

Copilot

To update $r1$ based on the difference between the input and the product $U1 \cdot \text{tensordot}(r1)$, you can use a gradient-based method. This is a common approach in machine learning for updating weights (in this case, $r1$) based on the error or difference between the predicted and actual values.

Here's a general idea of how you can do it:

1. Compute the difference between the input and the product $U1 \cdot \text{tensordot}(r1)$. This gives you the error or difference matrix.
2. Multiply this difference matrix with the transpose of $U1$ (or an equivalent manipulation). This step is essentially computing the gradient of the error with respect to $r1$.
3. Update $r1$ by adding this result to it. The exact method of updating can vary. For example, in gradient descent, you would typically multiply the gradient by a learning rate before adding it to $r1$.

Here's how you can do it in Python using NumPy:

```
import numpy as np

# Compute the product U1.tensordot(r1)
product = np.tensordot(U1, r1, axes=([-1], [0]))

# Compute the difference between the input and the product
difference = input - product

# Compute the transpose of U1 along the last three axes
U1_transpose = np.transpose(U1, (3, 0, 1, 2))

# Compute the dot product of the difference and U1_transpose
gradient = np.tensordot(difference, U1_transpose, axes=([0, 1, 2], [1, 2, 3]))

# Update r1
r1 += gradient
```

Please replace `input`, `U1`, and `r1` with your actual variables. Also, note that this is a basic implementation. In practice, you might want to include a learning rate and other optimizations. Let me know if you have any other questions! 😊