אימון הרשת

בשלב זה נפנה למימוש הרשת ואימונה לפי אלגוריתם מורד הגרדיאנט המקרי, תוך שימוש במלוא יכולותיה של הספרייה PyTorch. ראשית נגדיר את המודל,

```
from torch import nn
model = nn.Sequential(
          nn.Linear(784,10),     #z
          nn.LogSoftmax(dim=1)   #log(y)
          )
print(model)
```

פלט:

```
Sequential(
  (0): Linear(in_features=784, out_features=10, bias=True)
  (1): LogSoftmax(dim=1)
)
```

ונשים לב שבמקום פונקציית ה-Softmax בפלט הרשת אנו מחשבים את הלוגריתם שלה. הסיבה לכך היא שבתהליך האימון נשתמש בפונקציית המחיר של האנטרופיה הצולבת, עבורה עלינו לחשב הלוגריתם של הסתברויות הסיווג למחלקות, $Y=\left(y_0,\dots,y_9\right)$. בחישוב הלוגריתם באופן אנליטי אנו מקבלים תוצאה מדויקת יותר, שכן:

$$\log\left(\mathrm{softmax}\left(z_0,z_1,\dots,z_k\right)\right)=\log\left(\frac{1}{\sum_{n=0}^{k}e^{z_n}}\begin{pmatrix}e^{z_0}\\e^{z_1}\\\vdots\\e^{z_k}\end{pmatrix}\right)=\begin{pmatrix}\log\left(e^{z_0}\right)-\log\left(\sum_{n=0}^{k}e^{z_n}\right)\\\log\left(e^{z_1}\right)-\log\left(\sum_{n=0}^{k}e^{z_n}\right)\\\vdots\\\log\left(e^{z_k}\right)-\log\left(\sum_{n=0}^{k}e^{z_n}\right)\end{pmatrix}=\begin{pmatrix}z_0-\log\left(\sum_{n=0}^{k}e^{z_n}\right)\\z_1-\log\left(\sum_{n=0}^{k}e^{z_n}\right)\\\vdots\\z_k-\log\left(\sum_{n=0}^{k}e^{z_n}\right)\end{pmatrix}.$$

מעבר לחסכון פעולת האקספוננט והפיכתה בעזרת הלוגריתם, בחישוב זה אנו מתחמקים מבעיות יציבות נומריות אשר עלולות להיווצר כאשר $e^{z_k}\approx 0$. מלבד זאת, בשינוי זה אין אנו מפסידים דבר, שכן כל אימת שנרצה את פלט הרשת Y, נוכל להפעיל את האקספוננט ולקבלו.

בשלב הבא נגדיר את פונקציית המחיר, אשר כזכור עבור כל דגימה בסט האימון היא $H=-\log\left(y_n\right)$, כאשר n היא המחלקה אליה מסווגת הדגימה. זאת נממש בעזרת הפונקציה `nn.NLLLoss`: פונקציה זו מצפה לקבל כקלט את לוגריתם הסתברויות הסיווג של המודל, ומחזירה את מינוס הלוגריתם של פונקציית הנראות. בהתאם, פירוש ראשי התיבות בשמה הוא Negative Log Likelihood Loss. הפעלתה מיד לאחר LogSoftmax מניבה את האנטרופיה הצולבת, בה אנו מעוניינים להשתמש.

כל שנשאר עתה הוא להגדיר אובייקט אופטימיזציה, המקבל כקלט את פרמטרי המודל, ולהשתמש במתודות המובנות שלו בכדי לעדכן את הפרמטרים.

```
CE_loss = nn.NLLLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.1)
```

בעזרת אובייקטים אלו, נוכל לכתוב את לולאת האימון בתמציתיות. שימו לב בקטע הקוד הבא שבכל איטרציה של לולאת האימון יבוצעו הפעולות הבאות לפי הסדר:

1. טעינת minibatch של תמונות פרטי לבוש.
2. איפוס הגרדיאנט של הפרמטרים (שכן כברירת המחדל הגרדיאנט נצבר).

3. הזנת התמונות לתוך הרשת וחישוב הסתברויות חזויות.
4. חישוב פונקציית המחיר על minibatch זה.
5. חישוב הגרדיאנט המקרי בעזרת המתודה `backward()`.
6. עדכון ערכי הפרמטרים לפי הנוסחה $(W,b)=(W,b)-0.1\nabla C$ בעזרת המתודה `step()`. נוסחת עדכון זו נקבעה כאשר הגדרנו את אובייקט האופטימיזציה.

```
def iterate_batch():
  imgs, labels = next(iter(train_dataloader))
  imgs = imgs.flatten(start_dim=1)
  optimizer.zero_grad()
  y_model=model(imgs)
  loss=CE_loss(y_model,labels)
  loss.backward()
  optimizer.step()
```

נשאר לנו רק למדוד את ביצועי האלגוריתם לאורך האימון. המדד הבסיסי ביותר שאותו נרצה למדוד הוא מידת דיוק הסיווג של המודל (accuracy) – **אחוז הדגימות אשר המודל מסווג נכונה**. ראשית, נמיר את וקטור ההסתברויות המתקבל מהמודל לסיווג חד משמעי: נניח שהמודל מסווג כל דגימה נתונה אל המחלקה בעלת ההסתברות הגבוהה ביותר בפלט שלה, $\arg\max Y$. בעתיד נאפשר למודל להימנע מסיווג, למשל כאשר הסתברויות הסיווג כולן נמוכות, וכן נמדוד את ביצועי המודל בעזרת מדדים מתקדמים יותר. לשם כך יש להוסיף את שורות הקוד הבאות ללולאת האימון.

