ANN Class and Softmax

גלעד מרקמן

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

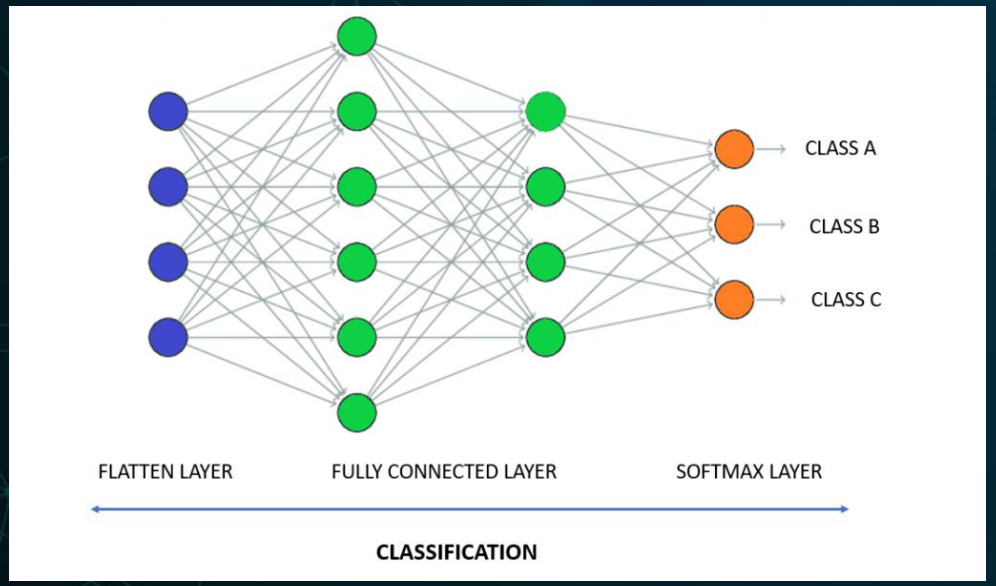
מטרת השיעור

- יצירת מחלקה מותאמת של רשת נוירונים במקום שימוש ב nn.Sequential.
- פתרון בעיית סיווג באמצעות פונקציית אקטיבציה softmax.
  - עד כה המודל נתן לנו תשובה לינארית (מספרית) או לוגית (כן / לא).
  - בשיעור זה נלמד כיצד ניתן לבנות מודל שיסווג לנו את התשובה למספר אפשרויות בדידות.
- נדגים את הנושא באמצעות בסיס הנתונים MNIST. הפעם נבנה מודל שיסווג את כל התמונות. כלומר, המודל יסווג כל תמונה לאיזה ספרה היא שייכת.
- במקרה זה המודל כולל 10 סיווגים (הספרות 0 – 9).

