**הגנה לפרוייקט – למידה חישובית**

Table of Contents

[איסוף נתונים 3](#_Toc440720545)

[בחינות ביצועים 3](#_Toc440720546)

[תחילת הדרך - מציאת הפרמטר C 3](#_Toc440720547)

[ביצועים עם מרכוז – בניית ה DB 4](#_Toc440720548)

[ביצועי Kernelים שונים 5](#_Toc440720549)

[10-Fold 5](#_Toc440720550)

[Neural Network 10](#_Toc440720551)

# איסוף נתונים

##### צורת יצירת Data set

יצירת ה Data Set לתהליך הלמידה מתבצעת בצורה פשוטה מאוד. מחלקת MyMnist.py מכילה בתוכה שתי פונקציות עקריות. הראשונה Read() אשר קוראת את המידע מ DB קיים, והשנייה CreateDb() אשר יוצרת את ה DB מתוך תיקייה ראשית המחולקת ל Labels. כך, כאשר נרצה להוסיף עוד תמונות לאימון או ניסוי נוכל לעשות זאת בפשטות.

המתודה CreateDB() קוראת את כל התמונות וכותבת אותם לשני קבצים בינארים – Images ו Labels.

##### PreProcessing

לאחר מספר ניסויי Hog על ה DB העברי, גיליתי שאין פיזור של החיזוי. המודל חזה רק אות אחת. לאחר חקירה מסויימת, הגעתי לשתי מסקנות עיקריות:

1. צבעים: שימוש בתמונות צבעוניות אינו תורם במקרה זה. על כל תמונה להיות בגווני אפור ואף יותר מזה ניתן לבצע זאת בשחור ולבן. לכן, השתמשתי בפונקציית Binary Threshold.
2. מירכוז: לעומת Mnist, ה DB שיצרתי אינו ממורכז ועל כן קשה לייצר ממנו מודל למידה אמין. חתכתי את החלקים בתמונה אשר צבועים בשחור ושיניתי את מידותיה ל 100x100.

# בחינות ביצועים

### תחילת הדרך - מציאת הפרמטר C

ביצעתי מספר ניסויים Linearים על האותיות ללא מירכוז התמונות. במהלכם עשיתי שינויים ב CValue ובכמות האימון וכמות הבחינה. אלו התוצאות:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Training | Test | Cvalue | %TP | %FP |
| Linear No CNT C0.1 | 700 | 99 | 0.1 | 67% | 40% |
| Linear No CNT C1 | 700 | 99 | 1 | 67% | 40% |
| Linear No CNT C10 | 700 | 99 | 10 | 67% | 40% |
| Linear No CNT C5 | 700 | 99 | 5 | 67% | 40% |
| Linear No CNT T200 C10 | 200 | 599 | 10 | 35% | 43% |
| Linear No Cnt T400 C0.0001 | 400 | 399 | 0.0001 | 59% | 29% |
| Linear No CNT T400 C0.1 | 400 | 399 | 0.1 | 59% | 29% |
| Linear No CNT T400 C1000 | 400 | 399 | 1000 | 59% | 29% |
| Linear No CNT T400 C5 | 400 | 399 | 5 | 59% | 29% |

לאחר חקירה נוספת, ניסתי למצוא Cvalueים אשר כן ישפיעו כל המודל. מצאתי כי יש להשתמש ב Cים קטנים מאוד בכדי לבנות מודל חדש. את הניסוי ביצעתי על 700 דוגמאות ובחנתי על 99 דוגמאות נוספות ואל מול כל ה 799:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | accuracy\_test | fp\_rate\_test | accuracy\_all | fp\_rate\_all | Training | Test | C |
| Linear T700 C1 | 67% | 40% | 96% | 4% | 700 | 99 | 1 |
| Linear T700 C0.0001 | 67% | 40% | 96% | 4% | 700 | 99 | 0.0001 |
| Linear T700 C0.000001 | 67% | 40% | 96% | 4% | 700 | 99 | 0.000001 |
| Linear T700 C0.0000001 | 65% | 15% | 82% | 9% | 700 | 99 | 0.0000001 |
| Linear T700 C0.00000001 | 37% | 15% | 51% | 14% | 700 | 99 | 0.00000001 |
| Linear T700 C0.000000001 | 2% | 3% | 3% | 3% | 700 | 99 | 0.000000001 |

ניכר מהניסיונות הללו כי C = 0.0000001 הוא הערך המתאים ביותר. עבור בדיקה על 400 דוגמאות:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | accuracy\_test | fp\_rate\_test | accuracy\_all | fp\_rate\_all | Training | Test | C |
| Linear T400 C0.0000001 | 58% | 18% | 75% | 12% | 700 | 99 | 0.0000001 |
| Linear T400 C0.000001 | 59% | 29% | 80% | 14% | 700 | 99 | 0.000001 |
| Linear T400 C0.00001 | 59% | 29% | 80% | 14% | 700 | 99 | 0.00001 |
| Linear T400 C0.0001 | 59% | 29% | 80% | 14% | 700 | 99 | 0.0001 |

וגם כאן נראה שזו בחירה לא רעה.

