למידת מכונה

Ragheb Ghazi

314892506

1) KNN אלגוריתם :

מימוש :מימשתי את class KNN שמורכב מ 3 פונקציות :

1- פונקציית אתחול בשם initialize שמקבלת כקלט את נתוני ה-x וה-y וגם את המספר k ומאתחלת אותם.
2- פונקציית אימון בשם train שמקבלת כקלט את sample של x , בשלב ראשון עשיתי חישוב של מרחקי הנקודות מהנקודה שקבלתי בקלט ואז עשיתי מיון למרחקי הנקודות לתוך מערך ואז עשיתי בדיקה לclasses הכי קרובים ובסוף הפעלתי את פונקציית most_common שכתבתי , פונקציה זאת סופרת את הפעמים של מופעי הערכים השונים ומחזירה את הערך של המספר הכי נפוץ.
3- פונקציית predict שמקבלת כקלט את סט האימון של X ועוברת על כל הנקודות ומפעילה את פונקציית train ומכניסה את התוצאות למערך predictions ומחזירה את המערך.

איך בחרתי את ה-K הטוב ביותר :

התחלתי ב-לבחור k רנדומלי , וראיתי ש בחירה בערך קטן של K מובילה לגבולות החלטה לא יציבים וגם בחירת ערך גדול של k מביאה ללא יציבות ואז ניסיתי לבדוק עם k בינוני שלא קטן ולא גדול ולפי הערכים שבדקתי סיכמתי שk צריך להיות ערך אי זוגי וגם שלא יהיה כפולה של מספר המחלקות (classes) ובסוף עשיתי בדיקה לכמה ערכי k שמקיימים תנאים אלה ובדקתי את אחוזי השגיאה בכל ערך והגעתי להחלטה ש k = 3 , k = 5 מביא לאחוז שגיאה קטן ככל הניתן אבל בחרתי בK=5 כיוון שהaccuracy שלו הייתה גבוהה מעט יותר.

2) Perceptron אלגוריתם :

מימוש : מימשתי את class Perceptron שמורכב מ 3 פונקציות :

1- פונקציית אתחול בשם initialize שמקבלת כקלט את נתוני ה-x וה-y וגם את הפרמטר-learning rate ומאתחלת אותם.
2- פונקציית אימון בשם train שמקבלת כקלט את הפרמטר epochs שמסמן מספר האיטרציות שהאלגוריתם צריך לפעול בהם. בהתחלה הגדרתי את ה bias באמצעות reshape ומספר ה-samples בסט האימון ואז הגדרתי את ה weights (3 וקטורים שמכילים אפסים). אחר כך עשיתי לולאה שנכנסים אליה כמספר הepochs שמקבלים בקלט וכל פעם שנכנסים ללולאה עושים shuffle לtrain_x,train_y ואז נעבור על כל זוג (x,y) ונעשה הבדיקות והעדכונים ל y,y_hat כמו שמתואר בתרגיל :

בתרגיל :

$$w_{t+1}^{y} = w_t^{y} + \eta * x$$

$$w_{t+1}^{\hat{y}} = w_t^{\hat{y}} - \eta * x$$

$$w_{t+1}^{i \neq \hat{y}, y} = w_{t+1}^{i \neq \hat{y}, y}$$

בקוד שלי :

```
weights[int(y_hat)] = weights[int(y_hat)] - u
weights[int(y)] = weights[int(y)] + u
```

```
weights[int(y)] = weights[int(y)]
weights[int(y_hat)] = weights[int(y_hat)]
```

ובסוף הפונקציה מחזירה את weights.

3- פונקציית predict שמקבלת כקלט את הweights ושורות של סט האימון וממלא את מערך ה predictions- בתוצאות שקיבלנו עבור train .

איך בחרתי את ה- learning rate ומספר epochs :

Epochs - הסתכלתי על ה- validation וה-training losses ועקבתי אחר הערכים שלהם. אם הvalidation loss הולך לגדול זה אומר התאמת יתר. ואז הבנתי שכדאי להגדיר את מספר ה epochs גבוה ככל האפשר ולהימנע מהתאמת יתר ובסוף הסתפקתי במספר 100 כי אם זה יותר אז הקוד לוקח הרבה זמן לרוץ וגם כך ה accuracy לא כל כך משתנה.

Learning rate – ראיתי שערך ברירת המחדל הוא 0.1 או 0.01 עבור ה learning rate , בהתחלה בחרתי את 0.1 ואז עשיתי כמה בדיקות של accuracy כאשר ה learning rate נע בין 0.01 ו 0.1 ולפי ה-validation ראיתי שהכי כדאי (מביא ל accuracy הכי טובה) זה לקחת את הממוצע ביניהם שזה 0.05 שעם זה קבלתי accuracy = 0.96.ي

3) SVM אלגוריתם :

מימוש :מימשתי את class SVM שמורכב מ 3 פונקציות :

1- פונקציית אתחול בשם initialize שמקבלת כקלט את נתוני ה-x וה-y וגם את פרמטרים learning rate ו lambada ומאתחלת אותם.

