למידת מכונה – מטלת בית 2

מגיש: לירון חיים

ת.ז: 206234635

דו"ח היפר-פרמטרים

אלגוריתם KNN:

ב-KNN, הפרמטר שנרצה לכייל הוא k, כמות השכנים הקרובים שלפיהם נקטלג את המידע. מצאתי את ה-k שממזער את כמות השגיאות עם הקוד בהמשך. הרעיון הוא לחלק את סט האימון שלנו לסגמנטים של וולידציה ואימון, להריץ את KNN עבור טווח k בין 1 ל-11, לסכום את כלל השגיאות לכל k בעזרת היסטוגרמה ולמצוא עבור איזה k (אינדקס) כמות השגיאות היא מינימלית.

כמו כן, שני שלבים בדרך היו נרמול ה-Data ולחשב את המרחק של הנקודות מהשכנים הקרובים אליהם. לשם כך, השתמשתי בשתי דרכי נרמול שונות (min-max ו- Z score) ובחרתי את האחת שהניבה תוצאות הכי טובות. באופן דומה גם בחירת פונקציית מרחק מתוך 3 אפשרויות (אוקלידי, מנהטן וקנברה), ובחרתי בקנברה.

```python
def choose_best_k(train, y_train_data, k_range=11, kfold=5):
    k_histogram = np.zeros(k_range)
    k_histogram[0] = sys.maxsize
    for r in range(kfold):
        # split data into train and valid sets, 5-fold
        new_train, new_y, valid_data, y_valid = k_fold_split(train, y_train_data, r, 5)
        # normalize data according to the train setS
        valid_data = normalize_data(valid_data, new_train, "zscore")
        new_train = normalize_data(new_train, new_train, "zscore")
        for k in range(1, k_range):
            # prediction
            y_hats = run_knn(new_train, valid_data, new_y, k)
            # check differences between predictions and real labels
            loss_array = y_hats - y_valid
            for diff in loss_array:
                if diff != 0:
                    k_histogram[k] += 1
    # outputs
    k_histogram = k_histogram / kfold
    best_k = np.argmin(k_histogram)
    success_rate_percent = (1 - (np.min(k_histogram) / len(y_valid))) * 100
    print("success rate: ", success_rate_percent, "%")
    print("loss: ", k_histogram[np.argmin(k_histogram)], "  k: ", best_k)
```

```
success rate:  91.66666666666666 %
loss:  4.0   k:  9
```

למידת מכונה – מטלת בית 2

מגיש: לירון חיים

ת.ז: 206234635

אלגוריתמי Multi-Class: Perceptron, SVM, Passive Aggressive

עבור מודל ה-Perceptron יש לכייל את פרמטר קצב הלמידה ומספר האפוקים שנעבור על ה-Data באימון.

עבור מודל ה-SVM גם כן יש לכייל אותם, יחד עם פרמטר נוסף של רגולריזציה – למדה.

עבור מודל ה- PA יש לכייל רק את מספר האפוקים לאימון.

השתמשתי בפונקציה אחת דומה שבודקת את ה-loss עבור כל אחד מהמודלים, וכך בחרתי ערכים מתאימים לפרמטרים השונים.

```python
def get_opt_hyper_parameters(train, y_data, lamda=0.1, max_epochs=200, learning_rate=0.1, k_fold_segments=5, it=1):
    e_histogram = np.zeros(max_epochs)
    e_histogram[0] = sys.maxsize
    sd_array = []

    for i in range(it):
        # shuffle data with random seed
        sd = np.random.randint(1000000)
        np.random.seed(sd)
        np.random.shuffle(train)
        np.random.seed(sd)
        np.random.shuffle(y_data)
        for k in range(k_fold_segments):
            # split data to train and validation (k-fold)
            new_train, new_y, valid_data, y_valid_data = k_fold_split(train, y_data, k, k_fold_segments)
            # normalize data
            valid_data = normalize_data(valid_data, new_train, "zscore")
            new_train = normalize_data(new_train, new_train, "zscore")
            # add bias
            new_train = np.hstack((np.ones((new_train.shape[0], 1)), new_train))
            valid_data = np.hstack((np.ones((valid_data.shape[0], 1)), valid_data))
            for e in range(1, max_epochs):
                # train
                weights = run_perceptron(new_train, new_y, sd, e, learning_rate)
                # weights = run_passive_aggressive(new_train, new_y, sd, e)
                # weights = run_svm(new_train, new_y, sd, e, lamda, learning_rate)
                # get predictions
                y_hats = get_validation_prediction(weights, valid_data)
                # sum errors
                for y, y_hat in zip(y_valid_data, y_hats):
                    if y_hat != y:
                        e_histogram[e] += 1
        # prepare outputs
        e_histogram /= k_fold_segments
        opt_epochs = np.argmin(e_histogram)
        opt_loss = e_histogram[opt_epochs]
        rate = 100 * (1 - (opt_loss / 48))
        sd_array.append([rate, sd])
        print("e: ", opt_epochs)
        print("loss: ", opt_loss)
        print("seed: ", sd)
        print("rate: ", rate)
    sd_array = sorted(sd_array, reverse=True)
    print(sd_array)
```

בדיקה עבור ה- Perceptron.

