Exercise 2. Support Vector Machine - Python Implementation Pattern Recognition

Sanchez Roberto
Roth Markus
Nikodemski Alexandre

Université de Fribourg

May 24, 2016

1 Description

With this exercise we want to build the foundation for the Pattern Recognition Framework. To do this we should still work on the MNIST dataset, with which we should be familiar by now. In this exercise we should aim to improve the recognition rate on the MNIST dataset using SVM.

The projects contains the following folders:

1 **Data:** Contains the MNIST handwritten digit database.

2 **Individual approaches:** Contains the individual collaboration approach of each member in the work-group.

3 **MLP with Theano:** The approach for MLP using the Theano library, this is an adaptation from: http://deeplearning.net/tutorial/mlp.html

4 **results_ex2 and results_ex3** The results for each exercise.

5 **Transformation:** This a feature extraction method proposed for the group, in order to decrease the elalpsed time of the each recognition.

The exercise 2 (SVM classifier) is executed by the script *ex2.py*. The dependent scripts are:

data_point: Opens the MNIST dataset form data.

classifiers: Create a kernel configuration for the classifier in order to perform for the cross-validation training routine and the corresponding evaluation of the classifier.

2 Important Functions / Scripts

2.1 Kernel construct and test kernel

Implements the framework necessary to perform the test of the SVM classifier according with the following parameters:

- C parameter, in range of [0, 10]
- Hyperplane Kernel function: Sigmoid, polynomial, Radial Basis and lineal.
- Decision function shapes: one-vs-the-rest (ovr), one-vs-one (ovo).
- Cache size

All those parameter are set up according to `http://scikit-learn.org/stable/modules/svm.html#svm-kernels`

2.2 Train classifier, test classifier

The training method uses the *fit* function for each kernel configuration in order to perform the training. The test method uses the *predict* function in order to measure the accuracy of the classifier

2.3 Cross validation

Cross validate uses the index operator ":" in order to perform the k-folder cross validation where k is the number of groups that we want to split the training set.

3 Methodology

3.1 Cross validation and testing

The script performs the cross validation along the training set. Once the training is finish, we perform the *test classifier* to observe the accuracy of the classifier according with the kernel that was used.

3.2 Feature extraction

Since the use of the 28 x 28 gray scale vector takes so much time in order to perform the cross-validation and the validation with the test set, we use a proposed feature extraction script that reduce the length of the vector from 28 x 28 features to 28 x 2 features (this proposed method is described in the transformation folder). We proceeded the following manner:

- Use a small part of the training set (i.e short_training.csv) and small part of the test set (short_testing.csv), we perform the training and the test validation.
- The test validation gives us the 92,6 % of accuracy, we proceeded in the same way for the new transformed feature vector, and we achieve the 86.3 % of accuracy.
- Because we observe a good performance over the transformed vector, and since in our opinion the accuracy is closed to the obtained precision. We decided transform the entire training and the test set (i.e. 1.2.Tr_train.csv and 1.2.Tr_test.csv) in order to decrease the elapsed time of each experiment.

4 Results

The following table shows the results according each experiment:

4.1 Polynomial Kernel

Over the train (25000 digits) and test_short data set (15000 digits):

	Degree(1)	Degree(2)	Degree(4)
C = 0.1	89.509	92.639	88.469
C = 1	89.509	92.639	88.469
C = 10	89.509	92.639	88.469

Over the 1.2.Tr_train (26999) and 1.2.Tr_test data set (15001):

	Degree(1)	Degree(2)	Degree(4)
C = 0.1	75.4	84.4	82.7
C = 1	78.1	82.9	82.7
C = 10	78.8	82.3	82.7

We achieve the best result using a kernel of second degree and C parameter equal to 0.1. Interesting to see is the fact that the accuracy doesn't change so much for the variations of the C parameter,

instead the degree of the kernel function is important. The best result we achieve in the total data set is 84.4%.

4.2 Radial Basis Kernel

Over the train_short (5000 digits) and test_short data set (2000 digits):

	$\gamma = 0.1$	$\gamma = 10$	$\gamma = 1000$	$\gamma = 10000$
C = 1	9.92	9.92	9.92	9.92
C = 10	9.92	9.92	9.92	9.92
C = 10000	9.92	9.92	9.92	9.92

Over the 1.2.Tr_train (26999) and 1.2.Tr_test data set (15001):

	$\gamma = 0.1$	$\gamma = 10$	$\gamma = 1000$	$\gamma = 10000$
C = 1	23.0	10.99	10.98	10.98
C = 10	25.63	10.99	10.98	10.98
C = 10000	25.63	10.98	10.98	10.98

We achieve the best result using the c parameter equal to 10 and γ equal to 0.1. Interesting to see is the fact that the accuracy doesn't change so much for the variations of γ parameter. The best accuracy that we achieved is 25.63%. Additional the more we training this model, the more accuracy we have.

We based our experiment according how the influence of the parameters c and γ affect the Radial Basis Kernel, this information is available in: `http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html#example-svm-plot-rbf-parameters-py`

4.3 Sigmoid Kernel

Over the 1.2.Tr_train (26999) and 1.2.Tr_test data set (15001):
The parameter C, here does not change so much the accuracy, therefore in our experiments $C = 1$, the γ parameter and the independent coefficient ρ change respectively:

	$\gamma = 1$	$\gamma = 100$	$\gamma = 10000$
$\rho = -1000$	71.28	82.69	82.752
$\rho = -10$	82.69	82.74	82.752
$\rho = 0$	82.75	82.75	82.757
$\rho = 10$	82.76	82.75	82.751
$\rho = 1000$	83.44	82.76	82.748

We achieve a good result when the independent coefficient ρ is high and the γ parameter is a small value, the best result was 83.44 %.

4.4 Lineal Kernel

Over the 1.2.Tr_train (26999) and 1.2.Tr_test data set (15001):

We configure the C parameter as follows:

	Accuracy
$C = 1$	77.6
$C = 100$	76.7
$C = 10000$	—

For values of C greater than 100, the calculation takes so much time. but we can observe that for small values of C, we get an accuracy of 77.6

5 Conclusions

- We achieve 93% of accuracy in the best case with the complete 28x28 vector feature using a polynomial kernel with degree 2 and $C = 1$.
- We observe that we achieve better results with a polynomial kernel. In our approach we test the polynomial kernel using the 28x28 vector feature, but due the fact that was so long, we extract two features by each row therefore we had a 2x28 feauture vector. We decrease considerably the elapsed time but the best accuracy was 84.4%. We conclude that this kernel can improve considerably if we use the complete 28x28 vector feature (i.e. 93%).
- From the experience of this experiment we observe the Radial Basis Kernel does not have good results, it could improve for big values of c and small values of γ, we achieve the best result of 25.63 % of accuracy.