

CSE641: Deep Learning (2023) End-Sem Examination

Duration: 2 hr

Total marks: 50

1. Activation-based questions:

- a. Consider the following trained neural network employed to classify a sentence as spam ($y=1$) or not spam ($y=0$). Why is this setup not correct for inference of an incoming text t encoded via feature vector x : [1]

$$\hat{y} = \sigma(\text{Relu}(W^T x))$$

- b. Considering the following step function $H(x)$, describe why it is not a good choice for the activation function, despite being a nonlinear function? [1]

$$H(x) := \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

- c. Consider the parametric soft plus activation function $S(x)$. [2+1]

$$S(x) = \alpha x + \ln(1 + e^x)$$

For very large values of x , $x \gg 0$ or $x \ll 0$ but **not** approaching infinity:

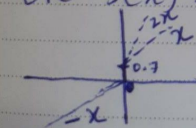
- i. What other activation function (say $f(x)$) does $S(x)$ mimic? What issue with $H(x)$ in 1.b does this said activation function ($f(x)$) solve?
 - ii. What other activation function does the first-order derivative $S'(x)$ mimic?
- a. Relu output will always be ≥ 0 and sigmoid for ≥ 0 will always be ≥ 0.5 which means that the output will always yield y as 1 i.e all mails will end up in spam.
- b. $H'(x) = 0$ for all values of x , which means no gradient update can take place.

$S(x) = \alpha x + \ln(1 + e^x)$
for $0 < \alpha < 1$

(i) when $x = 0$
 $S(x) = 0 + \ln(2) \approx 0.7$

when $x \gg 0$
 $\alpha x \approx x$
 $\ln(1 + e^x) \approx \ln(e^x) \approx x$
OR $S(x) \approx x + x \approx 2x$ ~~or x~~

when $x \ll 0$
 $\alpha x \approx -x$
 $\ln(1 + e^x) \approx \ln(1 + 0) \approx 0$
OR $S(x) = -x + 0 \approx -x$

 \approx Leaky Relu

c.

(ii) $S(x) = \alpha x + \ln(1 + e^x)$
 $S'(x) = \alpha + \frac{e^x}{1 + e^x}$
 $= \alpha + \frac{1}{1 + e^{-x}}$

Thus
 $S'(x) = \alpha + \text{sigmoid}(x)$
As $0 < \sigma(x) < 1$
 $\alpha < S'(x) < 1 + \alpha$.

Note from above discussion we observe that $S(x)$ gradient is not zero unlike $H'(x)$.

2. Discuss Inductive bias in GNN with an example. [2]

The inductive bias in GNN is that it favors the relationship among nodes instead of their sequential or spatial structure.

Dependency tree. See example in slide. GNN.

3. What is meant by “stochastic” in gradient descent (SGD)? Intuitively speaking, what role does shuffle/order of train samples play in SGD? [2]

“Stochastic” aka randomness in SGD comes into play as SGD giving a probabilistic approximation of the true gradient.

As SGD applies a greedy approach (optimise for single point), the order in which the points are selected affects the direction of gradient. Random shuffling allows for abruptness and prevent local minima.

4. For the following code snippet fill in the blank a? to h? so the code can perform a forward operation for a dense layer. [2]

```
class Dense:
    def __init__(self, number_inputs, number_neurons):
        self.weights = np.random.randn(a?, b?)
        self.biases = np.random.randn(c?, d?)
    def forward(self, inputs):
        self.output = np.e?(f?, g?) + h?
```

A: number_inputs, B: number_neurons, C:1, D: number_neurons, E: np.dot OR E: np.matmul, F: inputs, G: self.weights, H: self.biases.

Depending on how you used number_inputs, number_neurons and assumed inputs shape, you may have used Transpose as well inside the E() part. Also if you shuffled A and B then it should reflect in your final output as well. The self.output operation should not be a compilation error.

