Group exercise 2a

Loïc Rosset, Nanae Aubry, Kilian Ruchti, Lionel Ieri

- Use the provided training set to build your SVM.
- Apply the trained SVM to classify the test set.
- Investigate at least two different kernels and optimize the SVM parameters by means of cross-validation.

Import libraries

Reference :

- Cross validation from scikit-learn (https://scikit-learn.org/stable/modules/cross_validation.html)
- Grid-Search Cross Validation (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)

In [1]:

```
import csv
#numpy and panda for data structure
import numpy as np
import pandas as pd
#sklearn for svm and for cross validation
from sklearn import svm
from sklearn import metrics
from sklearn.model_selection import GridSearchCV
```

Read the provided dataset

In [2]:

```
#path to dataset
mnist_train = "./../dataset/csv/mnist_train.csv"
mnist_test = "./../dataset/csv/mnist_test.csv"
```

In [3]:

```
def read_data(filename):
    with open(filename, 'r') as csvfile:
        reader = csv.reader(csvfile)
        data = list(reader)
    #data into numpy array
    matrix = np.array(data, dtype = int)
    samples = matrix[:,1:]
    labels = matrix[:,0]
    return samples, labels
```

In [4]:

```
# Load the training and the test set
training_data, training_labels = read_data(mnist_train)
test_data, test_labels = read_data(mnist_test)
```

Computing the following code (section SVM form Scikit-learn) with the whole training dataset takes a while so here is, if needed for testing, a random sample of it :

In [5]:

```
random_ids = np.random.randint(0,training_data.shape[0],1000)
sample_data = training_data[random_ids]
sample_labels = training_labels[random_ids]
```

SVM from Scikit-learn

Reference : A practical guide to support vector classification (https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf)

SVM trained on a cross-validation of 5 folds and parameters found with Grid-search

In [6]:

```
#parameters sequences of C and gamma to be tested
c_seq = [pow(2,x) for x in range(-5, 15, 4)]
gamma_seq = [pow(2,x) for x in range(-15, 3, 4)]
```

In [7]:

```
parameters = {'kernel':('rbf','linear'), 'C':c_seq, 'gamma':gamma_seq}
nb_folds = 3
#Support Vector Classification
svc = svm.SVC(max_iter=100, cache_size=1000)
#Cross-validation
svmOptimized = GridSearchCV(svc, parameters, cv=nb_folds, verbose=10, n_jobs=10)
svmOptimized.fit(training_data, training_labels)
scores = pd.DataFrame.from_dict(svmOptimized.cv_results_)
print("--> Best parameters : ", svmOptimized.best_params_)
scores[['param_C', 'param_gamma', 'param_kernel', 'mean_test_score', 'std_test_score',
'rank_test_score']]
```

```
Fitting 3 folds for each of 50 candidates, totalling 150 fits

[Parallel(n_jobs=10)]: Using backend LokyBackend with 10 concurrent worker
s.
C:\ProgramData\Anaconda3\lib\site-packages\joblib\externals\loky\process_e
xecutor.py:706: UserWarning: A worker stopped while some jobs were given t
o the executor. This can be caused by a too short worker timeout or by a m
emory leak.
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=10)]: Done   5 tasks      | elapsed:  6.0min
[Parallel(n_jobs=10)]: Done  12 tasks      | elapsed: 12.4min
[Parallel(n_jobs=10)]: Done  21 tasks      | elapsed: 18.9min
[Parallel(n_jobs=10)]: Done  30 tasks      | elapsed: 25.9min
[Parallel(n_jobs=10)]: Done  41 tasks      | elapsed: 34.6min
[Parallel(n_jobs=10)]: Done  52 tasks      | elapsed: 42.8min
[Parallel(n_jobs=10)]: Done  65 tasks      | elapsed: 53.6min
[Parallel(n_jobs=10)]: Done  78 tasks      | elapsed: 62.3min
[Parallel(n_jobs=10)]: Done  93 tasks      | elapsed: 72.9min
[Parallel(n_jobs=10)]: Done 108 tasks      | elapsed: 85.6min
[Parallel(n_jobs=10)]: Done 125 tasks      | elapsed: 99.6min
[Parallel(n_jobs=10)]: Done 147 out of 150 | elapsed: 113.6min remaining:
2.3min
[Parallel(n_jobs=10)]: Done 150 out of 150 | elapsed: 114.8min finished

--> Best parameters :  {'C': 0.03125, 'gamma': 3.0517578125e-05, 'kernel':
'rbf'}

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\_base.py:231: Conve
rgenceWarning: Solver terminated early (max_iter=100).  Consider pre-proce
ssing your data with StandardScaler or MinMaxScaler.
  % self.max_iter, ConvergenceWarning)
```

