DFT comparison results.

chatGPT

2023 年 3 月 27 日

1 why $a_{2k} = \frac{2}{N}\sum_{n=0}^{\frac{N}{2}-1} x[n]e^{-j2k(2\pi/N)n}$?

Great question! To understand why $a_{2k} = \frac{2}{N}\sum_{n=0}^{\frac{N}{2}-1} x[n]e^{-j2k(2\pi/N)n}$, we need to look at the relationship between a_k and a_{2k}.

Using the formula for a_k, we can express a_{2k} as follows:

$a_{2k} = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-j2k(2\pi/N)n}$

Next, we can rearrange the exponential term by using the identity $e^{-j2k(2\pi/N)n} = e^{-jk(2\pi/N)(2n)}$, which gives us:

$a_{2k} = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-jk(2\pi/N)(2n)}$

Now, notice that the term inside the exponential is multiplied by 2n, which means that it takes on every even value from 0 to N-2 as n ranges from 0 to N-1. Therefore, we can rewrite the summation as:

$a_{2k} = \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1}(x[2n] + x[2n+1])e^{-jk(2\pi/N)(2n)}$

where we have split the original summation into two parts, one over the even-indexed samples and the other over the odd-indexed samples.

Next, we can use the fact that the signal is real-valued, which implies that $x[n] = x^*[N-n]$ for all n, where * denotes the complex conjugate. Substituting this in the equation for a_{2k}, we get:

$a_{2k} = \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1}(x[2n] + x^*[2(\frac{N}{2}-n)])e^{-jk(2\pi/N)(2n)}$

Now, we can separate the conjugate term from the original term in the summation, which gives us:

$a_{2k} = \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1} x[2n]e^{-jk(2\pi/N)(2n)} + \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1} x^*[2(\frac{N}{2}-n)]e^{-jk(2\pi/N)(2n)}$

Notice that the second summation is simply the complex conjugate of the first summation, but with the indices reversed. Therefore, we can write:

$a_{2k} = \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1} x[2n]e^{-jk(2\pi/N)(2n)} + \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1} x[2n]e^{jk(2\pi/N)(2(\frac{N}{2}-n))}$

Now, we can combine the two summations by grouping the first term of the second summation with the first summation, and the second term of the second summation with the second summation. This gives us:

$a_{2k} = \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1} x[2n](e^{-jk(2\pi/N)(2n)} + e^{jk(2\pi/N)(2(\frac{N}{2}-n))})$

Simplifying the exponentials inside the parentheses, we get:

$a_{2k} = \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1} x[2n](e^{-j2k(2\pi/N)n} + e^{-j2k(2\pi/N)(N-2n)})$

Now, we can use the fact that $e^{-j2k(2\pi/N)(N-2n)} = e^{-j2k(2\pi/N)(-2n)} = e^{-j2k(2\pi/N)(2n)}$, which gives us:

$a_{2k} = \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1} x[2n](e^{-j2k(2\pi/N)n} + e^{-j2k(2\pi/N)n})$

Simplifying further, we get:

$a_{2k} = \frac{2}{N}\sum_{n=0}^{\frac{N}{2}-1} x[2n]e^{-j2k(2\pi/N)n}$

Therefore, we have shown that $a_{2k} = \frac{2}{N}\sum_{n=0}^{\frac{N}{2}-1} x[2n]e^{-j2k(2\pi/N)n}$, which is the desired result.

2 easier explain

Sure! The equation $a_k = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-jk(2\pi/N)n}$ is a mathematical formula that calculates the k-th frequency component of a discrete-time signal $x[n]$. It does this by taking the sum of the products of the signal values $x[n]$ and complex exponential functions $e^{-jk(2\pi/N)n}$, where k and n are integers and N is the length of the signal.

The formula $a_{2k} = \frac{2}{N}\sum_{n=0}^{\frac{N}{2}-1} x[2n]e^{-j2k(2\pi/N)n}$ is a special case of the first formula, where we are calculating the 2k-th frequency component of the signal. This formula says that we only need to sum the products of every other sample of the signal $x[2n]$ with a complex exponential function $e^{-j2k(2\pi/N)n}$. This is because the complex exponential function has a periodicity of N, meaning that every N samples, it repeats. So, if we only use every other sample of the signal, we still get the same frequency component.

The factor of 2 in front of the summation is simply there to account for the fact that we are only using every other sample of the signal. We need to multiply the result by 2 to get the correct amplitude of the frequency component.

I hope that helps!

3 Seq2seq code

```
class Seq2Seq(FairseqEncoderDecoderModel):
    def __init__(self, args, encoder, decoder):
        super().__init__(encoder, decoder)
        self.args = args

    def forward(
        self,
        src_tokens,
        src_lengths,
        prev_output_tokens,
        return_all_hiddens: bool = True,
    ):
        """
        Run the forward pass for an encoder-decoder model.
        """
        encoder_out = self.encoder(
            src_tokens, src_lengths=src_lengths, return_all_hiddens=return_all_hiddens
        )
        logits, extra = self.decoder(
            prev_output_tokens,
            encoder_out=encoder_out,
            src_lengths=src_lengths,
            return_all_hiddens=return_all_hiddens,
        )
        return logits, extra
```