# FinalSVM

January 10, 2020

[18]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os

from sklearn.svm import SVC
from sklearn.datasets import fetch_openml

mnist = fetch_openml('mnist_784', version=1, cache=True)

X = pd.DataFrame(mnist.data)
Y= pd.Series(mnist.target).astype('int').astype('category')
```

[19]:
```
cleanTestX = X.tail(2000)
cleanTestY = Y.tail(2000)

X = X.head(10000)
Y = Y.head(10000)
```

[20]:
```
print("X data")
print(X.shape)
# keep in mind 28  by 28 pixel values
print ("Y data")
print(Y.shape)
#y / output only needs one colum saying which class the image fell into
```

```
X data
(10000, 784)
Y data
(10000,)
```

[21]:
```
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(X, Y, test_size = 0.20)
```

[22]:
```
##  Testing out linear kernal
svcLinear = SVC(kernel= 'linear')
svcLinear.fit(X_train,y_train)
```

```
predicted_Linear =svcLinear.predict(cleanTestX)
from sklearn.metrics import classification_report, accuracy_score


print(classification_report(cleanTestY,predicted_Linear))
accuracy_score(cleanTestY, predicted_Linear)
```

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       207
           1       0.98      0.99      0.98       230
           2       0.92      0.88      0.90       198
           3       0.93      0.90      0.91       207
           4       0.91      0.95      0.93       194
           5       0.90      0.91      0.91       169
           6       0.93      0.96      0.94       202
           7       0.94      0.94      0.94       215
           8       0.90      0.91      0.91       187
           9       0.93      0.87      0.90       191

    accuracy                           0.93      2000
   macro avg       0.93      0.93      0.93      2000
weighted avg       0.93      0.93      0.93      2000
```

[22]: 0.9305

[23]:
```
## Testing Polynomial Kernel
svcPoly = SVC(kernel= 'poly' )
svcPoly.fit(X_train,y_train)

predicted_Poly =svcPoly.predict(cleanTestX)

print(classification_report(cleanTestY,predicted_Poly))
accuracy_score(cleanTestY, predicted_Poly)
```

```
              precision    recall  f1-score   support

           0       0.97      0.98      0.97       207
           1       0.99      1.00      0.99       230
           2       0.96      0.93      0.94       198
           3       0.99      0.97      0.98       207
           4       0.95      0.98      0.97       194
           5       0.94      0.96      0.95       169
           6       0.96      0.99      0.97       202
           7       0.97      0.97      0.97       215
           8       0.96      0.95      0.95       187
           9       0.99      0.95      0.97       191
```

```
    accuracy                           0.97      2000
   macro avg       0.97      0.97      0.97      2000
weighted avg       0.97      0.97      0.97      2000
```

[23]: 0.968

[24]:
```
## Testing RBF kernel
svcRBF = SVC(kernel= 'rbf' )
svcRBF.fit(X_train,y_train)

predicted_RBF =svcRBF.predict(cleanTestX)

print(classification_report(cleanTestY,predicted_RBF))
accuracy_score(cleanTestY, predicted_RBF)
```

```
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       207
           1       1.00      1.00      1.00       230
           2       0.95      0.94      0.95       198
           3       0.99      0.98      0.99       207
           4       0.96      0.97      0.97       194
           5       0.99      0.95      0.97       169
           6       0.95      0.99      0.97       202
           7       0.98      0.97      0.98       215
           8       0.95      0.97      0.96       187
           9       0.98      0.96      0.97       191

    accuracy                           0.97      2000
   macro avg       0.97      0.97      0.97      2000
weighted avg       0.97      0.97      0.97      2000
```

[24]: 0.9725

[25]:
```
## Testing Sigmoid kernel
svcSigmoid = SVC(kernel= 'sigmoid' )
svcSigmoid.fit(X_train,y_train)

predicted_Sigmoid =svcSigmoid.predict(cleanTestX)

print(classification_report(cleanTestY,predicted_Sigmoid))
accuracy_score(cleanTestY, predicted_Sigmoid)
```

```
              precision    recall  f1-score   support
```

```
              0       0.83      0.89      0.86       207
              1       0.95      0.97      0.96       230
              2       0.81      0.79      0.80       198
              3       0.88      0.82      0.84       207
              4       0.88      0.94      0.91       194
              5       0.72      0.77      0.74       169
              6       0.94      0.88      0.91       202
              7       0.93      0.92      0.93       215
              8       0.78      0.73      0.75       187
              9       0.87      0.85      0.86       191

    accuracy                           0.86      2000
   macro avg       0.86      0.86      0.86      2000
weighted avg       0.86      0.86      0.86      2000
```

