The Hessian Matrix

Sargur Srihari

Definitions of Gradient and Hessian

• First derivative of a scalar function E(w) with respect to a vector $w = [w_1, w_2]^T$ is a vector called the *Gradient* of E(w)

$$\nabla E(\boldsymbol{w}) = \frac{d}{d\boldsymbol{w}} E(\boldsymbol{w}) = \begin{bmatrix} \frac{\partial E}{\partial w_1} \\ \frac{\partial E}{\partial w_2} \end{bmatrix}$$
 If there are M elements in the vector then Gradient is a M x 1 vector

• Second derivative of E(w) is a matrix called the *Hessian* of E(w)

$$H = \nabla \nabla E(\boldsymbol{w}) = \frac{d^2}{d\boldsymbol{w}^2} E(\boldsymbol{w}) = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} \\ \frac{\partial^2 E}{\partial w_2 \partial w_1} & \frac{\partial^2 E}{\partial w_2^2} \end{bmatrix}$$
 Hessian is a matrix with M^2 elements

Jacobian is a matrix consisting of first derivatives wrt a vector

Computing the Hessian using Backpropagation

- We have shown how backpropagation can be used to obtain first derivatives of error function wrt weights in network
- Backpropagation can also be used to derive second derivatives

$$rac{\partial^2 E}{\partial w_{_{ii}}w_{_{lk}}}$$

• If all weights and bias parameters are elements w_i of single vector \boldsymbol{w} then the second derivatives form the elements H_{ij} of Hessian matrix H where i,j ϵ $\{1,...W\}$ and W is the total no of weights and biases

Role of Hessian in Neural Computing

- Several nonlinear optimization algorithms for neural networks
 - are based on second order properties of error surface
- 2. Basis for fast procedure for retraining with small change of training data
- 3. Identifying least significant weights
 - For network pruning requires inverse of Hessian
- 4. Bayesian neural network
- Central role in Laplace approximation
 - Hessian inverse is used to determine the predictive distribution for a trained network
 - Hessian eigenvalues determine the values of hyperparameters
 - Hessian determinant is used to evaluate the model evidence

Evaluating the Hessian Matrix

- Full Hessian matrix can be difficult to compute in practice
 - quasi-Newton algorithms have been developed that use approximations to the Hessian
- Various approximation techniques have been used to evaluate the Hessian for a neural network
 - calculated exactly using an extension of backpropagation
- Important consideration is efficiency
 - With W parameters (weights and biases) matrix has dimension W x W
 - Efficient methods have $O(W^2)$

Methods for evaluating the Hessian Matrix

- Diagonal Approximation
- Outer Product Approximation
- Inverse Hessian
- Finite Differences
- Exact Evaluation using Backpropagation
- Fast multiplication by the Hessian

Diagonal Approximation

- In many case inverse of Hessian is needed
- If Hessian is approximated by a diagonal matrix (i.e., offdiagonal elements are zero), its inverse is trivially computed
- Complexity is O(W) rather than $O(W^2)$ for full Hessian

Outer product approximation

Neural networks commonly use sum-of-squared errors function

$$E = \frac{1}{2} \sum_{n=1}^{N} (y_n - t_n)^2$$

Can write Hessian matrix in the form

$$H \approx \sum_{n=1}^{n} b_n b_n^T$$

- Where $b_n = \nabla y_n = \nabla a_n$
- Elements can be found in $O(W^2)$ steps

Inverse Hessian

 Use outer product approximation to obtain computationally efficient procedure for approximating inverse of Hessian

Finite Differences

• Using backprop, complexity is reduced from $O(W^3)$ to $O(W^2)$

Exact Evaluation of the Hessian

- Using an extension of backprop
- Complexity is $O(W^2)$

Fast Multiplication by the Hessian

- Application of the Hessian involve multiplication by the Hessian
- The vector v^TH has only W elements
- Instead of computing H as an intermediate step, find efficient method to compute product