# CS F425 Deep Learning Final Test [AY 2021-22 Sem-I]

Department of CS & IS, BITS Pilani, K.K. Birla Goa Campus

IC: Tirtharaj Dash [tirtharaj@goa.bits-pilani.ac.in]

December 16, 2021 (Forenoon)

**Q1.** In some Machine Translation (MT) tasks, researchers observed that LSTM learns much better when the source sentences are reversed (target sentences are not reversed): LSTM's test perplexity dropped and test BLEU scores of decoded translation increased significantly. Which of the following could be possible explanation(s) for this result?

(a) It could have been caused by the introduction of many short-term dependencies to the input–output sequences in the dataset.

(b) The optimiser has an easier time "establishing communication" between the source and target sentences.

(c) Reversing the input sentences results in LSTMs with better memory utilisation.

(d) None of these

Ans: a,b,c
Reason: While we do not have a complete explanation to this phenomenon, we believe that it is caused by the introduction of many short term dependencies to the dataset. Normally, when we concatenate a source sentence with a target sentence, each word in the source sentence is far from its corresponding word in the target sentence. As a result, the problem has a large minimal time lag? By reversing the words in the source sentence, the average distance between corresponding words in the source and target language is unchanged. However, the first few words in the source language are now very close to the first few words in the target language, so the problem's minimal time lag is greatly reduced. Thus, backpropagation has an easier time "establishing communication" between the source sentence and the target sentence, which in turn results in substantially improved overall performance. Initially, we believed that reversing the input sentences would only lead to more confident predictions in the early parts of the target sentence and to less confident predictions in the later parts. However, LSTMs trained on reversed source sentences did much better on long sentences than LSTMs trained on the raw source sentences (see sec. 3.7), which suggests that reversing the input sentences results in LSTMs with better memory utilization. *[Sutskever et al. (2014): Sequence to sequence learning with neural networks. In NIPS.]*

**Q2.** In any sequence learning task, which of the following function is most appropriate to implement the gating mechanism in RNNs?

(a) `ReLU` or its variants

(b) `sigmoid`

(c) `tanh`

(d) `linear`

(e) `sin`

Ans: b
Reason: The gate has to open (1) or close (0) or something between (0,1).

**Q3.** Recall that a RNN takes in an input vector $\mathbf{x}(t)$ and a state vector $\mathbf{h}(t-1)$ and returns a new state vector $\mathbf{h}(t)$ and an output vector $\mathbf{y}(t)$. Refer the side figure. The equations are:

$$h(t) = f(w_1 x(t) + w_2 h(t-1) + b_2)$$
$$y(t) = g(w_3 h(t) + b_3)$$

where $f$ and $g$ are the activation functions at hidden and output layers, respectively.

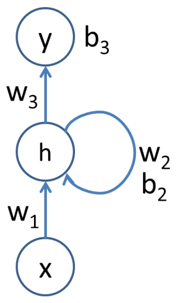
Now, given: $f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$; $\quad g(x) = x$; and, $h(0) = 0$.

Let's assume that the network is trained and the parameters (weights) of the model satisfy the following conditions:

$$b_3 = 0; \quad w_3 = 1; \quad b2 < 0;$$

and

other conditions



What are the other conditions on weight parameters such that the RNN model initially outputs 0, and as soon as it receives input 1, it starts producing 1 as output no matter what the input is? For example, if input sequence is 00100101, then output sequence would be 00111111.

(a) $w_1 + w_2 + b_2 \leq 0; \quad w_1 + b_2 \geq 0; \quad w_2 + b_2 \geq 0$

(b) $w_1 + w_2 + b_2 \geq 0; \quad w_1 + b_2 \leq 0; \quad w_2 + b_2 \leq 0$

(c) $w_1 + w_2 + b_2 \geq 0; \quad w_1 + b_2 \geq 0; \quad w_2 + b_2 \geq 0$

(d) $w_1 + w_2 + b_2 = 0; \quad w_1 + b_2 = 0; \quad w_2 + b_2 \geq 0$

(e) None of these

Ans: c
Reason: Calculate yourself!

**Q4.** Which of the following choice(s) can help alleviate the vanishing gradient problem in deep networks?

(a) Using residual connections in the network

(b) Using batch normalisation

(c) Using `ReLU` activations instead of `sigmoid`

(d) Using LSTMs instead of standard RNN cells

(e) None of these

Ans: a,b,c,d
Reason: a: allows the gradients to flow nicely due to the skip connections; b: it does not directly solve the vanishing gradient problem, but it stabilises the optimisation by not allowing the optimiser to get stuck in the local optima; c and d: straightforward.

**Q5.** You have constructed a very large and complex neural network model for representation learning. Learning adequately sparse representations for inputs has a lot of benefits in deep learning, such as extracting discriminating representation for any given input. Let $L(\mathbf{x}, \hat{\mathbf{x}})$ denote the loss function for your present model where $\mathbf{x}$ denotes the input. How will you modify this loss function such that your model is forced to learn sparse representation (select all that apply)?

(a) Modify the loss to $L' = L + \lambda \sum_{i=1}^{m} |h_i|$

(b) Modify the loss to $L' = L + \frac{\lambda}{2} \sum_{i=1}^{m} h_i^2$

(c) Modify the loss to $L' = L - \lambda \sum_{i=1}^{m} |h_i|$

(d) Modify the loss to $L' = L - \frac{\lambda}{2} \sum_{i=1}^{m} h_i^2$

(e) None of these

[Here: $m$ is the size of the representation layer, and $h_i$ is the value of the $i$th hidden unit, and $\lambda$ is the regularisation parameter.]

Ans: a,b
Reason: Sparse representations can be learned by enforcing a constraint on the number of hidden (representation) units activated given an input instance. This is done by penalizing the *activations* of the neural network, so that only a small subset of the neurons fire for any given data instance. The simplest way to achieve sparsity is to impose an $L_1$-penalty on the hidden units. Therefore, the original loss function $L$ is modified to the regularized loss function $L'$ as $L' = L + \lambda \sum_{i=1}^{m} |h_i|$ or $L' = L + \frac{\lambda}{2} \sum_{i=1}^{m} h_i^2$.

**Q6.** A contractive autoencoder is a heavily regularized encoder (normally, overcomplete) in which the hidden representation is not very sensitive to changes in input values. Consider an autoencoder with single hidden layer. Let the input vector be defined as $\mathbf{x} = (x_1, \ldots, x_d)$, and the hidden units be $\mathbf{h} = (h_1, \ldots, h_k)$ (usually, $k \geq d$).
The reconstruction loss for a single input instance is defined as:

$$L = \sum_{i=1}^{d} (x_i - \hat{x}_i)^2$$

where $\hat{\mathbf{x}}$ is the reconstructed version of the input.
The contractive autoencoder defines a new loss (let's call it $L'$) by adding a regularising term to $L$. Which of the following regularisations help(s) the contractive encoder achieve robustness? [choose all that apply]

(a) $L' = \sum_{i=1}^{d} (x_i - \hat{x}_i)^2 + \frac{\lambda}{2} \sum_{j=1}^{k} \left( \frac{\partial h_j}{\partial h_{j-1}} \right)^2$

(b) $L' = \sum_{i=1}^{d} (x_i - \hat{x}_i)^2 + \frac{\lambda}{2} \sum_{i=1}^{d} \sum_{j=1}^{k} \left( \frac{\partial h_j}{\partial x_i} \right)^2$

(c) $L' = \sum_{i=1}^{d} (x_i - \hat{x}_i)^2 + \frac{\lambda}{2} \sum_{i=1}^{d} \sum_{j=1}^{k} \frac{\partial h_j}{\partial x_i}$

(d) $L' = \sum_{i=1}^{d} (x_i - \hat{x}_i)^2 + \frac{\lambda}{2} \sum_{j=1}^{k} (\partial h_j)^2$

(e) None of these

Ans: b
Reason: This is standard definition of contractive loss. The derivative of $h$ w.r.t. $x$ conveys the fact that the representation is not sensitive to changes in the denominator (input). The answer (c) won't work as the derivative could be negative as well.

**Q7.** Consider a Generative Adversarial Network (GAN) the goal of which is to generate images of aeroplanes. Which of the following are TRUE?

(a) The aim of the discriminator is to discriminate between images of aeroplanes and non-aeroplanes.

(b) The aim of the generator is to learn the distribution of aeroplane images.

(c) Given a large sample of images of aeroplanes, after successful training of the GAN, the discriminator's loss converges to a constant value (lowest possible).

(d) Given a large sample of images of aeroplanes, after successful training of the GAN, the generator will be able to produce unseen images of aeroplanes.

(e) None of the above

Ans: b, c, d
Reason: (a) the goal is to discriminate real and fake samples; (b) correct; (c) it happens after stable training; (d) that's the whole point

**Q8.** Which of the following represent(s) the objective function for a Conditional-GAN? Here, $D$ denotes the discriminator function (a neural network), $G$ denotes the generator function, and $\mathbf{x}$ denotes a data instance.

