07_PW_Sol

November 7, 2018

1 TP 05 SVM

1.0.1 Date: 2017.10.18

1.0.2 Author: Christophe Gisler

Update to Python 3: Lorenz Rychener, Christophe Gisler

Imports

In [2]:
```
import _pickle as cPickle
import gzip
import time
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn import metrics

# get_ipython().magic(u'matplotlib inline')
```

2 Exercice 1

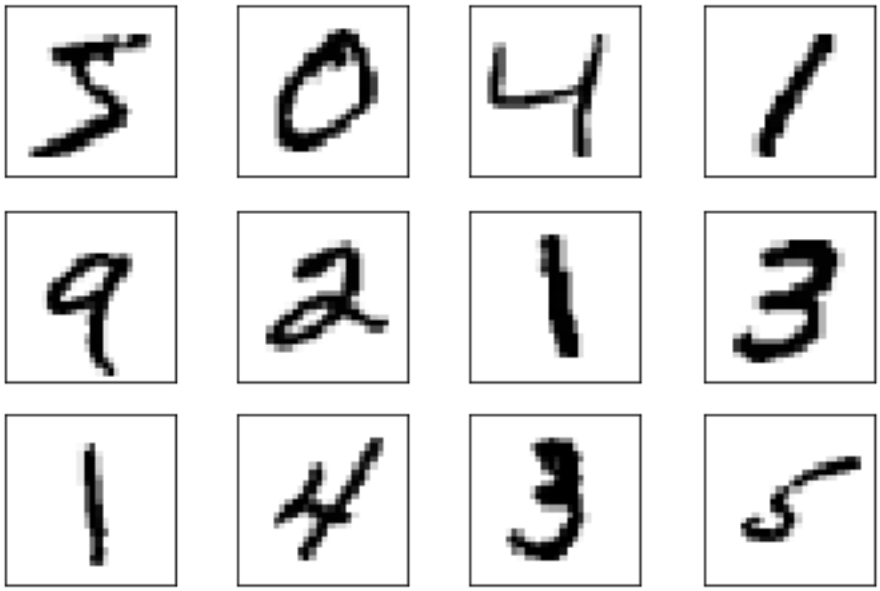
B) Load MNIST digit datasets (see http://deeplearning.net/tutorial/gettingstarted.html)

In [3]:
```
# Load training, validation and test datasets
f = gzip.open('mnist.pkl.gz', 'rb')
train_set, valid_set, test_set = cPickle.load(f, encoding='latin1')
f.close()

# Print the sizes of the various datasets
print("Training set: %d samples\nValidation set: %d samples (not used in the TP)\nTest
```

```
Training set: 50000 samples
Validation set: 10000 samples (not used in the TP)
Test set: 10000 samples
```

C) Visualize the images of some digits of the MNIST database

```
In [4]: # Plot a MNIST image (each image is represented as a 2-d numpy array).
        def plot_images(data_set, start_index, end_index, images_per_row):
            images = [np.reshape(f, (-1, 28)) for f in data_set[0][start_index:end_index] ]
            fig = plt.figure()
            n = end_index - start_index
            for i in range(n):
                ax = fig.add_subplot(n/images_per_row, images_per_row, i + 1)
                ax.matshow(images[start_index + i], cmap = matplotlib.cm.binary)
                plt.xticks(np.array([]))
                plt.yticks(np.array([]))
            plt.show()

        plot_images(train_set, 0, 12, 4)
```