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

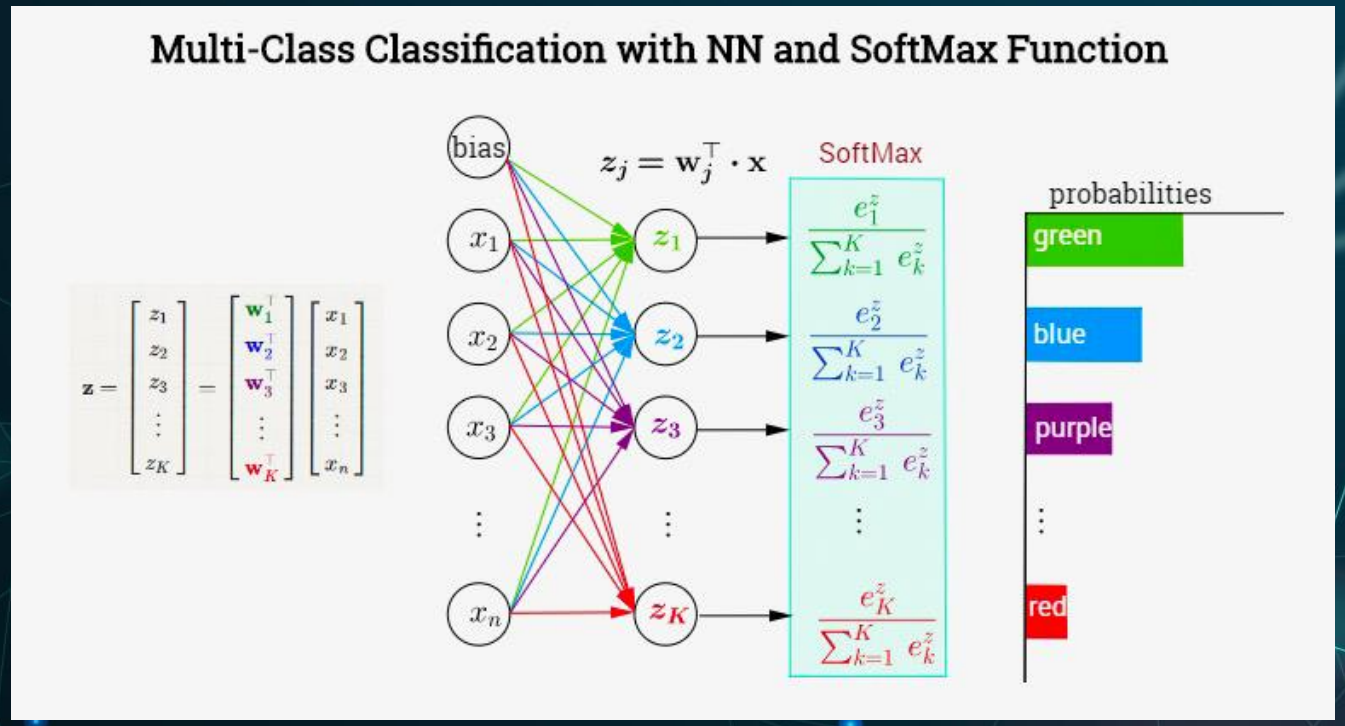
Softmax

- אנחנו מבקשים לבנות מודל שהתשובה שלו תיתן לנו את ההסתברות (הסיכוי) שהקלט שייך לקבוצה מסויימת.
- למשל, במקרה זה, אם הקלט שייך לקבוצה A אנחנו נרצה שהערך המספרי שיתקבל בנוירון A יהיה גבוה יותר מהערכים בקבוצות האחרות.