(a) $\max_D \min_G \{-\mathbb{E}_{\mathbf{x}\sim Data} y \log(D(\mathbf{x})) - \mathbb{E}_{\mathbf{z}\sim Noise}(1-y)\log(1-D(G(\mathbf{z})))\}$

(b) $\min_D \max_G \{-\mathbb{E}_{\mathbf{x}\sim Data} \log(D(\mathbf{x}|y)) - \mathbb{E}_{\mathbf{z}\sim Noise}\log(1-D(G(\mathbf{z}|y)))\}$

(c) $\min_D \max_G \{\mathbb{E}_{\mathbf{x}\sim Data} \log(D(\mathbf{x}|y)) + \mathbb{E}_{\mathbf{z}\sim Noise}\log(1-D(G(\mathbf{z}|y)))\}$

(d) $\min_D \max_G \{-\mathbb{E}_{\mathbf{x}\sim Data}(1-\log(D(\mathbf{x}))) - \mathbb{E}_{\mathbf{z}\sim Noise}\log(D(G(\mathbf{z})))\}$

(e) None of these

Ans: b
Reason: This is straightforward. Taught in class.

**Q9.** Variational Autoencoder (VAE) is a neural generative model. Here are a few computational steps that are involved behind the working of VAE (showing one pass through the computational graph assuming Gaussian prior and posterior):

| | | |
|---|---|---|
| S1 | Push $\mathbf{x}$ through the encoder | $\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x = M(\mathbf{x}), \Sigma(\mathbf{x})$ |
| S2 | Sample $\mathbf{z}$ | $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x)$ |
| S3 | Push $\mathbf{z}$ through the decoder | $\hat{\mathbf{x}} = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ |
| S4 | Compute reconstruction loss | $\mathcal{L}_{recon} = MSE(\mathbf{x}, \hat{\mathbf{x}})$ |
| S5 | Compute variational loss | $\mathcal{L}_{var} = -\mathrm{KL}[\mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x)\|\|\mathcal{N}(0, I)]$ |
| S6 | Compute VAE loss | $\mathcal{L} = \mathcal{L}_{recon} + \mathcal{L}_{var}$ |

Which of the above step will cause difficulty during backpropagation, and how is it solved in practice?

(a) S1 (solved by setting it to 1)

(b) S2 (solved by reparameterisation trick)

(c) S3 (solved by weight tying)

(d) S5 (solved by reparameterisation trick)

(e) None of the above

Ans: b

Reason: Backpropagation cannot handle sampling. Replace the step with $\mathbf{z} = \boldsymbol{\mu}_x + \boldsymbol{\sigma}_x \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$

**Q10.** The solution for the objective function for a GAN lies in the:

(a) Global minimum of the GAN objective

(b) Global maximum of the GAN objective

(c) Saddle point of the GAN objective

(d) None of these

Ans: c

Reason: Saddle point is the point where the generator achieves its maximum and the discriminator achieves its minimum

**Q11.** We want to cluster data using a neural network (unsupervised learning). Let's denote a set of data sample as $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where each pattern $\mathbf{x}_i$ is $d$-dimensional; $\mathbf{x}_i = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(d)})$.

Here is a naïve structure of the network consisting of only two layers: an input layer and an output layer. The input layer has $d$ neurons, and the output layer will have $K$ neurons, one for each cluster. The network is fully connected. A weight is denoted as $w_{ij}$ referring to the synaptic strength from input neuron $i$ to the output neuron $j$. The output uses linear activation.

Algorithm 1 provides the formal steps for constructing the clustering model. The algorithm can be said to have converged if one of the following conditions is met: (i) Weights do not change; or (ii) The cluster associated with each pattern in $\mathcal{X}$ remains same for different iteration.

```
Algorithm 1 Neuro-clustering
Require: $\mathcal{X}$, $K$
Ensure: $\mathbf{X} \neq \emptyset$, $K \neq 0$
  Randomly initialize $\mathbf{w}$ to non-zero
  $\alpha \leftarrow 0.99$ (Learning rate)
  while Until convergence do
    Randomly draw a $p$th pattern from $\mathcal{X}$
    $j \leftarrow 1$
    while $j \leq K$ do
      $y_j = \sum_{i=1}^{d} (x_p^{(i)} - w_{ij})^2$        ▷ Squared Euclidean distance
    end while
    $\boxed{\text{Box1}}$        ▷ winner neuron (represeting the cluster) in the output layer
    Learn the weights associated with this winner neuron:
                                   $\boxed{\text{Box2}}$
  end while
```

You have to fill up the missing part in the $\boxed{\text{boxes}}$ above.