### ביצועים עם מרכוז – בניית ה DB

לאחר בחינת הביצועים הלינארים בחנתי את אותם ביצועים על התמונות הממורכזות. ציפיתי לראות שינוי חיובי. זו התוצאה:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | accuracy\_test | fp\_rate\_test | accuracy\_all | fp\_rate\_all | Training | Test | C |
| Linear T700 C1 | 86% | 20% | 98% | 2% | 700 | 99 | 1 |
| Linear T700 C0.0001 | 86% | 20% | 98% | 2% | 700 | 99 | 0.0001 |
| Linear T700 C0.000001 | 86% | 20% | 98% | 2% | 700 | 99 | 0.000001 |
| Linear T700 C0.0000001 | 86% | 20% | 98% | 2% | 700 | 99 | 0.0000001 |
| Linear T700 C0.00000001 | 83% | 23% | 96% | 4% | 700 | 99 | 0.00000001 |
| Linear T700 C0.000000001 | 74% | 21% | 60% | 23% | 700 | 99 | 0.000000001 |

הנתון הזה הקפיץ משמעותית את אחוזי ההצלחה. לאחר מכן, התחלתי לאמן עבור התמונות הממורכזות עם C = 0.0000001

### ביצועי Kernelים שונים

לאחר יצירת ה DB המתאים וקביעת פרמטר C, המשכתי בבדיקת SVM POLY. ביצעתי אימון על 500 דוגמאות ובחינה אל מול 299 ואל מול כלל האימון:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | accuracy\_test | fp\_rate\_test | accuracy\_all | fp\_rate\_all | Training | Test | G |
| Kernel2 G10000000000 | 6% | 60% | 7% | 63% | 500 | 299 | 10000000000 |
| Kernel2 G0.000001 | 27% | 27% | 36% | 21% | 500 | 299 | 0.000001 |
| Kernel2 G0.00001 | 50% | 26% | 57% | 16% | 500 | 299 | 0.00001 |
| Kernel G0.001 | 80% | 22% | 93% | 7% | 500 | 299 | 0.001 |
| Kernel2 G0.1 | 80% | 22% | 93% | 7% | 500 | 299 | 0.1 |
| Kernel2 G1 | 80% | 22% | 93% | 7% | 500 | 299 | 1 |
| Kernel2 G100 | 80% | 22% | 93% | 7% | 500 | 299 | 100 |
| Kernel2 G1000000 | 80% | 22% | 93% | 7% | 500 | 299 | 1000000 |

הבחנתי בירידה באחוזי ההצלחה. בחנתי שוב מול 700 דוגמאות והשוותי מול Svm Linear:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kernel2 T700 G1 | 85% | 25% | 98% | 2% | 700 | 99 | 1 |
| Linear T700 C1 | 86% | 20% | 98% | 2% | 700 | 99 | 1 |

ואכן הייתה ירידה. המשכתי בבדיקה מול דרגה גבוהה יותר:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | accuracy\_test | fp\_rate\_test | accuracy\_all | fp\_rate\_all | Training | Test | G |
| Kernel3 G0.000001 | 54% | 17% | 63% | 8% | 500 | 299 | 0.000001 |
| Kernel3 G0.001 | 77% | 18% | 91% | 6% | 500 | 299 | 0.001 |
| Kernel3 G1 | 77% | 18% | 91% | 6% | 500 | 299 | 1 |
| Kernel3 G1000 | 77% | 18% | 91% | 6% | 500 | 299 | 1000 |
| Kernel3 G100000 | 2% | 13% | 3% | 13% | 500 | 299 | 100000 |

מכאן הבנתי שככל שנעלה יותר במעלת הKernel, נרד באחוזי ההצלחה. הנחתי שה Kernel ישמש יותר כאשר יהיו Features מתאימים.

