به نام خدا

درس: مقدمه‌ای بر یادگیری ماشین

استاد: دکتر صالح کلیبر

گزارش تمرین شماره ۳

سیّدمحمّدامین منصوری طهرانی

۹۴۱۰۵۱۷۴

کاهش بعد:

PCA: در هر ۳ روش اشاره شده هدف کاهش ابعاد ویژگی‌های داده‌ها است به نحوی که محاسبات را سریع‌تر و راحت‌تر کرده و فقط از اطلاعات مفیدتر بهره بگیریم و در استفاده از منابع محاسباتی صرفه‌جویی کنیم. در این روش بهتر است ابتدا داده‌ها را standardize کرده به این معنی که میانگین آن‌ها را از داده‌ها کم کنیم و همه را بر انحراف معیار داده‌ها تقسیم کنیم تا میانگین داده‌ها صفر و واریانس آن یک شود. **دقت شود در تمامی روش‌ها ابتدا کاهش بعد با توجه به داده‌های آموزش انجام می‌شود و سپس همان تبدیل بر روی داده تست انجام می‌شود و داده تست تبدیل شده به ورودی طبقه‌بندها داده می‌شود و لذا برای پیش‌بینی مناسب باید داده های تست جدید را به هم به scaler پیوست شده که برای standardize کردن است بدهید و هم تبدیل کاهش بعد را بر روی آن انجام داده و سپس برای پیش‌بینی به طبقه‌بند بدهید.** در این روش، ایده اصلی این است که جهت‌هایی را بیابد که واریانس داده‌ها در آن جهت‌ها بیشتر است. به عبارتی بیشینه اطلاعات ممکن را بدست می‌آورد.

به عبارت دیگر یک تبدیل متعامد است که یک مجموعه مشاهده احتمالاً وابسته را به مجموعه‌ای مستقل خطی می‌نگارد. این مجموعه محورهایی دارد که به آن‌ها جز اصلی یا principal component می‌گویند. خروجی PCA به نحوی است که اولین محور بیشترین واریانس را دارد و محور دوم دومین بزرگترین واریانس را و این روند ادامه می‌یابد. به این ترتیب برای مسأله ما که هر عکس ۷۸۴ بعد دارد، با اعمال PCA بر روی آن به تعداد بعد کمتر (حدود ۴۰) رسیده و این پیکسل‌ها بیشترین واریانس را نشان داده‌اند و برای محاسبات بعدی بیشترین اطلاعات (هم‌ارز واریانس) را دارند.

LDA: اگر به روش قبل دقت کرده باشیم، روشی بدون ناظر بوده (unsupervised) و اصطلاحاً به صورت کور، به دنبال جهت‌هایی می‌گردد که واریانس داده‌ها را بیشینه کنند. در روش LDA که به صورت با ناظر عمل می‌کند (supervised)، برچسب داده‌ها نیز در نظر گرفته شده و جهت‌هایی جستجو می‌شوند که جدایی بین داده‌های آموزش از کلاس‌های مختلف را بیشینه کنند. طبق مشاهدات تجربی برای آموزش با داده‌های کم روش PCA نسبت به این روش برتری دارد ولی معمول است که LDA به دنبال PCA بر روی داده‌ها اعمال شود. (در این تمرین عملکرد هرکدام جدا بررسی می‌شود.)

بنابراین به طور خلاصه روش مانند PCA است و جهت‌ها با اندکی تفاوت تعیین می‌شوند. نخست میانگین داده‌ها در ویژگی‌های مختلف محاسبه شده، ماتریس پراکندگی (بین کلاسی و داخل کلاسی) محاسبه شده و سپس بردارهای ویژه و مقدارهای ویژه متناظر آن‌ها بدست می‌آید. این مقادیر مرتب شده و k مقدار اول آن برای نمایش کاهش بعد یافته داده‌ها استفاده می‌شود و تبدیل بدست آمده بر داده‌های آموزش و تست اعمال می‌شود.

GDA: دو روش فوق همان‌طور که مشخص است، داده‌ها را به صورت خطی از هم جدا می‌کنند (جهت‌هایی که به وسیله ترکیب خطی جهت‌های principal component ها تعیین می‌شوند فضاهای خطی برای طبقه‌بندی ایجاد می‌کنند.) در این روش، با استفاده از ایده kernel ها که در درس نیز داشتیم، داده‌ها به فضای دیگری نگاشته شده که در آن فضا به صورت خطی جدایی‌پذیر باشند. با بدست آمدن نگاشت و جزهای اصلی در فضای جدید، داده‌های آموزش و تست تحت همان تبدیل‌ها تبدیل می‌شوند و بعد آن‌ها در فضای جدید کاهش می‌یابد. (در این تمرین با توجه به این‌که انتخاب روش کاهش بعد به عهده خودمان گذاشته شده و اجباری برای اعمال همه روش‌ها نیست، با توجه به این‌که داده‌های MNIST داده‌های نسبتاً خوبی می‌باشند، فقط از دو روش اول برای کاهش بعد استفاده می‌کنیم.)

طبقه‌بندی چند دسته‌ای:

در قسمت امتیازی تمرین قبل به این پرسش‌ها پاسخ داده شد و لذا جواب همان قسمت را در این‌جا می‌آورم.

One against all: برای هر کلاس یک طبقه‌بند آموزش دهیم که داده‌های مربوط به کلاس خودش را ۱ و سایر داده‌ها را صفر دسته‌بندی کند و این دسته‌بندی دارای یک درجه اطمینان باشد. در این صورت هر داده جدیدی که می‌آید توسط هر کلاس دسته‌بند برچسپ زده می‌شود و یک درجه اطمینان خواهد داشت (confidence score). برچسب این داده را بیشترین درجه اطمینان که از خروجی ۱۰ طبقه‌بند (برای رقم‌های MNIST) می‌آید تعیین می‌کند.

