Support Vector Machines: Methods and Applications: Exercise session I

{Vilen.Jumutc, Siamak.Mehrkanoon, Gervasio.Puertas, Johan.Suykens}@esat.kuleuven.be

0.1 Introduction

This document describes a step-by-step guide for the application of support vector machine methods. The goal of the exercise sessions is to provide you with experience in order to be able to solve future application problems.
The Matlab scripts and toolboxes, the related documents, referred papers and data-sets are available for academic purposes on Toledo: `http://toledo.kuleuven.be/`.

0.2 Written report about exercises and homework problems

The exam of the course H02D3a/H00H3a consists of an oral discussion on the basis of an individually written report about the 3 exercise sessions, **including also the homework problems**. In this report it is important to show that you have understanding about the problem and that you can work in a constructive manner towards getting solutions to given problems. This should be done e.g. by including plots of Matlab results **and discussing them within the report**. The course material found at the provided links is intended to help you in the completion process.

0.3 Absolute Crash Course

These exercises will be made using Matlab. For additional information on how to use this environment, see the website `http://www.mathworks.com/`. The documentation is available under the link `support` → `documentation`.

- Example commands for the Matlab prompt are notated as "`>> ...`", while comments are given as "`>> % ...`".
- For additional help on "`command`", type

  ```
  >> help command
  ```

- To clear all variables from the workspace, use

  ```
  >> clear
  ```

- Matlab is a high-level development environment, it is not required to bother about declarations, memory assignment and the like.
- One can pass the orders to Matlab via the command prompt. An alternative is to execute a series of commands in batch mode. For the latter, type

  ```
  >> edit script
  ```

An editor will open with the file `script.m`. After saving the file, one can execute the script:

```
>> script
```

- Matlab has quite extensive possibilities for visualization. Figures are made in a new *figure*-window, which can be created by

  ```
  >> H = figure;
  ```

  and can be referred to by its handle `H`. To close that window, type `>> close(H)`. There is always a *current* window, which is the one last used. To close the current window, type `>> close`. To close all windows, type `>> close all`. Information on different plotting commands can be found by the help of `plot`, `plot3`, `line`, `surf`,... .

- To install the `SVM` and `LS-SVMlab` toolboxes go to:

  Toledo website→ SVM Exercises Course → Course Documents

  download and unzip them in your personal account or hard disk, e.g. in the directories `C:/SVMcourse/SVM` and `C:/SVMcourse/LS-SVMlab`. Then set your Matlab path to:

  ```
  >> addpath('C:/SVMcourse/SVM');
  >> addpath('C:/SVMcourse/LS-SVMlab');
  ```

1 Exercise Session 1: Classification

1.1 A Simple Example: Two Gaussians

To get an intuitive idea what classification is about, the exercises start with a simple toy example. An artificial data-set is constructed from two classes of Gaussians with the same covariance matrices:

```
>> X1 =  1 + randn(50,2);
>> X2 = -1 + randn(51,2);
```

The last ";" avoids the content of the variables to be written out on the screen. The corresponding class labels are defined:

```
>> Y1 = ones(50,1);
>> Y2 = -ones(51,1);
```

To see the content of the variables, type:

```
>> X1
```

The dataset is copied into one input and output variable

```
>> X = [X1;X2];
>> Y = [Y1;Y2];
```

This data-set is shown on a figure using the command:

```
>> figure;
>> hold on;
>> plot(X1(:,1),X1(:,2),'ro');
>> plot(X2(:,1),X2(:,2),'bo');
>> hold off
```

where `'ro'` and `'bo'` define respectively the labels of the data of the positive class to be red and the labels of the data of the negative class to be blue.
Given this figure, can you make a geometric construction using lines to estimate the optimal classifier? Under which conditions do you think this construction is optimal/valid? In general it's a good idea to show insight in the following material by using such a visualization.

1.1.1 Linear Discriminant Analysis

File requirements: `lda.m`, `lda_script.m` available at
Toledo website→ SVM Exercises Course → Assignments → Session 1.

Apply a simple linear classification technique on this toy example. Given this technique, one can simply illustrate what's happening if the class centers will approach by the following loop:

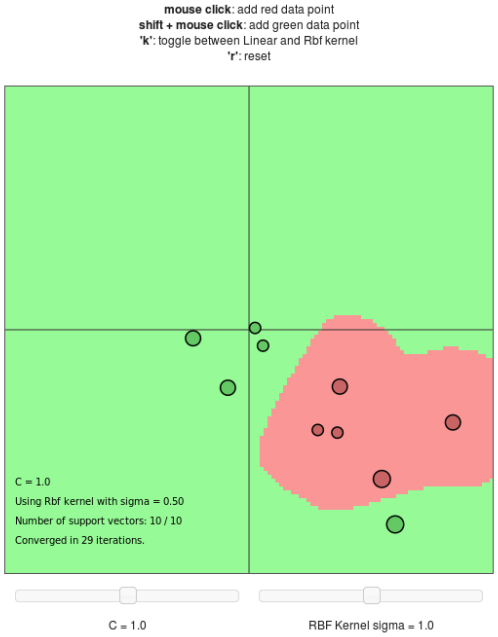
```
>> for i=1:10,

    % make the two clusters
    X1 =  (1-i/6) + randn(50,2);
    X2 = -(1-i/6) + randn(51,2);
    Y1 = ones(50,1);
    Y2 = -ones(51,1);
```

```
    X = [X1;X2];
    Y = [Y1;Y2];

    % a test data point
    Xtest = [0 0];

   % classify test point
   Ytest = lda(Xtest,X,Y);

   % visualize
   plot(X1(:,1), X1(:,2), 'ro'); hold on
   plot(X2(:,1), X2(:,2), 'bo');
   if Ytest>0,
     plot(Xtest(:,1), Xtest(:,2),'r*','markersize',20);
   else
     plot(Xtest(:,1), Xtest(:,2),'b*','markersize',20);
   end
   hold off

   pause
end
```

Compare this code with the sample script `lda_script.m` given at the website of the course and illustrate the main idea behind LDA.

1.2 The Support Vector Machine

Open your browser and type: `http://cs.stanford.edu/people/karpathy/svmjs/demo/`



Experiments to carry on:

1. Follow the instructions. Switch to using the linear kernel by toggling the "k" button. Adjust the existing datasets to have at least 10 data points for each class. What do you observe when you are adding data points to the classes? How drastically can classification boundaries change?
2. What if you add an outlying datapoint which lies on the wrong side of the classification boundary? How does it affect the classification hyperplane?
3. Try different values of `C` regularization hyperparameter. How does it affect the classification outcome? What is the role of it?
4. Follow the instructions and switch back to RBF kernel by toggling the "k" button. Compare to the classification outcome of the linear case.
5. Try to change the RBF kernel `sigma` hyperparameter. What is your intuition? How does it affect the classification boundaries? Now try to change both hyperparameters. What is the right choice of those if your data is almost linearly separable?
6. Create a linearly non-separable dataset with an overlapping region between classes (e.g. similar to the Gaussian clouds in the previous LDA experiment). Give comments on the role of the chosen kernel, the regularization parameter (`C`) and the kernel parameter (`sigma`).
7. What is the role of Support Vectors[1]? Change the data sets to make the number of support vectors increase/decrease. When does a particular datapoint become a Support Vector?
8. When does the corresponding importance[2] of a Support Vector change?

1.3 Using LS-SVMlab

We proceed with the least squares based variant of the support vector machine (LS-SVM). To try this method, the toolbox LS-SVMlab is used. If you have not downloaded it yet, please download:
Toledo website→ SVM Exercises Course → Course Documents → SVM course scripts

Enter `LSSVMlab` directory and run the classification demo by typing

```
>> democlass
```

Remark that by default the data is preprocessed internally for as well training as simulation. For additional information on the used preprocessing,

```
>> help prelssvm
```

This is important to remember as preprocessing directly affects the chosen hyper-parameters and the performance on the test set.

File requirements: `iris.mat`, `SampleScript_iris.m` available at
Toledo website→ SVM Exercises Course → Assignments → Session 1.

1. Repeat the same commands as given in the demo for training, simulation and plotting on the two-Gaussian example of the LDA experiment.

[1] large circles with the bold-lined circumference
[2] the importance is measured by the size of the bold-lined circle

2. Load the Iris dataset

   ```
   >> load iris
   ```

   Try out the linear kernel with the command

   ```
   >> [alpha,b] = trainlssvm({X,Y,'c',gam,[],'lin_kernel'});
   ```

   and make plot of the final classifier by the command of plotlssvm:

   ```
   >> plotlssvm({X,Y,'c',gam,[],'lin_kernel'},{alpha,b});
   ```

   What is the performance on the test set `Xt` and `Yt`? Try out the polynomial kernel with `degree` $= 1, 2, 3, \ldots$ and `t` $= 1$.

   ```
   >> [alpha,b] = trainlssvm({X,Y,'c',gam,[t;degree],'poly_kernel'});
   ```

   What happens when you are changing the degree of a polynomial kernel? Explain the obtained results. Does it correspond to the changes in `sigma` hyperparameter of the RBF kernel in the previous example?

3. Let us focus on the RBF kernel with kernel parameter the bandwidth σ^2. Try out a good range of different `sig2`'s as kernel parameters. For each individual value of `sig2`, the corresponding LS-SVM is evaluated on the test set. Make a figure of the `sig2`'s with their corresponding test set performance.

4. Now, take a look at the regularization constant `gam`. Fix a reasonable choice for the `sig2` of the RBF kernel and again compare a range of `gam`'s by plotting the corresponding test set performances. What is a *good* range for `gam`?

Compare your results with the sample script `SampleScript_iris.m` on the website of the course on Toledo.

1.3.1 The Choice of the Hyper-parameters

The intuition developed in the previous section is now used to motivate automated tuning algorithms.

1. A straightforward way to find good values for the parameters is to divide the given data in a *training set* and a *validation set*. The latter is used to measure the generalization performance of the model trained on the former. Motivate why this validation set used to optimize a number of (tuning-) parameters cannot used again to measure the final model.

   ```
   % set the parameters to some value
   >> gam = 0.1;
   >> sig2 = 20;

   % generate random indices
   >> idx = randperm(size(X,1));

   % create the training and validation sets
   % using the randomized indices
   >> Xtrain = X(idx(1:80),:);
   >> Ytrain = Y(idx(1:80));
   >> Xval = X(idx(81:100),:);
   >> Yval = Y(idx(81:100));
   ```

Train an LS-SVM model with Xtrain and Ytrain:

```
>> [alpha,b] = trainlssvm({Xtrain,Ytrain,'c',gam,sig2,'RBF_kernel'});
```

and evaluate it on Xval:

```
>> estYval = simlssvm({Xtrain,Ytrain,'c',gam,sig2,'RBF_kernel'}, ...
{alpha,b},Xval);
```

Calculate the performance in terms of misclassification error on the validation set (using estYval and Yval). Make a plot of the performance with respect to several values of `gam` and `sig2` (e.g. `gam` = 1, 10, 100, `sig2` = 0.1, 1, 10). Give comments about the results.

2. Perform crossvalidation using 10 folds:

```
>> performance = crossvalidate({X,Y,'c',gam,sig2,'RBF_kernel'}, ...
                                10,'misclass');
```

Think about a clear and intuitive way to represent this technique. Why should one prefer this method over a simple validation? Change `crossvalidate` procedure for `leaveoneout`. Is it giving better results? In which cases one would prefer each?

3. Use `tunelssvm` procedure to optimize the hyperpatameters. Execute:

```
>> model = {X,Y,'c',[],[],'RBF_kernel','csa'};
>> [gam,sig2,cost] = tunelssvm(model,'simplex', ...
                                'crossvalidatelssvm',{10,'misclass'});
```

Try to change different parameters like `'csa'` (Coupled Simulated Annealing) vs. `'ds'` (Randomized Directional Search) and `'simplex'` (Nelder-Mead method) vs. `'gridsearch'` (brute force grid search). What differences do you observe? Why in some cases the obtained hyperparameters differ a lot?

4. One alternative way to judge a classifier is by using the Receiver Operating Characteristic (ROC) curve of a binary classifier (see `>> help roc`). The higher the area under the curve, the better the classifier separates the data. For making the ROC curve on the training data use `roc` command. Using the training set is not recommended, do you know why? An ROC plot of a validation set `Xval` and `Yval` is made by

```
>> [alpha,b] = trainlssvm({X,Y,'c',gam,sig2,'RBF_kernel'});
>> [Ysim,Ylatent] = simlssvm({X,Y,'c',gam,sig2,'RBF_kernel'},...
                             {alpha,b},Xv);
>> roc(Ylatent,Yval);
```

Note: In the case the data is preprocessed (zero mean, unit variance), a good initial guess of `sig2` of the RBF kernel is the dimension of the input space.

1.4 Homework Problems

Illustrate your skills on support vector machine learning of synthetic and real-life datasets. Inspirations for typical questions which are relevant for the analysis of the following case studies can be found in the previous exercises. Answer the following questions related to LS-SVM classification while analyzing the subsequent data-sets.

1. Look at the plotted data. What seems to be important properties of the data?
2. Try a linear model. Do you think it's sufficient?
3. Try the RBF kernel and tune its parameter. Is the tuning acceptable? Run several times the `tunelssvm` routine. Do the tuned parameters `gam` and `sig2` change? What about the performance?
4. Consider the `ROC` curve. What does it say about the choice between 2. and 3.?
5. Judge your final model. Is the methodology perfectly suited for this data-set?

1.4.1 The Ripley Data-Set

This classical data-set allows one to visualize the different steps of the analysis in a straightforward way.

1.4.2 Breast Cancer Data-set

The training scenario for this data-set consists of 400 datapoints. Each datapoint has 30 real-valued features. The task is to predict whether the experiment is benign or malignant. The test set consists of 169 datapoints. Additional information and the description of the dataset can be found in: `http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.names`.

1.4.3 Diabetes Database

This data-set consists of 300 datapoints (refined set from UCI repository) for training and 168 datapoints for testing. Additional information and the description of the dataset can be found in: `http://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.names`.