5. What are $Q(s,a)$ and $\pi(s)$? Define them. Write their typical loss functions for the two classes of reinforcement algorithms. [2+2]

$Q(s, a)$ is the value function: Given a state s and an action a , estimate the expected reward if we take action a from the state s . [1]

$\pi(s)$ is the policy function: Given a state s , find the best action that can give us the best reward. [1]

Loss functions:

Value/Q learning-based:

$$\mathcal{L} = \mathbb{E} \left[\left\| \overbrace{\left(r + \gamma \max_{a'} Q(s', a') \right)}^{\text{target}} - \overbrace{Q(s, a)}^{\text{predicted}} \right\|^2 \right]$$

Policy learning-based:

$$\text{loss} = -\log P(a_t | s_t) R_t$$

log-likelihood of action
reward

6. For a sequence $S = \{s_1, s_2, \dots, s_{10}\}$ where each $s_i \in \mathbb{R}^{100}$, we wish to compute the sentiment (positive or negative) using a single hidden layer RNN model with sigmoid function at the output layer. Compute the number of parameters we need to learn in a default settings (of Keras/Tensorflow/Pytorch). [4]

Structure of the model:

Input: 100 + 1 (bias) neurons

Hidden: h + 1 (bias) neurons

output : 1 neuron (sigmoid)

Weight matrix:

Input to hidden: 101 * h

Hidden to hidden: (h+1) * h

Hidden to output: (h+1) * 1

Total: $\{101 * h\} + \{(h + 1) * h\} + \{(h+1) * 1\}$

If you assumed a value of h and gave the answer in absolute value that too is valid as long as bias was considered. Additionally, if you assumed input embedding matrix instead of a single input embedding that too will be considered.

By default bias=True for pytorch and TF/Keras

Reference: https://www.tensorflow.org/api_docs/python/tf/keras/layers/SimpleRNN

<https://pytorch.org/docs/stable/generated/torch.nn.RNN.html>

<https://towardsdatascience.com/counting-no-of-parameters-in-deep-learning-models-by-hand-8f1716241889>

7. Write the objective function for a typical GAN. Mention and elaborate on the convergence issue with it and how do we solve it (with eqn). [5]

MinMax Objective function

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

Issue with generator: Initially, even though the generator has not learned the distribution properly, it will receive lower gradients; hence, its convergence will also be slow.

Solution: Remove the complement part (i.e., 1 -) from the gradient-ascent and convert it into gradient-descent such that the generator will receive higher gradients initially and it has a better chance to converge quickly.

GANs - Issues with generator

- Gradient-descent on Generator

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(z^{(i)})))$$



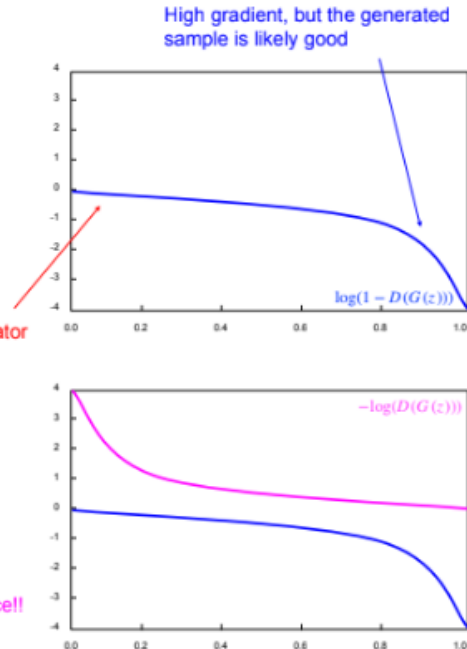
- Gradient-ascent on Generator

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log(D(G(z^{(i)})))$$

https://arxiv.org/pdf/1605.04497v1.pdf

Generated sample is likely fake, so the generator needs to learn; however, the gradient is flat

Works well in practice!!



8. At the inference time, we only use ____\$1____ in VAEs and ____\$2____ in auto encoders. [1+1+2+2]

- Discuss the possibility of using only ____\$1____ in auto encoder.
- Discuss the possibility of using both encoder and decoder in VAE.

Note:

- \$1 and \$2 are placeholders and you have to mention them.
- For both points, if it's possible, how can we do it and what would be the consequences, if any. If not possible, why not.