בדיקה עבור ה- PA.

בדיקה עבור ה- SVM.

למידת מכונה – מטלת בית 2

מגיש: לירון חיים

ת.ז: 206234635

הפונקציה מאתחלת היסטוגרמה שתמנה את כמות השגיאות בכל עבור כל מודל לפי מספר האפוקים שאומן. גם כאן מצאתי כי האופציה לנרמל את ה-Data ע"י Z score היא הטובה ביותר. בנוסף לכך, הוספת עמודת bias של אחדות שיפרה את התוצאות.

השתמשתי ב- Cross Validation וחילקתי את הסט ל5 חלקים, השתמשתי בכל פעם בחלק אחד עבור סט ולידציה. רצתי על טווח אפוקים כרצוני ובכל איטרציה אימנתי מודל חדש במספר האפוקים שאיטרציה זו מייצגת.

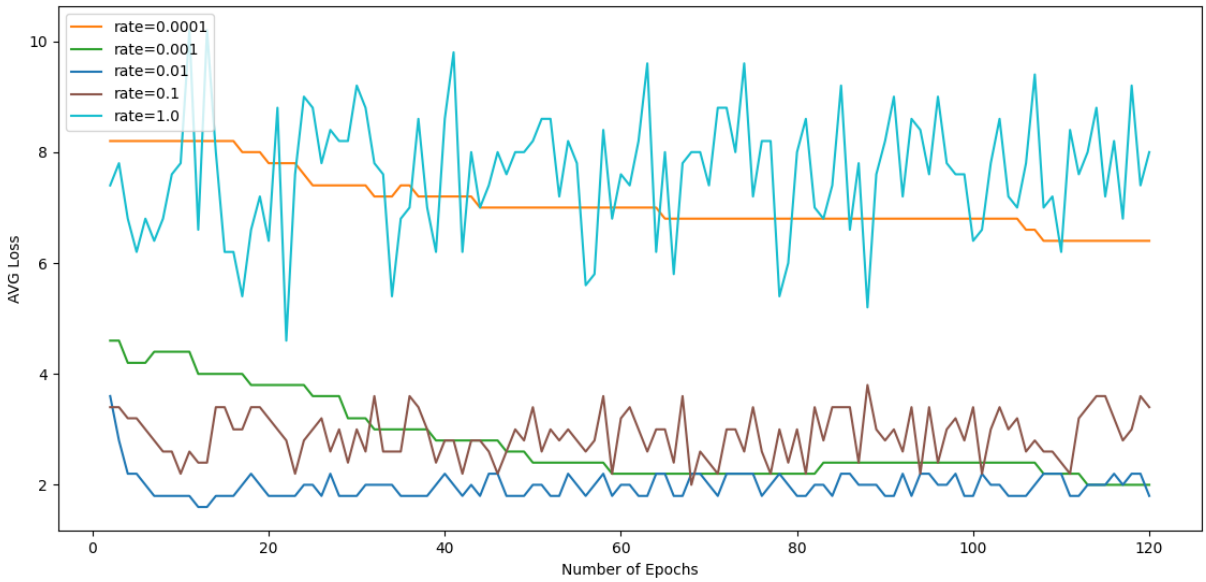
קטע הקוד המגריל seed עבור ה-shuffle נועד לערבב לי את המידע לפני שאחלק אותו.

בנוסף לכך, אני מבצע feature selection. הרצתי את כל המתואר כמספר הפיצ'רים ובכל פעם הורדתי פיצ'ר אחר (עמודה אחרת). קיבלתי תוצאות טובות עבור הורדת העמודה האחרונה.

בחירת הלמדה והקצב הלמידה

אסביר כיצד בחרתי את הפרמטרים באלגוריתם SVM כיוון שיש לבחור עבורו גם למדה וגם קצב, והדבר דומה בPerceptron.

קיים "משחק" בין הפרמטרים של הלמדה וקצב הלמידה. הלמדה דואגת לנו לרגולריזציה, משנה לנו את עוצמת הענישה על תיוגים נכונים הנמצאים ב margin מקו החלטה. ככל שלמדה מקבל ערך גבוה יותר – כך נעניש יותר ונגדיל את ה-Loss. מנגד, אם הערך יהיה נמוך מדי נוכל להגיע למצב של overfitting. ניתן לבחור ערכים בכפולות של 10 (באזור ה0.01) ולבדוק את מזלנו. יצרתי את הגרף הבא כדי לקבל יותר מושג איזה loss אקבל עם ערכי למדה שונים. העקומה התכלת מייצגת ערך גבוה מדי ואילו הכתום והירוק מייצגים overfitting. לכן החלטתי ללכת עבור בערך 0.01 של העקומה הסגולה.



