Transformer in PyTorch

SYDE 599 Deep Learning F23

November 9, 2023

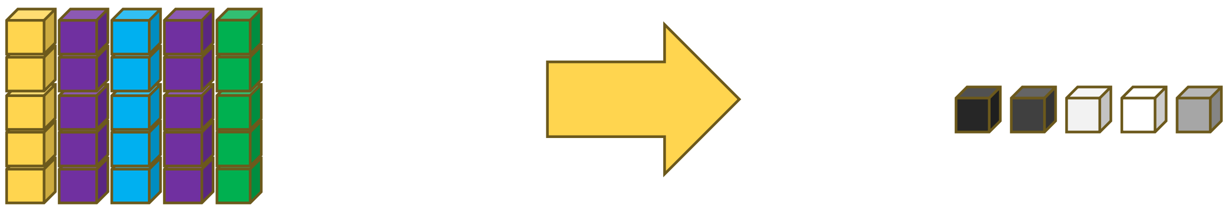
UNIVERSITY OF WATERLOO

RNA project problem

- "[Y]ou will be predicting the reactivity of an RNA sequence to two chemical modifiers DMS and 2A3"
  - What are the two inputs?
  - What is the output?
  - What are the possible values for the inputs and outputs?
  - What kind of machine learning task is this?

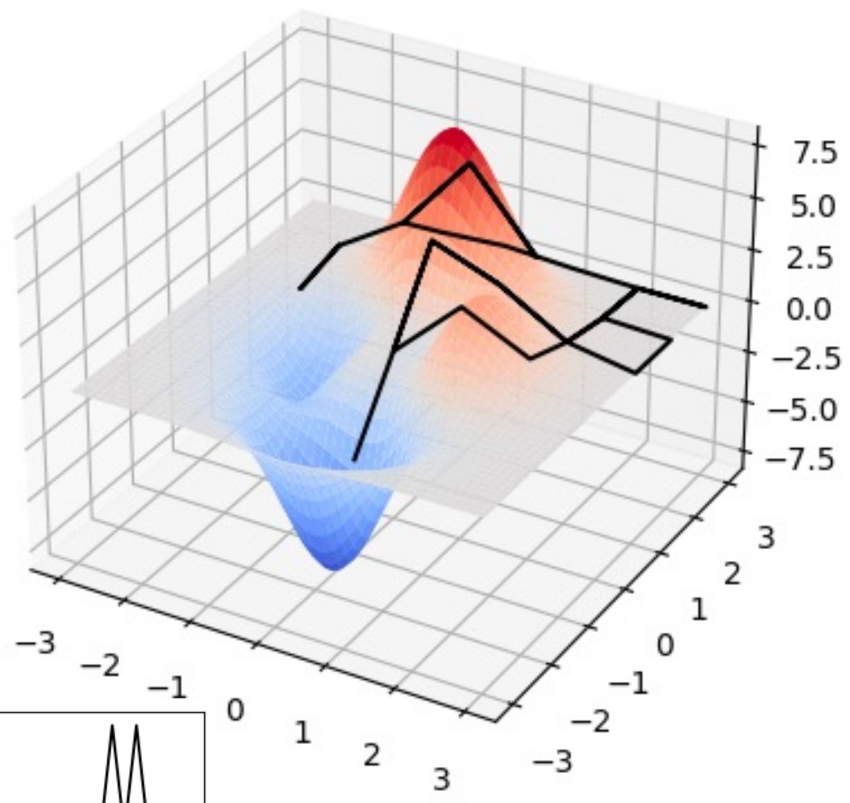
RNA project problem

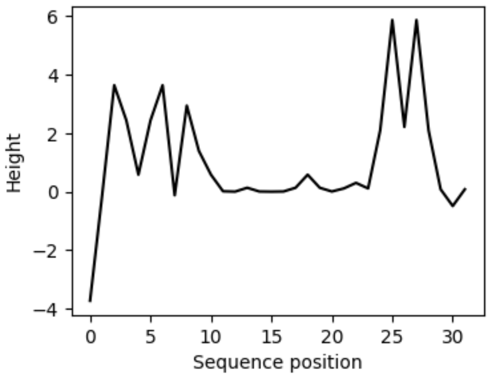
- [“A”, “U”, “C”, “U”, “G”, …] + [“DMS”] → [0.02, 0.31, 0.93, 1.03, 0.54, …]
- This is a sequence regression problem!
- Inputs of shape (B, N, D), outputs of shape (B, N)



Ant hill traversal problem

['D', 'U', 'U', 'R', 'R', 'L', 'L', 'D', 'R', 'R', 'U', 'R', 'U', 'L', 'U', 'R', 'L', 'D', 'D', 'U', 'U', 'L', 'L', 'L', 'D', 'R', 'R', 'L', 'L', 'L', 'D', 'U']





- The data was generated from a random sequence of moves of an ant traversing the XY-plane over this function landscape.
- The trajectory stops if $|x| > 3$ or $|y| > 3$, or after 32 steps
- The inputs are a discrete set of moves (Up, Down, Left, Right) and the targets are the continuous altitudes at that step
- What kind of problem is this?

Transformers for sequence regression

- Considerations for applying transformers to sequence regression:
  - How will we convert the inputs into vectors for the model?
  - How will we construct the output layers to predict a sequence of target values?
  - How will we deal with different length sequences?
  - How will we encode sequence position?
  - How will we construct our loss function for this problem?
  - Which transformer architecture (encoder/decoder) should we use?