One against one: در این حالت به ازای k دسته (در مسأله ما ۱۰ دسته) ، از هر جفت دسته یک نمونه برمی داریم و یک طبقه‌بند باینری آموزش دهیم که این دو دسته را تمیز دهد. بنابراین به تعداد انتخاب ۲ از k طبقه بند باینری داریم. (در مسأله ما انتخاب ۲ از ۱۰ تا) به هنگام پیش‌بینی همه این طبقه‌بندهای جفت جفت بر داده جدید اعمال می‌شوند و دسته‌ای که بیشترین تعداد برچسب یک را از این طبقه‌بندهای باینری از آن خود کند، برچسب داده جدید را تعیین خواهد کرد.

مسأله:

خلاصه توضیح درباره کد و روش‌ها: در ادامه با توجه به این‌که کد تا حد زیادی کامنت‌گذاری شده‌است، به ذکر توضیحات کوتاهی درباره هر قطعه کد اکتفا می‌کنیم و خروجی آن بخش را گزارش می‌کنیم.

```python
# ################################################ Functions ################################################


def load_images_from_folder(folder):
    images = []

    # os.listdir('path') returns name of the folders and files in 'path'.

    # for loop is on the name of the files in folder. Hence folder should be something like ./training/4 so that
    # os.listdir returns name of all png files inside this folder
    for filename in os.listdir(folder):
        img = Image.open(os.path.join(folder, filename))
        images.append(img)

    # Output images are in a list
    return images


# Also glob can be used to load *.extension i.e. *.png


def multi_class_confusion_matrix_conversion(cm):
    true_positive = []
    for index in range(0, len(cm)):
        true_positive.append(cm[index][index])
    false_negative = np.sum(cm, axis=1, dtype=int) - true_positive
    false_positive = np.sum(cm, axis=0, dtype=int) - true_positive
    true_negative = np.sum(cm) - (true_positive + false_negative + false_positive)
    return true_positive, true_negative, false_positive, false_negative
```

در این قطعه‌کد دو تابع مشاهده می‌شود. تابع اول به وسیله نام فایل‌های فولدرهای مختلف آن‌ها در یک لیست پایتون ذخیره می‌کند و این لیست را باز می‌گرداند. تابع دوم یک ماتریس درهم‌ریختگی برای داده چند کلاسه را گرفته و درایه‌های قطری آن را به عنوان true positive ذخیره می‌کند. سپس روی سطرها و ستون‌های مختلف جمع زده و از مقدار قطر اصلی آن کم می‌کند تا false negative و false positive بدست آیند. True negative نیز جمع همه درایه‌ها منهای مجموع سه داده قبلی می‌باشد و در پایان، تابع این ۴ مقدار را به عنوان خروجی برمی‌گرداند. برای محاسبه accuracy و specificity به این مقادیر نیاز داریم چون با دستور مستقیم پایتون بدست نمی‌آیند.

در تابع بعدی رسم ROC curve انجام می‌شود و با گرفتن برچسب‌های پیش‌بینی شده و برچسب‌های واقعی این کار انجام می‌شود. دقت می‌کنیم که برای رسم roc چند کلاسه، داده‌ها با دستور get.dummies به صورت one hot تبدیل می‌شوند. به ازای هر کلاس، مقادیر نرخ true positive و false positive که برای رسم roc لازم هستند بدست می‌آیند و مساحت زیر نمودار برای آن‌ها به وسیله دستور auc محاسبه می‌شود.

```
def roc_curve_plot(y_test, y_pred, n_classes, title):
    y_score = y_pred
    fpr = dict()
    tpr = dict()
    roc_auc = dict()
    for i in range(n_classes):
        fpr[i], tpr[i], _ = roc_curve(np.array(pd.get_dummies(y_test))[:, i], np.array(pd.get_dummies(y_score))[:, i])
        roc_auc[i] = auc(fpr[i], tpr[i])
    # Compute micro-average ROC curve and ROC area
    fpr["micro"], tpr["micro"], _ = roc_curve(np.array(pd.get_dummies(y_test)).ravel(),
                                              np.array(pd.get_dummies(y_score)).ravel())
    roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
    # Compute macro-average ROC curve and ROC area
    lw = 2
    # First aggregate all false positive rates
    all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

    # Then interpolate all ROC curves at this points
    mean_tpr = np.zeros_like(all_fpr)
    for i in range(n_classes):
        mean_tpr += np.interp(all_fpr, fpr[i], tpr[i])

    # Finally average it and compute AUC
    mean_tpr /= n_classes

    fpr["macro"] = all_fpr
    tpr["macro"] = mean_tpr
    roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

    # Plot all ROC curves
    plt.figure()
    plt.plot(fpr["micro"], tpr["micro"],
             label='micro-average ROC curve (area = {0:0.2f})'
                   ''.format(roc_auc["micro"]),
             color='deeppink', linestyle=':', linewidth=4)

    plt.plot(fpr["macro"], tpr["macro"],
             label='macro-average ROC curve (area = {0:0.2f})'
```

```
    # Finally average it and compute AUC
    mean_tpr /= n_classes

    fpr["macro"] = all_fpr
    tpr["macro"] = mean_tpr
    roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

    # Plot all ROC curves
    plt.figure()
    plt.plot(fpr["micro"], tpr["micro"],
             label='micro-average ROC curve (area = {0:0.2f})'
                   ''.format(roc_auc["micro"]),
             color='deeppink', linestyle=':', linewidth=4)

    plt.plot(fpr["macro"], tpr["macro"],
             label='macro-average ROC curve (area = {0:0.2f})'
                   ''.format(roc_auc["macro"]),
             color='navy', linestyle=':', linewidth=4)

    colors = cycle(['aqua', 'darkorange', 'cornflowerblue'])
    for i, color in zip(range(n_classes), colors):
        plt.plot(fpr[i], tpr[i], color=color, lw=lw,
                 label='ROC curve of class {0} (area = {1:0.2f})'
                       ''.format(i, roc_auc[i]))

    plt.plot([0, 1], [0, 1], 'k--', lw=lw)
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(title)
    plt.legend(loc="lower right")
    plt.show()

    return
```