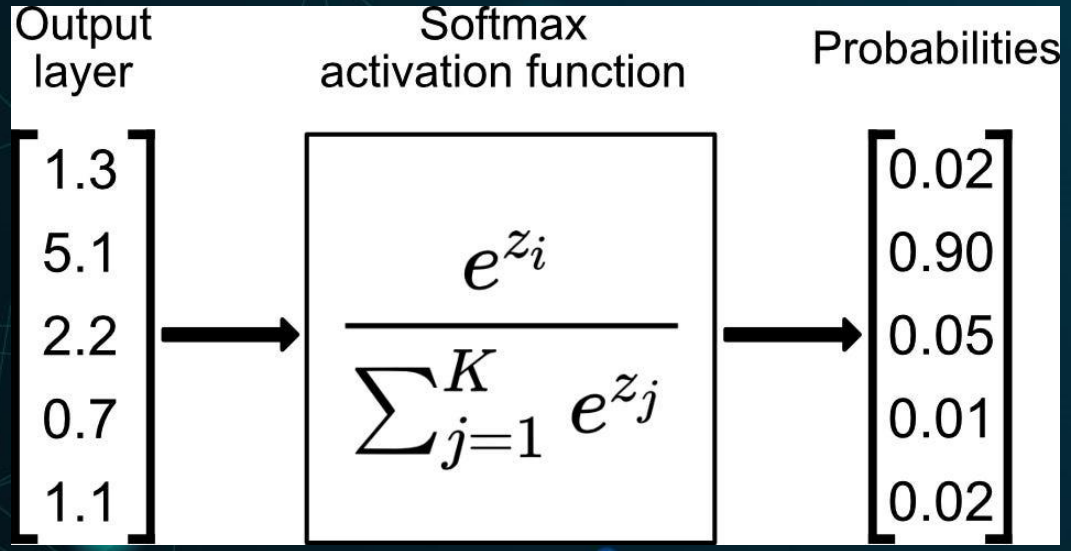
רשת נוירונים לסיווג

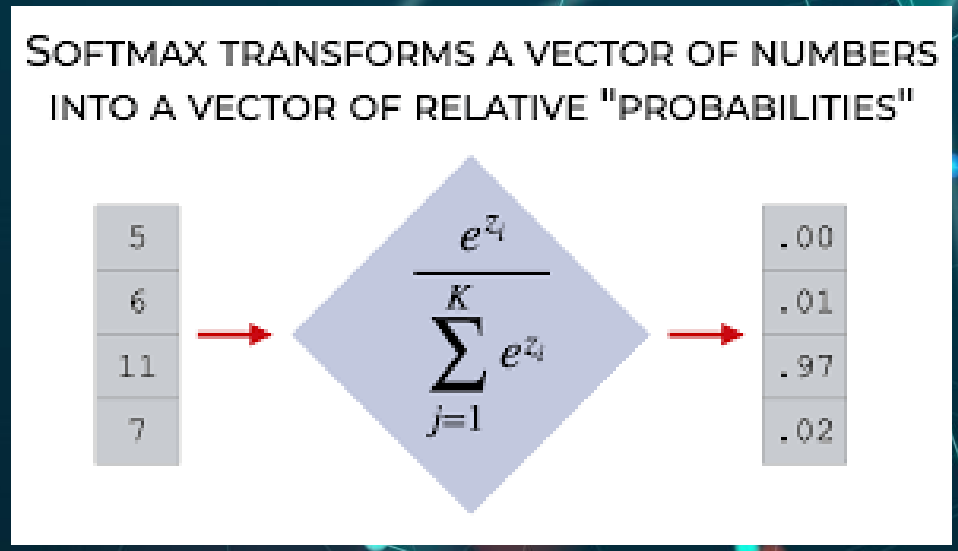
- ברשת הנוירונים לסיווג לקבוצות הפלט הוא כמספר הקבוצות האפשריות.
- כל ערך מתאר את הסיכוי שהדגימה שייכת לקבוצה זו.
- האינדקס של הערך הגבוה ביותר מתאר את הקבוצה לה שייכת הדגימה.



softmax

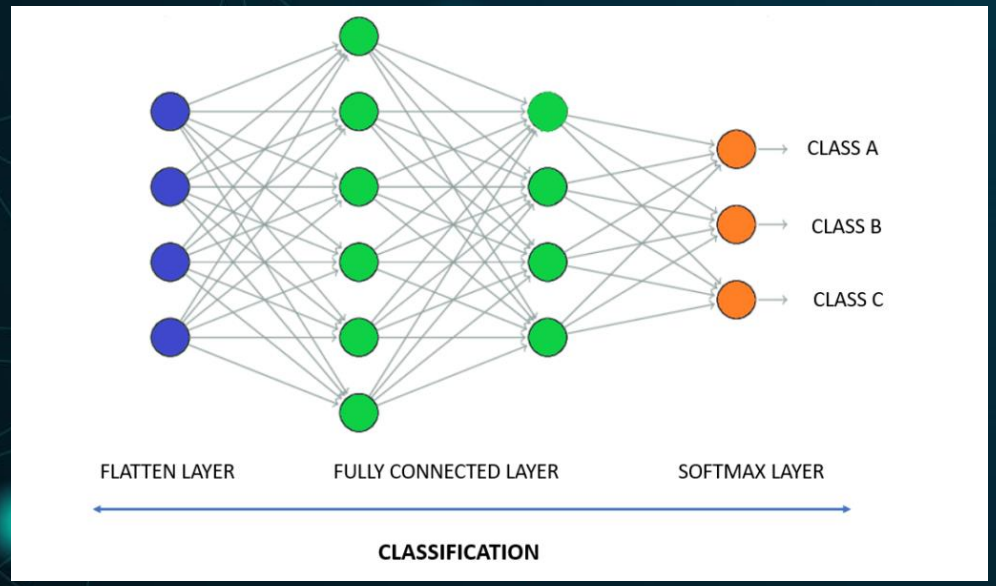
- פונקציית האקטיבציה softmax מבצעת חישוב חלוקה לקטגוריות הסתברותיות:
  - סכום הערכים המתקבלים הוא תמיד 1.
  - כל ערך מסמן את ההסתברות כי הקלט שייך לקבוצה הרלוונטית. אם למשל הערך הוא 0.90 המודל סבור כי ב- 90% הקלט שייך לקבוצה זו.





