



קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

PyTorch Tensors

גלעד מרקמן





כללי Tensors

- ספריית PyTorch מבוססת על Tensors, שהינם מבני נתונים מיוחדים ללימוד מכונה.
- Tensor הינו למעשה מערך בעל מספר מימדים, הכולל יכולות נוספות המיוחדות ללמידת מכונה, ובין היתר:
 - א. חישובים מתמטיים מהירים על מערכים בעלי מספר מימדים.
 - ב. שימוש ב GPU המבצע חישובים במקביל ומקצר זמנים באופן משמעותי.
 - ג. חישוב נגזרות Gradients באופן אוטומטי באמצעות ספריית autograd.
 - ד. בניית רשתות נוירונים עמוקות באמצעות הספרייה pytorch.nn.
- בשיעור זה נלמד את יסודות השימוש ב Tensors.
[קישור למחברת colab](#)



יצירת Tensor

ניתן ליצור Tensor במספר אופנים:

- From List:

```
data = [[1,2], [3,4]]  
x = torch.Tensor(data)
```

- From numpy array:

```
np_array = np.array(data)  
x_np = torch.from_numpy(np_array)
```

- With Values:

```
shape = (2,3)  
rand_tensor = torch.rand(shape)  
rand_int_tensor = torch.randint(0,9,shape)  
ones_tensor = torch.ones(shape)  
zeros_tensor = torch.zeros(shape)
```

- Empty tensor:

- ערכי זבל

```
empty_tensor = torch.empty(2,2,3)
```

- Full tensor:

```
full_tensor = torch.full((2, 3), 0.5)
```

- יוצר מערך בגודל המבוקש עם הערך שניתן לו = 0.5

```
x:  
  tensor([[1., 2.],  
         [3., 4.]])  
x_np:  
  tensor([[1, 2],  
         [3, 4]], dtype=torch.int32)  
Random Tensor:  
  tensor([[0.0421, 0.1818, 0.2277],  
         [0.1043, 0.6380, 0.7588]])  
Random int Tensor:  
  tensor([[0, 4, 1],  
         [4, 3, 7]])  
Ones Tensor:  
  tensor([[1., 1., 1.],  
         [1., 1., 1.]])  
Zeros Tensor:  
  tensor([[0., 0., 0.],  
         [0., 0., 0.]])
```



המאפיינים של Tensor

- המאפיינים של Tensor כוללים את המבנה שלו, סוג הנתונים, ומיקום האחסון שלו (האם CPU או GPU):

```
tensor = torch.rand(3,4)

print(tensor.shape)
print(tensor.dtype)
print(tensor.device)
```

```
torch.Size([3, 4])
torch.float32
cpu
```

קביעת מאפייני Tensor

- בעת יצירת Tensor ניתן לקבוע את המאפיינים שלו:

```
tensor_float32 = torch.tensor(data, dtype=torch.float32, device="cuda")
tensor_float16 = torch.rand(size=(3,3), dtype=torch.float16, device="cpu")
print("tensor_float32: \n",tensor_float32.dtype, tensor_float32.device,"\n")
print("tensor_float16: \n",tensor_float16.dtype, tensor_float16.device)
```

```
tensor_float32:
  torch.float32 cuda:0
```

```
tensor_float16:
  torch.float16 cpu
```





פעולות על Tensors

- ניתן להעביר Tensor מה CPU ל- GPU ולהפך.
- הפעולות מעתיקות את ה Tensor ואנו נשארים עם העתק של הנתונים.

```
if torch.cuda.is_available():  
    tensor_cuda = tensor_float16.to("cuda")  
    tensor_cpu = tensor_cuda.to("cpu")  
  
print("tensor_cuda: ", tensor_cuda.device)  
print("tensor_cpu: ", tensor_cpu.device)
```

```
tensor_cuda:  cuda:0  
tensor_cpu:  cpu
```



