Math 5467 – Homework 4

Instructions:

- Complete the problems below, and submit them though the Google form for Homework 4.
- If you use LaTeX to write up your solutions, upload them as a pdf file. Students who use LaTeX to write up their solutions will receive bonus points on the homework assignment.
- If you choose to write your solutions and scan them, please either use a real scanner, or use a smartphone app that allows scanning with you smartphone camera. It is not acceptable to submit images of your solutions, as these can be hard to read.

Problems:

1. (ResNet) The traditional neural network architecture

$$f_k = \sigma_k(W_k f_{k-1} + b_k), \quad k = 1, \ldots, L,$$

often yields worse performance for deeper networks with more layers compared to shallower networks. The main issue it that training is difficult, due to vanishing gradients or gradient blowup (where the gradients either become very small and training does not progress, or become very large and training is unstable). This is not so surprising; consider the case where $\sigma_k(t) = t$ is the identity and the biases $b_k = 0$ vanish. Then

$$f_L(x) = W_L W_{L-1} \cdots W_2 W_1 x.$$

The L-fold product is very sensitive to the spectral norms of the matrices; when the eigenvalues are larger than one in magnitude it blows up exponentially, while when they are less than one it decay to zero exponentially.

The *Residual Neural Network (ResNet)* architecture [1] is a recent development in deep learning that solves this problem by changing the architecture to

$$f_k = f_{k-1} + W_{k,1}\sigma_k(W_{k,2} f_{k-1} + b_k), \quad k = 1, \ldots, L. \tag{1}$$

The idea is to have each layer learn the *residual* $f_k - f_{k-1}$, which allows the network to easily skip layers, by setting $f_k = f_{k-1}$. Thus, a deeper network with ResNet architecture should not perform worse than a shallower network. The ResNet architecture should remind you of the discretization of an ordinary differential equation (ODE), and many recent works have exploited this to explain the stability of ResNet.

Each ResNet layer has two weight matrices $W_{k,1}$ and $W_{k,2}$ and a bias b_k. Derive the back propagation equations to compute $\frac{\partial \mathcal{L}}{\partial W_{k,1}}$, $\frac{\partial \mathcal{L}}{\partial W_{k,2}}$, and $\frac{\partial \mathcal{L}}{\partial b_k}$ for ResNet. You should closely follow Theorem 8.4.1 from the class notes, with appropriate changes for ResNet. You can assume that all layers have the same number of hidden units so that f_k and f_{k-1} have the same dimensions.

2. A graph convolutional neural network layer is given by

$$H_k = \sigma(AH_{k-1}W_k),$$

where A is the $n \times n$ adjacency matrix for the graph, n is the number of nodes in the graph, H_k is an $n \times N$ matrix where each row is a feature vector attached to a node in the graph, and W_k is an $N \times N$ tunable weight matrix. The matrices H_k are the outputs of each layer, and play the role of the vectors f_k in fully connected neural networks. You can assume the weight matrix A is symmetric, so $A = A^T$. Derive the back-propagation equation

$$\frac{\partial \mathcal{L}}{\partial H_{k-1}} = A\left(\sigma'(AH_{k-1}W_k) \odot \frac{\partial \mathcal{L}}{\partial H_k}\right) W_k^T.$$

The symbol $\odot$ is the coordinatewise product of two matrices, so

$$(A \odot B)(i,j) = A(i,j)B(i,j).$$

Hint: Start by writing

$$H_k(i,j) = \sigma\left(\sum_{\ell=1}^{N}\sum_{m=1}^{n} A(i,m)H_{k-1}(m,\ell)W_k(\ell,j)\right),$$

and use the chain rule

$$\frac{\partial \mathcal{L}}{\partial H_{k-1}(p,q)} = \sum_{i=1}^{n}\sum_{j=1}^{N} \frac{\partial L}{\partial H_k(i,j)} \frac{\partial H_k(i,j)}{\partial H_{k-1}(p,q)}.$$

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.