در قطعه‌کدی که در ادامه می‌آید عملیات لود کردن داده‌ها اتفاق می‌افتد. سپس برای کارهای بعدی با دستور as array به فرمت آرایه دو بعدی درمی‌آیند. برچسب آن‌ها را نیز با توجه به این‌که به ترتیب از فولدر training لود کرده‌ایم، به ترتیب تعیین می‌کنیم و ذخیره می‌کنیم. در انتهای این بخش یک تصویر به عنوان نمونه نمایش داده می‌شود.

```
# ############################################## Loading Data ################################################

# 0, 1, 2 respectively correspond to selecting PCA, LDA and GDA. To get the results for each of those, please simply
# change this selector and run the code again.
selector = 0
n_classes = 10

data = []

for i in tqdm(range(10)):
    data_temp = load_images_from_folder('./training/' + str(i))
    data.append(data_temp)

array_data = []
for i in tqdm(range(len(data))):
    class_data = data[i]
    for j in range(len(data[i])):
        temp = np.asarray(class_data[j])
        array_data.append(temp)

array_data = np.asarray(array_data)

data_labels = []
for i in range(10):
    temp = list(np.zeros(len(data[i])) + i)
    data_labels = data_labels + temp

# If you ever read a file using csv_read you can quickly identify anomalies using data.describe()! Such simple that is.

# data_labels = np.asarray(data_labels)
# To use and visualize images as color coded 2-D arrays, use the following piece of code for each element of the list.
# A = np.asarray(a[1])

# Ro visualize images as image (not an array) use the following code
plt.imshow(array_data[1])
plt.show()
```

در قطعه کد بعدی، ابتدا با دستوری از پایتون، داده‌ها به نسبت ۲۰ درصد و ۸۰ درصد به ترتیب‌برای اعتبارسنجی و آموزش تقسیم می‌شوند. سپس با فیلتر میانه نویز آن‌ها را کم می‌کنیم. این فیلتر برای نویز فلفل و نمک مناسب می‌باشد، هم‌چنین این‌جا چون پیکسل‌های عددها در کنار هم و با پیوستگی می‌باشند، به نظر می‌رسد اگر نویزی باشد، پیکسلی که به عدد مربوط است به احتمال زیاد وقتی توسط میانه اطرافش تعیین شود به درستی تصحیح شده و پیکسلی که واقعاً به background مربوط بوده‌است با این روش به دسته پس‌زمینه تصحیح می‌شود. هم‌چنین این روش مرزها را حفظ می‌کند و آن‌ها را نرم نمی‌کند که نسبت به فیلتر گاوسی دو بعدی یک مزیت محسوب می‌شود. چون کل هر عکس ۲۸ در ۲۸ پیکسل است، احتمال دادم از بین رفتن مرزها اثر مخربی در یادگیری داشته باشد و برای همین این روش (فیلتر میانه) را برای عکس‌ها انتخاب کردم. این فیلتر بر روی عکس‌های آموزش و یادگیری اعمال شد. سپس شئ scaler را برای standardize کردن داده‌ها (که در بالاتر اشاره شد) تعریف کردم و آن را بر روی داده آموزش برازش کردم. این تبدیل بر روی داده‌های تست و آموزش اعمال شد تا همه یک‌دست شوند و برای کاهش بعد و طبقه‌بندی آماده شوند.

```
# ############################### Training and Validation Partitioning (%20 and %80) ###############################

train_img, test_img, train_lbl, test_lbl = train_test_split(array_data, data_labels, test_size=1 / 5.0,
                                                            random_state=0)

# ################################################ Denoising ################################################

# Median Filtering

train_img = ndimage.median_filter(train_img, 5)
test_img = ndimage.median_filter(test_img, 5)

# plt.imshow(im_med)
# plt.show()

# ############################################ Dimension Reduction ############################################

# Reshaping

train_img = train_img.reshape([len(train_img), 784])
test_img = test_img.reshape([len(test_img), 784])

# Standardizing the data

scaler = StandardScaler()
# Fit on training set only.
scaler.fit(train_img)

pickle.dump(scaler, open('Scaler', 'wb'))
# Apply transform to both the training set and the test set.
train_img = scaler.transform(train_img)
test_img = scaler.transform(test_img)
```

