

Multilayer Perceptron

הפרויקט עוסק בזיהוי מגדר לפי קטעי קול שהוקלטו והומרו למאפיינים ווקליים. זיהוי המגדר נעשה ע"י Multilayer Perceptron שממומש בעזרת Tensorflow.

הפרויקט חולק לארבעה חלקים כאשר בין החלק השני לשלישי הגדרנו ובנינו את מודל רשת הנוירונים שלנו.

חלק ראשון(read data set):

החלק הראשון עוסק בקריאת ה data set של המודל (במודל שלנו הוא מוצג כקובץ csv) על ידי שימוש בספרייה pandas. בגלל שהפרמטר label במודל מוצג כערכים male, female אז הוחלפו הערכים שיהיו 0,1 בהתאמה ואז עורבבו השורות על מנת שה-dataset לא יהיה ממויין (בקובץ csv מופיעים קודם כל הדגימות של הגברים ורק לאחר מכן הדגימות של הנשים) ואז המידע הוצב במשתנה voice.

חלק שני(split data set):

החלק השני עוסק בלקיחת המידע שהושג בחלק הראשון וחילוקו ל train ו-test. החילוק נעשה על ידי הכנסת הנתונים שנמצאים במשתנה voice לתוך שני מערכים מהספרייה numpy. מערך ראשון data_y מכיל את כל הערכים שנמצאים בפרמטר label ומערך שני data_x מכיל את שאר הערכים שלא הוכנסו מקודם. אחרי החילוק של שני המערכים כל מערך חולק לשני מערכים ביחס של 30-70. ארבעת המערכים הוצבו לתוך המשתנים , train_data_x , train_data_y , test_data_x , test_data_y.

בין החלק השני לשלישי הוגדרו המשתנים של הספרייה tensorflow בצורה שהמודל יוכל לחשב את המודל של הרשת נוירונים בעזרת הכלים שהוגדרו ב-tensorflow. מייד לאחר מכן, אנו בנינו את המודל של הרשת נוירונים והגדרנו אותה. בבניית הרשת "שיחקנו" בכל פעם עם העומק של הרשת (כמות השכבות) ובכמות הנוירונים שבכל שכבה. יש לציין שהבחנו שכאשר אנו העמסנו כל שכבה בהרבה נוירונים (256 נוירונים) המודל לא הצליח ללמוד. לאחר בדיקות וניסויים הבנו שהמודל שלנו יהיה טוב אם יהיו לנו 4 hidden layers:

שכבה נסתרת ראשונה: 20 נוירונים

שכבה נסתרת שנייה: 40 נוירונים

שכבה נסתרת שלישית: 40 נוירונים

שכבה נסתרת רביעית: 4 נוירונים

וכמובן, שכבת הקלט מכילה נויירונים ככמות ה-features שלנו ושכבת הפלט מכילה נויירון אחד.

בכל ה-hidden layers השתמשנו בפונקציית אקטיבציה reLU ע"מ לפשט את חישוב הגרדיאנט עבור tensorflow ומכיוון שהיא מתאימה לבעיות קלאסיפיקציה כמו שיש לנו. כמו כן, דאגנו לבחור את פונקציית האקטיבציה עבור הנוירון שבשכבת הפלט להיות סיגמוייד.

בנינו את הרשת ע"י שימוש ב-tensorflow.layers.dense שמוסיפה שכבה חדשה לרשת שלנו (עם כמות הנוירונים שאנו מעוניינים ופונקציית אקטיבציה כפי שהחלטנו).

חלק שלישי(model training):

בחלק השלישי נלקחו הנתונים שנמצאים ב- train_data_x , train_data_y , ועל סמך הנתונים האלו המודל התחיל לחשב את פונקציה ה-logistic על מנת למצוא את המשקלים הטובים ביותר. בתוכנית נעשו 10000 חזרות על הנתונים עד שהמודל סיים את שלב האימון.

חלק רביעי(model testing):

בחלק הרביעי אחרי מציאת המשקלים בחלק השלישי יבדק עד כמה המודל מדויק על ידי שימוש במידע הנותר שלא נעשה בו שימוש לצורך למידה ב- test_data_x , test_data_y על ידי הרצה של דגימה והשוואה בין התוצאה של המודל לבין התוצאה האמיתית. במהלך ה-test אנו מחשבים את ה-Loss וה-Accuracy ובסופו אנו מציגים אותם.

