National University of Singapore
School of Computing
CS3244: Machine Learning
Solution to Tutorial 09

RNNs and Explainable AI(XAI)

1. **RNN and BPTT** Here, we'll be computing gradients via *Backpropagation Through Time* (BPTT). The Forward Pass of a RNN can be characterised as follows(Here σ denotes *softmax* function):

$$\mathbf{h}_t = g^{[h]}((\mathbf{W}^{[xh]})^\top \mathbf{x}_t + (\mathbf{W}^{[hh]})^\top \mathbf{h}_{t-1}) \tag{1}$$
$$\hat{\mathbf{y}}_t = g^{[y]}((\mathbf{W}^{[hy]})^\top \mathbf{h}_t) \tag{2}$$
$$\hat{\mathbf{o}}_t = \sigma(\hat{\mathbf{y}}_t) \tag{3}$$

The loss L is the Cross Entropy Loss:

$$L = -\sum_t^T \mathbf{y}_t \cdot \log(\hat{\mathbf{o}}_t) \tag{4}$$

For simplicity, let's call the final time step loss, $E_T = -\mathbf{y}_T \log(\hat{\mathbf{o}}_T)$. The objective of BPTT is to update the parameters $\mathbf{W}^{[xh]}$, $\mathbf{W}^{[hh]}$, and $\mathbf{W}^{[hy]}$.

(a) Use Chain Rule to find an expression for $\frac{\partial E_T}{\partial \mathbf{W}^{[hh]}}$. (Note, there is no need to expand the term $\frac{\partial \mathbf{h}_{T-1}}{\partial \mathbf{W}^{[hh]}}$ further)
The answer is a multiplication of the terms via Chain Rule:

$$\frac{\partial E_T}{\partial \mathbf{W}^{[hh]}} = \frac{\partial E_T}{\partial \hat{\mathbf{o}}_T} \times \frac{\partial \hat{\mathbf{o}}_T}{\partial \hat{\mathbf{y}}_T} \times \frac{\partial \hat{\mathbf{y}}_T}{\partial \mathbf{h}_T} \times \frac{\partial \mathbf{h}_T}{\partial \mathbf{W}^{[hh]}} \tag{5}$$
$$\frac{\partial E_T}{\partial \mathbf{W}^{[hh]}} = \frac{\partial E_T}{\partial \hat{\mathbf{o}}_T} \times \frac{\partial \hat{\mathbf{o}}_T}{\partial \hat{\mathbf{y}}_T} \times \frac{\partial \hat{\mathbf{y}}_T}{\partial \mathbf{h}_T} \times \left(\frac{\partial g^{[h]}(\mathbf{x}_T, \mathbf{h}_{T-1})}{\partial \mathbf{W}^{[hh]}} + \frac{\partial g^{[h]}(\mathbf{x}_T, \mathbf{h}_{T-1})}{\partial \mathbf{h}_{T-1}} \times \frac{\partial \mathbf{h}_{T-1}}{\partial \mathbf{W}^{[hh]}} \right) \tag{6}$$

Figure 1 shows the gradient flow.

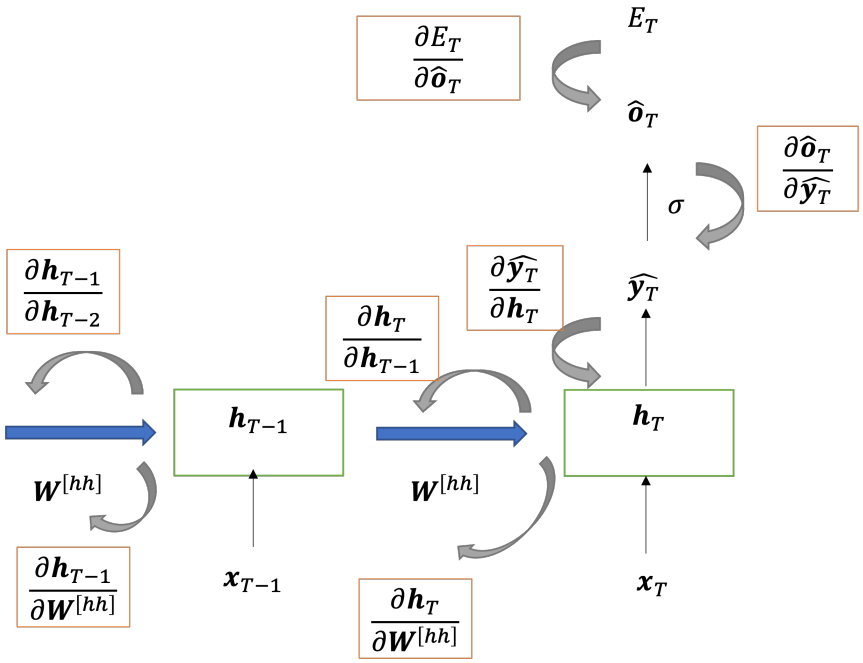


Figure 1: Gradient Flow

(b) Out of the various terms above, which one do you think is responsible for the *Vanishing Gradient Problem*?
The term $\frac{\partial \mathbf{h}_T}{\partial \mathbf{W}^{[hh]}}$ is responsible for VGP. This is because the partial derivative of this term has an dependence on h_{T-1} from the previous timestep.