C) Build final balanced training and test sets

```
In [5]: def select_samples(data_set, samples_per_class):
            Xs = []
            ys = []
            tot = 0
            ctrPerClass = {}
            for i in range(len(data_set[0])):
                X = data_set[0][i]
                y = data_set[1][i]
                ctrPerClass.setdefault(y, 0)
```

```
            if ctrPerClass[y] < samples_per_class:
                Xs.append(X)
                ys.append(y)
                ctrPerClass[y] = ctrPerClass[y] + 1
                tot += 1
            if tot >= 10 * samples_per_class:
                break
        return Xs, ys

    # Build final training set that will be composed of N samples for each class (i.e. dig
    #X_train = train_set[0]
    #y_train = train_set[1]
    train_samples_per_class = 200
    X_train, y_train = select_samples(train_set, train_samples_per_class)
    print("Final training set: %d samples (%d samples per class, 10 classes)" % (len(X_tra

    # Build final test set (just take test_set as it is)
    #X_test  = test_set[0]
    #y_test  = test_set[1]
    test_samples_per_class = 100
    X_test, y_test = select_samples(test_set, test_samples_per_class)
    print("Final test set: %d samples (%d samples per class, 10 classes)" % (len(X_test), t

    # Normalization (standardization) of training and test sets
    scaler = StandardScaler()
    scaler.fit(X_train)
    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)
```

```
Final training set: 2000 samples (200 samples per class, 10 classes)
Final test set: 1000 samples (100 samples per class, 10 classes)
```