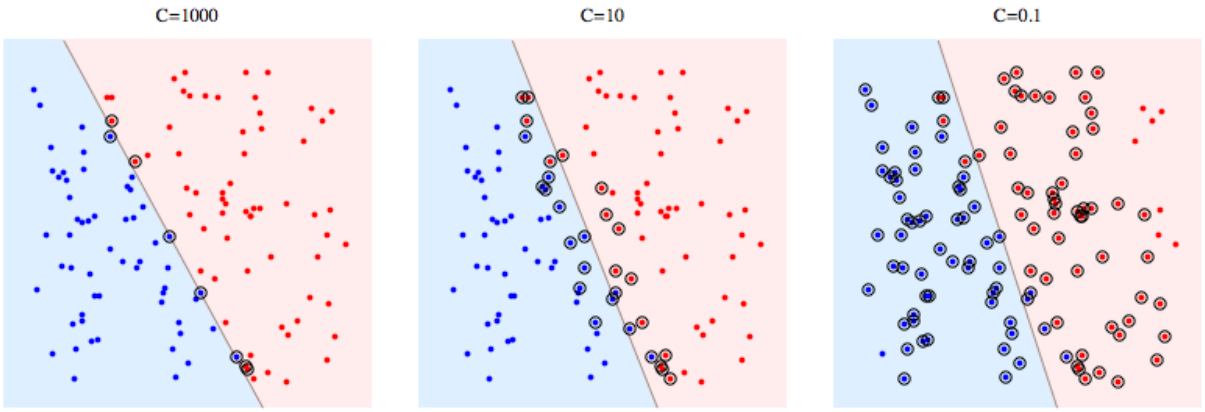
در قطعه‌کد زیر عملیات کاهش بعد انجام می‌شود. در قسمت اول، روش PCA با نگه‌داشتن ۵۰ درصد واریانس کل داده‌ها، جزهای اصلی را یافته و داده‌های آموزش و تست را بر روی آن تصویر می‌کند. (تعداد جزهای اصلی متناظر ۵۰ درصد واریانس کل نیز چاپ می‌شود.) در قسمت دوم همین عملیات برای LDA انجام می‌شود و همان‌طور که اشاره شد برای GDA این‌کار را انجام نمی‌دهیم. کد آن صرفاً به عنوان نمونه گذاشته‌است.

```
# ######################## PCA ########################

# Making an instance of the Model
pca = PCA(.5)

pca.fit(train_img)

train_img_pca = pca.transform(train_img)
test_img_pca = pca.transform(test_img)
print('number of principal components preserved after data reduction is:', pca.n_components_)

# ######################## LDA ########################

clf = LinearDiscriminantAnalysis()
clf.fit(train_img, train_lbl)

train_img_lda = clf.fit_transform(train_img, train_lbl)
test_img_lda = clf.fit_transform(test_img, test_lbl)

# ######################## GDA ########################

clf_gda = QuadraticDiscriminantAnalysis()
clf_gda.fit(train_img, train_lbl)

train_img_gda = ...
test_img_gda = ...
```

```
# ############################################## Classifiers ####################################################

target_names = ['class 0', 'class 1', 'class 2', 'class 3', 'class 4', 'class 5', 'class 6', 'class 7',
                'class 8', 'class 9']

# ############################################## SVM Classifier #################################################
c = 5
gamma = 0.05

decision_function = 'ovo'

if decision_function == 'ovr':
    one_versus_all = True
elif decision_function == 'ovo':
    one_versus_all = False

svm_model = svm.SVC(verbose=False, gamma=gamma, C=c, decision_function_shape=decision_function)
```

در قطعه‌کد بالا وارد قسمت طبقه‌بندها می‌شویم. نام هدف‌ها برای گرفتن گزارش مقادیر f1 score و recall و precision با دستور مربوطه آن‌ها است.

برای طبقه‌بند SVM، یک پارامتر c تعریف می‌کنیم که هرچه بیشتر باشد حاشیه‌های کوچکتری برای SVM برمی‌گزیند (شکل زیر).



γ پارامتر کرنل گاوسی زیر می‌باشد. مقدار کم آن متناظر واریانس زیاد بوده و حتی اگر فاصله دو داده زیاد باشد، در طبقه‌بندی همدیگر اثر دارند و برعکس هرچه زیادتر باشد معادل واریانس کمتر بوده که یعنی اثر وسیعی نمی‌گذارد (توان نمایی منفی بزرگ است).

$$K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2), \gamma > 0$$

در این مسأله مقادیر انتخاب شده در شکل مشاهده می‌شوند. یک عبارت هم برای تابع تصمیم بین one against all و one against one تعریف شده و با تعویض آن ادامه خروجی‌های کد تعویض می‌شوند. در آخر قطعه‌کد بالا نیز شئ مدل SVM ساخته می‌شود.

```
if selector == 0:  # PCA
    if one_versus_all:
        # Model Fitting
        svm_model.fit(train_img_pca, train_lbl)
        # Calculating the Score
        score = svm_model.score(test_img_pca, test_lbl)
        # Predicting Labels
        predicted_label = svm_model.predict(test_img_pca)
        # Calculating Confusion Matrix
        SVM_CM_ovr_PCA = confusion_matrix(test_lbl, predicted_label)
        print('score for PCA data reduction and SVM classifier (One versus all) is:', score)
        # Getting the summary for precision, recall, f1 score
        report = classification_report(test_lbl, predicted_label, target_names=target_names)
        # Getting the summary for accuracy and specificity
        TP, TN, FP, FN = multi_class_confusion_matrix_conversion(SVM_CM_ovr_PCA)
        TPTNFPFN = {'TP': TP, 'TN': TN, 'FP': FP, 'FN': FN}
        accuracy = (TP + TN)/(TP + TN + FP + FN)
        specificity = TN/(TN + FP)
        # Plotting ROC curves
        roc_curve_plot(test_lbl, predicted_label, n_classes, 'ROC curve for SVM-one versus all-PCA')
        # Saving the model for future use
        pickle.dump(svm_model, open('SVM-ov-all-PCA', 'wb'))
```

