OSE QUESTIONS

Name: Biswajit Palit

Matriculation Number: 50071214

Q1. List five different tasks that belong to the field of natural language processing.

A1. a)Text Classification b)Named Entity Recognition c)Question Answering d)Machine Translation e)Text Generation

Q2. What is the fundamental difference between econometrics/statistics and supervised machine learning

A2. The primary difference between Econometrics/statistics and supervised machine learning is in their respective objectives as to why they are performed. Econometrics/ statistics is mostly used to establish causality and inferencial relationships between variables. Supervised machine learning on the other hand focusses more on predicting unseen data, rather than understanding the causal relationships.

Q3.Can you use stochastic gradient descent to tune the hyperparameters of a random forrest. If not, why?

A3. Stochastic Gradient descent cannot be implemented to directly tune the hyperparameters of a random forrest. The reason for this is the nature of Random Forests and the way they are trained. Stochastic Gradient Descent (SGD) is a gradient-based optimization technique commonly used for finding the global minima while training machine learning models, especially in the context of neural networks and linear models. Random Forest, on the other hand is an ensemble of decision trees and doesn't have a differentiable objective function that SGD requires for optimization.

Q4. What is imbalanced data and why can it be a problem in machine learning?

A4. Imbalanced data refers to a situation in which the classes or categories in a dataset are not represented equally. This can lead to a skewed distribution of classes within the dataset. For example, in a binary classification problem, if there are 95% instances of class A and only 5% instances of class B, the data is imbalanced. In this case the model may suffer from a high degree of bias favouring the majority class. Imbalanced data can result in models that don't generalize well to new, unseen data, particularly for the minority class, leading to occurrence of skewed patterns, high false negative rates and misleading metrics like achieving high accuracy, while predicting only the majority class.

Q5. Why are samples split into training and test data in machine learning?

A5. Samples are split into training and test data in machine learning to assess how well a trained model generalizes to new, unseen data. Commonly, the data is split into three sets: training, validation, and test sets to evaluate the model's performance on unseen data. Training data is used to train the model's parameters, while the test data is used to assess how well the model generalizes to new, unseen examples. This separation

helps ensure that the model's performance is reliable and that it hasn't simply memorized the training data, which could lead to poor performance on real-world data.

Q6. Describe the pros and cons of word and character level tokenization.

A6. Word-level tokenization captures the semantic meaning of individual words, which is crucial for understanding the context and sentiment of the text, while character level tokenisation fails to capture the semantic essence of the individual words. Also in case of a large dataset word level tokenisation reduces the dimensionality of the text data which makes the data more manageable for the machine learning algorithms. Character Level tokenisation significantly increases dimensionality of the dataset, which also significantly increases computational requirements.
Character level tokenisation is useful for handling rare words that might not be in the vocabulary of word level tokenization. It is also useful in handling typos in the dataset. Character level tokenization is more useful than word in case of complex scripts especially where the boundaries are not so well defined.

Q7. Why does fine-tuning usually give you a better performing model than feature extraction?

A7. Fine-tuning tends to produce better results than feature extraction due to its ability to adapt a pre-trained model for a specific task. While both techniques utilize pre-trained knowledge, fine-tuning goes a step further by utilizing the features and training them using hyperparameters to better suit the nuances of the target task, resulting in improved accuracy. It is helpful when the new task is not very similar to the pre-trained task.

Q8. What are advantages over feature extraction over fine-tuning

A8. In feature extraction, we take a pre-trained model and remove the final layers designed for the specific task on which the model was originally trained. We then use the remaining layers as a feature extractor. These layers have already learned to recognize meaningful features in the data, so we can extract those features for the specific task. So it is ideal for tasks closely related to the pre-trained model's original task. This makes feature extraction quicker for training the model.

Q9. Why are neural networks trained on GPUs or other specialized hardware?

A9. Neural networks are trained on GPU primarily because it involves numerous matrix multiplications and mathematical operations. Given GPU's high bandwidth, it can perform simultaneous training of several models which is crucial for handling the large amounts of data used in deep learning without causing bottlenecks and at the same time being time efficient.

Q10. How can you write pytorch code that uses a GPU if it is available but also runs on a laptop that does not have a GPU.