### 10-Fold

לאחר הבדיקות הללו, הגדלתי את ה Data Set משמעותית והתחלתי בבדיקה בעזרת K-Fold. כעת, אני בודק את יכולות האימון של המודל שלי מול כל חלקי ה Data Set. בנוסף, בגלל הגדילה המשמעותית של ה Data Set והכפלת הפעולות המתבצעות, זמן העיבוד היה גדול מהרגיל. על כן, החלטתי להקטין את גודל התמונות ל 28x28 (בדומה ל Mnist). להלן הביצועים:

#### Linear:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | accuracy\_test | fp\_rate\_test | C |
| Linear C1 | 80% | 17% | 1 |
| Linear C0.0001 | 81% | 20% | 0.0001 |
| Linear C1000000 | 79% | 21% | 1000000 |
| Linear C0.000001 | 77% | 23% | 0.000001 |
| Linear C1000 | 72% | 28% | 1000 |

#### Kernel:

|  |  |  |
| --- | --- | --- |
| Name | accuracy\_test | fp\_rate\_test |
| Kernel D2 C0.001 G1000 | 71% | 22% |
| Kernel D2 C0.001 G1 | 75% | 22% |
| Kernel D2 C0.001 G100000 | 71% | 22% |
| Kernel D2 C0.001 G0.001 | 72% | 26% |
| Kernel D2 C1 G100000 | 75% | 21% |
| Kernel D2 C1 G1 | 75% | 21% |
| Kernel D2 C1 G0.0001 | 74% | 22% |
| Kernel D2 C1 G0.01 | 73% | 24% |
| Kernel D2 C1 G1000 | 72% | 25% |
| Kernel D2 C10000 G0.01 | 80% | 20% |
| Kernel D2 C10000 G0.0001 | 73% | 23% |
| Kernel D2 C10000 G10000 | 74% | 25% |
| Kernel D2 C10000 G1 | 74% | 25% |

#### HOG

##### Linear

|  |  |  |
| --- | --- | --- |
| Name | accuracy\_test | fp\_rate\_test |
| HOG 14x14 d10 C1.0 | 82% | 17% |
| HOG 14x14 d10 C1e-06 | 84% | 17% |
| HOG 14x14 d10 C0.01 | 80% | 20% |
| HOG 14x14 d10 C100.0 | 79% | 22% |
| HOG 14x14 d10 C100000.0 | 75% | 22% |
| HOG 14x14 d36 C1e-06 | 84% | 14% |
| HOG 14x14 d36 C100000.0 | 84% | 16% |
| HOG 14x14 d36 C1000.0 | 81% | 17% |
| HOG 14x14 d36 C0.001 | 79% | 20% |
| HOG 14x14 d36 C1.0 | 79% | 22% |
| HOG 4x4 d10 C0.001 | 88% | 10% |
| HOG 4x4 d10 C100.0 | 87% | 11% |
| HOG 4x4 d10 C100000.0 | 85% | 12% |
| HOG 4x4 d10 C1e-06 | 87% | 13% |
| HOG 4x4 d10 C1.0 | 85% | 15% |
| HOG 4x4 d36 C1e-06 | 87% | 11% |
| HOG 4x4 d36 C100000.0 | 87% | 13% |
| HOG 4x4 d36 C0.001 | 85% | 13% |
| HOG 4x4 d36 C100.0 | 85% | 14% |
| HOG 4x4 d36 C1.0 | 86% | 14% |
| HOG 7x7 d10 C0.001 | 89% | 12% |
| HOG 7x7 d10 C1.0 | 86% | 14% |
| HOG 7x7 d10 C100.0 | 84% | 15% |
| HOG 7x7 d10 C100000.0 | 82% | 17% |
| HOG 7x7 d10 C1e-06 | 84% | 18% |
| HOG 7x7 d36 C100.0 | 89% | 10% |
| HOG 7x7 d36 C1e-06 | 89% | 11% |
| HOG 7x7 d36 C100000.0 | 86% | 11% |
| HOG 7x7 d36 C1.0 | 87% | 12% |
| HOG 7x7 d36 C0.001 | 85% | 14% |

##### Kernel

|  |  |  |
| --- | --- | --- |
| Name | accuracy\_test | fp\_rate\_test |
| HOG 7x7 d36 C1000.0 G1000000.0 | 89% | 10% |
| HOG 7x7 d36 C1.0 G1.0 | 88% | 10% |
| HOG 7x7 d36 C11.0 G1000000.0 | 90% | 11% |
| HOG 7x7 d36 C100000 G1000.0 | 89% | 11% |
| HOG 7x7 d36 C1.0 G1e-06 | 88% | 12% |

Confusion Matrix סופי עבור HOG 7x7 d36 C11.0 G1000000.0:



### Neural Network

בכדי לממש Deep Learning השתמשתי במודול Keras. המודול הזה משתמש בתשתית Theano בכדי לייצר מימוש מהיר של יצירת Neural Network. לאחר שהגדרתי את המודול, השתמשתי במבנה רשת קיים לפתרון ה Mnist (<https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py>). לאחר הרצה ראשונה, יצרתי משקולות מתאימות עבור ה Data Set שלי (בנוי בצורה שונה מה Data Set של Mnist). לאחר יצירת המשקולות הראשונה, השתמשתי במשקולות הללו על מנת להמשיך את הלמידה. הרצתי את פעולת יצירת הרשת שוב עד אשר ה Loss הפסיק להתכנס. אלו התוצאות על בדיקה של עוד כ -1800 נתונים:

|  |  |  |  |
| --- | --- | --- | --- |
| Test Number | accuracy\_test | Loss Start | Loss End |
| Test 1 | 62% | 3.319 | 0.482 |
| Test 2 | 63% | 0.482 | 0.056 |
| Test 3 | 64% | 0.056 | 0.007 |
| Test 4 | 64% | 0.007 | 0.001 |