Out[7]:

	param_C	param_gamma	param_kernel	mean_test_score	std_test_score	rank_test_score
0	0.03125	3.05176e-05	rbf	0.841167	0.020858	1
1	0.03125	3.05176e-05	linear	0.719783	0.007997	3
2	0.03125	0.000488281	rbf	0.285150	0.007569	29
3	0.03125	0.000488281	linear	0.719783	0.007997	3
4	0.03125	0.0078125	rbf	0.099250	0.000041	36
5	0.03125	0.0078125	linear	0.719783	0.007997	3
6	0.03125	0.125	rbf	0.099150	0.000000	41
7	0.03125	0.125	linear	0.719783	0.007997	3
8	0.03125	2	rbf	0.099150	0.000000	41
9	0.03125	2	linear	0.719783	0.007997	3
10	0.5	3.05176e-05	rbf	0.799767	0.043498	2
11	0.5	3.05176e-05	linear	0.719783	0.007997	3
12	0.5	0.000488281	rbf	0.286333	0.007463	28
13	0.5	0.000488281	linear	0.719783	0.007997	3
14	0.5	0.0078125	rbf	0.099250	0.000041	36
15	0.5	0.0078125	linear	0.719783	0.007997	3
16	0.5	0.125	rbf	0.099150	0.000000	41
17	0.5	0.125	linear	0.719783	0.007997	3
18	0.5	2	rbf	0.099150	0.000000	41
19	0.5	2	linear	0.719783	0.007997	3
20	8	3.05176e-05	rbf	0.253583	0.008082	30
21	8	3.05176e-05	linear	0.719783	0.007997	3
22	8	0.000488281	rbf	0.184983	0.004821	33
23	8	0.000488281	linear	0.719783	0.007997	3
24	8	0.0078125	rbf	0.099250	0.000041	36
25	8	0.0078125	linear	0.719783	0.007997	3
26	8	0.125	rbf	0.099150	0.000000	41
27	8	0.125	linear	0.719783	0.007997	3
28	8	2	rbf	0.099150	0.000000	41
29	8	2	linear	0.719783	0.007997	3
30	128	3.05176e-05	rbf	0.253583	0.008082	30
31	128	3.05176e-05	linear	0.719783	0.007997	3
32	128	0.000488281	rbf	0.184983	0.004821	33
33	128	0.000488281	linear	0.719783	0.007997	3
34	128	0.0078125	rbf	0.099250	0.000041	36
35	128	0.0078125	linear	0.719783	0.007997	3
36	128	0.125	rbf	0.099150	0.000000	41

	param_C	param_gamma	param_kernel	mean_test_score	std_test_score	rank_test_score
37	128	0.125	linear	0.719783	0.007997	3
38	128	2	rbf	0.099150	0.000000	41
39	128	2	linear	0.719783	0.007997	3
40	2048	3.05176e-05	rbf	0.253583	0.008082	30
41	2048	3.05176e-05	linear	0.719783	0.007997	3
42	2048	0.000488281	rbf	0.184983	0.004821	33
43	2048	0.000488281	linear	0.719783	0.007997	3
44	2048	0.0078125	rbf	0.099250	0.000041	36
45	2048	0.0078125	linear	0.719783	0.007997	3
46	2048	0.125	rbf	0.099150	0.000000	41
47	2048	0.125	linear	0.719783	0.007997	3
48	2048	2	rbf	0.099150	0.000000	41
49	2048	2	linear	0.719783	0.007997	3

Classify test set with trained SVM

In [8]:

```
labels_pred = svmOptimized.predict(test_data)
print("Accuracy of svm classifier with optimized parameter values: ", metrics.accuracy_
score(test_labels, labels_pred)*100,"%")
```

```
Accuracy of svm classifier with optimized parameter values:  83.86 %
```