$$\frac{\partial \mathbf{h}_T}{\partial \mathbf{W}^{[hh]}} = \left(\frac{\partial g^{[h]}(\mathbf{x}_T, \mathbf{h}_{T-1})}{\partial \mathbf{W}^{[hh]}} + \frac{\partial g^{[h]}(\mathbf{x}_T, \mathbf{h}_{T-1})}{\partial \mathbf{h}_{T-1}} \times \frac{\partial \mathbf{h}_{T-1}}{\partial \mathbf{W}^{[hh]}} \right) \tag{7}$$

This happens all the way back in time till the first timestep. With every small number $0 < x < 1$ multiplied to the chain, the resulting gradient becomes even smaller. By the time the network tries to update the initial few layers (in time), the gradient would practically be 0. This means your network manages to update only the parameters from recent timesteps while ignoring the ones way back in time.

This further results in the *Short-term Dependency Problem* where the RNN's memory only goes back a few timesteps. It cannot keep track of things from the distant past. Think of this as analogous to asking "do you remember what you ate for breakfast 8 days back?". The answer is probably "No" because a lot more recent information about your life has *drowned* out the memory of breakfast 8 days back.

2. **LIME**

(a) In LIME, we use a predictor which is a quadratic function rather than a line. How do you sample points for the decision boundary? Show how quadratic LIME can improve over a linear LIME.
Sample data points by radius or priority. Figure 2 illustrates the improvement of a quadratic LIME. Image Credits

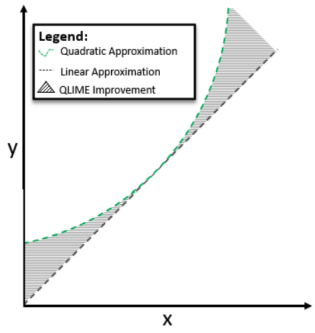


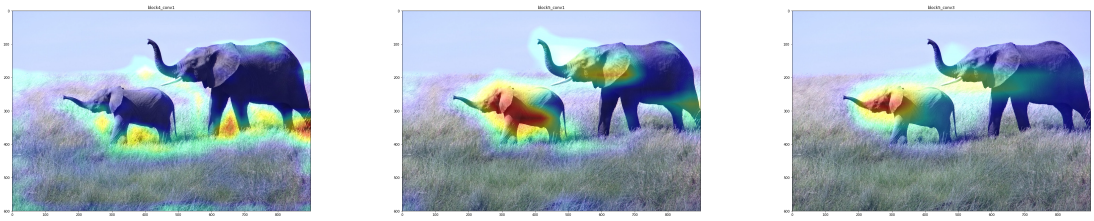
Figure 2: Caption

(b) What are the disadvantages of using LIME for explanation?
1 - Finding a proper neighbourhood is difficult. For every new region, we may have to find new kernel widths.
2 - Repeating the sampling process can produce different explanations.
3 - Locally perturbed data may not be realistic.

3. **XAI** Fig 4 illustrates CAM (Class Activation Mapping) / the heatmaps of various convolutional layers of the input image shown in Fig 3. Here, we use a large neural network, VGG16 which has 5 blocks of convolutions and each block has 3 convolutional layers. VGG16 is used to classify the images into number of classes. Elephant is one of those classes.

Figure 3: Input Image

(a) Block 1 Conv 1 (b) Block 2 Conv 1 (c) Block 3 Conv 1



(d) Block 4 Conv 1 (e) Block 5 Conv 1 (f) Block 5 Conv3

Figure 4: Heatmap through various Convolutional layers of VGG16 - Image Credit

(a) Comment on the heatmaps with respect to the image classification.
Despite being a classification network, VGG16 is able to locate the elephant in the picture. Moreover, the decisions are highly influenced by the face and ear portions of the data.

(b) In each layer, different segment of the image is activated. Why those activations are different? You can assume that *Block5 Conv3* is the decision making layer.
This is because earlier layers have filters which are looking at various parts of the image. But the class prediction largely depends on the activations of the layers just before prediction and hence, visualizing heatmaps of the layer just before prediction layer *Block5 Conv3* tells us which portion of the input image led to a particular prediction.