در این قطعه‌کد، selector که در ابتدای کل کد نیز تعریف شده، انتخاب می‌کند که روش کاهش بعد PCA باشد یا LDA. در حالت اول، اگر متغیر True one_versus_all شده باشد وارد این بلوک می‌شویم و مدل آموزش داده شده و دقت آن بر روی داده اعتبارسنجی محاسبه می‌شود. برچسب‌های آن نیز پیش‌بینی می‌شود و با دستوری که مشاهده می‌شود ماتریس درهم‌ریختگی SVM_CM_ovr_PCA که بیان‌گر این ماتریس برای حالت یک در مقابل همه و کاهش بعد PCA محاسبه می‌شود. دقت چاپ می‌شود. فایل متنی report حاوی ۳ کمیت ذکر شده در بالا بدست آمده، (در تصاویر زیر نیز این مقادیر قابل مشاهده خواهند بود.) سپس با تابعی که تبدیل ماتریس درهم‌ریختگی به tp, fp, tn, fn این مقادیر بدست می‌آیند. دو کمیتی که با دستور بدست نیامده بود محاسبه شده و در یک دیکشنری برای مشاهده در کنار هم ریخته شده، نمودار ROC با تابعی که بالاتر توضیح داده شد رسم شده و نهایتاً مدل برای استفاده‌های بعدی ذخیره می‌شود.

ادامه کد تکرار همین کارها برای حالتی که LDA برای کاهش بعد استفاده شده و حالتی که در SVM، طبقه‌بندی بر اساس یک در مقابل یک باشد انجام شده‌است و لذا از تکرار آن‌ها خودداری می‌کنیم. در ادامه گزارش مدل‌های مختلف آورده شده‌است. خود مدل‌ها نیز پیوست شده‌اند.

ماتریس درهم‌ریختگی مدل‌ها:

استفاده از PCA:

SVM برای one against one:

SVM_CM_ovo_PCA

	0	1	2	3	4	5	6	7	8	9
0	561	0	606	0	0	0	1	0	0	0
1	0	1258	67	1	2	0	0	2	1	0
2	0	0	1222	1	0	0	0	0	0	0
3	0	0	631	600	0	3	0	1	0	0
4	0	0	520	0	635	0	0	0	0	6
5	0	0	720	6	0	393	1	0	0	0
6	0	0	554	0	0	1	666	0	1	0
7	0	0	543	0	1	0	0	684	1	3
8	0	3	552	2	0	2	0	0	540	0
9	0	1	425	8	2	0	0	8	1	764

K-NN:

KNN_CM_PCA

	0	1	2	3	4	5	6	7	8	9
0	1157	1	3	1	0	1	3	1	0	1
1	0	1321	4	1	1	0	1	1	2	0
2	10	7	1169	11	1	1	8	8	8	0
3	2	3	12	1178	1	17	0	9	10	3
4	1	6	10	0	1108	2	6	1	2	25
5	6	1	5	19	5	1055	12	2	6	9
6	4	2	1	1	2	13	1198	0	1	0
7	0	10	6	1	9	1	0	1183	0	22
8	6	5	7	14	6	27	5	2	1020	7
9	4	2	4	6	17	4	0	37	8	1127

Random Forest:

RF_CM_PCA

	0	1	2	3	4	5	6	7	8	9
0	777	200	96	46	38	8	3	0	0	0
1	0	1189	70	39	18	11	4	0	0	0
2	2	4	767	295	114	29	7	5	0	0
3	0	0	9	752	342	87	36	9	0	0
4	0	1	1	11	743	278	89	33	5	0
5	0	1	7	19	114	833	128	16	2	0
6	0	3	11	14	45	202	939	8	0	0
7	0	2	2	6	20	53	148	959	41	1
8	0	2	4	15	33	88	196	347	413	1
9	0	0	2	4	12	30	66	148	427	520

استفاده از LDA:

SVM برای one against one:

KNN_CM_LDA | SVM_CM_ovo_LDA

	0	1	2	3	4	5	6	7	8	9
0	994	0	49	7	5	21	8	0	81	3
1	0	112	8	1058	3	5	0	0	143	2
2	14	11	1058	49	8	12	27	15	26	3
3	46	95	53	696	5	110	9	64	70	87
4	3	1	13	1	940	11	18	7	14	153
5	31	2	13	39	11	885	38	30	63	8
6	11	0	44	6	10	20	1100	0	24	7
7	3	1	18	74	24	15	0	1024	3	70
8	15	16	35	38	12	72	14	4	877	16
9	7	0	15	16	164	11	1	314	24	657

K-NN:

KNN_CM_LDA

	0	1	2	3	4	5	6	7	8	9
0	1069	0	34	8	3	26	12	0	16	0
1	0	112	10	931	1	6	1	0	269	1
2	35	21	987	42	12	9	57	16	40	4
3	87	216	39	604	4	108	12	68	77	20
4	3	2	14	2	871	19	16	4	10	220
5	51	14	11	43	25	835	34	35	64	8
6	22	3	39	3	12	32	1091	0	13	7
7	4	2	18	54	27	9	1	1028	5	84
8	26	29	27	29	11	70	12	5	865	25
9	12	4	10	13	256	16	2	275	30	591

Random Forest:

RF_CM_LDA

	0	1	2	3	4	5	6	7	8	9
0	287	265	201	100	45	129	137	3	1	0
1	0	1	211	813	227	50	13	10	6	0
2	0	4	345	499	252	84	24	12	3	0
3	2	3	95	442	428	142	65	48	10	0
4	0	0	2	13	356	400	218	91	64	17
5	3	5	19	54	203	566	210	50	9	1
6	1	1	10	22	47	194	925	18	3	1
7	0	0	5	11	57	105	132	863	46	13
8	0	4	8	29	50	99	194	273	442	0
9	0	0	9	11	25	76	119	381	429	159