הפלט של רשת נוירונים לסיווג

- הפלט של רשת הנוירונים במקרה של פונקציית האקטיבציה softmax יכלול מטריצה כדלקמן:
  - כל שורה היא דגימה אחת.
  - כל עמודה היא קבוצה אחת. העמודה עם הערך הגבוה ביותר היא הקבוצה לה הדגימה שייכת בהסתברות הגבוהה ביותר.



vbnet

Sample	Class 1	Class 2	Class 3
1	0.25	0.15	0.60
2	0.10	0.50	0.40
3	0.35	0.25	0.40
4	0.20	0.30	0.50
5	0.40	0.20	0.40

one hot encoding

- one hot encoding – דרך לסיווג לקבוצות.
- הפלט כולל מספרים כמספר הקבוצות, כאשר כל מספר הוא 0 או 1.
- התשובה תכלול תמיד אפסים ו- 1 במקום המסמן את הקבוצה.
- בדוגמה למטה הסימון של הקבוצות הן:
  - הצבע האדום הוא [1,0,0]
  - הצבע הצהוב הוא [0,1,0]

Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1
0	1	0

one hot encoding & torch.max

- הפונקציה torch.max(tensor) מחזירה לנו את הערך המקסימלי בטנסור.
- הפונקציה torch.max(tensor, dim=1) המקבלת מטריצה תחזיר לנו שני ערכים:
  - (א) הערך המירבי בכל שורה. (ב) האינדקס של הערך המירבי בכל שורה.
- הפלט של רשת הנוירונים במקרה של softmax כולל לכל דגימה שורה של ערכים, שהאינדקס של הערך הגבוה ביותר הוא התשובה המצופה.
-

Sample	Class 0	Class 1	Class 2
1	0.25	0.15	0.60
2	0.10	0.50	0.40
3	0.35	0.25	0.40
4	0.20	0.30	0.50
5	0.40	0.20	0.40

```
torch.return_types.max(
  values=tensor([0.60, 0.50, 0.40, 0.50, 0.40]),
  indices=tensor([2, 1, 2, 2, 0])
)
```

```scss
(tensor([0.60, 0.50, 0.40, 0.50, 0.40]), tensor([2, 1, 2, 2, 0]))
```

פונקציית המחיר Categorial Cross Entropy

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

- פונקציית המחיר nn.CrossEntroptLoss(), כוללת בתוכה את פונקציית האקטיבציה softmax, ופועלת כך:
  - הפונקציה מקבלת מטריצה כמו בשקופית הקודמת.
  - בנוסף הפונקציה מקבלת מערך (שורה) של התשובות הנכונות. תשובה אחת לכל דגימה - target.
- הפונקציה מחשבת את הטעות.
- כאשר, במקרה זה התוצאות של דגימות 1,2,4,5 נכונות. בדגימה 3 יש טעות.

Sample	Class 0	Class 1	Class 2
1	0.25	0.15	0.60
2	0.10	0.50	0.40
3	0.35	0.25	0.40
4	0.20	0.30	0.50
5	0.40	0.20	0.40

```
tensor([2, 1, 0, 2, 0])
```

דוגמה MNIST

- נשנה את הקוד שעשינו בשיעור הקודם אשר נתן לנו תשובה בינארית (כן / לא) האם תמונה היא שייכת לקטגוריה מסויימת אם לאו. הפען המודל יתן לנו את הקטגוריה אליה שייכת התמונה (במקרה שלנו הספרה).

- נתאר את השינויים בקוד בלבד.