[25]:
```
0.861
```

[26]:
```
### NOW TO FIND A MORE STRUCTURED APPROACH TO TUNING BY MAKING USE OF A GRID␣
↪SEARCH
```

[27]:
```
from sklearn.model_selection import GridSearchCV
```

[ ]:
```

```

[28]:
```
# defining parameter range
param_grid = {'C': [1,2,3],
              'gamma': [0.0000002, 0.0000003, 0.0000004, 0.00000044],
              'kernel': ['rbf']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

grid.fit(X_train, y_train)

predicted_yNew =grid.predict(X_val)
print(accuracy_score(y_val, predicted_yNew))
print("****** ------- ********")
print(classification_report(y_val,predicted_yNew))

print(grid.best_estimator_)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
[CV] C=1, gamma=2e-07, kernel=rbf …
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[CV] … C=1, gamma=2e-07, kernel=rbf, score=0.952, total=  18.0s
[CV] C=1, gamma=2e-07, kernel=rbf …
```

```
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:   18.0s remaining:    0.0s

[CV] … C=1, gamma=2e-07, kernel=rbf, score=0.961, total=  17.9s
[CV] C=1, gamma=2e-07, kernel=rbf …

[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:   35.9s remaining:    0.0s

[CV] … C=1, gamma=2e-07, kernel=rbf, score=0.963, total=  18.0s
[CV] C=1, gamma=2e-07, kernel=rbf …
[CV] … C=1, gamma=2e-07, kernel=rbf, score=0.953, total=  18.2s
[CV] C=1, gamma=2e-07, kernel=rbf …
[CV] … C=1, gamma=2e-07, kernel=rbf, score=0.954, total=  18.4s
[CV] C=1, gamma=3e-07, kernel=rbf …
[CV] … C=1, gamma=3e-07, kernel=rbf, score=0.956, total=  22.1s
[CV] C=1, gamma=3e-07, kernel=rbf …
[CV] … C=1, gamma=3e-07, kernel=rbf, score=0.966, total=  22.5s
[CV] C=1, gamma=3e-07, kernel=rbf …
[CV] … C=1, gamma=3e-07, kernel=rbf, score=0.969, total=  27.9s
[CV] C=1, gamma=3e-07, kernel=rbf …
[CV] … C=1, gamma=3e-07, kernel=rbf, score=0.960, total=  23.0s
[CV] C=1, gamma=3e-07, kernel=rbf …
[CV] … C=1, gamma=3e-07, kernel=rbf, score=0.960, total=  21.8s
[CV] C=1, gamma=4e-07, kernel=rbf …
[CV] … C=1, gamma=4e-07, kernel=rbf, score=0.959, total=  26.5s
[CV] C=1, gamma=4e-07, kernel=rbf …
[CV] … C=1, gamma=4e-07, kernel=rbf, score=0.969, total=  26.5s
[CV] C=1, gamma=4e-07, kernel=rbf …
[CV] … C=1, gamma=4e-07, kernel=rbf, score=0.973, total=  26.4s
[CV] C=1, gamma=4e-07, kernel=rbf …
[CV] … C=1, gamma=4e-07, kernel=rbf, score=0.960, total=  27.5s
[CV] C=1, gamma=4e-07, kernel=rbf …
[CV] … C=1, gamma=4e-07, kernel=rbf, score=0.961, total=  27.9s
[CV] C=1, gamma=4.4e-07, kernel=rbf …
[CV] … C=1, gamma=4.4e-07, kernel=rbf, score=0.960, total=  29.8s
[CV] C=1, gamma=4.4e-07, kernel=rbf …
[CV] … C=1, gamma=4.4e-07, kernel=rbf, score=0.971, total=  31.0s
[CV] C=1, gamma=4.4e-07, kernel=rbf …
