CSCI-GA 2572 Deep Learning - Homework 4

Due: October 29 @ 11:59pm

Name: Stephen Spivack (ss7726)

1 Theory

1.1 Attention

(a) Given queries $Q \in \mathbb{R}^{d\times n}, K \in \mathbb{R}^{d\times m}, V \in \mathbb{R}^{t\times m}$, what is the output H of the standard dot-product attention? (You can use the $softargmax_\beta$ function directly. It is applied to the column of each matrix).

$H = softargmax_\beta(K^T Q)V$, where the dimensionality is $\mathbb{R}^{t\times n}$.

(b) Explain how the scale β influence the output of the attention? And what β is conveniently to use?

The β parameter influences the softness of the distribution. For example, if $\beta = 0$ then it becomes the argmax function, resulting in hard attention, focused on a single key. On the other hard, if $\beta = 1$ then out attention weights form a more spread out distribution. Scaling prevents the dot products from becoming too large, with a popular choice being $\beta = \frac{1}{\sqrt{d}}$.

(c) One advantage of the attention operation is that it is really easy to preserve a value vector v to the output h. Explain in what situation, the outputs preserves the value vectors. Also, what should the scale β be if we just want the attention operation to preserve value vectors. Which of the four types of attention we are referring to? How can this be done when using fully connected architectures?

Attention preserves value vectors when it focuses on just a few, or even a single, value(s) in the vector V. Simply set β to be a relatively high value, corresponding to hard attention, as discussed in the previous questions. Given a fully-connected architecture, we can achieve this by using all connections in the attention mechanism.

(d) On the other hand, the attention operation can also dilute different value vectors v to generate new output h. Explain in what situation the outputs is spread version of the value vectors. Also, what should the scale β be if we just want the attention operation to diffuse as much as possible. Which of the four types of attention we are referring to? How can this be done when using fully connected architectures?

Attention dilutes value vectors when it is spread across multiple values in the vector V. Simply set β to be a relatively low value, corresponding to soft attention, as discussed previously. Given a fully-connected architecture we can achieve this by using all connections in the attention mechanism.

(e) If we have a small perturbation to one of the k (you could assume the perturbation is a zero-mean Gaussian with small variance, so the new $\hat{k} = k + \epsilon$), how will the output of the H change?

A small perturbation to one of the keys will result in changes in output H that are proportional to the dot production of ϵ and the query vector.

(f) If we have a large perturbation that it elongates one key so the $\hat{k} = \alpha k$ for $\alpha > 1$, how will the output of the H change?

The change in output H is proportional to the dot product of the perturbation and the query vector, like in the previous case.

1.2 Multi-headed Attention

(a) Given queries $Q \in \mathbb{R}^{d \times n}$, $K \in \mathbb{R}^{d \times m}$ and $V \in \mathbb{R}^{t \times m}$, what is the output H of the standard multi-headed scaled dot-product attention? Assume we have h heads.

The output H is computed by concatenating and linearly transforming the outputs of each head:

$H = Concat(head_1, head_2, ..., head_h)W^O$

where the output of each head is:

$head_i = softmax(\frac{QW_i^Q(KW_i^K)^T}{\sqrt{d}})(VW_i^V)$

(b) Is there anything similar to multi-headed attention for convolutional networks? Explain why do you think they are similar.

Yes, we see parallels between multi-headed attention in transformers and the use of multiple filters in convolutional networks, in that they are both used for feature extraction for representation learning.

1.3 Self Attention

(a) Given an input $C \in \mathbb{R}^{e \times n}$, what is the queries Q, the keys K and the values V and the output H of the standard multi-headed scaled dot-product self-attention? Assume we have h heads. (You can name and define the weight matrices yourself).

$Q = C \cdot W_Q$, where $W_Q \in \mathbb{R}^{e \times (h \times d)}$ is a learnable matrix.

$K = C \cdot W_K$, where $W_K \in \mathbb{R}^{e \times (h \times d)}$ is a learnable matrix.

$V = C \cdot W_V$, where $W_V \in \mathbb{R}^{e \times (h \times d)}$ is a learnable matrix.

$H = \text{Concat}(head_1, head_2, ..., head_h)W^O$,

where each head is:

$\text{head}_i = \text{softmax}\left(\frac{QW_i^Q(KW_i^K)^T}{\sqrt{d}}\right)(VW_i^V)$, and $W^O \in \mathbb{R}^{(h \times d) \times e}$ is the output projection matrix.

(b) Explain when we need the positional encoding for self-attention and when we don't.

It's needed when the order of the elements in the sequence matters, such as in natural language. When the order does not matter it is not needed.

(c) Show us one situation that the self attention layer behaves like an identity layer or permutation layer.

The output might behave like the input when each element attends only to itself such that the weights become peaked.

(d) Show us one situation that the self attention layer behaves like an "running" linear layer (it applies the linear projection to each location). What is the proper name for a "running" linear layer.

This is a linear transformer, where the linear projections are applied independently to each location.

(e) Show us one situation that the self attention layer behaves like an convolution layer with a kernel larger than 1. You can assume we use positional encoding.

This can occur when positional encoding allows for the consideration of local neighborhoods in the sequence, which is similar to how a convolutional layer with a larger kernel size behaves.

1.4 Transformer

(a) Explain the primary differences between the Transformer architecture and previous sequence-to-sequence models (such as RNNs and LSTMs).

Transformers rely on attention instead of recurrent connections. Instead of encoding inputs sequentially like GRUs or LSTMs, transformers do so in parallel such that dependencies within elements of the sequence are captured with self-attention.

(b) Explain the concept of self-attention and its importance in the Transformer model.

Self-attention allows the model to weigh the importance of different positions in the input sequence simultaneously. This allows the model to pick up on global dependencies, unlike RNN models.

(c) Describe the multi-head attention mechanism and its benefits.

First, the multi-head attention mechanism linearly projects input queries, keys, and values into sets processed by separate attention heads in parallel, enabling the model to attend to different locations of the input sequence. Next, the concatenated outputs enhance the model's ability to jointly attend to separate information.

(d) Explain the feed-forward neural networks used in the model and their purpose.

They are used in the transformer as a sub-layer in each encoder and decoder layer, the purpose of which is to apply non-linear transformations to each position's representation independently.

(e) Describe the layer normalization technique and its use in the Transformer architecture.

It is used to normalize the outputs of each sub-layer, which mitigates against covariance shift, improving overall efficiency and speed of training.

1.5 Vision Transformer

(a) What is the key difference between the Vision Transformer (ViT) and traditional convolutional neural networks (CNNs) in terms of handling input images? Can you spot a convolution layer in the ViT architecture?

CNNs process the entire image through convolutional layers, whereas ViT splits the image into patches and processes them as sequences; each patch is linearly embedded and fed into a transformer.

(b) Explain the differences between the Vision Transformer and the Transformer introduced in the original paper.

The original transformer processed sequences of tokens (for sequential data) whereas ViT splits the image into patches as the input tokens (for grid-structured data).

(c) What is the role of positional embeddings in the Vision Transformer model, and how do they differ from positional encodings used in the original Transformer architecture?

In the original, positional encodings are added to input embeddings to correspond to order of the tokens; in ViT, these positional embeddings are added to correspond to the spatial information of the input.

(d) How does the Vision Transformer model generate the final classification output? Describe the process and components involved in this step.

ViT generates classification by using the token's representation as an aggregated representation of the entire image, which is then fed into the classification head for mapping to the classes.