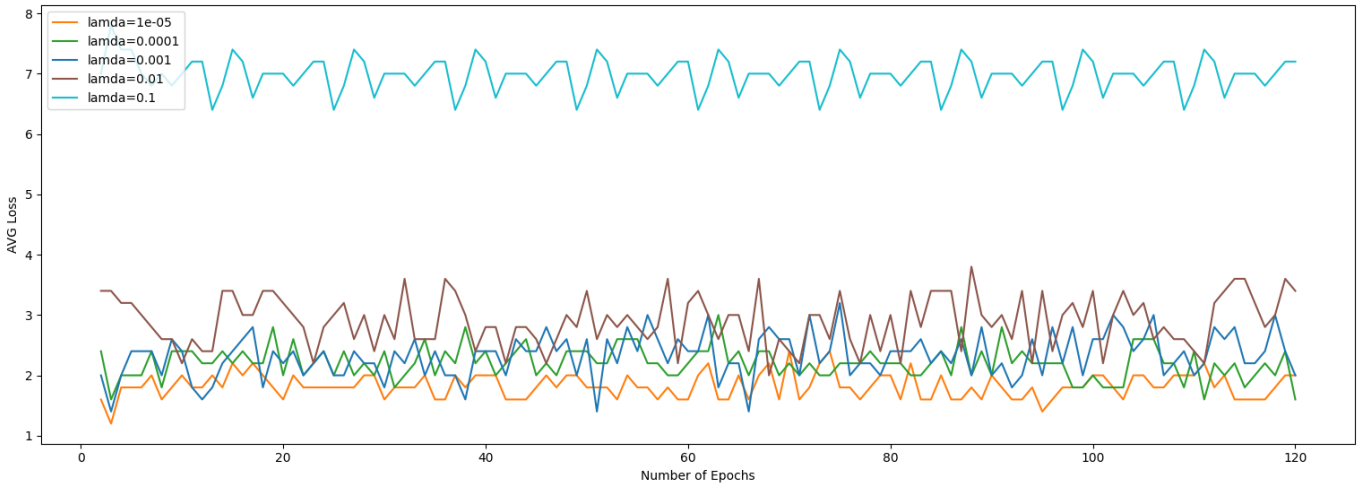
הגרף הנ"ל נוצר עבור ריצה עם learning rate = 0.1.

למידת מכונה – מטלת בית 2

מגיש: לירון חיים

ת.ז: 206234635

גם עבור פרמטר הלמידה יצרתי גרף:



נזכור שעבור ערך גבוה בפרמטר הלמידה נוכל "לדלג" על נקודות מינימום של ערכי loss ונפספס מזיעורים אפשריים. לעומת זאת, ערך נמוך מדי יכול לעקב אותנו במציאת מינימום ולא נמצא אותו במספר אפוקים נמוך ומספק. לכן נבחר ערך מאלו הנמוכים בגרף.

כל הערכים שבחרתי נמצאים בקובץ הקוד הראשי שהוגש, כחלק מהקריאה לפונקציות האימון.

למידת מכונה – מטלת בית 2

מגיש: לירון חיים

ת.ז: 206234635

נספחים:

פונקציית הCross Validation. חלוקת המידע ל k קבוצות ובחירת סגמנט עבור ה validation.

```
def k_fold_split(train, y_data, segment, k=5):
    size = len(train)
    diff = size % k
    segment_size = size / k
    segment_start = segment * segment_size
    segment_end = (segment + 1) * segment_size
    if segment + 1 == k and diff != 0:
        segment_end -= diff
    div_valid = []
    div_train = []
    div_y_train = []
    div_y_valid = []
    for i in range(len(train)):
        if segment_start <= i < segment_end:
            div_valid.append(train[i])
            div_y_valid.append(y_data[i])
        else:
            div_train.append(train[i])
            div_y_train.append(y_data[i])
    return np.asarray(div_train), np.asarray(div_y_train), np.asarray(div_valid), np.asarray(div_y_valid)
```

פונקציות מרחקים

```
def euc_distance(a, b):
    return np.linalg.norm(a - b, ord=2)


def manhattan_distance(a, b):
    return np.sum(abs(a - b))


def canberra_distance(a, b):
    can_sum = 0
    for aa, bb in zip(a, b):
        d = (abs(aa) + abs(bb))
        if 0 == d:
            continue
        can_sum += abs(aa - bb) / d
    return can_sum
```