3 Exercice 2 : Classification of digits based on raw pixel values using SVM and different kernels

Use SVM classifier with different kernels and parameters to recognize digits

In [6]:
```
# Set the number of folds for Cross-Validation and SVM tuning
cv_folds = 10

# Create different SVM classifier(s)
classifiers = {"svm-Linear" : GridSearchCV(SVC(kernel='linear', cache_size=7000), cv=cv
                       param_grid={"C": [1, 10, 100, 1000]}),
               "svm-Polynomial" : GridSearchCV(SVC(kernel='poly', degree=2, cache_size=
                       param_grid={"C": [1, 10, 100, 1000],
                                   "gamma": [1e-2,1e-3, 1e-4, 1e-5]}),
                "svm-RBF" : GridSearchCV(SVC(kernel='rbf', gamma=0.1, cache_size=7000)
```

```
                                param_grid={"C": [1, 10, 100, 1000],
                                            "gamma": [1e-2,1e-3, 1e-4, 1e-5]})
                        }
```

```
In [7]: for name, classifier in classifiers.items():
            t0 = time.time()
            print("%s:" % name)

            # Train SVM classifier on the training set
            classifier.fit(X_train, y_train)
            print("- Best classifier found using %d-fold CV (grid search) in %0.3fs:\n  %s" %

            # Predict number labels using the trained classifier on the test set
            y_pred = classifier.predict(X_test)

            # Print classification results with confusion matrix
            print("Classification report for classifier %s:\n%s\n" % (classifier, metrics.class
            print("\nConfusion matrix:\n%s" % metrics.confusion_matrix(y_test, y_pred))

            # Print computation time
            print("- Computation time: %0.3fs\n" % (time.time() - t0))
```

```
svm-Linear:
- Best classifier found using 10-fold CV (grid search) in 29.934s:
  SVC(C=1, cache_size=7000, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
Classification report for classifier GridSearchCV(cv=10, error_score='raise-deprecating',
       estimator=SVC(C=1.0, cache_size=7000, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'C': [1, 10, 100, 1000]}, pre_dispatch='2*n_jobs',
       refit=True, return_train_score='warn', scoring=None, verbose=0):
              precision    recall  f1-score   support

           0       0.95      0.95      0.95       100
           1       0.93      0.99      0.96       100
           2       0.78      0.90      0.83       100
           3       0.88      0.73      0.80       100
           4       0.82      0.89      0.85       100
           5       0.79      0.84      0.82       100
           6       0.95      0.83      0.89       100
           7       0.86      0.89      0.88       100
           8       0.88      0.77      0.82       100
           9       0.86      0.87      0.87       100
```

```
   micro avg       0.87      0.87      0.87      1000
   macro avg       0.87      0.87      0.87      1000
weighted avg       0.87      0.87      0.87      1000



Confusion matrix:
[[95  0  2  0  2  1  0  0  0  0]
 [ 0 99  0  0  0  0  0  0  1  0]
 [ 0  1 90  1  1  0  1  2  3  1]
 [ 0  2  6 73  0 12  1  3  2  1]
 [ 0  1  1  1 89  0  2  0  0  6]
 [ 1  1  1  3  2 84  0  3  3  2]
 [ 2  0  6  0  3  5 83  0  1  0]
 [ 0  1  2  1  2  1  0 89  0  4]
 [ 2  2  6  3  4  3  0  3 77  0]
 [ 0  0  2  1  6  0  0  3  1 87]]
- Computation time: 30.804s

svm-Polynomial:
- Best classifier found using 10-fold CV (grid search) in 249.503s:
  SVC(C=10, cache_size=7000, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=2, gamma=0.001, kernel='poly',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
Classification report for classifier GridSearchCV(cv=10, error_score='raise-deprecating',
       estimator=SVC(C=1.0, cache_size=7000, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=2, gamma='auto_deprecated',
  kernel='poly', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'C': [1, 10, 100, 1000], 'gamma': [0.01, 0.001, 0.0001, 1e-05]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0):
              precision    recall  f1-score   support

           0       0.95      0.91      0.93       100
           1       0.92      1.00      0.96       100
           2       0.95      0.88      0.91       100
           3       0.90      0.83      0.86       100
           4       0.85      0.83      0.84       100
           5       0.85      0.94      0.89       100
           6       0.97      0.92      0.94       100
           7       0.94      0.90      0.92       100
           8       0.80      0.86      0.83       100
           9       0.88      0.91      0.90       100
```

```
   micro avg       0.90      0.90      0.90      1000
   macro avg       0.90      0.90      0.90      1000
weighted avg       0.90      0.90      0.90      1000



Confusion matrix:
[[ 91   0   1   0   1   4   2   0   1   0]
 [  0 100   0   0   0   0   0   0   0   0]
 [  0   1  88   0   1   0   0   1   8   1]
 [  0   3   0  83   0   5   0   2   5   2]
 [  0   1   2   3  83   1   1   0   3   6]
 [  1   0   0   1   2  94   0   1   0   1]
 [  2   0   1   0   1   4  92   0   0   0]
 [  0   3   0   1   4   0   0  90   0   2]
 [  2   1   1   4   3   3   0   0  86   0]
 [  0   0   0   0   3   0   0   2   4  91]]
- Computation time: 250.706s

svm-RBF:
- Best classifier found using 10-fold CV (grid search) in 57751.087s:
  SVC(C=100, cache_size=7000, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
Classification report for classifier GridSearchCV(cv=10, error_score='raise-deprecating',
       estimator=SVC(C=1.0, cache_size=7000, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.1, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'C': [1, 10, 100, 1000], 'gamma': [0.01, 0.001, 0.0001, 1e-05]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0):
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       100
           1       0.99      0.99      0.99       100
           2       0.76      0.95      0.84       100
           3       0.92      0.80      0.86       100
           4       0.91      0.89      0.90       100
           5       0.87      0.89      0.88       100
           6       0.98      0.87      0.92       100
           7       0.88      0.87      0.87       100
           8       0.89      0.85      0.87       100
           9       0.87      0.89      0.88       100

   micro avg       0.90      0.90      0.90      1000
```

```
   macro avg       0.90      0.90      0.90      1000
weighted avg       0.90      0.90      0.90      1000



Confusion matrix:
[[97  0  2  0  1  0  0  0  0  0]
 [ 0 99  0  0  0  0  0  0  1  0]
 [ 0  0 95  0  0  0  0  1  3  1]
 [ 0  0  5 80  0  8  0  3  3  1]
 [ 0  0  1  0 89  0  1  1  1  7]
 [ 1  0  2  2  0 89  0  3  2  1]
 [ 2  0  8  0  1  2 87  0  0  0]
 [ 0  1  7  1  0  1  0 87  0  3]
 [ 2  0  3  3  3  2  1  1 85  0]
 [ 0  0  2  1  4  0  0  3  1 89]]
- Computation time: 57752.333s
```