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

Class ANN

- נבנה מחלה היורשת מ- nn.Model.
- נגדיר כמאפיינים את השכבות של הרשת.
- נוסיף פעולה forward המבצעת את החלול קדימה. חובה להגדיר פעולה זו.
- לבסוף נגדיר אובייקט Model בהתאם למחלקה. הפקודה Model(X) תפעיל את forward אוטומטית, ואין לנו צורך לקרוא לפעולה Model.Forward(X).

```
Model = nn.Sequential(
    nn.Linear(input_size,hidden_size,device=device),
    nn.ReLU(),
    nn.Linear(hidden_size, num_classes, device=device),
)
```

```
class ANN_Model(nn.Module):
  def __init__(self):
    super().__init__()
    self.linear1 = nn.Linear(input_size,hidden_size,device=device)
    self.linear2 = nn.Linear(hidden_size, num_classes, device=device)

  def forward(self, x):
    x = self.linear1(x)
    x = F.relu(x)
    x = self.linear2(x)
    return x

Model = ANN_Model().to(device)
```

Loss & Optimizer

- נשים לב כי הפונקציה nn.CrossEntropyLoss כוללת כבר את פוקנציית האקטיבציה softmax. על כן, אין צורך להוסיף במודל שלנו את פונקציית האקטיבציה במפורש.

```
class ANN_Model(nn.Module):
  def __init__(self):
    super().__init__()
    self.linear1 = nn.Linear(input_size,hidden_size,device=device)
    self.linear2 = nn.Linear(hidden_size, num_classes, device=device)

  def forward(self, x):
    x = self.linear1(x)
    x = F.relu(x)
    x = self.linear2(x)
    return x

Model = ANN_Model().to(device)
```

```
# Loss = nn.MSELoss()
# Loss = nn.BCELoss()
Loss = nn.CrossEntropyLoss () # applies nn.LogSoftmax + nn.NLLLoss, No softmax in last layer

# init optimizer
# optim = torch.optim.SGD(Model.parameters(), lr=learning_rate,momentum=0.9)
optim = torch.optim.Adam(Model.parameters(), lr=learning_rate)
```

אימון המודל

```python
n_total_steps = len(train_loader)

for epoch in range(epochs):
  for i, (images, lables) in enumerate(train_loader):

    # origin shape: [50, 1, 28, 28]
    # resized: [50, 784]
    images = images.reshape(-1, 28*28).to(device)
    lables = lables.to(device)

    # forward
    Y_predict = Model(images)

    # backward
    loss = Loss(Y_predict, lables)
    loss.backward()

    # update wights
    optim.step()

    if i % 100 == 0:
      print(f"epoch= {epoch} i= {i+epoch * n_total_steps} loss={loss.item():.4f} ")

    # zero wights
    optim.zero_grad()
```

```
epoch= 0 i= 0 loss=2.3216
epoch= 0 i= 100 loss=0.4033
epoch= 0 i= 200 loss=0.1351
epoch= 0 i= 300 loss=0.6652
epoch= 0 i= 400 loss=0.0907
epoch= 0 i= 500 loss=0.1559
epoch= 0 i= 600 loss=0.1802
epoch= 0 i= 700 loss=0.0778
epoch= 0 i= 800 loss=0.2260
epoch= 0 i= 900 loss=0.1913
epoch= 0 i= 1000 loss=0.1954
epoch= 0 i= 1100 loss=0.3005
epoch= 1 i= 1200 loss=0.0997
epoch= 1 i= 1300 loss=0.2428
epoch= 1 i= 1400 loss=0.0093
epoch= 1 i= 1500 loss=0.2789
epoch= 1 i= 1600 loss=0.1163
epoch= 1 i= 1700 loss=0.0021
epoch= 1 i= 1800 loss=0.1887
epoch= 1 i= 1900 loss=0.0072
epoch= 1 i= 2000 loss=0.5900
epoch= 1 i= 2100 loss=0.3409
epoch= 1 i= 2200 loss=0.1062
epoch= 1 i= 2300 loss=0.0678
```

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

בדיקת המודל

- הגענו לדיוק של 95.41% בזיהוי התמונות

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

```
with torch.no_grad():
    n_correct = 0
    n_samples = 0
    for images, lables in test_loader:
        images = images.reshape(-1, 28*28).to(device)
        lables = lables.to(device)
        __,y_predict = torch.max(Model(images),1)
        n_samples += lables.size(0)
        n_correct += (y_predict == lables).sum().item()

    acc = 100 * n_correct / n_samples
    print(f'Accuracy of the network on the {n_samples} test images: {acc} %')
```

```
Accuracy of the network on the 10000 test images: 95.41 %
```