محمدرضا عباسی ۹۸۴۲۲۱۰۱ جواب تمرین سوم

جواب سوال 1 و 2 و 3)

ماشین بردار پشتیبان (svm)

ماشین بردار پشتیبان SVM ، یک الگوریتم نظارت‌شده یادگیری ماشین است که هم برای مسائل طبقه‌بندی و هم مسائل رگرسیون قابل استفاده است؛ با این حال از آن بیشتر در مسائل طبقه‌بندی استفاده می‌شود. در الگوریتم SVM ، هر نمونه داده را به عنوان یک نقطه در فضایn-بعدی روی نمودار پراکندگی داده‌ها ترسیم کرده(n تعداد ویژگی‌هایی است که یک نمونه داده دارد) و مقدار هر ویژگی مربوط به داده‌ها، یکی از مؤلفه‌های مختصات نقطه روی نمودار را مشخص می‌کند. سپس، با ترسیم یک خط راست، داده‌های مختلف و متمایز از یکدیگر را دسته‌بندی می‌کند. ماشین بردار پشتیبان مرزی است که به بهترین شکل دسته‌های داده‌ها را از یکدیگر جدا می‌کند.
بردارهای پشتیبان به زبان ساده، مجموعه ای از نقاط در فضای n-بعدی داده ها هستند که مرز دسته ها را مشخص می کنند و مرزبندی و دسته بندی داده ها براساس آنها انجام می شود و با جابجایی یکی از آنها، خروجی دسته بندی ممکن است تغییر کند.

می خواهیم کلاسیفایر SVM را با استفاده از پکیج sklearn بر روی سه فیچر خواسته شده با استفاده از داده های آموزشی ترین کنیم:
می توانیم برای این کار از کرنل های مختلف SVM استفاده کنیم .در صورت استفاده نکردن از kernel به صورت پیش فرض 'rbf' انتخاب می شود.
(defult = 'rbf' .)

', defult = 'rbf {‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’} kernel

برای اطمینان از این موضوع ما یک بار بدون استفاده از کرنل و هم با استفاده از کرنل rbf جواب ها را مقایسه می کنیم.
خروجی در هر دو یکسان خواهد بود:

```
# Create a Radial basis function SVM classifier
svclassifier = svm.SVC()
svclassifier.fit(x_train, y)
y_train, y_test = tts( y , test_size = 0.4965, random_state = 0)
pred = svclassifier.predict(x_test)
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
[[  0 252   0   0]
 [  0 247   0   0]
 [  0 245   0   0]
 [  0 244   0   0]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       252
           1       0.25      1.00      0.40       247
           2       0.00      0.00      0.00       245
           3       0.00      0.00      0.00       244

    accuracy                           0.25       988
   macro avg       0.06      0.25      0.10       988
weighted avg       0.06      0.25      0.10       988
```

```
# Create a Radial basis function SVM classifier
svclassifier = svm.SVC(kernel='rbf')
svclassifier.fit(x_train, y)
y_train, y_test = tts( y , test_size = 0.4965, random_state = 0)
pred = svclassifier.predict(x_test)
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
[[  0 252   0   0]
 [  0 247   0   0]
 [  0 245   0   0]
 [  0 244   0   0]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       252
           1       0.25      1.00      0.40       247
           2       0.00      0.00      0.00       245
           3       0.00      0.00      0.00       244

    accuracy                           0.25       988
   macro avg       0.06      0.25      0.10       988
weighted avg       0.06      0.25      0.10       988
```

اکنون می خواهیم با استفاده از کرنل های متفاوت دیگر ('sigmoid' , 'poly' , 'linear') خروجی را مقایسه کنیم:

```
# Create a polynomial SVM classifier
svclassifier = svm.SVC(kernel='poly')
svclassifier.fit(x_train, y)
y_train, y_test = tts( y , test_size = 0.4965, random_state = 0)
pred = svclassifier.predict(x_test)
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
[[ 0   0   0 252]
 [ 0   0   0 247]
 [ 0   0   0 245]
 [ 0   0   0 244]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       252
           1       0.00      0.00      0.00       247
           2       0.00      0.00      0.00       245
           3       0.25      1.00      0.40       244

    accuracy                           0.25       988
   macro avg       0.06      0.25      0.10       988
weighted avg       0.06      0.25      0.10       988
```

```
# Create a linear SVM classifier
svclassifier = svm.SVC(kernel='linear')
svclassifier.fit(x_train, y)
y_train, y_test = tts( y , test_size = 0.4965, random_state = 0)
pred = svclassifier.predict(x_test)
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
[[ 0   0   0 252]
 [ 0   0   0 247]
 [ 0   0   0 245]
 [ 0   0   0 244]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       252
           1       0.00      0.00      0.00       247
           2       0.00      0.00      0.00       245
           3       0.25      1.00      0.40       244

    accuracy                           0.25       988
   macro avg       0.06      0.25      0.10       988
weighted avg       0.06      0.25      0.10       988
```

```
# Create a sigmoid SVM classifier
svclassifier = svm.SVC(kernel='sigmoid')
svclassifier.fit(x_train, y)
y_train, y_test = tts( y , test_size = 0.4965, random_state = 0)
pred = svclassifier.predict(x_test)
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
[[ 0  0 34 218]
 [ 0  0 21 226]
 [ 0  0 22 223]
 [ 0  0 27 217]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       252
           1       0.00      0.00      0.00       247
           2       0.21      0.09      0.13       245
           3       0.25      0.89      0.38       244

    accuracy                           0.24       988
   macro avg       0.11      0.24      0.13       988
weighted avg       0.11      0.24      0.13       988
```

از آنجایی که هر چه تعداد تشخیص‌های نادرست برنامه بیشتر باشد Recall آن کمتر می‌شود و هر چه مواردی که باید بدست می‌آمدن ولی پیش بینی نشدن بیشتر باشد Precision کاهش پیدا می‌کند پس ما به دنبال این هستیم که کرنلی را انتخاب کنیم که بیشترین Recall و Precision را دارا باشد.

همان طور که ملاحظه می کنید در کلاس ۳ کرنل های linear و poly بیشترین Recall و Precision را دارند و در کلاس ۲ بیشترین مقدار Recall و Precision متعلق به کرنل sigmoid می باشد.