[CV] … C=1, gamma=4.4e-07, kernel=rbf, score=0.973, total=  31.8s
[CV] C=1, gamma=4.4e-07, kernel=rbf …
[CV] … C=1, gamma=4.4e-07, kernel=rbf, score=0.961, total=  31.1s
[CV] C=1, gamma=4.4e-07, kernel=rbf …
[CV] … C=1, gamma=4.4e-07, kernel=rbf, score=0.962, total=  30.1s
[CV] C=2, gamma=2e-07, kernel=rbf …
[CV] … C=2, gamma=2e-07, kernel=rbf, score=0.958, total=  18.3s
[CV] C=2, gamma=2e-07, kernel=rbf …
[CV] … C=2, gamma=2e-07, kernel=rbf, score=0.968, total=  18.2s
[CV] C=2, gamma=2e-07, kernel=rbf …
[CV] … C=2, gamma=2e-07, kernel=rbf, score=0.966, total=  18.8s
[CV] C=2, gamma=2e-07, kernel=rbf …
```

```
[CV] … C=2, gamma=2e-07, kernel=rbf, score=0.958, total=  18.6s
[CV] C=2, gamma=2e-07, kernel=rbf …
[CV] … C=2, gamma=2e-07, kernel=rbf, score=0.959, total=  18.6s
[CV] C=2, gamma=3e-07, kernel=rbf …
[CV] … C=2, gamma=3e-07, kernel=rbf, score=0.961, total=  22.6s
[CV] C=2, gamma=3e-07, kernel=rbf …
[CV] … C=2, gamma=3e-07, kernel=rbf, score=0.971, total=  22.7s
[CV] C=2, gamma=3e-07, kernel=rbf …
[CV] … C=2, gamma=3e-07, kernel=rbf, score=0.969, total=  22.5s
[CV] C=2, gamma=3e-07, kernel=rbf …
[CV] … C=2, gamma=3e-07, kernel=rbf, score=0.960, total=  22.9s
[CV] C=2, gamma=3e-07, kernel=rbf …
[CV] … C=2, gamma=3e-07, kernel=rbf, score=0.961, total=  22.8s
[CV] C=2, gamma=4e-07, kernel=rbf …
[CV] … C=2, gamma=4e-07, kernel=rbf, score=0.962, total=  27.0s
[CV] C=2, gamma=4e-07, kernel=rbf …
[CV] … C=2, gamma=4e-07, kernel=rbf, score=0.973, total=  27.2s
[CV] C=2, gamma=4e-07, kernel=rbf …
[CV] … C=2, gamma=4e-07, kernel=rbf, score=0.974, total=  28.8s
[CV] C=2, gamma=4e-07, kernel=rbf …
[CV] … C=2, gamma=4e-07, kernel=rbf, score=0.963, total=  28.2s
[CV] C=2, gamma=4e-07, kernel=rbf …
[CV] … C=2, gamma=4e-07, kernel=rbf, score=0.961, total=  28.2s
[CV] C=2, gamma=4.4e-07, kernel=rbf …
[CV] … C=2, gamma=4.4e-07, kernel=rbf, score=0.964, total=  30.9s
[CV] C=2, gamma=4.4e-07, kernel=rbf …
[CV] … C=2, gamma=4.4e-07, kernel=rbf, score=0.975, total=  29.8s
[CV] C=2, gamma=4.4e-07, kernel=rbf …
[CV] … C=2, gamma=4.4e-07, kernel=rbf, score=0.974, total=  29.0s
[CV] C=2, gamma=4.4e-07, kernel=rbf …
[CV] … C=2, gamma=4.4e-07, kernel=rbf, score=0.962, total=  28.5s
[CV] C=2, gamma=4.4e-07, kernel=rbf …
[CV] … C=2, gamma=4.4e-07, kernel=rbf, score=0.961, total=  28.9s
[CV] C=3, gamma=2e-07, kernel=rbf …
[CV] … C=3, gamma=2e-07, kernel=rbf, score=0.961, total=  17.3s
[CV] C=3, gamma=2e-07, kernel=rbf …
[CV] … C=3, gamma=2e-07, kernel=rbf, score=0.969, total=  18.3s
[CV] C=3, gamma=2e-07, kernel=rbf …
[CV] … C=3, gamma=2e-07, kernel=rbf, score=0.965, total=  17.4s
[CV] C=3, gamma=2e-07, kernel=rbf …
[CV] … C=3, gamma=2e-07, kernel=rbf, score=0.960, total=  17.1s