(a) Box1: $c = \arg\min_j \{y_1, y_2, \ldots, y_k\}$; Box2: $w_{ic} \leftarrow w_{ic} + \alpha \left( x_p^{(i)} - w_{ic} \right)$

(b) Box1: $c = \arg\max_j \{y_1, y_2, \ldots, y_k\}$; Box2: $w_{ic} \leftarrow w_{ic} + \alpha \left( x_p^{(i)} - w_{ic} \right)$

(c) Box1: $c = \arg\min_j\{y_1, y_2, \ldots, y_k\}$; Box2: $w_{ic} \leftarrow w_{ic} - \alpha \left(x_p^{(i)} - w_{ic}\right)$

(d) Box1: $c = \arg\max_j\{y_1, y_2, \ldots, y_k\}$; Box2: $w_{ic} \leftarrow w_{ic} - \alpha \left(x_p^{(i)} - w_{ic}\right)$

(e) None of the above

Ans: a
Reason: The idea is to update the weights such that the network achieves the lowest (ideally, 0) Euclidean distance between the winner neuron and the data instance. This is a common learning strategy in unsupervised learning in neural network; for example: Self-organising maps

**Q12.** Consider a two-player game like GANs with objective function $L(x, y)$, and we want to compute $\min_x \max_y L(x, y)$. When is $\min_x \max_y L(x, y)$ equal to $\max_y \min_x L(x, y)$?

(a) $L$ needs to be convex in the maximisation variables and concave in the minimisation variables.

(b) $L$ needs to be convex in the minimisation variables and concave in the maximisation variables.

(c) $L$ needs to be convex in the minimisation variables and concave in the minimisation variables.

(d) $L$ needs to be convex in the maximisation variables and concave in the maximisation variables.

(e) None of these

Ans: b
Reason: The max-min is always less than or equal to min-max. The best example is to consider the function sin(x + y), for which the min-max is 1, but the max-min is -1. In general, the function needs to be convearex in the minimization variables and concave in the maximization variables for the two to be equal. This is a well known result in minimax optimization.

**Q13.** Attention-based sequence models have the greatest advantage over standard sequence models (i.e. models which do not use an attention mechanism) when:

(a) When the input sequences are long

(b) When the input sequences are short

(c) Attention models are not affected by sequence lengths

(d) Attention models behave identically while dealing with short and long sequences

(e) None of these

(Here "long" and "short" need not be quantified. Understand these in the context of sequence learning and as discussed in the lectures.)

Ans: a
Reason: ... they help overcome some challenges with Recurrent Neural Networks(RNNs) such as performance degradation with increase in length of the input and the computational inefficiencies resulting from sequential processing of input (source: `https://arxiv.org/pdf/1904.02874.pdf`)

**Q14.** Computation of which of the following is intractable in the context of a variational autoencoder (VAE)? (This intractability leads to variational inference in a VAE.)

(a) $p(\mathbf{x}|\mathbf{z})$

(b) $p(\mathbf{x})$

(c) $p(\mathbf{z}|\mathbf{x})$

(d) $p(\mathbf{z})$

(e) None of these

Ans: b
Reason: $p(\mathbf{x})$ is calculated as:

$$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

**Q15.** For a VAE with latent dimension $[1 \times 3]$, consider that the encoder outputs $\boldsymbol{\sigma} = [0.1, 0.2, 0.1]$ and sampled $\boldsymbol{\epsilon} = [0.01, 0.02, 0.03]$ from $\mathcal{N}(\mathbf{0}, I)$. What is the encoder's output mean vector to produce the latent vector $\mathbf{z} = [0.3, 0.1, 0.5]$?

(a) $[0.040, 0.040, 0.080]$

(b) $[0.301, 0.104, 0.503]$

(c) $[0.299, 0.096, 0.497]$

(d) $[0.200, -0.100, 0.400]$

(e) None of these

Ans: c
Reason: $\boldsymbol{\mu} = z - \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}$

............................... ✂ ............................... ✂ ...............................

FILL YOUR CHOICES IN THE TABLE BELOW

**BITS ID:** **NAME:**

| | | | | |
|---|---|---|---|---|
| **1.** | **2.** | **3.** | **4.** | **5.** |
| **6.** | **7.** | **8.** | **9.** | **10.** |
| **11.** | **12.** | **13.** | **14.** | **15.** |