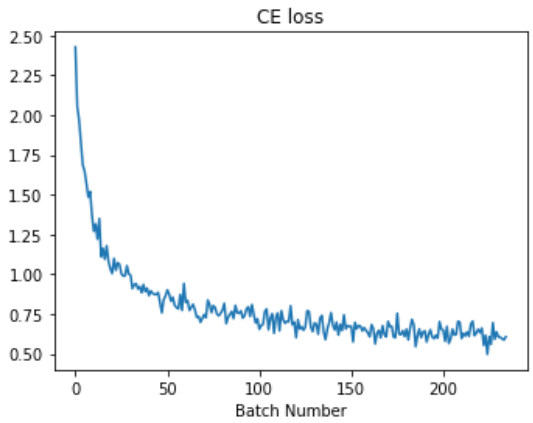
```
def iterate_batch():
 .
 .
 .
  predicted_labels = y_model.argmax(dim=1)
  acc = (predicted_labels == labels).sum()/len(labels)
  return loss.detach(), acc.detach()
```

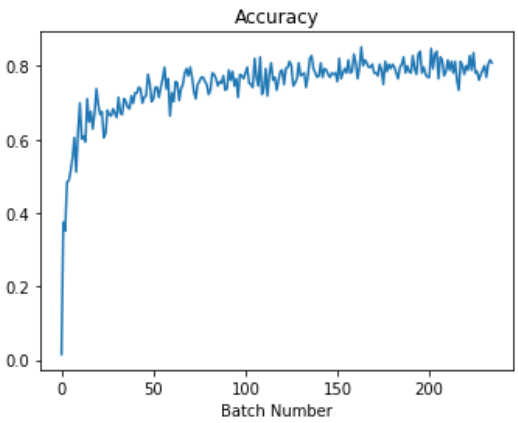
מכיוון שברצוננו להשתמש במדדי הביצועים לציור גרפים בהמשך, ומכיוון שאין טעם (ואף אין תמיכה בכך) לעקוב אחרי הפעולות אשר מבצעת הספרייה matplotlib בעזרת מערכת ה-Autograd, אנו מנתקים אותם ממערכת זו לפני החזרתם בעזרת המתודה `detach()`.

עתה נחלק את הנתונים ל-batches בגודל 256 דגימות ונבצע אימון לאורך epoch יחיד, מעבר יחיד על כל אוסף הנתונים.

```
train_dataloader = DataLoader(
    train_data_transformed, batch_size=256, shuffle=True)
batches=len(train_dataloader)
batch_loss=torch.zeros(batches)
batch_acc=torch.zeros(batches)
for idx in range(batches):
    batch_loss[idx], batch_acc[idx] = iterate_batch()
```

באיור הבא אנו מציירים את פונקציית ההפסד ומידת הדיוק, ובו ניתן לראות שהרשת אכן למדה לסווג את הנתונים, עד כדי דיוק של 0.8. יש לשים לב כמו כן לתנודות הרנדומליות במדדים, שכן בכל איטרציה הם מחושבים על פני batch יחיד, אשר אינו מייצג בדיוק מושלם את אוסף הנתונים כולו.





בשלב הבא יש לחזור על פעולה זו מספר פעמים, ולבחון את ביצועי המודל על פני נתונים שלא השתמשנו בהם לאימון. בכדי לחסוך בפעולות ולקצר את זמן הריצה, בדיקה זו יש לבצע בסוף כל epoch בלבד.

שאלות לתרגול

1. הסבירו מהי הבעיה הנומרית אשר עלולה להיווצר בחישוב האנטרופיה הצולבת כאשר $e^{z_k} \approx 0$.
2. החליפו את פונקציית המחיר של המודל למידת הדיוק (Accuracy), ונסו לאמן את הרשת מחדש. היכן נכשל התהליך? האם תוכלו להסביר באופן תיאורטי למה מידת הדיוק לא מתאימה כפונקציית מחיר לאלגוריתם מבוסס גרדיאנט?
3. כתבו את לולאת האימון המלאה, לאורך epochs, וכן הוסיפו לה את בדיקת ביצועי המודל בסוף כל epoch על פני כל סט הבדיקה.
   תזכורת: את סט הבדיקה ניתן לטעון בדרך דומה לסט האימון, על ידי שינוי הפרמטר `train` של פונקציית טעינת הנתונים ל-`False`.
4. חזרו על האימון עם minibatches בגדלים: 1,4,16,512, והשוו את התוצאות. באיזה גודל batch תעדיפו להשתמש?