פעולות אינדקסים

- ניתן לגשת לאיבר ב Tensor באמצעות האינדקס שלו.
- ניתן לחתוך חלק מה Tensor בדרך הבאה:

```
tensor = torch.ones(4, 4)
tensor[3,3] = 5
print(f"second row: {tensor[1]}")
print(f"Third column: {tensor[:, 2]}")
print(f"Last column: {tensor[..., -1]}")
tensor[:,1] = 0 # zero second column
print(tensor)
print(tensor[[0, 1, 3]]) # print 1st, 2nd & 4th row
print(tensor[:,[0, 1, 3]]) # print 1st, 2nd & 4th column
```

```
tensor([[1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 5.]])
tensor([[1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 5.]])
tensor([[1., 0., 1.],
        [1., 0., 1.],
        [1., 0., 1.],
        [1., 0., 5.]])
```

פעולות מתמטיות



קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

- ניתן לבצע ביעילות מרובה פעולות מתמטיות על tensor:

```
x = torch.ones(2,3)
y = torch.full((2,3), 2)
z1 = x + y
print (z1)
z2 = x * y
print (z2)
z3 = y**2 + 5
print(z3)
```

```
tensor([[3., 3., 3.],
        [3., 3., 3.]])
tensor([[2., 2., 2.],
        [2., 2., 2.]])
tensor([[9, 9, 9],
        [9, 9, 9]])
```




פעולות המחזירות ערך בודד

- פעולות על Tensor המחזרות ערך בודד יחזירו Tensor עם איבר אחד. על מנת להפוך אותו למספר רגיל, יש להשתמש בפעולה `.item()`.

```
data = torch.tensor([[1,2,3],[4,5,6]], dtype=torch.float32)
sum = data.sum()
print("sum: ", sum)
mean = data.mean()
print("mean: ", mean.item())
```

```
sum: tensor(21.)
mean: 3.5
```

- שימוש ב**פונקציה** `item()` על Tensor עם מספר איברים יגרום לשגיאת ריצה.



חישוב in-place

- חישובים המשנים את ה Tensor המקורי נקראים in-place. ב pytorch הסימון שלהם הוא עם קו תחתי בסיום הפעולה:

```
x = torch.tensor([[1,2,3],[4,5,6]])  
print (x, "\n")  
x.add_(5)  
x.t_()  
print (x)
```

```
tensor([[1, 2, 3],  
        [4, 5, 6]])
```

```
tensor([[ 6,  9],  
        [ 7, 10],  
        [ 8, 11]])
```



Transpose

- פעולה Transpose הופכת את השורות לעמודות.

```
x = torch.tensor([[1,2,3],[4,5,6]])  
print (x)  
print (x.T)
```

```
tensor([[1, 2, 3],  
        [4, 5, 6]])  
tensor([[1, 4],  
        [2, 5],  
        [3, 6]])
```

- שימו לב, לא מדובר בסיבוב של המטריצה, אלא השורה 0 הופכת לעמודה 0, השורה 1 לעמודה 1 וכך הלאה.



torch.reshape

- הפעולה reshape נועדה לשנות את המימדים של הטנסור, ובלבד שמספר האיברים בשני הטנסורים זהה.
- אם נציין מימד של -1 חישוב ערכו יעשה אוטומטית לפי הערכים של המימדים האחרים.

```
a tensor([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]) torch.Size([12])
a1 tensor([[ 0],
           [ 1],
           [ 2],
           [ 3],
           [ 4],
           [ 5],
           [ 6],
           [ 7],
           [ 8],
           [ 9],
           [10],
           [11]]) torch.Size([12, 1])
a2 tensor([[ 0, 1, 2],
           [ 3, 4, 5],
           [ 6, 7, 8],
           [ 9, 10, 11]]) torch.Size([4, 3])
a3 tensor([[[ 0, 1, 2],
            [ 3, 4, 5]],
           [[ 6, 7, 8],
            [ 9, 10, 11]]]) torch.Size([2, 2, 3])
a4 tensor([[[ 0, 1, 2],
            [ 3, 4, 5]],
           [[ 6, 7, 8],
            [ 9, 10, 11]]]) torch.Size([2, 2, 3])
```

```
a = torch.arange(12)
a1 = a.reshape(-1,1)
a2 = a.reshape(4,3)
a3 = torch.reshape(a1, (2,2,3))
a4 = a.reshape(2, -1, 3)
print("a", a, a.shape)
print("a1", a1, a1.shape)
print("a2", a2, a2.shape)
print("a3", a3, a3.shape)
print("a4", a4, a4.shape)
```