Confusion Matrix:



### מהלך ההגנה

במהלך ההגנה נתקלתי בבעיה בזיהוי האותיות שניתנו. זה קרה בגלל קריאה לא נכונה של התמונות (ניסיתי לקרוא אותם לתוך מערך בגודל 28x28 למרות שנקראו כבר ככה). לאחר פתירת הבעיה הזו, אלו התוצאות:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data | Predicted | | Label | | TP |
| 101.jpg | 4 | 4 | א | 10 | 0 |
| 102.jpg | ז | 16 | ז | 16 | 1 |
| 103.jpg | ה | 14 | ה | 14 | 1 |
| 104.jpg | ן | 26 | ק | 33 | 0 |
| 105.jpg | ר | 34 | ר | 34 | 1 |
| 106.jpg | ה | 14 | ה | 14 | 1 |
| 107.jpg | ט | 18 | ט | 18 | 1 |
| 108.jpg | ר | 34 | ה | 14 | 0 |
| 109.jpg | י | 19 | ר | 34 | 0 |
| 110.jpg | צ | 31 | ד | 13 | 0 |
| 111.jpg | 0 | 37 |  | 37 | 1 |
| 112.jpg | ח | 17 | ב | 11 | 0 |
| 113.jpg | ח | 17 | ת | 36 | 0 |
| 114.jpg | 0 | 37 | מ | 23 | 0 |
| 115.jpg | 0 | 37 | ע | 28 | 0 |
| 116.jpg | 0 | 37 | א | 10 | 0 |
| 117.jpg | ר | 34 | ר | 34 | 1 |
| 118.jpg | ט | 18 | ט | 18 | 1 |
| 119.jpg | מ | 23 | מ | 23 | 1 |
| 120.jpg | כ | 20 | כ | 20 | 1 |
| 121.jpg | מ | 23 | מ | 23 | 1 |
| 122.jpg | ש | 35 | ש | 35 | 1 |
| 123.jpg | ה | 14 | ה | 14 | 1 |
| 124.jpg | ף | 30 | מ | 23 | 0 |
| 125.jpg | 0 | 37 |  | 37 | 1 |
| 126.jpg | ל | 22 | א | 10 | 0 |
| 127.jpg | א | 10 | ב | 11 | 0 |
| 128.jpg | ה | 14 | כ | 20 | 0 |
| 129.jpg | 0 | 37 | ק | 33 | 0 |
| 130.jpg | ה | 14 | ה | 14 | 1 |
| 131.jpg | א | 10 | א | 10 | 1 |
| 132.jpg | א | 10 | א | 10 | 1 |
| 133.jpg | א | 10 | ג | 12 | 0 |
| 134.jpg | ן | 26 | ע | 28 | 0 |
| 135.jpg | ב | 11 | ב | 11 | 1 |
| 136.jpg | ל | 22 | א | 10 | 0 |
| 137.jpg | ה | 14 | ה | 14 | 1 |
| 138.jpg | נ | 25 | כ | 20 | 0 |
| 139.jpg | ש | 35 | ש | 35 | 1 |
| 140.jpg | ב | 11 | ה | 14 | 0 |
| 141.jpg | ח | 17 | ת | 36 | 0 |
| 142.jpg | ת | 36 | ל | 22 | 0 |
| 143.jpg | ש | 35 | ש | 35 | 1 |
| 144.jpg | מ | 23 | מ | 23 | 1 |
| 145.jpg | ו | 15 | י | 19 | 0 |
| 146.jpg | נ | 25 | נ | 25 | 1 |
| 147.jpg | מ | 23 | ק | 33 | 0 |
| 148.jpg | ס | 27 | ס | 27 | 1 |
| 149.jpg | ג | 12 | ג | 12 | 1 |
| 150.jpg | ה | 14 | ה | 14 | 1 |
| 151.jpg | 2 | 2 | פ | 29 | 0 |
| 152.jpg | מ | 23 | מ | 23 | 1 |
| 153.jpg | צ | 31 | צ | 31 | 1 |
| 154.jpg | ש | 35 | ש | 35 | 1 |
| 155.jpg | כ | 20 | כ | 20 | 1 |

ה Accuracy שהתקבל הינו: 55%