Date: Thursday, March 18, 2021. 10:30 am - 12:00 noon

There are 5 questions. You need to attempt all questions. Each Question carries 5 points. Max Points: 25

Answer to each question must start on a new page. You need to justify all your answers. Answers without justification may not get full points.

1. **Image Translation:** Consider the task of image translation: given an image with text written in language A, create another image which is a copy of the original image but with the text translated to a language B. Note that original image may have additional background information which should be replicated as is in the transformed image. Consider performing this task in a supervised setting.

   (a) Describe this task in terms of the input and output pairs: what will your input be and what will your output be? You should be as detailed as possible. What concerns will determine your modeling choices, i.e., whether an CNN based model is more suited for this task, a recurrent network, or a combination. Justify you answer.

   (b) Describe the broad outline of your model (you do not have to go in specific details of the architecture), how you would process the input to get the desired output. Your answer should be described in terms of a pipeline of relevant modules.

   (c) Now suppose you have additional supervised data for going from sentences in language A to sentences in language B. How would you use this additional data to learn a better model for the above problem?

2. **Back-propagation**: Consider a fully connected feed forward network with T number of hidden layers with T being even. Assume that the number of units in each odd numbered hidden layer is the same, and is given by n_1. Similarly, let the number of units in each even numbered hidden layer is the same, and is given by n_2. Let the output layer be given as r-sized softmax applied over a linear transformation of the output of the last hidden layer, i.e., softmax$(W_o h_o + b_o)$, where symbols are as defined in the class. Let the parameters in the layer t and t+2, i.e., $W^{(t)}$, $W^{(t+2)}$ be tied to each other, for all values of t, ~~$1 \leq t \leq T-2$~~ $2 \leq t \leq T-2$. In other words, $W^{(3)} = \cdots = W^{(T-1)}$ and similarly, $W^{(2)} = W^{(4)} = \cdots = W^{(T)}$. Assume that each hidden layer learns a bias of its own. Note that the weights $W^{(1)}$ in the first layer are not tied to any other weights. ~~Let the weight parameters corresponding to the input (layer) be given by W_I.~~ For simplicity, assume that we are learning over a single training example (x,y), where $x \in \mathcal{R}^n$ and $y \in \{1, \cdots, r\}$.

   (a) Assuming a negative log-likelihood based loss, write the expression for the loss in the above model.

(b) Derive the expression for gradient of the loss with respect to the parameters $W^{(1)}$, $W^{(2)}$,$W^{(3)}$, the bias terms $b^{(t)}$ in each layer, and hidden representations $h^{(t)}$ for all values of t. ~~,and weights W_i corresponding to the input.~~

(c) Fully describe the back-propagation algorithm for the above network.

3. **Cross-Entropy based Loss:** Given two distributions P and Q over a K valued discrete random variable u, cross entropy between P and Q is defined as: $-\sum_{k=1}^{K} Q(u_l) log P(u_l)$. Consider learning a model for the problem of object classification, i.e., given an image, we would like to identify objects present in the image. Let the number of possible labels be given by r. Assume learning with a batch of examples given as $\{x^{(i)}, y^{(i)}\}_{i=1}^{b}$.

   (a) Let each image contain exactly one among the given set of objects. Define the appropriate loss function for this setting in terms of a softmax layer. Express your loss in terms of cross entropy between two suitably defined distributions. You should clearly describe the architecture of your output layer, and the parameters thereof for this setting - you need to justify your construction.

   (b) We will generalize the above setting to when one or more objects can be present in any given image (Each $y^{(i)}$ now represents the set of object labels in the image corresponding to $x^{(i)}$). Describe the changes to your architecture in the output layer to handle this setting. Define the loss function. Express your loss in terms of cross entropy over suitably defined distributions.

4. **L1 and L2 regularization:** Let the (un-regularized) cost function for a learning problem be denoted by $J(\theta)$. Consider learning with a combination of L1 and L2 regularization with the weights of the corresponding regularization terms given by α_1 and α_2, respectively. We will use the quadratic approximation to the cost function at a given value of parameters θ for the following derivation. Mathematically, describe the impact of using the above regularization on the learned parameters, vis-a-vis the parameters which would have been learned when training an un-regularized model. You should describe you answer in terms of the eigenvalues of the Hessian matrix H of the original loss function $J(\theta)$, as well as, α_1 and α_2. Does this model induce sparsity, push the model parameters towards zero, or both. Justify your answer in terms of the expression that you drive. You can assume that H is diagonal.

5. **Gradient Clipping:** Mathematically, show that the problem of exploding gradients can be particularly nasty when dealing with recurrent networks. You can make the assumption of a network with linear activation units of the form $g(z) = \alpha z$ ($\alpha > 0$). One of the ways to ameliorate this problem is to use clipped gradients. Describe precisely your implementation of the clipping algorithm during back-propagation. Describe the gradient clipping in terms of purely using a smaller learning rate for specific learning iterations. You should mathematically justify your answer.

   Gradient clipping can also be very useful for avoiding jumping off the cliffs. Describe, with the help of a picture, the issue with jumping off the cliffs, and how gradient clipping might help. Do you see any potential pitfalls for using gradient clipping as opposed to plain gradient descent?