Music VAE

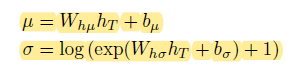
Encoder :

The encoder is a two layer bidirectional LSTM

Reason for using LSTM : RNN did not work well for longer sequence thats why LSTM were used. LSTM has cell state which passed the information. The ability to keep in memory or remove information is managed by the gates.

Reason for bidirectional LSTM : Using unidirectional will only keep in memory what it has previously seen. But in some cases the current value depends on both what is previously observed and what comes next. Thus for MusicVAE bidirectional lstm is used because interpolation between two music notes is done meaning the first music note and the last music note are both to be considered for finding the intermediate music

The output from the first layer lstm is given as input to the second layer. Finally the second layer produces two states one for each direction. These two are concatenated to form one H\_t. this is then given to two fully connected layers to produce the latent distribution parameters



Reason for concatenating two h states : the latent feature should describe in both the direction . If one h is used then only information from one side will be used which should not be the case fr music interpolation

The latent vector is passed through fully connected layer followed by tanh

Decoder :

Has two parts conductor and decoder :

Conductor : Two layer unidirectional lstm

Reason for using unidirectional conductor : The latent features z has combined information from both direction therefore it is sufficient to use unidirectional conductor

The input sequence is assumed to be segmented into U parts . The conductor produces embeddings for each subsequence

Each embedding is individually passed through **shared fully connect**ed layer followed by tanh (Why shared here alone ? )The output from here becomes the input to the decoder

Decoder : Two layer lstm

It produces sequence of distribution autoregressively for each of the subsequence with softmax.

At each decoder current conductor embedding is concatenated with previous output. The state is propagated in the decoder only within the subsequence to prevent the posterior collapse. Could say that the lstm in the decoder keeps in memory of the previously seen states only within the subsequence. For the next subsequence the input state is again from the conductor. By doing this the decoder is forced to make use of the latent variables. Experiments with passing the output back to the conductor didn't work Reason : conductor may depend only on the previous outputs seen and the latent variables wont be used. Thus by propagating only within a sequence, decoder for each sequence is initialised by the embeddings from the conductor which in turn depends on the latent distribution

Loss function mentioned is ELBO. I believe For each output from the decoder the loss is calculated and all the loss is summed up to give the final loss. Based on this the weights are updated. (does backprop happen all the way to encoder ? )

Code :

All the model configurations are mentioned in configs.py. This was the starting point for me to know how the model was set up.

Major file that tells the working are in lstm\_models.py. (**HierarchicalLstmDecoder** )