מסקנות:

במהלך ה-train (2217 דוגמאות, עם 10000 חזרות) שנעשה על המודל דאגנו להדפיס את ה-Loss וה-Accuracy ע"מ שנוכל להבחין בתהליך הלמידה והתוצאות שהתקבלו הן:

start training the model

Acc: 50.34% Loss: 0.688 Step: 0

Acc: 78.98% Loss: 0.534 Step: 500

Acc: 86.42% Loss: 0.467 Step: 1000

Acc: 90.26% Loss: 0.436 Step: 1500

Acc: 91.88% Loss: 0.422 Step: 2000
Acc: 91.97% Loss: 0.414 Step: 2500
Acc: 92.87% Loss: 0.409 Step: 3000
Acc: 93.19% Loss: 0.405 Step: 3500
Acc: 93.69% Loss: 0.401 Step: 4000
Acc: 94.14% Loss: 0.396 Step: 4500
Acc: 95.04% Loss: 0.391 Step: 5000
Acc: 94.18% Loss: 0.396 Step: 5500
Acc: 95.04% Loss: 0.390 Step: 6000
Acc: 95.40% Loss: 0.387 Step: 6500
Acc: 95.62% Loss: 0.385 Step: 7000
Acc: 95.81% Loss: 0.384 Step: 7500
Acc: 95.53% Loss: 0.383 Step: 8000
Acc: 95.85% Loss: 0.382 Step: 8500
Acc: 95.99% Loss: 0.382 Step: 9000
Acc: 96.12% Loss: 0.383 Step: 9500
Acc: 96.08% Loss: 0.380 Step: 10000

finish training the model

ניתן להבחין שטרם תחילת הלמידה של המודל אחוז הדיוק קרוב ל-50%, שמשמעותו בבעיית קלאסיפיקציה בינארית – ניחוש. לאחר סיום הלמידה המודל הגיע לאחוז דיוק גבוה מאוד: 96.08%.

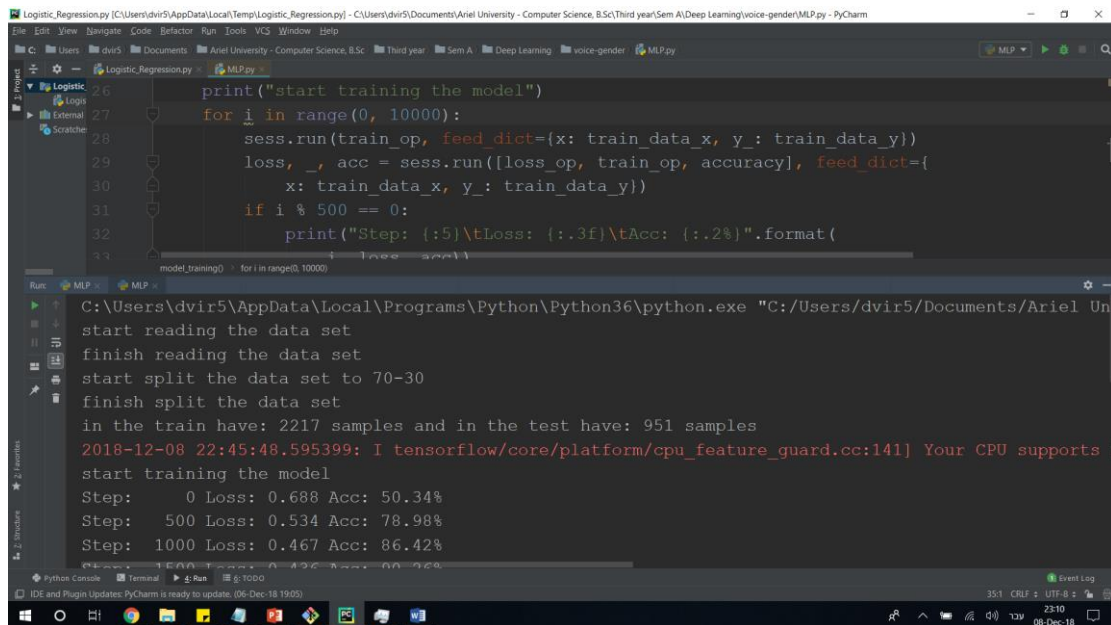
לאחר שביצענו את שלב ה-test (עם 951 דוגמאות) על המודל קיבלנו את התוצאות הבאות:

Accuracy: 93.27% Loss: 0.433

בהשוואה למודל ה-Logistic Regression שמימשנו במטלה הקודמת, קיים הפרש של למעלה מ-13% באחוז הדיוק לאחר ה-test. במודל Logistic Regression קיבלנו אחוז דיוק 80.7% ובמודל רשת הנוירונים שמימשנו במטלה זו קיבלנו אחוז דיוק 93.27%, שזה שיפור משמעותי.

יש לציין שערכנו מספר ניסויים בעיצוב במודל של הרשת נוירונים. במהלך הניסויים הבחנו שכאשר אנו מעמיסים יתר על המידה בכמות הנוירונים שבכל שכבה ביחד לכמות ה-features שלנו המודל לא הצליח ללמוד, וכאשר הוספנו להעמיק את הרשת זמן הלימוד התארך. בסוף הצלחנו למצוא את האיזון ולבנות מודל שיהיה מספיק חזק ללמידת הבעייה ולדעת לסווג קול למיגרד באחוז דיוק גבוה.