Transformers for sequence regression

- How will we convert the inputs into vectors for the model?
  - Inputs should be encoded as integers, typically reserve 0 to represent padding
  - Transformers use embedding layers to convert sequences of integers into sequences of vectors
- How will we construct the output layers to predict a sequence of target values?
  - We should use a linear layer with no activation applied to each sequence vector
- How will we deal with different length sequences?
  - We often pad input sequences with padding elements (zeros) up to a maximum sequence length and mask inputs when computing attention
  - We want all inputs to have the same sequence length so we can make a regular tensor and process the entire batch in parallel

Transformers for sequence regression

- How will we encode sequence position?
  - Fixed sinusoidal encodings or learned position embeddings are common choices
- How will we construct our loss function for this problem?
  - Regression uses MSE loss, and we should only compute loss over non-padding sequence elements
- Which transformer architecture (encoder/decoder) should we use?
  - Encoder (bi-directional), decoder (auto-regressive), and encoder-decoder should all work
  - Encoder-only may have stronger representation capabilities if we don't need to do generation

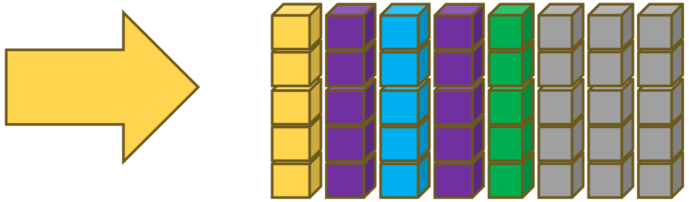
Typical transformer architecture

- Input layers
  - Input embeddings + sequence encoding
  - LayerNorm
- Processing layers
  - Transformer blocks
- Output layers
  - LayerNorm
  - Linear layer(s)
  - Output activation, depending on task

Torch embedding layer

- nn.Embedding
  - Takes arguments for number of unique inputs (size of embedding vocabulary), vector dimension (D), and padding index (usually 0)
  - Forward takes a sequence of type `torch.LongTensor` and shape (B, N) and outputs a sequence of vectors of type `torch.FloatTensor` and shape (B, N, D)

[1, 4, 2, 4, 3, 0, 0, 0]



Torch transformer

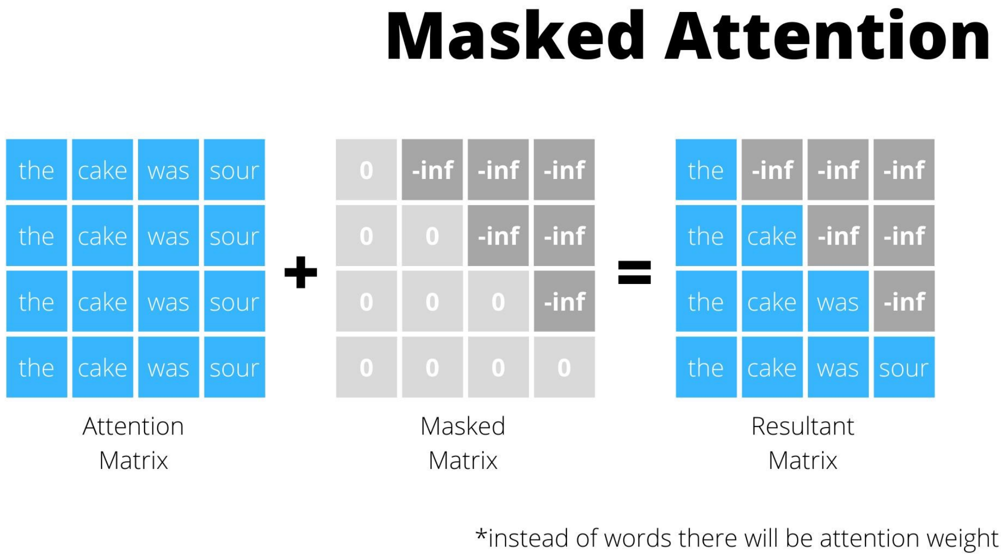
- `nn.TransformerEncoderLayer`, `nn.TransformerDecoderLayer`
  - Parameters related to repeated transformer block structure such as number of heads, model dimension, FFN dimension, activation function, etc.
- `nn.TransformerEncoder`, `nn.TransformerDecoder`, `nn.Transformer`
  - Encoder, decoder, and encoder-decoder architectures respectively
  - Repeats the layer module(s) above sequentially in the specified architecture
  - Takes arguments for the layer module(s) and the number of layers

Torch transformer

- `forward()` arguments depend on architecture
  - `src` represents the encoder input sequence of shape (B, N, D)
  - `tgt` represents the decoder input sequence of shape (B, N, D)
  - Outputs are shape (B, N, D)
  - `<src, tgt>_mask` is an additive mask of shape (N, N) with values of 0, or –inf applied to the attention values. It's typically autoregressive and applied to the tgt sequence for decoders
    - There is a convenience function `nn.Transformer.generate_square_subsequent_mask` to generate the autoregressive mask
  - `<src, tgt>_key_padding_mask` is a Boolean mask of shape (B, N) that is True when a sequence element is a padding element and False otherwise. It informs which sequence elements are valid to attend to and which are just padding.

Transformer masks

- Causal (auto-regressive) attention mask, shape (N, N)



- Padding mask, shape (B, N)

```
x = [[3, 4, 0, 0, 0]
     [1, 4, 2, 0, 0]
     [2, 2, 3, 1, 1]]
```

F	F	T	T	T
F	F	F	T	T
F	F	F	F	F

Transformer for ant hill transversal problem

- Input sequence of moves ("U", "D", "L", "R) is already encoded into integers and padded with 0's to maximum sequence length of 32
- Output sequence of float heights is already padded with 0's to maximum sequence length of 32
- We will build a transformer encoder and train it on this problem