בס"ד

למידת מכונה Report

knn

על מנת למצוא את הערכים מתאימים עבור k השכנים,נרמול ומרחק ביצעתי את הדברים הבאים:
תחילה בדקתי אם ישנה עמודה שכדאי להוריד –

```
def remove_one_best(x, y):
    print('non remove col')
    find_k(x, y)
    col = x.shape[1]
    for i in range(col):
        x_copy = x.copy()
        x_copy=np.delete(x_copy,i,axis=1)
        print('remove',i)
        find_k(x_copy, y)
```

לאחר מכן בדקתי איזה מרחקים ונרמול נותנים לי את מספר שגיאות הנמוך ביותר על ולידציה-

```
def find_k(train_x, train_y):
    norm = max_min
    dis = euclidean
    print('norm=maxmin,dis=euclidan')
    iter_find_k(train_x,train_y,norm,dis)
    norm = to_norm_z
    dis = euclidean
    print('norm=z_score,dis=euclidan')
    iter_find_k(train_x, train_y, norm, dis)
    norm = to_norm_z
    dis = manhattan
    print('norm=z_score,dis=manhattan')
    iter_find_k(train_x, train_y, norm, dis)
    norm = max_min
    dis = manhattan
    print('norm=min-max,dis=manhattan')
    iter_find_k(train_x, train_y, norm, dis)
    norm = to_norm_z
    dis = canberra
    print('norm=zscore,dis=canberra')
    iter_find_k(train_x, train_y, norm, dis)
    dis = canberra
    norm = max_min
    print('norm=min-max,dis=canberra')
    iter_find_k(train_x, train_y, norm, dis)
```

חישוב של סוגי מרחקים :

```
def euclidean(p1, p2):
    return np.linalg.norm(p1 - p2, ord=2)


def canberra(p1, p2):
```

בס"ד

```
    sum = 0
    size = len(p1)
    for i in range(size):
        if p1[i] == 0 and p2[i] == 0:
            val = 0
        else:
            val = abs(p1[i] - p2[i]) / (abs(p1[i]) + abs(p2[i]))
        sum += val
    return sum


def manhattan(p1, p2):
    return np.linalg.norm(p1 - p2, ord=1)
```

חישוב של סוגי נירמולים:

```
def max_min(arrry_train, array_valid):
    dict_f = {}
    if len(arrry_train) > 0:
        for x in range(len(arrry_train[0])):
            dict_f.update({x: []})
    for xi in arrry_train:
        for i in range(len(xi)):
            l = []
            l.extend(dict_f[i])
            l.append(xi[i])
            dict_f.update({i: l})
    for key, value in dict_f.items():
        min_f = min(value)
        max_f = max(value)
        dict_f.update({key: [min_f, max_f]})
    line_train = len(arrry_train)
    col_train = 0
    if line_train > 0:
        col_train = len(arrry_train[0])
    line_valid = len(array_valid)
    col_valid = 0
    if line_valid > 0:
        col_valid = len(array_valid[0])
    arr_train_new = np.zeros((line_train, col_train))
    arr_valid_new = np.zeros((line_valid, col_valid))
    for x in range(line_train):
        for y in range(col_train):
            if dict_f[y][0] == dict_f[y][1]:
                arr_train_new[x][y] = 0
                if x < line_valid and y < col_valid:
                    arr_valid_new[x][y] = 0
            else:
                arr_train_new[x][y] = (arrry_train[x][y] - dict_f[y][0]) /
(dict_f[y][1] - dict_f[y][0])
                if x < line_valid and y < col_valid:
                    arr_valid_new[x][y] = (array_valid[x][y] -
dict_f[y][0]) / (dict_f[y][1] - dict_f[y][0])
    return np.asarray(arr_train_new), np.asarray(arr_valid_new)
```

בס"ד

```
def to_norm_z(arrry_train, array_valid):
    size_col = 0
    size_row = len(arrry_train)
    line_valid = len(array_valid)
    col_valid = 0
    if line_valid > 0:
        col_valid = len(array_valid[0])
    if size_row > 0:
        size_col = len(arrry_train[0])
    arr_valid_new = np.zeros((line_valid, col_valid))
    arr_train_new = np.zeros((size_row, size_col))
    avg = np.mean(arrry_train, axis=0)
    std = np.std(arrry_train, axis=0)
    for i in range(size_row):
        for j in range(size_col):
            arr_train_new[i][j] = (arrry_train[i][j] - avg[j]) / std[j]
            if i < line_valid and j < col_valid:
                arr_valid_new[i][j] = (array_valid[i][j] - avg[j]) / std[j]
    return np.asarray(arr_train_new), np.asarray(arr_valid_new)
```

בדקתי את accurncy על יד 50 הרצות של 11 בחירות שונות של k-fold .

```
def iter_find_k(x, y,norm,dis):
    dict = {}
    it = 50
    for i in range(it):
        min_k(x, y, norm, dis, dict, 11)
    min_iter = min(dict, key=dict.get)
    best_iter = []
    for key, value in dict.items():
        if value == dict[min_iter]:
            best_iter.append(key)
    loss1 = dict[min_iter]
    print(loss1, (1 - (loss1 / (240 * it))) * 100, best_iter)
```

הפונקציה min_k הריצה 11 פעם k-fold (כאשר יש שאפל על מידע כל פעם ב valid_train_partition)

```
def min_k(train_x, train_y, norm, dis, dict, n):
    for i in range(1, n + 1):
        p = valid_train_partition(train_x, train_y, norm, 1 / 5)
        size = len(p)
        for j in range(size):
            x_norm = p[j][0]
            y = p[j][1]
            x_valid_norm = p[j][2]
            y_valid = p[j][3]
            number = 0
            if i in dict:
                number = dict[i]
            dict.update({i: number + validation(x_norm, y, x_valid_norm,
y_valid, i, dis)})
```

הפונקציה **valid_train_partition** תוסבר בהמשך. הפונקציה validtion מריצה את knn על כל שורה בולידשיין ומחזירה כמה טעיות היו:

בס"ד

```
def validation(train_x, train_y, x_vaild, y_vaild, k, dis):
    errors = 0
    size = len(x_vaild)
    for i in range(size):
        y_tag = knn(x_vaild[i], train_x, train_y, k, dis)
        if y_vaild[i] != y_tag:
            errors += 1
    return errors
```

סה"כ יוצא אחוזי הצלחה גבוהים עם נרמול z-score ומרחק Canberra על ולידשיין שלי:

```
norm=zscore,dis=canberra
906 92.45 [9]
```

92.45%

ובסמביט על טסט חדש 96.667%

perceptron

כדי למצוא את הפרמטרים מספר איטרציות וארטא חילקתי את ה trian לvalidition 20% וtrian 80% (kfold כאשר K=5).

החלוקה מתבצעת בפונקציה הבאה:

```
def valid_train_partition(tx, ty, norm, k):
    size = len(tx)
    partition = []
    shuffle_xy = shuffle(tx.copy(), ty.copy())
    x_shuffle = shuffle_xy[0]
    y_shuffle = shuffle_xy[1]
    number = int(k * size)
    for i in range(0, size, number):
        x = []
        y = []
        x_valid = []
        y_valid = []
        for j in range(0, i):
            x.append(x_shuffle[j])
            y.append(y_shuffle[j])

        for j in range(i, i + number):
            x_valid.append(x_shuffle[j])
            y_valid.append(y_shuffle[j])
        for j in range(i + number, size):
            x.append(x_shuffle[j])
            y.append(y_shuffle[j])
        x_norm, x_valid_norm = norm(x, x_valid)

        partition.append([np.asarray(x_norm), np.asarray(y),
np.asarray(x_valid_norm), np.asarray(y_valid)])
    return partition
```

מחזיר מערך של חלוקות שכל רשימה במערך היא חלוקה מסוימת ל trian וvalidtion.

הגרלתי מספר לseed, מספר איפוקים, ואת rata ע"י הפונקציה הבאה :(זה מגריל עוד פרמטרים שלא משתמשים בהם לאלגוריתם הנ"ל):

```
def rand_parameters():
    return
```

```
[np.random.choice(1000),np.random.randint(1,100),np.random.choice(rata,
1)[0],np.random.choice(rata, 1)[0]]
```

לאחר מכן כדי לבדוק את עמודה שהכי כדאי להוריד יצרתי פונקציה שבודקת מה ה accurncy בולדיישין שלי בהינתן שהורדתי עמודה אחת מהtrain (שמכיל בתוכו את validation):

```
def remove_one_best(x, y):
    print('non remove col')
    the_good_iteration(x, y)
    col = x.shape[1]
    for i in range(col):
        x_copy = x.copy()
        x_copy=np.delete(x_copy,i,axis=1)
        print('remove',i)
        the_good_iteration(x_copy, y)
```

הפונקציה the_good_iteration מריצה את המודל מספר פעמים על מנת לכייל את הפרמטרים:

```
def the_good_iteration(x, y):
    list_dict=[]
    it = 10
    for j in range(10):
        dict = {}
        parameter = rand_parameters()
        for i in range(it):
            start_parameter_algo(perceptron, x, y, parameter, dict)
        list_dict.append((dict,parameter))
    best_iter = []
    best_dict = []
    for t in list_dict:
        print(t[1])
        min_iter = min(t[0], key=t[0].get)
        best_dict.append(t[0][min_iter])
    minimum=min(best_dict)
    index_dict = best_dict.index(minimum)
    dict = list_dict[index_dict][0]
    for key, value in dict.items():
        if value == minimum:
            best_iter.append(key)
    loss1 = minimum
    print(loss1, (1 - (loss1 / (240 * it))) * 100,
best_iter,list_dict[index_dict][1])
```

מריצה 10 פעמים על פרמטרים שונים את מודל 10 פעמים ובודקת את אחוז accurncy המקסימלי, מספר איטרציות נותנת אותו ופרמטרים נותנים אותו ומדפיסה את כלל דברים. מריצה את start_parameter_algo עם אלגוריתם parcepton המעדכן את מילון כאשר key זה מספר איטרציה ו value זה הloss, לכל הגרלה של פרמטרים יש מילון שונה ובסוף מוצאים את min loss מבין כל המילונים.

```
def start_parameter_algo(algo, train_x, train_y, parameter, dict_loss):
    norm = to_norm_z
    indexs = parameter[1]
    np.random.seed(parameter[0])
    p = valid_train_partition(train_x, train_y, norm, 1 / 5)
    size = len(p)
    for j in range(size):
        x_norm = p[j][0]
        y = p[j][1]
```

```
        x_valid_norm = p[j][2]
        y_valid = p[j][3]
        for i in range(1, indexs+1):
            parameter[1] = i
            list_w = algo(x_norm, y, parameter)
            number = 0
            if i in dict_loss:
                number = dict_loss[i]
            dict_loss.update({i: number + loss(x_valid_norm, y_valid,
list_w)})
```

מריץ את המודל לפי מספר איטרציות שהוא מקבל בפרמטרים ומכניס את תוצאות של הloss למילון. הרצה של מודל מתבצעת כאשר `algo(x_norm, y, parameter)` מחזיר את הw לפי אלגוריתם שנתנו לו פרספטון ולאחר מכן בודק את הloss על ולידשיין לפי פונקציית הloss . בעקרון הtrain משנה את הw בהתאם למספר אטרקציות שנותנים לו. מצאתי שפרמטרים שנותנים לי את ה-accuracies הגובה ביותר הם: seed=813 מספר איטרציות הוא 63 והrata=0.0001 והורדה של עמודה רביעית .בולידשיין ובהגשה בסמביט אלה היפר פרמטרים שנתנו את תוצאות הטובות ביותר.

כפי שניתן לראות בין אלגוריתם svm, perceptron, PA מה ששונה זה זה פרמטר שנותנים `start_parameter_algo` ולכן הם עוברים את אותו תהלך בדיוק שתיארנו להנל רק אם אלגוריתם שונה למציאת הw

Pa

כפי שאמרנו קודם התהליך כמעט אותו דבר, ההבדל הוא ששולחים את pa בתור האלגוריתם ב `the_good_iteration` בקריאה לפונקציה `start_parameter_algo`.

הפרמטרים שמצאתי הם שמספר איטרציות הוא 21 ,seed=865 והורדת העמודה אחרונה. בולידשיין ובהגשה בסמביט אלה היפר פרמטרים שנתנו את תוצאות הטובות ביותר.

Svm

כפי שאמרנו קודם התהליך כמעט אותו דבר, ההבדל הוא ששולחים את svm בתור האלגוריתם ב `the_good_iteration` בקריאה לפונקציה `start_parameter_algo`.

הפרמטרים שנותנים לי את הaccourncy המקסימלי הם seed=7553 ,32 איטרציות, rata=0.1 ,lamda=0.01 והורדת העמודה אחרונה. בולידשיין ובהגשה בסמביט אלה היפר פרמטרים שנתנו את תוצאות הטובות ביותר.