4 Exercice 3 :) Impact of preprocessing and feature extraction

A) Preprocessing steps : binarization

```
In [8]: def binarization(X):
            return np.array(X > 0, dtype=int)
```

```
In [9]: # Set the number of folds for Cross-Validation and SVM tuning
        cv_folds = 10

        # Create different SVM classifier(s)
        classifiers = {"svm-Linear" : GridSearchCV(SVC(kernel='linear', cache_size=7000), cv=cv
                                param_grid={"C": [1, 10, 100, 1000]}),
                       "svm-Polynomial" : GridSearchCV(SVC(kernel='poly', degree=2, cache_size=
                                param_grid={"C": [1, 10, 100, 1000],
                                            "gamma": [1e-2,1e-3, 1e-4, 1e-5]}),
                        "svm-RBF" : GridSearchCV(SVC(kernel='rbf', gamma=0.1, cache_size=7000)
                                param_grid={"C": [1, 10, 100, 1000],
                                            "gamma": [1e-2,1e-3, 1e-4, 1e-5]})
                       }

        for name, classifier in classifiers.items():
            t0 = time.time()
            print ("%s:" % name)

            # Train SVM classifier on the training set
            classifier.fit(binarization(X_train), y_train)
            print ("- Best classifier found using %d-fold CV (grid search) in %0.3fs:\n  %s" %
```

```
            # Predict number labels using the trained classifier on the test set
            y_pred = classifier.predict(binarization(X_test))

            # Print classification results with confusion matrix
            print("Classification report for classifier %s:\n%s\n" % (classifier, metrics.class
            print("\nConfusion matrix:\n%s" % metrics.confusion_matrix(y_test, y_pred))

            # Print computation time
            print("- Computation time: %0.3fs\n" % (time.time() - t0))
```

```
svm-Linear:
- Best classifier found using 10-fold CV (grid search) in 34.102s:
  SVC(C=1, cache_size=7000, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
Classification report for classifier GridSearchCV(cv=10, error_score='raise-deprecating',
       estimator=SVC(C=1.0, cache_size=7000, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'C': [1, 10, 100, 1000]}, pre_dispatch='2*n_jobs',
       refit=True, return_train_score='warn', scoring=None, verbose=0):
              precision    recall  f1-score   support

           0       0.94      0.94      0.94       100
           1       0.93      1.00      0.97       100
           2       0.84      0.92      0.88       100
           3       0.89      0.78      0.83       100
           4       0.90      0.86      0.88       100
           5       0.80      0.86      0.83       100
           6       0.94      0.89      0.91       100
           7       0.85      0.89      0.87       100
           8       0.89      0.77      0.82       100
           9       0.83      0.88      0.85       100

   micro avg       0.88      0.88      0.88      1000
   macro avg       0.88      0.88      0.88      1000
weighted avg       0.88      0.88      0.88      1000



Confusion matrix:
[[ 94   0   1   0   1   1   3   0   0   0]
 [  0 100   0   0   0   0   0   0   0   0]
 [  0   2  92   0   1   0   0   1   3   1]
```

```
 [  0   1   1  78   0  13   1   3   2   1]
 [  0   0   0   0  86   0   2   1   1  10]
 [  1   1   2   3   1  86   0   2   2   2]
 [  2   0   5   0   0   3  89   0   1   0]
 [  0   2   5   0   1   1   0  89   0   2]
 [  3   1   3   5   3   3   0   3  77   2]
 [  0   0   0   2   3   0   0   6   1  88]]
- Computation time: 35.152s

svm-Polynomial:
- Best classifier found using 10-fold CV (grid search) in 249.934s:
  SVC(C=10, cache_size=7000, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=2, gamma=0.01, kernel='poly',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
Classification report for classifier GridSearchCV(cv=10, error_score='raise-deprecating',
       estimator=SVC(C=1.0, cache_size=7000, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=2, gamma='auto_deprecated',
  kernel='poly', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'C': [1, 10, 100, 1000], 'gamma': [0.01, 0.001, 0.0001, 1e-05]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0):
              precision    recall  f1-score   support

           0       0.93      0.97      0.95       100
           1       0.96      1.00      0.98       100
           2       0.92      0.92      0.92       100
           3       0.94      0.81      0.87       100
           4       0.89      0.91      0.90       100
           5       0.85      0.90      0.87       100
           6       0.96      0.92      0.94       100
           7       0.86      0.93      0.89       100
           8       0.89      0.81      0.85       100
           9       0.86      0.89      0.88       100

   micro avg       0.91      0.91      0.91      1000
   macro avg       0.91      0.91      0.91      1000
weighted avg       0.91      0.91      0.91      1000



Confusion matrix:
[[ 97   0   0   0   0   2   1   0   0   0]
 [  0 100   0   0   0   0   0   0   0   0]
 [  0   0  92   0   1   0   1   2   3   1]
 [  0   1   1  81   0  10   1   3   2   1]
```

```
 [  0   0   0   0  91   0   1   1   0   7]
 [  2   0   1   1   0  90   0   3   1   2]
 [  2   0   2   0   1   2  92   0   1   0]
 [  0   2   1   1   2   0   0  93   0   1]
 [  3   1   3   3   3   2   0   2  81   2]
 [  0   0   0   0   4   0   0   4   3  89]]
- Computation time: 250.904s

svm-RBF:
- Best classifier found using 10-fold CV (grid search) in 193.253s:
  SVC(C=10, cache_size=7000, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.01, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
Classification report for classifier GridSearchCV(cv=10, error_score='raise-deprecating',
       estimator=SVC(C=1.0, cache_size=7000, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.1, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'C': [1, 10, 100, 1000], 'gamma': [0.01, 0.001, 0.0001, 1e-05]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0):
              precision    recall  f1-score   support

           0       0.97      0.98      0.98       100
           1       0.99      1.00      1.00       100
           2       0.92      0.93      0.93       100
           3       0.92      0.85      0.89       100
           4       0.94      0.90      0.92       100
           5       0.87      0.93      0.90       100
           6       0.96      0.92      0.94       100
           7       0.89      0.93      0.91       100
           8       0.88      0.84      0.86       100
           9       0.87      0.92      0.89       100

   micro avg       0.92      0.92      0.92      1000
   macro avg       0.92      0.92      0.92      1000
weighted avg       0.92      0.92      0.92      1000



Confusion matrix:
[[ 98   0   0   0   0   1   1   0   0   0]
 [  0 100   0   0   0   0   0   0   0   0]
 [  0   0  93   0   1   0   1   1   3   1]
 [  0   0   0  85   0   8   1   3   2   1]
 [  0   0   0   0  90   0   1   1   1   7]
```

```
 [  0   0   1   2   0  93   0   2   1   1]
 [  2   0   2   0   1   2  92   0   1   0]
 [  0   1   3   1   0   0   0  93   0   2]
 [  1   0   2   4   2   3   0   2  84   2]
 [  0   0   0   0   2   0   0   3   3  92]]
- Computation time: 194.634s
```