2- בדומה ל perceptron פונקציית אימון בשם train שמקבלת כקלט את הפרמטר epochs שמסמן מספר האיטרציות שהאלגוריתם צריך לפעול בהם. בהתחלה הגדרתי את ה bias באמצעות reshape ומספר הsamples- בסט האימון ואז הגדרתי את ה weights (3 וקטורים שמכילים אפסים). אחר כך עשיתי לולאה שנכנסים אליה כמספר הepochs שמקבלים בקלט וכל פעם שנכנסים ללולאה עושים shuffle לtrain_x,train_y ואז נעבור על כל זוג (x,y) ונעשה הבדיקות והעדכונים ל y,y_hat כמו שמתואר בתרגיל ובנוסף נחשב את ערך הloss :

בתרגיל :

$$w_{t+1}^{y} = (1-\eta\lambda)w_t^{y} + \eta * x$$

$$w_{t+1}^{\hat{y}} = (1-\eta\lambda)w_t^{\hat{y}} - \eta * x$$

$$w_{t+1}^{i\neq\hat{y},y} = (1-\eta\lambda)w_{t+1}^{i\neq\hat{y},y}$$

בקוד שלי :

```
weights[int(y_hat)] = (1 - u1) * weights[int(y_hat)] - u2
weights[int(y)] = (1 - u1) * weights[int(y)] + u2
```

```
if i != int(y):
    if i != int(y_hat):
        weights[i] = weights[i] * (1 - u1)
```

ובנוסף , אחרי כל 10 איטרציות הכפלתי את ה learning rate בעצמו כדי לשמור על accuracy גדולה ביותר. ובסוף מחזירים את weights.

3- פונקציית predict שמקבלת כקלט את הweights ושורות של סט האימון וממלא את מערך ה predictions- בתוצאות שקיבלנו עבור train .

איך בחרתי את ה-lambada וה- learning rate ומספר epochs :

Lambada- עבור ערכים גבוהים יותר של הלמבדה קיימת אפשרות גבוהה יותר של התאמה יתר, בעוד שעבור ערכים נמוכים יותר של הלמבדה יש אפשרויות גבוהות יותר של תת התאמה.

ואז עשיתי כמה בדיקות לaccuracy עם ערכי למבדה שונים ובסוף בחרתי את lambada = 0.1 שעבורו קבלתי את accuracy = 0.9708.

את ה- epochs , learning rate כמו בperceptron :

Epochs – כמו נימוק קודם של perceptron

Learning rate – כמו נימוק קודם של perceptron

4) Passive Aggressive (PA) אלגוריתם :

מימוש :מימשתי את class SVM שמורכב מ 3 פונקציות :

1- פונקציית אתחול בשם initialize שמקבלת כקלט את נתוני ה-x וה-y ומאתחלת אותם.
2- בדומה ל perceptron פונקציית אימון בשם train שמקבלת כקלט את הפרמטר epochs שמסמן מספר האיטרציות שהאלגוריתם צריך לפעול בהם. בהתחלה הגדרתי את ה bias באמצעות reshape ומספר הsamples- בסט האימון ואז הגדרתי את ה weights (3 וקטורים שמכילים אפסים). אחר כך עשיתי לולאה שנכנסים אליה כמספר הepochs שמקבלים בקלט וכל פעם שנכנסים ללולאה עושים shuffle לtrain_x,train_y ואז נעבור על כל זוג (x,y) ונעשה הבדיקות והעדכונים ל y,y_hat כמו שמתואר בתרגיל :

בתרגיל :

$$w_{t+1}^{y} = w_t^{y} + \tau * x$$

$$w_{t+1}^{\hat{y}} = w_t^{\hat{y}} - \tau * x$$

$$w_{t+1}^{i \neq \hat{y}, y} = w_{t+1}^{i \neq \hat{y}, y}$$

בקוד שלי :

```
weights[int(y)] = weights[int(y)] + n_loss * x
weights[int(y_hat)] = weights[int(y_hat)] - n_loss * x
```

```
weights[int(y)] = weights[int(y)]
weights[int(y_hat)] = weights[int(y_hat)]
```

ובסוף מחזירים את weights.

3- פונקציית predict שמקבלת כקלט את הweights ושורות של סט האימון וממלא את מערך ה- predictions בתוצאות שקיבלנו עבור train .

איך בחרתי את מספר ה -epochs :

Epochs - כמו נימוק קודם של perceptron