Natural Language Processing: Assignment 1

Isaac Jefferson Lee

Email: isalee@student.ethz.ch

Question I

Question I, Part a)

So we want to show that:

$$\nabla^2 f(\boldsymbol{x}) = \begin{bmatrix} \nabla(\boldsymbol{e_1}\nabla f(\boldsymbol{x})^T) \\ \dots \\ \nabla(\boldsymbol{e_n}\nabla f(\boldsymbol{x})^T \end{bmatrix}.$$

The LHS (Left Hand Side) can be written as:

$$\text{LHS} = \nabla^2 f(\boldsymbol{x}) = \begin{bmatrix} f_{x_1x_1} & f_{x_1x_2} & \dots & f_{x_1x_n} \\ f_{x_2}f_{x_1} & f_{x_2x_2} & \dots & f_{x_2x_n} \\ \dots & \dots & \dots & \dots \\ f_{x_nx_1} & f_{x_nx_2} & \dots & f_{x_nx_n} \end{bmatrix}.$$

I.e $[\nabla^2 f(\boldsymbol{x})]_{ij} = f_{x_ix_j}$ Now we consider the i^{th} row of the RHS (Right Hand Side):

$$\nabla(\boldsymbol{e}_i\nabla f(\boldsymbol{x})^T).$$

Recall that $\nabla f(\boldsymbol{x}) = [f_{x_1}, f_{x_2}, ..., f_{x_n}] \implies \boldsymbol{e}_i\nabla f(\boldsymbol{x})^T = f_{x_i}$

$$\implies i^{\text{th}}\text{row of RHS} = \nabla f_{x_i}.$$

So we can define $g(\boldsymbol{x}) := f_{x_i}$ and it follows that:

$$\text{RHS} = \nabla g(\boldsymbol{x}) = \begin{bmatrix} g_{x_1}, & g_{x_2}, & \dots, & g_{x_n} \end{bmatrix}.$$

$$\implies j^{\text{th}}\text{column of RHS} = g_{x_j} = \frac{\partial}{\partial x_j}(f_{x_i}) = f_{x_jx_i}.$$

So then if we assume continuity of the second order derivatives, we can apply *Schwarzes Theorem*, which gives symmetry of second order partial derivatives to see that

$$[\text{LHS}]_{ij} = [\text{LHS}]_{ji} = [\text{RHS}]_{ij}.$$

QED.

It is obvious by the above mentioned symmetry of second order partial derivatives that the Hessian matrix is symmetric, thus it is clear that:

$$i^{\text{th}}\text{column of Hessian} = (\nabla(\boldsymbol{e}_i\nabla f(\boldsymbol{x})^T))^T.$$

Question I, Part b)

From the identity above we can derive the following algorithm for computing the hessian:

1. **STEP I:** Use backpropagation on f to find it's **symbolic** first order derivatives.
2. **STEP 2:** For each f_{x_i} first order (symbolic) partial derivative, again apply the backpropagation algorithm (this time with forward propogation and numerical outputs) to get the i^{th} row of the hessian.

Since we have n first order partial derivatives, which we can evaluate in $O(m)$ time for which we need to draw the corresponding computational graph and re-apply backpropogation (also in $O(m)$ time), it follows that computing the Hessian with this method will have a time complexity of $O(n * m)$. This is clearly more efficient than calculating the Hessian the "naive way" i.e entry by entry, which would clearly have a runtime complexity of $O(n^2 * m)$.

Question I, Part c)

For the third order tensor, we can just apply the same idea, with

$$\nabla(\boldsymbol{e}_j \nabla(\boldsymbol{e}_i \nabla f(\boldsymbol{x})^T)^T).$$

If we visualize the 3rd order tensor as a 3D grid of values, then this will give us a vertical "column" of values, with

$$[\nabla(\boldsymbol{e}_j \nabla(\boldsymbol{e}_i \nabla f(\boldsymbol{x})^T)^T)]_k = f_{x_i x_j x_k}.$$

Therefore per "vertical column" we will have a runtime of $O(m)$ and there are n^2 vertical columns so therefore we will have a runtime complexity of $O(n^2 * m)$.

In general it is clear that if we apply this method to get the k^{th} order tensor then we will have a runtime of $O(n^{k-1} * m)$.

Question 2

https://colab.research.google.com/drive/10OqHG-bIPm1uSX3v47s3x1rCIj46vc40?usp=sharing