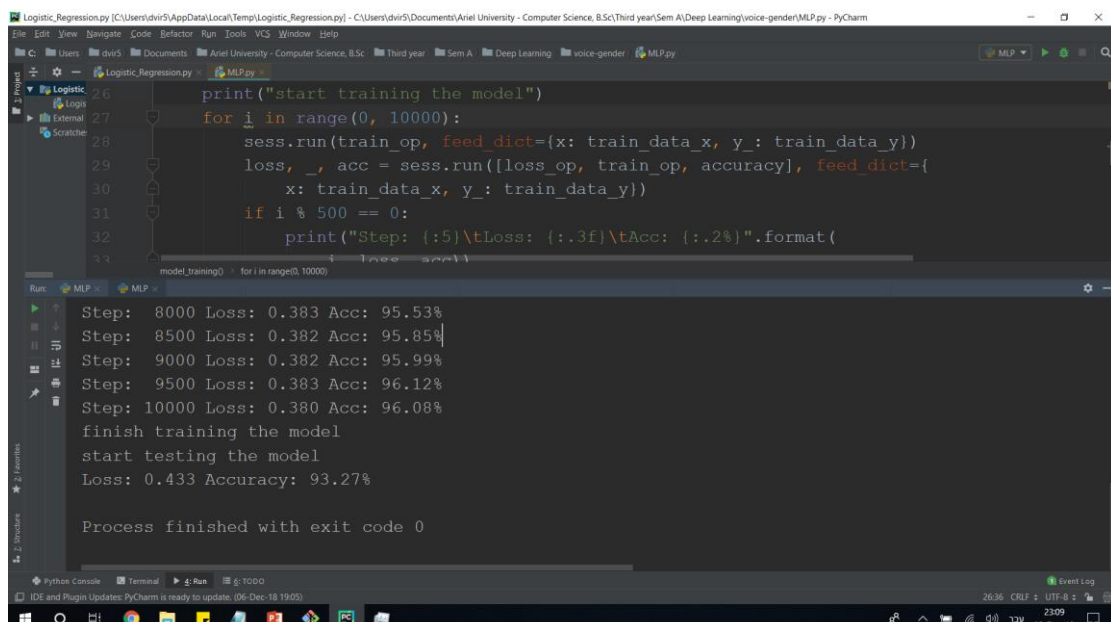
צילומי מסך:



```
Logistic_Regression.py [C:\Users\dvir5\AppData\Local\Temp\Logistic_Regression.py] - C:\Users\dvir5\Documents\Ariel University - Computer Science, B.Sc\Third year\Sem A\Deep Learning\voice-gender\MLP.py - PyCharm
```

```
26 print("start training the model")
27 for i in range(0, 10000):
28     sess.run(train_op, feed_dict={x: train_data_x, y: train_data_y})
29     loss, _, acc = sess.run([loss_op, train_op, accuracy], feed_dict={
30         x: train_data_x, y: train_data_y})
31     if i % 500 == 0:
32         print("Step: {:5}\tLoss: {:.3f}\tAcc: {:.2%}".format(
33             i, loss, acc))
```

```
Run: MLP x MLP x
C:\Users\dvir5\AppData\Local\Programs\Python\Python36\python.exe "C:/Users/dvir5/Documents/Ariel University - Computer Science, B.Sc/Third year/Sem A/Deep Learning/voice-gender/MLP.py"
start reading the data set
finish reading the data set
start split the data set to 70-30
finish split the data set
in the train have: 2217 samples and in the test have: 951 samples
2018-12-08 22:45:48.595399: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2 AVX AVX2 FMA
start training the model
Step:    0 Loss: 0.688 Acc: 50.34%
Step:   500 Loss: 0.534 Acc: 78.98%
Step:  1000 Loss: 0.467 Acc: 86.42%
```



```
Logistic_Regression.py [C:\Users\dvir5\AppData\Local\Temp\Logistic_Regression.py] - C:\Users\dvir5\Documents\Ariel University - Computer Science, B.Sc\Third year\Sem A\Deep Learning\voice-gender\MLP.py - PyCharm
```

```
26 print("start training the model")
27 for i in range(0, 10000):
28     sess.run(train_op, feed_dict={x: train_data_x, y: train_data_y})
29     loss, _, acc = sess.run([loss_op, train_op, accuracy], feed_dict={
30         x: train_data_x, y: train_data_y})
31     if i % 500 == 0:
32         print("Step: {:5}\tLoss: {:.3f}\tAcc: {:.2%}".format(
33             i, loss, acc))
```

```
Run: MLP x MLP x
Step:  8000 Loss: 0.383 Acc: 95.53%
Step:  8500 Loss: 0.382 Acc: 95.85%
Step:  9000 Loss: 0.382 Acc: 95.99%
Step:  9500 Loss: 0.383 Acc: 96.12%
Step: 10000 Loss: 0.380 Acc: 96.08%
finish training the model
start testing the model
Loss: 0.433 Accuracy: 93.27%

Process finished with exit code 0
```