B) Feature extraction steps : Vertical projections

```
In [10]: def vertical_projection(X):
             images = np.array([np.reshape(f, (-1, 28)).sum(0) for f in X])
             return images

In [11]: # Set the number of folds for Cross-Validation and SVM tuning
         cv_folds = 10
         # Create different SVM classifier(s)
         classifiers = {"svm-Linear" : GridSearchCV(SVC(kernel='linear', cache_size=7000, max_
                                param_grid={"C": [0.001, 0.01, 0.1, 1]}),
                        "svm-RBF" : GridSearchCV(SVC(kernel='rbf', gamma=0.1, cache_size=7000
                                param_grid={"C": [0.001, 0.01, 0.1, 1],
                                            "gamma": [1e-2,1e-3, 1e-4, 1e-5]})
                        }

In [12]: for name, classifier in classifiers.items():
             t0 = time.time()
             print("%s:" % name)

             # Train SVM classifier on the training set
             classifier.fit(vertical_projection(X_train), y_train)
             print("- Best classifier found using %d-fold CV (grid search) in %0.3fs:\n  %s" %

             # Predict number labels using the trained classifier on the test set
             y_pred = classifier.predict(vertical_projection(X_test))

             # Print classification results with confusion matrix
             print("Classification report for classifier %s:\n%s\n" % (classifier, metrics.cla
             print("\nConfusion matrix:\n%s" % metrics.confusion_matrix(y_test, y_pred))

             # Print computation time
             print("- Computation time: %0.3fs\n" % (time.time() - t0))

svm-Linear:


/Users/gislerc/.virtualenvs/ml/lib/python3.6/site-packages/sklearn/svm/base.py:244: Convergence
  % self.max_iter, ConvergenceWarning)
```

```
- Best classifier found using 10-fold CV (grid search) in 2.307s:
  SVC(C=0.001, cache_size=7000, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=1000, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
Classification report for classifier GridSearchCV(cv=10, error_score='raise-deprecating',
       estimator=SVC(C=1.0, cache_size=7000, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=1000, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'C': [0.001, 0.01, 0.1, 1]}, pre_dispatch='2*n_jobs',
       refit=True, return_train_score='warn', scoring=None, verbose=0):
              precision    recall  f1-score   support

           0       0.43      0.56      0.48       100
           1       0.55      0.94      0.69       100
           2       0.25      0.13      0.17       100
           3       0.24      0.16      0.19       100
           4       0.37      0.42      0.39       100
           5       0.40      0.10      0.16       100
           6       0.24      0.26      0.25       100
           7       0.39      0.31      0.34       100
           8       0.32      0.39      0.35       100
           9       0.31      0.40      0.35       100

   micro avg       0.37      0.37      0.37      1000
   macro avg       0.35      0.37      0.34      1000
weighted avg       0.35      0.37      0.34      1000



Confusion matrix:
[[56  2  6  0 19  0 13  1  3  0]
 [ 0 94  0  0  2  0  0  0  2  2]
 [16  5 13  8 10  8  7 10 12 11]
 [ 4 14  8 16  4  1 10 17  7 19]
 [13  7  3  4 42  1  5  3  3 19]
 [15  9  8  2  7 10 19  4 17  9]
 [15  5  4  0  8  4 26  2 30  6]
 [ 0 19  6 22  3  0  5 31  3 11]
 [ 9  4  4  5  2  1 19  3 39 14]
 [ 3 12  0  9 16  0  4  9  7 40]]
- Computation time: 2.378s

svm-RBF:
- Best classifier found using 10-fold CV (grid search) in 12.534s:
  SVC(C=1, cache_size=7000, class_weight=None, coef0=0.0,
```

```
  decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
  max_iter=1000, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
Classification report for classifier GridSearchCV(cv=10, error_score='raise-deprecating',
       estimator=SVC(C=1.0, cache_size=7000, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.1, kernel='rbf',
  max_iter=1000, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'C': [0.001, 0.01, 0.1, 1], 'gamma': [0.01, 0.001, 0.0001, 1e-05]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0):
              precision    recall  f1-score   support

           0       0.54      0.73      0.62       100
           1       0.63      0.97      0.76       100
           2       0.37      0.26      0.31       100
           3       0.41      0.29      0.34       100
           4       0.45      0.57      0.50       100
           5       0.45      0.19      0.27       100
           6       0.40      0.53      0.46       100
           7       0.57      0.45      0.50       100
           8       0.48      0.32      0.38       100
           9       0.45      0.55      0.50       100

   micro avg       0.49      0.49      0.49      1000
   macro avg       0.47      0.49      0.46      1000
weighted avg       0.47      0.49      0.46      1000



Confusion matrix:
[[73  0  3  0  7  0 11  3  2  1]
 [ 0 97  1  0  1  0  0  0  0  1]
 [14  3 26 12  3 12  6  8  8  8]
 [ 6 13 10 29  8  1  9  6  4 14]
 [ 4  6  1  5 57  2  8  0  1 16]
 [17  5 14  2  8 19 14  4 11  6]
 [ 8  5  5  1 11  1 53  7  5  4]
 [ 0 16  5 10 10  1  2 45  2  9]
 [12  3  4  8  4  6 20  3 32  8]
 [ 1  7  1  3 19  0  9  3  2 55]]
- Computation time: 12.631s
```