view



קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

- הפעולה view זהה כמעט לחלוטין ל reshape. ההבדלים האם:
 - הפעולה מחזירה את אותם נתונים ולא מעתיקה אותם.
 - ניתן גם לבצע שינוי של סוג הנתונים.

```
v = torch.arange(12)
v1 = v.view(-1,1)
v2 = v.view(4,3)
v3 = v.view(torch.float32)
v4 = v.view(-1, 1, 3)
print ("v", v.shape, v.dtype)
print ("v1", v1.shape)
print ("v2", v2.shape)
print ("v3", v3.shape, v3.dtype)
print ("v4", v4.shape)
```

```
v torch.Size([12]) torch.int64
v1 torch.Size([12, 1])
v2 torch.Size([4, 3])
v3 torch.Size([24]) torch.float32
v4 torch.Size([4, 1, 3])
```

torch.permute

• הפעולה torch.permute מחליפה את סדר המימדים בטנסור. לדוגמה:

```
c = torch.randn(1,3,2)
c1 = torch.permute(c, (2, 0, 1))
print(c.size())
print(c1.size())
print(c)
print(c1)
```

```
torch.Size([1, 3, 2])
torch.Size([2, 1, 3])
tensor([[[[-0.2126, -0.3432],
          [-0.5466, -0.0683],
          [-0.8641,  2.2388]]]])
tensor([[[[-0.2126, -0.5466, -0.8641]],
          [[-0.3432, -0.0683,  2.2388]]]])
```



torch.cat



קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

- הפעולה `cat` (מהמילה `Concatenates` – שירשור) מחברת שני טנסורים. כך ניתן להוסיף עמודות או שורות של נתונים לטנסור.

```
x1 = torch.ones(1,4)
x2 = torch.full((1,4) , 2)
print (x1, x2)
x3 = torch.cat((x1,x2),0)
x4 = torch.cat((x1,x2),1)
print (x3)
print (x4)
```

```
tensor([[1., 1., 1., 1.]]) tensor([[2, 2, 2, 2]])
tensor([[1., 1., 1., 1.],
        [2., 2., 2., 2.]])
tensor([[1., 1., 1., 1., 2., 2., 2., 2.]])
```



המרה של Tensor ל- numpy

- הייצוג בזכרון של Tensor ו- ndarray מאוד דומים, ולכן המרה של סוג אחד לשני אינה משנה את הזכרון של הנתונים.
- במקרה של המרה כזו הנתונים משותפים לשני המצביעים, ושינוי של אחד יביא לשינוי של השני.
- כתוצאה מכך, ההמרה מאוד מהירה.

```
t = torch.full((5,), 2.2)
print(t)

tensor([2.2000, 2.2000, 2.2000, 2.2000, 2.2000])

n = t.numpy()
t.mul_(5)
print (n)

[11. 11. 11. 11. 11.]
```

```
t1 = torch.from_numpy(n)
print (t1)

tensor([11., 11., 11., 11., 11.])
```

סיכום



קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

- למדנו כיצד לעבוד עם Tensors

- בשיעור הבא נלמד את ספריית autograd שבאמצעותה ניתן לעשות שימוש במאפיין החשוב ביותר של Tensors, והוא היכולת לחשב נגזרות.



קריית החינוך
פארק המדע
בית לערכים
למציאות ולחדשנות