[CV] C=3, gamma=2e-07, kernel=rbf …
[CV] … C=3, gamma=2e-07, kernel=rbf, score=0.960, total=  17.4s
[CV] C=3, gamma=3e-07, kernel=rbf …
[CV] … C=3, gamma=3e-07, kernel=rbf, score=0.961, total=  20.9s
[CV] C=3, gamma=3e-07, kernel=rbf …
[CV] … C=3, gamma=3e-07, kernel=rbf, score=0.971, total=  20.9s
[CV] C=3, gamma=3e-07, kernel=rbf …
```

```
[CV] … C=3, gamma=3e-07, kernel=rbf, score=0.971, total=  21.1s
[CV] C=3, gamma=3e-07, kernel=rbf …
[CV] … C=3, gamma=3e-07, kernel=rbf, score=0.961, total=  21.1s
[CV] C=3, gamma=3e-07, kernel=rbf …
[CV] … C=3, gamma=3e-07, kernel=rbf, score=0.962, total=  22.0s
[CV] C=3, gamma=4e-07, kernel=rbf …
[CV] … C=3, gamma=4e-07, kernel=rbf, score=0.964, total=  26.6s
[CV] C=3, gamma=4e-07, kernel=rbf …
[CV] … C=3, gamma=4e-07, kernel=rbf, score=0.973, total=  27.4s
[CV] C=3, gamma=4e-07, kernel=rbf …
[CV] … C=3, gamma=4e-07, kernel=rbf, score=0.974, total=  26.4s
[CV] C=3, gamma=4e-07, kernel=rbf …
[CV] … C=3, gamma=4e-07, kernel=rbf, score=0.963, total=  26.2s
[CV] C=3, gamma=4e-07, kernel=rbf …
[CV] … C=3, gamma=4e-07, kernel=rbf, score=0.961, total=  26.0s
[CV] C=3, gamma=4.4e-07, kernel=rbf …
[CV] … C=3, gamma=4.4e-07, kernel=rbf, score=0.964, total=  28.1s
[CV] C=3, gamma=4.4e-07, kernel=rbf …
[CV] … C=3, gamma=4.4e-07, kernel=rbf, score=0.975, total=  29.1s
[CV] C=3, gamma=4.4e-07, kernel=rbf …
[CV] … C=3, gamma=4.4e-07, kernel=rbf, score=0.973, total=  29.1s
[CV] C=3, gamma=4.4e-07, kernel=rbf …
[CV] … C=3, gamma=4.4e-07, kernel=rbf, score=0.962, total=  29.3s
[CV] C=3, gamma=4.4e-07, kernel=rbf …
[CV] … C=3, gamma=4.4e-07, kernel=rbf, score=0.961, total=  28.4s

[Parallel(n_jobs=1)]: Done  60 out of  60 | elapsed: 24.3min finished

0.975
****** ------- ********
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       187
           1       1.00      0.98      0.99       219
           2       0.98      0.99      0.99       202
           3       0.98      0.96      0.97       228
           4       0.98      0.97      0.98       195
           5       0.96      0.99      0.97       171
           6       0.97      0.96      0.97       199
           7       0.95      0.98      0.97       207
           8       0.98      0.98      0.98       201
           9       0.95      0.95      0.95       191

    accuracy                           0.97      2000
   macro avg       0.97      0.98      0.97      2000
weighted avg       0.98      0.97      0.98      2000

SVC(C=2, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=4.4e-07, kernel='rbf',
```

```
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

[29]:
```
# defining parameter range
param_gridPoly = {'degree': [2,3,4,5],
             'kernel': ['poly']}