گزارش مقادیر True Positive, True Negative, False Positive, False Negative مدل‌ها:

استفاده از PCA:

SVM برای one against one:

	0	1	2	3	4	5	6	7	8	9
0	561	1258	1222	600	635	393	666	684	540	764
1	10832	10665	6159	10747	10834	10874	10776	10757	10897	10782
2	0	4	4618	18	5	6	2	11	4	9
3	607	73	1	635	526	727	556	548	559	445

K-NN:

	0	1	2	3	4	5	6	7	8	9
0	1157	1321	1169	1178	1108	1055	1198	1183	1020	1127
1	10799	10632	10725	10711	10797	10814	10743	10707	10864	10724
2	33	37	52	54	42	66	35	61	37	67
3	11	10	54	57	53	65	24	49	79	82

Random Forest:

	0	1	2	3	4	5	6	7	8	9
0	777	1189	767	752	743	833	939	959	413	520
1	10830	10456	10575	10316	10103	10094	10101	10202	10426	10789
2	2	213	202	449	736	786	677	566	475	2
3	391	142	456	483	418	287	283	273	686	689

استفاده از LDA:

SVM برای one against one:

	0	1	2	3	4	5	6	7	8	9
0	994	112	1058	696	940	885	1100	1024	877	657
1	10702	10543	10529	9477	10597	10603	10663	10334	10453	10442
2	130	126	248	1288	242	277	115	434	448	349
3	174	1219	165	539	221	235	122	208	222	552

K-NN:

	0	1	2	3	4	5	6	7	8	9
0	1069	112	987	604	871	835	1091	1028	865	591
1	10592	10378	10575	9640	10488	10585	10631	10365	10377	10422
2	240	291	202	1125	351	295	147	403	524	369
3	99	1219	236	631	290	285	131	204	234	618

Random Forest:

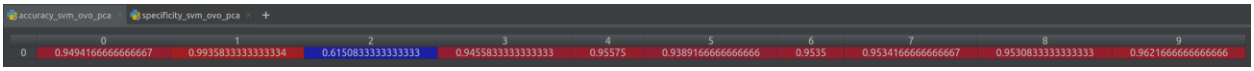
	0	1	2	3	4	5	6	7	8	9
0	287	1	345	442	356	566	925	863	442	159
1	10826	10387	10217	9213	9505	9601	9666	9882	10330	10759
2	6	282	560	1552	1334	1279	1112	886	571	32
3	881	1330	878	793	805	554	297	369	657	1050

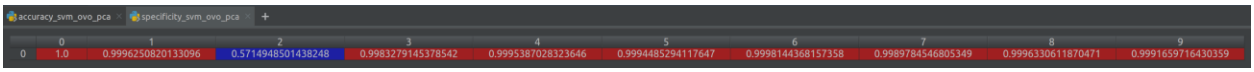
گزارش Accuracy, Precision, Recall, Specificity, F1 score مدل‌ها:

استفاده از PCA:

SVM برای one against one:

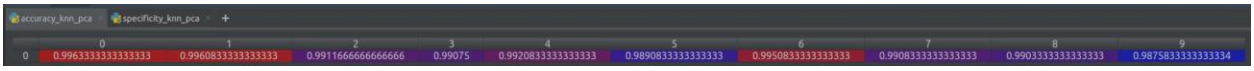
	precision	recall	f1-score	support
class 0	1.00	0.48	0.65	1168
class 1	1.00	0.95	0.97	1331
class 2	0.21	1.00	0.35	1223
class 3	0.97	0.49	0.65	1235
class 4	0.99	0.55	0.71	1161
class 5	0.98	0.35	0.52	1120
class 6	1.00	0.55	0.70	1222
class 7	0.98	0.56	0.71	1232
class 8	0.99	0.49	0.66	1099
class 9	0.99	0.63	0.77	1209
micro avg	0.61	0.61	0.61	12000
macro avg	0.91	0.60	0.67	12000
weighted avg	0.91	0.61	0.67	12000





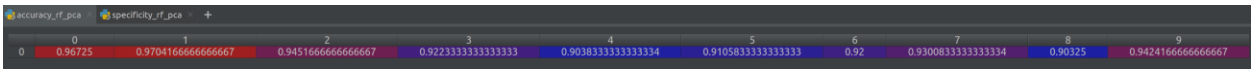
K-NN:

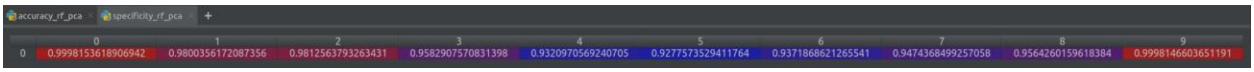
	precision	recall	f1-score	support
class 0	0.97	0.99	0.98	1168
class 1	0.97	0.99	0.98	1331
class 2	0.96	0.96	0.96	1223
class 3	0.96	0.95	0.96	1235
class 4	0.96	0.95	0.96	1161
class 5	0.94	0.94	0.94	1120
class 6	0.97	0.98	0.98	1222
class 7	0.95	0.96	0.96	1232
class 8	0.96	0.93	0.95	1099
class 9	0.94	0.93	0.94	1209
micro avg	0.96	0.96	0.96	12000
macro avg	0.96	0.96	0.96	12000
weighted avg	0.96	0.96	0.96	12000



