IE643: Deep Learning - Theory and Practice **July-Dec 2024**

Practice Mid-sem Exam: (18^{th} Sep 2024) Time: 90 minutes

1 Instructions (Please read carefully!)

You are free to refer to your hand-written (and/or paper-printed) class notes and scribes that you carry with you. However you are not allowed to access textbooks, e-books, tablets, laptops, calculators, electronic devices and internet. No discussion is allowed.

Answer all questions. Write your answers clearly; explain the assumptions you make and indicate the reasoning behind each claim you make. Use results discussed in class; if you use new results, you must justify them with proper proofs. You can score a maximum of **50** marks in this exam.

2 Questions

1. Assume a feed forward neural network (or MLP) such that at layer ℓ, there are N_ℓ neurons and these neurons are named $n_1^\ell, n_2^\ell, \ldots, n_{N_\ell}^\ell$. To improve regularization and to reduce overfitting when training the feed forward neural network, the following strategy is sometimes used. In a hidden layer ℓ, a probability vector $\boldsymbol{\theta}$ of size $N_\ell \times 1$ is constructed where every entry θ_i of $\boldsymbol{\theta}$ is sampled from uniform distribution over the interval $[0, 1]$. If $\theta_i < 0.5$, then it is assumed that the neuron n_i^ℓ will not take part in the forward and backward passes, otherwise it will. The resultant architecture can be visualized by removing incoming connections to n_i^ℓ from every neuron $n_j^{(\ell-1)}$ from the $(\ell-1)$-th layer, and by removing all outgoing connections from n_i^ℓ to every neuron $n_m^{(\ell+1)}$ in the $(\ell+1)$-th layer. The following questions are related to this aspect.

   (a) [**3 marks**] Illustrate the above situation using an appropriate figure and explain.

   (b) [**5 marks**] Though it might be easy to visualize using your illustration, in terms of implementation, it might not be feasible to consider arrays or matrices of dynamically changing sizes based on missing neurons. Hence we would wish to preserve the size of the weight, activation and dot product matrices. Towards this, suggest appropriate mathematical expressions which would help to compute the vector of activations $\mathbf{a}^\ell$ of size $N_\ell \times 1$, whose j-th entry a_j^ℓ denotes the activation at j-th neuron n_j^ℓ in ℓ-th layer, by appropriately modifying the expressions discussed in class. Assume that no neuron is removed in $(\ell-1)$-th layer. You can introduce additional vectors or matrices as necessary. If you introduce new notations not present in class lectures, explain them carefully. Justify why your expressions will help in computation of forward pass activations for the situation described in the question above.

   (c) [**5 marks**] Using your answer to part (b), suggest a general expression for forward pass where the neurons in all hidden layers can probabilistically take part in the forward pass. Explain your expressions.

   (d) [**7 marks**] Using your expression in part (c), can you come up with equations for backpropagation? Explain and give proper justifications why your expressions would help in effectively propagating the gradients without dynamically changing the sizes of the weight, activation and dot product matrices and vectors.

2. Recall that the directional derivative of a function $f : \mathbb{R}^d \to \mathbb{R}$ at a point $\mathbf{x} \in \mathbb{R}^d$ along the direction $\mathbf{q} \in \mathbb{R}^d$ (where $\mathbf{q}$ is not the zero vector) is defined as follows: $f'(\mathbf{x}; \mathbf{q}) = \lim_{\alpha \downarrow 0} \frac{f(\mathbf{x}+\alpha\mathbf{q})-f(\mathbf{x})}{\alpha}$. The following questions are related to this definition.

   (a) [**3 marks**] Explain what the notation $\alpha \downarrow 0$ means in the definition. Why is $\alpha \uparrow 0$ not considered explicitly in the definition?

   (b) [**2 marks**] Explain why the direction $\mathbf{q}$ is considered to be non-zero in the definition.

(c) **[4 marks]** Suppose if we wish to replace the limits $\lim_{\alpha\downarrow 0}$ into $\lim_{\alpha\uparrow 0}$ in the definition. Explain how your expression (within the limits) for directional derivative will then change to take care of the change in limits.

(d) **[6 marks]** Using your new expression in part (c), give a proof that if $\mathbf{q}$ is a descent direction (that is, if $f'(\mathbf{x};\mathbf{q}) < 0$), then there exists an appropriate $\epsilon > 0$ such that for every $\alpha \in [a, b)$ for suitable values of a, b, we have $f(\mathbf{x} + \alpha\mathbf{q}) < f(\mathbf{x})$.

3. Consider a multi-layer perceptron with an input layer with 5 neurons and an output layer with 5 neurons. Consider three hidden layers L_1, L_2, L_3 where L_1 and L_3 have 3 neurons and L_2 has 2 neurons. Let B_1 denote the matrix of weights connecting the input layer and layer L_1, and B_2 denote the matrix of weights connecting layers L_1 and L_2. Let A_2 denote the matrix of weights connecting the layers L_2 and L_3 and let A_1 denote the matrix of weights connecting the layer L_3 with the output layer. Assume that all hidden layers and output layer use linear activations. Answer the following:

   (a) **[2 marks]** Depict the neural network structure in a figure with appropriate notations.

   (b) **[3 marks]** Discuss the expression for forward pass and the expression for the objective error function E, assuming a squared error loss.

   (c) **[5 marks]** Using the discussions in class, illustrate if the error E is convex with respect of vec(A_1) for fixed A_2, B_1, B_2 matrices.

4. **[5 marks] (open-ended)** Consider a data set D of images and corresponding class labels. You have designed a MLP for D which gives decent performance on your laptop. Note that to train a MLP on D, we can use vectors for the images obtained by flattening the images. Now you want to train your MLP on your friend's computer on another data set D' similar to D. Unfortunately, the memory capacity is very limited in your friend's computer for storing the weights of your MLP . Recall that we can use an auto-encoder to extract low-dimensional representations of the flattened images. A natural way to perform the training on your friend's computer would be to first train an appropriate auto-encoder (which can fit into the memory) on the images of D' to obtain low dimensional representations of the flattened images and then use the low dimensional representations for training another MLP for classification, which might require very few weights than your MLP. However this two step process is time-consuming and you want to integrate this process. Suggest a MLP architecture and a training process (with details of loss function and training algorithm) which can integrate this two step process into a single step process. You can assume that there is sufficient memory on your friend's computer for your modified design of MLP.

END OF QUESTIONS