A10. A way to write a pytorch code that utilizes GPU, if available, but also runs on a laptop which does not have GPU.

```
import torch
```

```
# Check if GPU is available, else use CPU
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

The code checks if a GPU is available using torch.cuda.is_available(). If a GPU is available, the tensor is created on the GPU using .to("cuda"). If no GPU is available, the tensor is created on the CPU using .to("cpu").

Q11. How many trainable parameters would the neural network in this video have if we remove the second hidden layer but leave it otherwise unchanged?

A11. In the video, there are 784 pixels. Let us say they have grayscale value a. For each neuron in the third layer(assuming the second layer is removed) 784 weights are assigned and multiplied with the a score of the pixels. So 784 weights for each of the 16 neurons. Therefore 784 X 16 in the first step. The same process repeated in the second step. 16 weights for each of the 10 neurons making 16 X 10 weights. Now for each neuron there is a bias term. Therefore in the first step we will add 16 bias terms and in the second step we add 10 bias terms. Therefore the total number of trainable parameters are 784 X 16 + 16 X 10 + 16 + 10 = 12730 parameters.

Q12. Why are nonlinearities used in neural networks? Name at least three different nonlinearities.

A12. Nonlinearities, also known as activation functions, are used in neural networks for two main reasons: to enable the network to capture complex relationships between inputs and outputs, and to introduce nonlinear relationships, allowing it to extract meaningful features as we go deeper into the model which helps to generalize and accurately predict new unseen data.
Examples of non linearity in neural networks are:

1. Rectified Linear Unit (ReLU): This activation returns the input value if it's positive and zero otherwise.
2. Sigmoid: Maps input values to the range [0, 1], often used in binary classification problems.
3. Hyperbolic Tangent (tanh): Similar to sigmoid, but maps input values to the range [-1, 1].
4. Softmax: Converts a vector of values into a probability distribution, commonly used in the output layer for multi-class classification.

Q13. Some would say that softmax is a bad name. What would be a better name and why?

A13. "Softmax" is a common term in machine learning and neural networks. It's an activation function(nonlinearities) used in the output layer of a neural network to convert a vector of raw scores (also known as logits) into a probability distribution. It is often used when we want the neural network to make predictions among several mutually exclusive classes. Mathematically, the softmax function takes the logits, exponentiates each element to make them positive, and then normalizes the values to sum up to 1. This normalized vector represents the probabilities of different classes in a multi-class classification problem. Softmax is a pretty relevant name, in my opinion, because it has relevance to the mathematics behind it. Basically, 'soft' refers to the transformation of the raw scores into probabilities and smoothening them into a probability distribution. However 'max' may refer to the fact that the class with the highest score gets the

maximum probability, without changing the order of the classes. In this sense, the “max” part of the name can be a bit misleading, because there is no maximization happening here. An alternative name could be “Normalized Exponential Probability(nepr)”.

Q14. What is the purpose of DataLoaders in pytorch?

A14. In PyTorch, DataLoaders are a component that helps manage and streamline the process of loading and preprocessing data for training and evaluation of machine learning models. They combine a dataset with settings for batch size, shuffling, and parallel loading, making it easier to iterate through data during model training and evaluation. Shuffling randomizes the train set which prevents the model to overlearn and helps in better generalization. It separately takes train and eval datas, and separately computes the computation matrix for each dataset.

Q15. Name a few different optimizers that are used to train deep neural networks

A15. Few optimizers used to train deep neural networks are as follows:
Stochastic Gradient Descent (SGD): The classic optimization algorithm that updates the model's parameters by considering the gradient of the loss function for each individual data point in a batch.
Adam (Adaptive Moment Estimation): An adaptive learning rate optimization algorithm that combines the advantages of both Adagrad and RMSProp. It adapts the learning rate based on the past gradients and squared gradients.
AdamW: An adaptation of Adam that includes weight decay to prevent overfitting.

Q16. What happens when the batch size during the optimization is set too small?

A16. In case of a small batch size, the number of iterations per epoch increases but the time taken per iteration increases. Hence the training time for one epoch remains relatively constant irrespective of the batch size.
In case of a large dataset, if the batch size is small it may lead to delayed convergence, less effective GPU utilization, possible overfitting and poor generelization due to increased variance in one sample leading to fluctuations in optimization process. On the other hand in a small dataset, small batch size is desirable due to efficient convergence and low variability. Hoverver choice of batch size must also depend on the structure of the dataset. In case of an imbalance data, a smaller batch size may capture the observations of the minority class, which might be ignored when the batch size is big.

Q17. What happens when the batch size during the optimization is set too large?

