Deep Learning Toolkit *(Einsum)*

Rowel Atienza, PhD
University of the Philippines
github.com/roatienza

2022

Outline

- Environment, Code Editor
- Python
- Tensor libraries – numpy, <u>**einsum**</u>, einops
- PyTorch, Timm
- Huggingface (HF), Gradio, Streamlit
- HF Accelerator, GitHub
- Machines – Colab, DeepNote, Kaggle, SageMaker
- Other tools

Einstein Summation or Einsum

https://numpy.org/doc/stable/reference/generated/numpy.einsum.html

https://rockt.github.io/2018/04/30/einsum

https://towardsdatascience.com/einsum-an-underestimated-function-99ca96e2942e

Motivation: A Lot of Tensor Operations in Deep Learning

Multilayer Perceptron (MLP) 1st Layer

Input: $\boldsymbol{x} \in \mathbb{R}^4$, Weights: $\boldsymbol{W}_0$, Biases: $\boldsymbol{b}_0$, Parameters: $\boldsymbol{\theta_0} = \{\boldsymbol{W}_0, \boldsymbol{b}_0\}$, Activation function: $\sigma(\cdot)$

$$\boldsymbol{f}_1(\boldsymbol{x};\ \boldsymbol{\theta}_0) = \sigma_1(\boldsymbol{W}_0\boldsymbol{x} + \boldsymbol{b}_0)$$

$$\boldsymbol{f}_1(\boldsymbol{x};\ \boldsymbol{\theta}_0) = \sigma_1\left(\begin{bmatrix} W_{00} & \cdots & W_{03} \\ \vdots & \ddots & \vdots \\ W_{k0} & \cdots & W_{k3} \end{bmatrix}\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_{00} \\ \vdots \\ b_{k0} \end{bmatrix}\right)$$

Numpy vs Einsum APIs

Operation	Numpy	Einsum
Matrix Multiply	`matmul`	`einsum`
Diagonal	`diag`	`einsum`
Sum along an axis	`sum`	`einsum`
Transpose	`transpose`	`einsum`
Dot, Inner and Outer Products	`dot, inner, outer`	`einsum`
Trace	`trace`	`einsum`

One Einsum API for all

```
from numpy import einsum
```

Or

```
from torch import einsum
```

Or

```
from tensorflow import einsum
```

Example Data

```
w = np.arange(6).reshape(2,3).astype(np.float32)
x = np.ones((3,1), dtype=np.float32)
w: [[0. 1. 2.]
    [3. 4. 5.]]
x: [[1.]
    [1.]
    [1.]]
```

In other words:

$$\boldsymbol{w} = \begin{bmatrix} 0. & 1. & 2. \\ 3. & 4. & 5. \end{bmatrix}$$

$$\boldsymbol{x} = \begin{bmatrix} 1. \\ 1. \\ 1. \end{bmatrix}$$

$\boldsymbol{wx}$

Numpy

```
y = np.matmul(w, x)
```

Einsum

```
y = einsum('ij,jk->ik',w,x)
```

Einstein Summation (Einsum)

$$(A \cdot B)_{i,k} = \sum_j A_{i,j} \cdot B_{j,k}$$

$$(A \cdot B)_{i,k} = A_{i,j} \cdot B_{j,k}$$

```
einsum('ij,jk->ik',A,B)
```

Example Data

```
w = np.arange(6).reshape(2,3).astype(np.float32)
x = np.ones((1,3), dtype=np.float32)
w: [[0. 1. 2.]
    [3. 4. 5.]]
x: [[1. 1. 1.]]
```

In other words:

$$\boldsymbol{w} = \begin{bmatrix} 0. & 1. & 2. \\ 3. & 4. & 5. \end{bmatrix}$$

$$\boldsymbol{x} = \begin{bmatrix} 1. & 1. & 1. \end{bmatrix}$$

$\boldsymbol{wx^T}$

Numpy

```
y = np.matmul(w, np.transpose(x))
```

Einsum

```
y = einsum('ij,kj->ik', w, x)
```

Example Data

```
w = np.arange(9).reshape(3,3).astype(np.float32)
```

```
w = [[0. 1. 2.]
     [3. 4. 5.]
     [6. 7. 8.]]
```

$$diagonal(\boldsymbol{w}): \boldsymbol{w} = \begin{bmatrix} 0. & 1. & 2. \\ 3. & 4. & 5. \\ 6. & 7. & 8. \end{bmatrix}$$

Numpy

```
d = np.diag(w)

d: [0. 4. 8.]
```

Einsum

```
d = einsum('ii->i', w)

d: [0. 4. 8.]
```

$$trace(\boldsymbol{w}): \boldsymbol{w} = \begin{bmatrix} \boxed{0.} & 1. & 2. \\ 3. & \boxed{4.} & 5. \\ 6. & 7. & \boxed{8.} \end{bmatrix} \quad \sum w_{i,i}$$

Numpy

```
t = np.trace(w)
```

```
t: 12.0
```

Einsum

```
t = einsum('ii->', w)
```

```
t: 12.0
```

Sum all elements column wise: : $\boldsymbol{w} = \begin{bmatrix} \boxed{\begin{matrix}0.\\3.\\6.\end{matrix}} & \boxed{\begin{matrix}1.\\4.\\7.\end{matrix}} & \boxed{\begin{matrix}2.\\5.\\8.\end{matrix}} \end{bmatrix}$ $\sum_i w_{i,j}$

Numpy

```
s = np.sum(w, axis=0)
```

```
s = [ 9., 12., 15.]
```

Einsum

```
s = einsum('ij->j', w)
```

```
s = [ 9., 12., 15.]
```

$$\boldsymbol{w}^T$$

Numpy

```
t = np.transpose(w)
```

Einsum

```
t = einsum("ij->ji", w)
```

Example Data

```
a = np.ones((3,), dtype=np.float32)
b = np.ones((3,), dtype=np.float32) * 2
a: [1. 1. 1.]
b: [2. 2. 2.]
```

Dot, inner, outer products

Numpy

```
d = np.dot(a,b)
i = np.inner(a,b)
o = np.outer(a,b)
```

Einsum

```
d = einsum("i,i->", a, b)
i = einsum("i,i->", a, b)
o = einsum("i,j->ij", a, b)
```

End

https://github.com/roatienza/Deep-Learning-Experiments/blob/master/versions/2022/tools/python/einsum_demo.ipynb