Random Forest:

	precision	recall	f1-score	support
class 0	1.00	0.67	0.80	1168
class 1	0.85	0.89	0.87	1331
class 2	0.79	0.63	0.70	1223
class 3	0.63	0.61	0.62	1235
class 4	0.50	0.64	0.56	1161
class 5	0.51	0.74	0.61	1120
class 6	0.58	0.77	0.66	1222
class 7	0.63	0.78	0.70	1232
class 8	0.47	0.38	0.42	1099
class 9	1.00	0.43	0.60	1209
micro avg	0.66	0.66	0.66	12000
macro avg	0.70	0.65	0.65	12000
weighted avg	0.70	0.66	0.66	12000

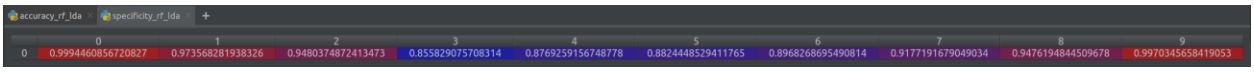




استفاده از LDA:

Random Forest:

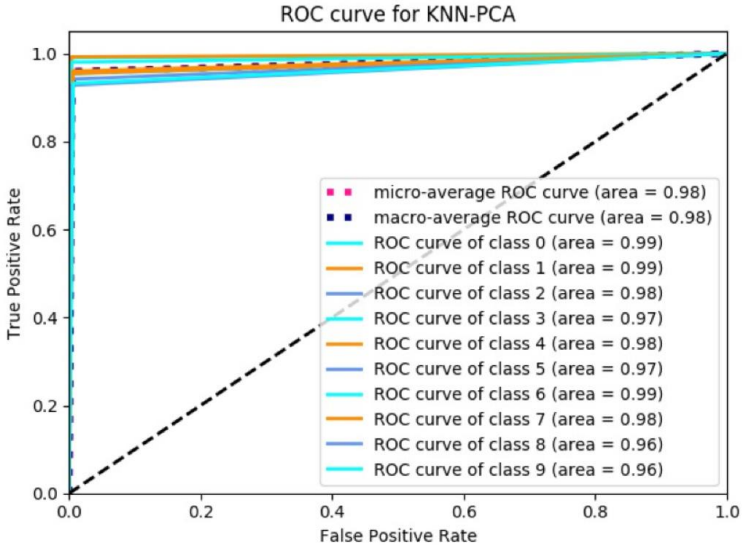
	precision	recall	f1-score	support
class 0	0.98	0.25	0.39	1168
class 1	0.00	0.00	0.00	1331
class 2	0.38	0.28	0.32	1223
class 3	0.22	0.36	0.27	1235
class 4	0.21	0.31	0.25	1161
class 5	0.31	0.51	0.38	1120
class 6	0.45	0.76	0.57	1222
class 7	0.49	0.70	0.58	1232
class 8	0.44	0.40	0.42	1099
class 9	0.83	0.13	0.23	1209
micro avg	0.37	0.37	0.37	12000
macro avg	0.43	0.37	0.34	12000
weighted avg	0.43	0.37	0.34	12000



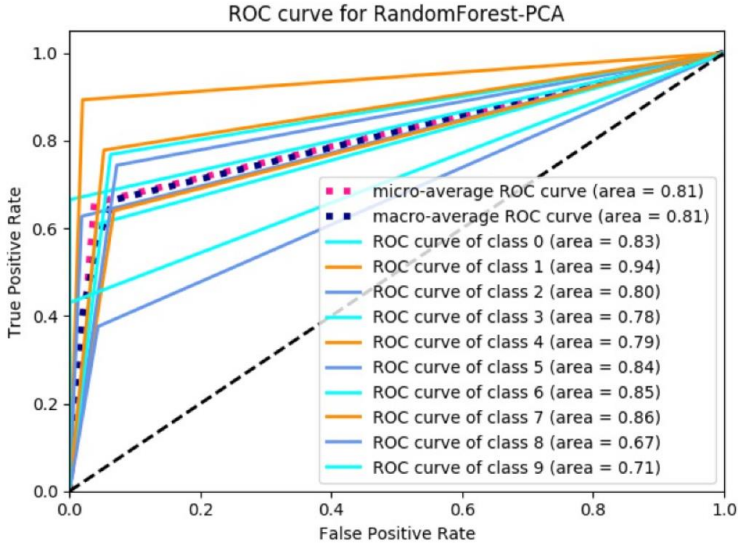
نمودار ROC مدل‌ها:

استفاده از PCA:

K-NN:

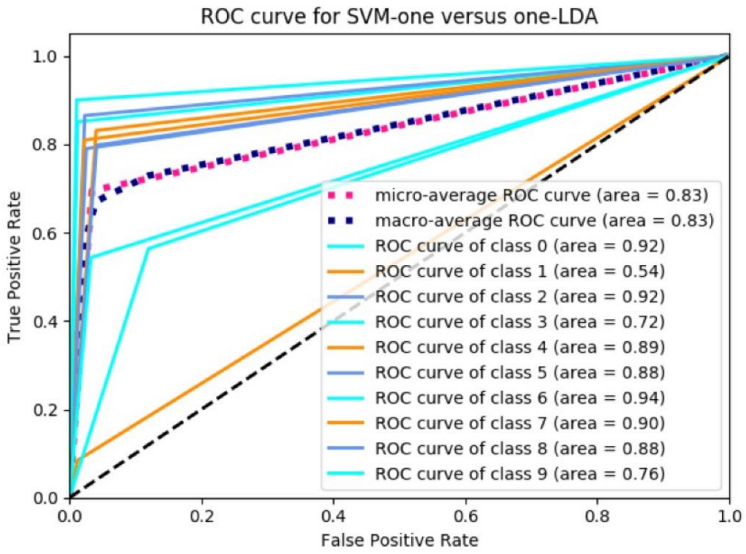


Random Forest:

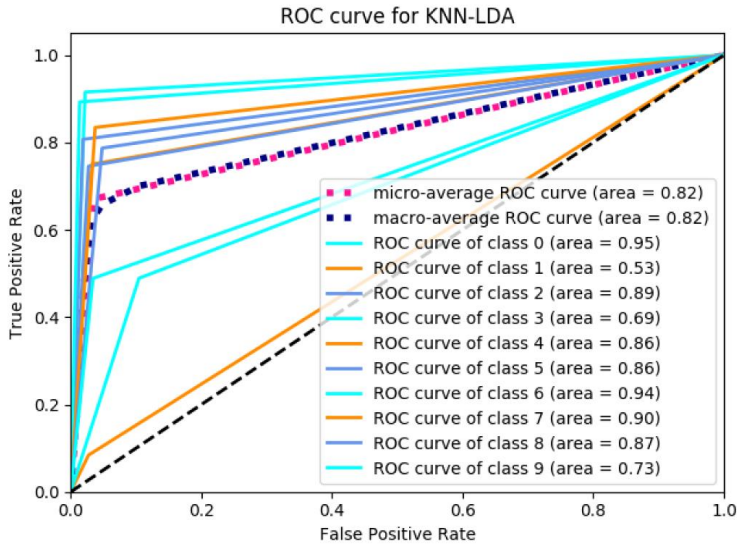


استفاده از LDA:

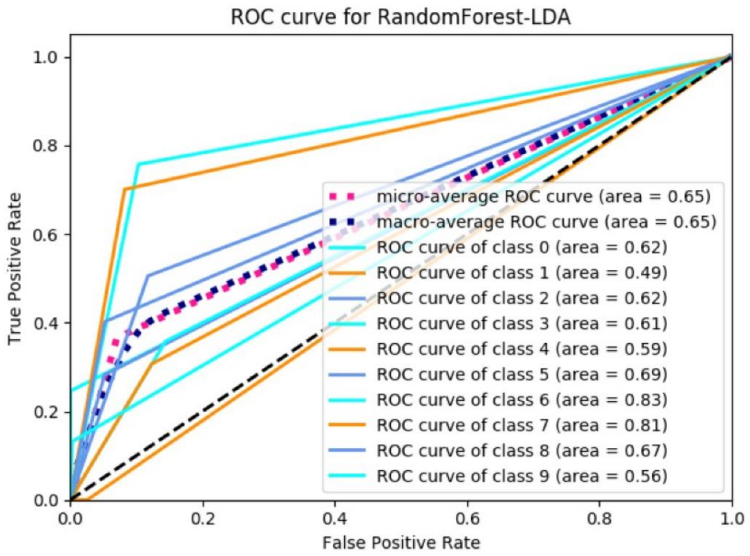
SVM برای one against one:



K-NN:



Random Forest:



نحوه محاسبه همه موارد بالا در کدها توضیح داده شد. مدل‌ها به ازای استفاده از کاهش بعد PCA و LDA آموزش داده شدند. همان‌طور که مشاهده می‌شود طبقه‌بند SVM با کاهش بعد PCA خوب کار نمی‌کند ولی وضعیت برای LDA بهتر است (با دقت در ماتریس در هم ریختگی) اما در تشخیص رقم ۲ با مشکل جدی مواجه است. اما روش‌های طبقه‌بندی K-NN و Random Forest با کاهش بعد PCA به خوبی کار می‌کنند. باید در نظر داشت که برای کاهش بعد PCA از حدود ۹۰ ویژگی استفاده شده است و با تعداد ویژگی‌های کمتر نتایج مطلوبی بدست نیامد. هم‌چنین برای SVM روش یک در برابر یک نتایج بهتری بدست داد و لذا از آوردن نتایج روش یک در برابر همه خودداری می‌کنیم. روش Random forest با تعداد ۵۰ تخمین‌زن پیاده‌سازی شد.

توجه: لطفاً در نمره‌دهی دقت با ارفاق عمل بفرمایید. من یک روز تمام همه موارد را بررسی کردم و تغییر دادم تا به دقت بهتری برسم ولی بهترین حالت موارد فوق بود که مدل همه نیز پیوست شده‌است. روش‌هایی که امتحان کردم، شامل موارد زیر بود:

فیلتر گاوسی پس از فیلتر میانه، هر فیلتر به تنهایی، استفاده نکردن از فیلتر، اعمال LDA بر روی خروجی PCA، نرمال کردن ورودی PCA و LDA و غیر نرمال کردن (برای PCA نرمال کردن اثر فاحشی داشت). عوض کردن کرنل SVM. یک مشکل بزرگ زمان آموزش طولانی SVM بود که هر بار حدود ۵۰ دقیقه به طول می‌انجامید. این باعث محدود شدن بررسی شد. همه این موارد اعمال شد ولی نتیجه به ۹۸ درصد که در کلاس حل تمرین گفته شد، برای همه طبقه‌بندها نرسید و لذا خواهشمندم در نمره دقت مقداری آسان بگیرید.

در ضمن با توجه به طولانی بودن زمان اجرا، کل قسمت طبقه‌بندی کامنت شده‌است و خواهشمندم برای اجرا آن را از حالت کامنت خارج بفرمایید.

با تشکر