\rho_k} > \epsilon \|b\|_2 \text{ and } k < k_{\max} \\
& \qquad k = k + 1 \\
& \qquad \text{if } k = 1 \\
& \qquad\qquad p = r
\end{aligned}
$$

Conjugate Gradient Method II

else

$$\beta = \rho_{k-1}/\rho_{k-2}$$

$$p = r + \beta p$$

end

$$w = Ap$$

$$\alpha = \rho_{k-1}/p^T w$$

$$x = x + \alpha p$$

$$r = r - \alpha w$$

$$\rho_k = \|r\|_2^2$$

end

Conjugate Gradient Method III

- Note that

$$r = b - Ax$$

  indicates the error
- We can see that Ap is the only matrix-vector product at each step
- Others are vector operations

Matrix-vector Products I

- Earlier we have shown that the Gauss-Newton matrix is

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

- We have

$$G\boldsymbol{v} = \frac{1}{C}\boldsymbol{v} + \frac{1}{l}\sum_{i=1}^{l}\left((J^i)^T\left(B^i(J^i\boldsymbol{v})\right)\right). \tag{1}$$

Matrix-vector Products II

- If we can calculate

$$J^i \boldsymbol{v} \text{ and } (J^i)^T(\cdot)$$

  then G is never explicitly stored
- Therefore, we can apply the conjugate gradient (CG) method by a sequence of matrix-vector products.
- But is this approach really feasible?
- We show that memory can be an issue