Support Vector Machines: Methods and Applications
Exercise Session II

Siamak Mehrkanoon Dries Hendrikx Johan A. K. Suykens

{siamak.mehrkanoon, dries.hendrikx, johan.suykens}@esat.kuleuven.be

Introduction

The Matlab scripts and toolboxes, the related documents, referred papers and data-sets are available for academic purposes on Toledo: `http://toledo.kuleuven.be/`.

2 Exercise Session 2: Function Estimation and Time-series Prediction

2.1 The Support Vector Machine for Regression

For this exercise the SVM toolbox is used. Please download it directly from Toledo: Toledo website → SVM Exercises Course → Course Documents → SVM toolbox. Execute:

```
>> uiregress
```

Construct a data-set of around 20 data-points. Which kernel is best suited for your data-set? Construct a data-set where a linear kernel is better than any other kernel. What is the influence of different values of `e`? (try small values *e.g.*, 0.1, 0.25, 0.5...) and of `Bound` (*i.e.* hyperparameter `C` in the previous exercise session)? Try values with larger increments *e.g.* 0.01, 0.1, 1, 10, 100. Where does the sparsity property come in? In what respect is SVM regression different from a classical Least Squares fit?

2.2 A Simple Example: Sum of Cosines

For the remainder of this exercise session, LS-SVMlab is used. Run the demo `demofun`. We start with the following synthetic example:

```
>> X = (-10:0.1:10)';
>> Y = cos(X) + cos(2*X) + 0.1.*randn(length(X),1);
```

The training/validation and test sets are created:

```
>> Xtrain = X(1:2:length(X));
>> Ytrain = Y(1:2:length(Y));
>> Xtest = X(2:2:length(X));
>> Ytest = Y(2:2:length(Y));
```

The following commands can be used to make an LS-SVM model with the RBF kernel and arbitrary values of the hyper-parameters:

```
>> gam = 100;
>> sig2 = 1.0;
>> [alpha,b] = trainlssvm({Xtrain,Ytrain,'f',gam,sig2,'RBF_kernel'});
```

The results on the training set can be visualized via:

```
>> plotlssvm({Xtrain,Ytrain,'f',gam,sig2,'RBF_kernel'},{alpha,b});
```

The results of applying the trained model on the test set are calculated using:

```
>> YtestEst = simlssvm({Xtrain,Ytrain,'f',gam,sig2,'RBF_kernel'},...
   {alpha,b},Xtest);
```

and visualized with:

```
>> plot(Xtest,Ytest,'.');
>> hold on;
>> plot(Xtest,YtestEst,'r+');
>> legend('Ytest','YtestEst');
```

Make similar plots showing the results on training and test sets with different values of `gam` and `sig2`. Do you think there is one pair of optimal hyper-parameters?

2.3 Hyper-parameter Tuning

The functions `crossvalidate` and `leaveoneout` can be used to estimate the performance of a set of hyper-parameters:

```
>> cost_crossval = crossvalidate({Xtrain,Ytrain,'f',gam,sig2},10);
>> cost_loo = leaveoneout({Xtrain,Ytrain,'f',gam,sig2});
```

The optimization of the hyper-parameters is done with the function `tunelssvm`. For example,

```
>> optFun = 'gridsearch';
>> globalOptFun = 'csa';
>> [gam,sig2,cost] = tunelssvm({Xtrain,Ytrain,'f',[],[],'RBF_kernel', ...
                    globalOptFun},optFun,'crossvalidatelssvm',{10,'mse'})
```

tunes the hyper-parameters using 10-fold crossvalidation using a gridsearch approach and Coupled Simulated Annealing. Train a model with tuned hyper-parameters and display the results on the training data:

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

Run the previous `tunelssvm` command several times. What can you say about the values of the hyperparameters and the results? Change `optFun` to `'simplex'` (`>> optFun = 'simplex';`) and run the tuning once more. What is the difference between `'simplex'` and `'gridsearch'`? Try to change `globalOptFun` to `'ds'` (Randomized Directional Search). Is there any significant deviation in results? Which method is faster and why?

2.4 Application of the Bayesian Framework

2.4.1 Regression

Let's consider the same toy example. The Bayesian framework now can be used to tune and to analyze the LS-SVM regressor. The basic result from the Bayesian framework for the LS-SVM is the derivation of *the probability that the data points are generated by the given model*. This is called the posterior probability. This probability criterion is expressed as a number. There are 3 variants: the posterior with respect to the model parameters `alpha` and `b`, the posterior with respect to the regularization constant `gam` and the posterior with respect to the choice of the kernel and its parameter. The cost (negative logarithm of the posteriors) is calculated by the function call

```
>> sig2 = 0.05; gam = 10;
>> criterion_L1 = bay_lssvm({Xtrain,Ytrain,'f',gam,sig2},1)
>> criterion_L2 = bay_lssvm({Xtrain,Ytrain,'f',gam,sig2},2)
>> criterion_L3 = bay_lssvm({Xtrain,Ytrain,'f',gam,sig2},3)
```

The model can be optimized with respect to these criteria:

```
>> [~,alpha,b] = bay_optimize({Xtrain,Ytrain,'f',gam,sig2},1);
>> [~,gam] = bay_optimize({Xtrain,Ytrain,'f',gam,sig2},2);
>> [~,sig2] = bay_optimize({Xtrain,Ytrain,'f',gam,sig2},3);
```

Try to think about a clear schematic visualization of this three-level principle. For regression, the error bars can be computed using Bayesian inference, *e.g.*

```
>> sig2e = bay_errorbar({Xtrain,Ytrain,'f',gam,sig2},'figure');
```

2.4.2 Classification

For classification, it is also possible to get probability estimates. Load the Iris data-set:

```
>> clear;
>> load iris;
>> gam = 5; sig2 = 0.75;
```

The probabilities that a certain data point belongs to the positive class given the model is calculated by:

```
>> bay_modoutClass({X,Y,'c',gam,sig2},'figure');
```

How do you interpret the colors? (hint: activate the color-bar of the figure). Change the values of `gam` and `sig2`. What is the influence of the given hyper-parameters on this figure?

2.4.3 Automatic Relevance Determination

The Bayesian framework can also be used to select the most relevant inputs by Automatic Relevance Determination (ARD). The following procedure uses this criterion for *backward selection* for a three dimensional input selection task:

```
>> X = 10.*rand(100,3)-3;
>> Y = cos(X(:,1)) + cos(2*(X(:,1))) + 0.3.*randn(100,1);
>> [selected, ranking] = bay_lssvmARD({X,Y,'class',gam,sig2});
```

Can you visualize the results on a simple figure? Can you think of a similar way to do input selection by using the `crossvalidate` function instead of the Bayesian framework?

2.5 Robust Regression

In situations where the data is corrupted with non-Gaussian noise or outliers, it becomes important to incorporate robustness into the estimation. Consider the following simple example:

```
>> X = (-10:0.2:10)';
>> Y = cos(X) + cos(2*X) + 0.1.*rand(size(X));
```

Outliers are added via:

```
>> out = [15 17 19];
>> Y(out) = 0.7+0.3*rand(size(out));
>> out = [41 44 46];
>> Y(out) = 1.5+0.2*rand(size(out));
```

Train and plot an LS-SVM regression model with `gam = 100, sig2 = 0.1`. What is the influence of the outlying data points?

Let's now train a robust LS-SVM model using the object-oriented interface and robust crossvalidation.

```
>> model = initlssvm(X,Y,'f',[],[],'RBF_kernel');
>> costFun = 'rcrossvalidatelssvm';
>> wFun = 'whuber';
>> model = tunelssvm(model,'simplex',costFun,{10,'mae'},wFun);
>> model = robustlssvm(model);
>> plotlssvm(model);
```

Discuss the results. Why in this case is the *mean absolute error* (mae) preferred over the classical *mean squared error* (mse)? Try alternatives to the weighting function `wFun` (*e.g.* `'whampel'`,`'wlogistic'`,`'wmyriad'`). Check the user's guide for more information.

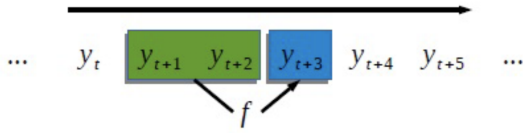

Figure 1: Schematic representation of the nonlinear auto-regressive model.

2.6 Homework Problem

2.6.1 Introduction: Time-series Prediction

The linear auto-regressive (AR) model of a process Z_t with $t = 1, 2, \ldots, \infty$ is given by

$$\hat{z}_t = a_1 z_{t-1} + a_2 z_{t-2} + \cdots + a_n z_{t-n} \ , \tag{1}$$

with $a_i \in \mathbb{R}$ for $i = 1, \ldots, n$ and n the model order. At the same time, the nonlinear variant (NAR) is described as:

$$\hat{z}_t = f(z_{t-1}, z_{t-2}, \ldots, z_{t-n}) \ . \tag{2}$$

A depiction of these processes can be found in Figure 1. Remark that in this way, *the time-series identification can be written as a classical black-box regression modeling problem*:

$$\hat{y}_t = f(x_t) \ , \tag{3}$$

with $y_t = z_t$ and $x_t = [z_{t-1}, z_{t-2}, \ldots, z_{t-n}]$. In a nutshell, any provided sequence `Z` can be mapped into a regression problem. In our case, the required transformation can be done via *e.g.* the command `windowize`:

```
>> order  = 10;
>> W = windowize(Z,1:order+1);
>> X = W(:,1:order);
>> Y = W(:,end);
```

Now a model can be built using these data points:

```
>> gam = 10;
>> sig2 = 10;
>> model = {X,Y,'f',gam,sig2,'RBF_kernel'};
>> [alpha,b] = trainlssvm(model);
```

It is straightforward to predict the next data points using the `predict` function of the LSSVMlab toolbox. In order to call the function, we first have to define the starting point of the prediction:

```
>> Xs = Z(end-order+1:end,1);
```

This is the last point of the training set. The test set `Ztest` presents data points after this point, which we will try to predict. This can be implemented as follows:

```
>> nb = 50;
>> prediction = predict(model,Xs,nb);
```

where `nb` indicates how many time points we want to predict. Here, we define this number equal to the number of data points in the test set. Finally, the performance of the predictor can be checked visually:

```
>> figure
>> hold on
>> plot(Ztest, 'k')
>> plot(prediction, 'r')
```

In this figure, the data points of the test set, that is, the actual data points that we want to predict are indicated in black, while the prediction is presented in red. Try to assess the performance of the prediction: do you think it's good? Is there still some room for improvements? As in the previous section, `gam` and `sig2` can be optimized by using a validation procedure like 10-fold crossvalidation. In the same way, one can optimize the `order` parameter. Try to change the order of the model and evaluate the mean squared error (MSE) on the test set. How does the order of the model affect the obtained performance?

2.6.2 Application: Santa Fe Laser Dataset

We would like you to apply now this procedure to the the Santa Fe laser time series data-set and answer the following set of questions. **Please, justify all of your answers as thoroughly as possible**. Keep in mind that one of the skills being evaluated during the final examination is also your ability to creatively and constructively address problems with which you may not be entirely familiar with the help of the tools learned during the course.

Before proceeding with the experiments on the Santa Fe Laser dataset ensure that your data is already split to the training (`Z`) and test (`Ztest`) sets as it was described in the previous section.

Questions:

- Does `order=50` for the utilized auto-regressive model sound like a good choice?
- Would it be sensible to use the performance of this recurrent prediction on the *validation set* to optimize the *hyper-parameters* and the *model order*?