A17. In case of a large batch size, the number of iterations per epoch increases but the time taken per iteration reduces. Hence the training time for one epoch remains relatively constant irrespective of the batch size. In case of a large dataset, having a big batch size can be better because in each sample, the model can train more observation. This would make the training more efficient. In case of a small dataset, if the batch size is too big, there may arise a chance of underfitting leading to loss of generalization. The model might perform well on the training data, but on new unseen data, the model might fail to perform well. Moreover there is a possibility that with very large batch sizes the model fixates on the local minima instead of searching for the global minima, hence leading to slower convergence.

Q18. Why can the feed-forward neural network we implemented for image classification not be used for language modelling?

A18. Language models use words , characters, embedding layers which tokenizes the strings to numeric tokens. While image classification models takes the whole image, divides them into small parts, and takes the pixels of the small parts as numeric data points into 2D or 3D arrays. So in case of image classification, the model takes the numerics to be the input vector while language models have to keep the words in storage to understand the semantics and sentiment. Hence feed forward neural network for image classification can not be implemented for language modelling.

Q19. Why is an encoder-decoder architecture used for machine translation (instead of the simpler encoder only architecture we used for language modelling)

A19. While encoding models, like those used in sentiment analysis or text classification, focus on creating context vectors to capture information about the input text, they typically stop at this step. For machine translation tasks, the encoded output(which captures the context of the text) is used as input for the decoder for it to understand the context of the text in order to generate a meaningful sequence of texts in a different language. In normal classification tasks, we stop at the step of creating the context vector because it enables us to do all the tasks that we want. But in case of machine translation, we have to go one more step forward, where we take the context vector and pass it through the decoder to get the meaningful output in the other language.

Q20. Is it a good idea to base your final project on a paper or blogpost from 2015? Why or why not?

A20. Whether it is sensible to base a project on a blogpost or paper from 2015 will depend on certain factors such as relevance, advancement of the field since then, validation and reproducibility. For example, if I use some model from 2015, I will not be able to utilize the BERT model, which was introduced in 2018. Now whether it is sensible to do that will depend on the tasks I want to perform, that is it is completely task-specific.

Q21. Do you agree with the following sentence: To get the best model performance, you should train a model from scratch in Pytorch so you can influence every step of the process.

A21. Whether to train the model from scratch depends on the kind of dataset I am working with and whether that kind of dataset has been pre-trained on a specific model. In case I have a dataset that does not correspond to any of the pre-trained models, it is better to go through the process of a feed-forward neural network for the model to adapt to the particular dataset. Moreover, a pre-trained model might comprise bias, so if I am unsure as to whether the pre-trained model has bias its better to train it from scratch specific to my task. However, it should also be kept in mind that the process of training from scratch is extremely time-consuming and computationally expensive, so the decision to do so will also depend on that.

Q22. What is an example of an encoder-only model?

A22. An encoder only task example is text classification. An example of an encoder-only model is bert-base-uncased or bert-large-uncased model. Any model of BERT is an encoder model as it is designed as a pre-trained language representation model for text understanding and various natural language processing tasks.

Q23. What is the vanishing gradient problem and how does it affect training?

A23. The vanishing gradient problem occurs while training deep neural networks with many layers. The gradients are the derivatives of the loss function with respect to the parameters of the model. So when the gradients, while being back-propagated through the layers, become close to 0, we encounter the vanishing gradient problem. When this problem arises, it can lead to slow convergence, and in the worst case might lead to no convergence at all.

Q24. Which model has a longer memory: RNN or Transformer?

A24. In terms of memory, transformers generally have longer memory than recurrent neural networks (RNNs). This is primarily due to the self-attention mechanisms and positional embeddings used in transformer models. These mechanisms allow transformers to capture the relationships and dependencies between words or characters in a text across longer distances in a more effective manner than traditional RNNs. Transformers excel at handling long-range dependencies, making them better suited for tasks that require a broader context understanding. In this sense, transformers have better and longer memory as compared to Recurrent Neural Network

Q25. What is the fundamental component of the transformer architecture?

A25. The fundamental component of the transformer architecture is the attention mechanism. This mechanism plays a pivotal role in enabling the transformer model to capture intricate dependencies and relationships between various elements within an input sequence. It accomplishes this by computing attention scores, which quantify the similarity between elements in the sequence. These attention scores are then employed to create weighted combinations of the input data, effectively capturing context and relationships in a highly efficient manner.