UNIVERSITY OF BONN, DEPARTMENT OF ECONOMICS

OSE - Introduction to Deep Learning

Mudabbira Mushtary
September 10, 2023

1 Questions and Answers

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

Translation, summarization, classification, topic modelling and relation extraction.

1.2 What is the fundamental difference between econometrics/statistics and suprevised machine learning

Econometrics set out to build probabilistic models designed to describe economic phenomena, while machine learning uses algorithms capable of learning from their mistakes, generally for classification purposes (sounds, images, etc.).

1.3 Can you use stochastic gradient descent to tune the hyperparameters of a random forest? If not, why?

No, stochastic gradient descent can not be used to tune the hyperparameters of a random forest because random forests have discrete hyperparameters (e.g. tree depth, number of trees, number of features, etc.)

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

A classification data set with skewed class proportions is called imbalanced. Imbalanced data is a common problem in machine learning, which brings challenges to feature correlation, class separation and evaluation, and results in poor model performance.

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

In machine learning, data splitting is typically done to avoid overfitting.

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

- Pros(Word-level Tokenization):

  Word-level tokenization captures the semantic meaning of words, making it suitable for tasks where word semantics are important, such as sentiment analysis or machine translation. In addition, it results in more interpretable representations because words are human-readable. This can be beneficial for model debugging and understanding.

- Cons(Word-level Tokenization):

  It struggles with out-of-vocabulary words, which are words not present in the vocabulary. Handling OOV words can be challenging, and they may be replaced with a special token or subword-level representation. Additionally, Word-level tokenization may not capture fine-grained morphological information, which can be important in languages with complex word forms (e.g., agglutinative languages like Finnish or Turkish).

- Pros(Character-level Tokenization):

  Character-level tokenization is more robust to OOV words since any word can be represented as a sequence of characters. This is particularly advantageous for languages with a large vocabulary or when dealing with domain-specific terms. Moreover, character-level tokenization is language-agnostic and works for any language, making it suitable for multilingual or low-resource language scenarios. It also captures subword information, which can be useful for tasks like Named Entity Recognition (NER), part-of-speech tagging, and morphological analysis.

- Cons(Character-level Tokenization):

  Character-level tokenization results in longer sequences compared to word-level tokenization. This can increase computational demands and training time. In addition, it may not capture the semantic meaning of words as well as word-level tokenization. It treats words as sequences of characters without considering their meanings.

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

Fine-tuning tunes the final parts of the network to better suit the problem at hand. Here, a feature extraction would mean training, or continuing training, the first layers as well, which is usually not desirable considering the computational cost.

1.8 What are advantages over feature extraction over fine-tuning?

Feature extraction and fine-tuning are two common approaches in transfer learning, especially in the context of deep learning and neural networks. They each have their advantages and are

suited for different scenarios. Here are the advantages of feature extraction over fine-tuning: reduced computational cost, reduced data requirements, and reduced risk of overfitting.

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

Neural networks form the basis of deep learning (a neural network with three or more layers) and are designed to run in parallel, with each task running independently of the other. This makes GPUs more suitable for processing the enormous data sets and complex mathematical data used to train neural networks.

1.10 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.

```
import torch

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

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

It's important to note that the original network was able to capture complex features and patterns in the handwritten digits. Removing a hidden layer from the network may lead to a reduction in the network's ability to effectively represent these patterns. Specifically, if the second layer in the network shown in the video is removed, it could result in the loss of recognition of various relevant "little edges" that correspond to different numbers.

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

Nonlinearities, also known as activation functions, are used in neural networks to introduce nonlinearity to the model. Without nonlinearities, the entire neural network would be equivalent to a linear transformation, which severely limits its expressive power. Nonlinearities allow neural networks to capture complex relationships and patterns in data. Here are three different nonlinearities commonly used in neural networks: Rectified Linear Unit (ReLU), Sigmoid, Hyperbolic Tangent (Tanh)

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

The term "softmax" is often misunderstood, as it is not a smooth maximum function, but rather a smooth approximation to the arg max function. The arg max function returns the

index of a vector's largest element. Therefore, some experts prefer the more precise term "softargmax," but "softmax" is the conventional term used in machine learning.

1.14 What is the purpose of DataLoaders in pytorch?

The PyTorch DataLoader class is a powerful utility class that is essential for loading and preprocessing data for deep learning models. It provides efficient data loading, data augmentation, flexibility, and shuffling capabilities, making it a popular choice for deep learning practitioners.

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

Gradient Descent. Stochastic Gradient Descent (SGD) Mini Batch Stochastic Gradient Descent (MB-SGD) SGD with momentum. Nesterov Accelerated Gradient (NAG) Adaptive Gradient (AdaGrad) AdaDelta. RMSprop.

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

The batch size can be understood as a trade-off between accuracy and speed. Smaller batch sizes can provide better accuracy, but can be computationally expensive and time-consuming

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

The batch size can be understood as a trade-off between accuracy and speed. Large batch sizes can lead to faster training times but may result in lower accuracy and overfitting.

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

The feed-forward neural network lacks the mechanisms necessary to handle the sequential and contextual nature of language, which is why specialized architectures like RNNs and transformers are used for language modeling tasks.

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

Encoder-decoder architectures can handle inputs and outputs that both consist of variable-length sequences and thus are suitable for sequence-to-sequence problems such as machine translation. The encoder takes a variable-length sequence as input and transforms it into a state with a fixed shape.

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

It may not be wise to rely solely on a paper or blogpost from 2015 for the final project, as transformer models were not yet available at that time. The methods used in the paper or blogpost may be outdated by now. However, it could still be helpful to compare non-transformer based approaches with current transformer based ones to identify differences in performance.

1.21 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.

The choice to either train a model from the beginning or utilize transfer learning depends on the individual situation and objectives. Although starting from scratch gives one complete command over the model, it may not always yield the best performance, particularly if there are available pre-trained models that are relevant to the task at hand. A frequent and efficient method in various machine learning and deep learning tasks is to fine-tune pre-trained models.

1.22 What is an example of an encoder-only model?

BERT

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

Vanishing gradient problem is a phenomenon that occurs during the training of deep neural networks, where the gradients that are used to update the network become extremely small or "vanish" as they are backpropogated from the output layers to the earlier layers.

1.24 Which model has a longer memory: RNN or Transformer?

Compared with RNN-based models, Transformer has shown superior performance in capturing long-term dependencies

1.25 What is the fundamental component of the transformer architecture?

The fundamental component of the transformer architecture is the "self-attention" mechanism, often referred to simply as "attention." The attention mechanism is a key innovation that allows transformers to capture relationships between different positions in a sequence of data, making them highly effective for tasks involving sequential or structured data, such as natural language processing.