Support Vector Machines ML HW05

Gerald Baulig 0780827 – 2019/05/20

This is the report and to the machine learning homework 05 and documentation to the related source code. This work contains three parts:

1. One script for classifying handwritten digits from 0 to 4 via Support Vector Machine (SVM), and finding best parameter-set via grid-search,
2. another script for clustering 2D-point-clouds plotting the predicted classification, the support vectors and in bonus the decision boarders,
3. and last but not least, this report.

Please respect the recommended environment the installation instructions and usage-advisements to ensure valid results.

You are welcome to visit: https://github.com/bugerry87/mlhw05

Environment

The test environment (and only tested environment) should the following properties:

- Windows 10
- Python3.7.3 (64bit)

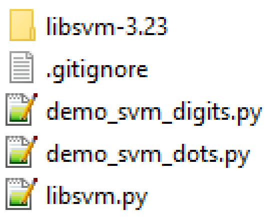
Installation

Before starting the scripts, please make sure that the following software is installed:

- Python3.7.3 (64bit) – with the packages:
    - argparse (should be in python core)
    - numpy
    - matplotlib

Furthermore, the LIBSVM 3.23 needs to be downloaded and extracted into the project directory:

- Please download LIBSVM 3.23 form https://www.csie.ntu.edu.tw/~cjlin/libsvm/
- Extract it into the project directory where the demo scripts and the file libsvm.py is located:



libsvm-3.23
.gitignore
demo_svm_digits.py
demo_svm_dots.py
libsvm.py

- If you are using Linux or MAC OS please refer to the LIBSVM install instructions.

The libsvm.py will temporary adds the LIBSVM's python interface to the working environment, whereas the python interface adds the DLL of LIBSVM. Do not break the given path hierarchy.

Usage

This project contains two runnable scripts:

1. demo_svm_digits.py
    a. Classifies handwritten digits from 0 to 4 via C-SVM.
    b. Performs a Grid-Search on a subset and a given set of parameters.
    c. During the Grid-Search different predefined kernels can be used including:
        - -t 0: linear kernel
        - -t 1: polynomial kernel
        - -t 2: RBF kernel
        - -t 3: Sigmoid kernel (not tested)
        - -t 4: linear+RBF kernel (not supported yet)
    d. The best parameter set will be one more time applied to the training of the whole dataset and a final evaluation will be performed.
    e. Additionally, a training and evaluation session with the custom linear+RBF kernel will be performed.
    f. This script contains all requirements of Part1.
2. demo_svm_dots.py
    a. Clusters 2D-point-clouds via C-SVM.
    b. Runs a training and evaluation session for each given parameter-set.
    c. Plots each session result in a separate subplot.
    d. Colorize the prediction (purple – cyan – yellow), the support vectors (red), and the decision-border of each cluster.
    e. All kernels are supported (see: demo_svm_digits.py).

Simple Usage

Open a terminal of your choice in the project (tested with PowerShell) and run following command:

```
python demo_svm_digits.py
```

or

```
python demo_svm_dots.py
```

These commands will run the scripts with predefined default parameters. Check the default settings and the modifiable parameters via:

```
python demo_svm_digits.py -h
```

or

```
python demo_svm_dots.py -h
```

Advanced Usage – demo_svm_digits

If you want to run demo_svm_digits.py with other grid-search parameters, please run i.e.:

```
python demo_svm_digits.py -t 0 2 -c 1 2 3
```

This command will run a grid-search on linear kernel (0) and RBF kernel (2) with costs of (1,2,3). So, it will run 6 cross-validations in total one for each parameter combination.

Only the parameters -t, -c, -g, -r and -d are supported. Please refer to the LIBSVM documentation for more information.

Advanced Usage – demo_svm_dots

If you want to plot other SVM configurations with demo_svm_dots.py, please run i.e.:

```
python demo_svm_dots.py -p ‚-t 0 -c 2‘ ‚-t 1 -c 5 -d 3‘
```

This command will create 2 plots according to the number of parameter-strings given to -p. In this example the first plot will use a linear kernel with the cost of 2. The second plot use a polynomial kernel with a cost of 5 and a polynomial order of 3 degrees.

All parameters of LIBSVM should be supported (not tested). Please refer to the LIBSVM documentation for more information.

Documentation

The source code contains 2 more helper scripts:

1. libsvm.py

2. mics.py
   Provides general utility functions.

libsvm.py

Loads the LIBSVM library and provides utility functions according to the LIBSVM.

linearRBF(u, v, g = 0.01) -> Gram-Matrix

Calculates and returns the Gram-Matrix of a linear+RBF kernel for a given dataset (u,v) and converts to a LIBSVM-like format.

prediction(X, Y, m, p, isKernel=False) -> y, acc, z

A wrapper function for svm_predict. Applies the linearRBF kernel if required.

grid_search(X, Y, param_dict, viz_func=None) -> best svm_parameters

Performs a grid search with a set of parameters given by 'param_dict' and returns the best parameter-set according to the cross-validation.

These are the most relevant functions to implement Part1 and Part2 of the homework.

Evaluation of Kernels on Handwritten Digits

The Grid-Search of demo_svm_digits performs on linear, polynomial and RBF kernels. The cross-validation shows following average accuracy:

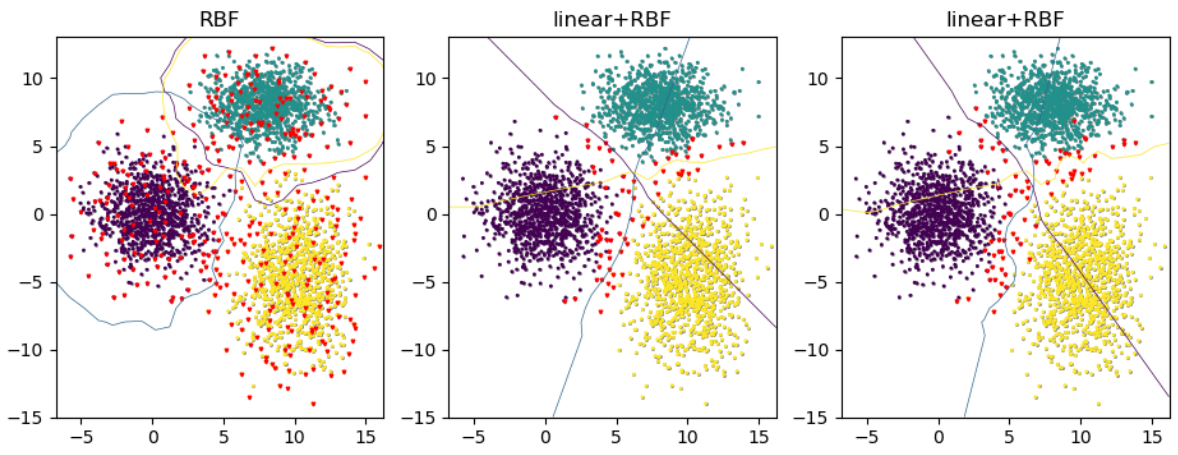
Linear	~90%
Polynomial	~80%
RBF	~25%

The linear kernel method has a surprising high accuracy to a very low computation time compared to the complex RBF kernel. Since RBF wraps a decision border around points that are spatial close to each other (with high similarity), this observation indicates that handwritten digits can be badly seen as a problem of spatial similarity.

The best accuracy of 95.96% we obtain with the parameter-set:

```
-t 0 -c 0.01 -d 4 -g 0.25 -r 1        #Note that '-d 4 -g 0.25 -r 1' have no effect on linear kernel!
```

The custom linear+RBF kernel with a parameter-set of: '-t 4 -c 4 -g 0.5 -h 0' performs with an accuracy of 95.36%. In the result of demo_svm_dots.py it points out that linear+RBF is a compromise of both kernels, while the linear decision-border dominates. The influence of the RBF part depends on gamma. The higher gamma is, the more the decision-border approximates to RBF.



Left: The plot of RBF only, middle: linear+RBF with gamma=0.1, right: linear+RBF with gamma=10