\$1: Decoder

\$2: Encoder

- Decoder in auto-encoder at the inference:** It does not make sense as there is no way we can compute a latent representation z for an input – remember encoder is not available.
 - Encoder-Decoder in VAE:** VAE is a generative model. It means, at the inference time, it should generate output from noise/distribution (or from nowhere in simple terms). Having an encoder means we need an input, which will violate the objective of generative models.
9. In a bi-modal setup (e.g., A and B), mention the input modalities for the triplets of query, key, and value in a typical self-attention module. For example, if “key” takes the A

modality, then what modalities should query and value take? Give all possible and valid cases. [6]

Note: There will be a negative marking (0.5) for every invalid case.

The question asks for self-attention that means inputs to key, value, and query must be the same. Depending on the number of modalities, only following are the valid cases.

Unimodality:

Query: A, Key: A, Value: A [2]

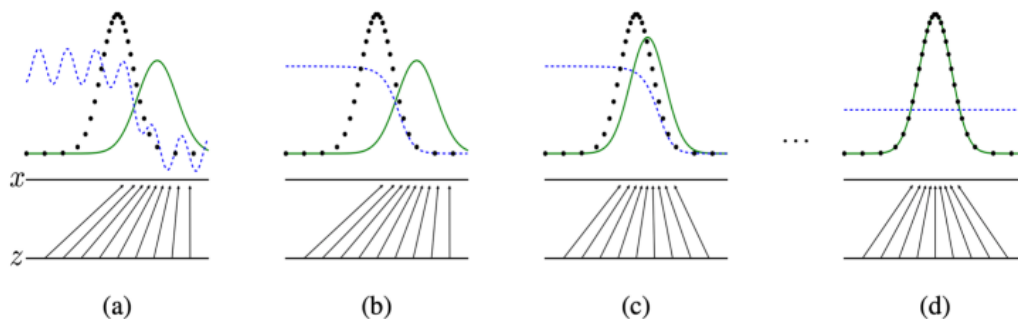
Query: B, Key: B, Value: B [2]

Bi-modality:

Query: A+B, Key: A+B, Value: A+B [2]

Note: A+B means some combination of A and B.

10. Appropriately describe the following images considering the plots. Make necessary assumptions, if any. **Note:** We don't need equations. [2*4]



Blue dotted-line: Discriminator output $D(x)$

Black-dotted line: Real data distribution (p_{data})

Green solid line: Generated data distribution. (p_g)

1 mark each for correctly listing the graphs, and 1 mark each for writing a line or two about it.

a. Initialization of GAN:

i. Discriminator output: Random or partially correct classifier.

b. Update on Discriminator:

- i. In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D(x) = p_{data}(x) / \{p_{data}(x) + p_g(x)\}$.
- c. Update on generator:
 - i. Generated data distribution moves closer to the real data distribution: After an update to G, gradient of D has guided $G(z)$ to flow to regions that are more likely to be classified as data.
- d. Convergence:
 - i. Generated data distribution matches the distribution of real data. After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g \sim p_{data}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(x) = 0.5$

11. Attempt either part a or b, only the first answer will be checked.

- a. Define the types of density function modeling in generative models. Mention and justify the types for the following classes of models: VAEs, GANs, PixelCNN/RNN. **[6]**

Implicit and Explicit density function modellings.

Explicit: When the real data distribution plays a direct role in learning the generated distribution through the gradient of loss. E.g., VAEs, Pixel RNN/CNN

Implicit: Otherwise, E.g., GAN. The generator does not know the real data. It learns through a signal from the discriminator as to whether its data was deemed real or fake.

‘OR’

- b. Summarize w.r.t. the *input*, *output*, *architecture highlight*, and *objective function* for the following models (equations are not necessary). **[2+2+2]**
 - i. Vision Transformer [Dosovitskiy et al., 2021]
 - ii. Visual BERT [Li et al., 2019]
 - iii. ViLBERT [Lu et al., 2019]

See slides. Multimodality.