gridP = GridSearchCV(SVC(), param_gridPoly, refit = True, verbose = 3)

gridP.fit(X_train, y_train)

predicted_yP =gridP.predict(X_val)
print(accuracy_score(y_val, predicted_yP))
print("****** ------- ********")
print(classification_report(y_val,predicted_yP))

print(gridP.best_estimator_)
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
[CV] degree=2, kernel=poly …
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[CV] … degree=2, kernel=poly, score=0.951, total=  14.6s
[CV] degree=2, kernel=poly …
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:   14.6s remaining:    0.0s
[CV] … degree=2, kernel=poly, score=0.956, total=  14.6s
[CV] degree=2, kernel=poly …
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:   29.2s remaining:    0.0s
[CV] … degree=2, kernel=poly, score=0.956, total=  14.6s
[CV] degree=2, kernel=poly …
[CV] … degree=2, kernel=poly, score=0.949, total=  14.6s
[CV] degree=2, kernel=poly …
[CV] … degree=2, kernel=poly, score=0.953, total=  14.7s
[CV] degree=3, kernel=poly …
[CV] … degree=3, kernel=poly, score=0.939, total=  17.2s
[CV] degree=3, kernel=poly …
[CV] … degree=3, kernel=poly, score=0.953, total=  17.1s
[CV] degree=3, kernel=poly …
[CV] … degree=3, kernel=poly, score=0.950, total=  17.1s
[CV] degree=3, kernel=poly …
[CV] … degree=3, kernel=poly, score=0.946, total=  17.1s
[CV] degree=3, kernel=poly …
[CV] … degree=3, kernel=poly, score=0.950, total=  17.2s
[CV] degree=4, kernel=poly …
[CV] … degree=4, kernel=poly, score=0.924, total=  20.6s
[CV] degree=4, kernel=poly …
```

```
[CV] … degree=4, kernel=poly, score=0.917, total=  20.6s
[CV] degree=4, kernel=poly …
[CV] … degree=4, kernel=poly, score=0.920, total=  20.5s
[CV] degree=4, kernel=poly …
[CV] … degree=4, kernel=poly, score=0.920, total=  20.8s
[CV] degree=4, kernel=poly …
[CV] … degree=4, kernel=poly, score=0.935, total=  20.6s
[CV] degree=5, kernel=poly …
[CV] … degree=5, kernel=poly, score=0.866, total=  24.4s
[CV] degree=5, kernel=poly …
[CV] … degree=5, kernel=poly, score=0.875, total=  24.4s
[CV] degree=5, kernel=poly …
[CV] … degree=5, kernel=poly, score=0.874, total=  25.5s
[CV] degree=5, kernel=poly …
[CV] … degree=5, kernel=poly, score=0.887, total=  25.5s
[CV] degree=5, kernel=poly …
[CV] … degree=5, kernel=poly, score=0.884, total=  24.2s

[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:  6.4min finished

0.955
****** ------- ********
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       187
           1       0.96      0.98      0.97       219
           2       0.98      0.97      0.97       202
           3       0.96      0.94      0.95       228
           4       0.94      0.94      0.94       195
           5       0.95      0.95      0.95       171
           6       0.96      0.95      0.96       199
           7       0.96      0.96      0.96       207
           8       0.96      0.94      0.95       201
           9       0.90      0.93      0.92       191

    accuracy                           0.95      2000
   macro avg       0.95      0.95      0.95      2000
weighted avg       0.96      0.95      0.96      2000

SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=2, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

[30]:
```
#On Test data not on validation
predicted_yPFinal =gridP.predict(cleanTestX)
print("Poly Trained results \n␣
 ↪"+classification_report(cleanTestY,predicted_yPFinal))
```

```
Poly Trained results
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       207
           1       1.00      1.00      1.00       230
           2       0.95      0.94      0.94       198
           3       1.00      0.97      0.98       207
           4       0.97      0.98      0.98       194
           5       0.96      0.95      0.96       169
           6       0.96      0.99      0.97       202
           7       0.98      0.97      0.97       215
           8       0.95      0.95      0.95       187
           9       0.98      0.95      0.97       191

    accuracy                           0.97      2000
   macro avg       0.97      0.97      0.97      2000
weighted avg       0.97      0.97      0.97      2000
```

[31]:
```
#On Test data not on validation
predicted_yRBGFinal =grid.predict(cleanTestX)
print("RBG Trained results \n␣
 ↪"+classification_report(cleanTestY,predicted_yRBGFinal))
```

```
RBG Trained results
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       207
           1       1.00      1.00      1.00       230
           2       0.95      0.97      0.96       198
           3       1.00      0.98      0.99       207
           4       0.96      0.97      0.97       194
           5       0.99      0.96      0.97       169
           6       0.97      0.99      0.98       202
           7       0.98      0.97      0.97       215
           8       0.96      0.97      0.97       187
           9       0.98      0.96      0.97       191

    accuracy                           0.98      2000
   macro avg       0.98      0.98      0.98      2000
weighted avg       0.98      0.98      0.98      2000
```

[ ]: