



## **SVMBridge: Using SVM Solvers from R**

**Aydın Demircioğlu**  
Ruhr-Universität Bochum

**Hanna Houphouet**  
Ruhr-Universität Bochum

**Daniel Horn**  
TU Dortmund

**Tobias Glasmachers**  
Ruhr-Universität Bochum

**Bernd Bischl**  
TU München

**Claus Weihs**  
TU Dortmund

---

### **Abstract**

Most SVM Solver do not come with a wrapper in R. In case the SVM Solver is written in C++, it can be linked to a package via Repp. Although possible, this entails a lot of work to do, and is not possible, if the SVM Solver is written in other languages. Alternatively, one can call the SVM Solver from within R by a system command. SVMBridge eases this calls by providing a framework and ready wrappers for several SVM Solvers like LASVM, SVMperf, LLSVM and BVM/CVM.

*Keywords:* support vector machines, command line.

---

## **1. Introduction**

### **1.1. Support Vector Machines**

Support Vector machines in general.

### **1.2. SVM packages in R**

There are only a few SVM libraries readily available from within R. This includes the kernlab, the e1071 and the lasvmR packages. All of these link existing C++ sources directly from R and need more or less extensive wrappers to allow the user direct access to the options of the underlying SVM solver.

Other SVM solver, like the BudgetedSVM package, which contains BSGD and LLSVM, or SVMperf cannot be called directly from R. Instead, a system command has to be issued. To

ease this calls, the **SVMBridge** package has been developed.

## 2. The SVMBridge Package

### 2.1. Wrapper

A wrapper consists of several routines: It must provide routines for assembling the command line call for training as well as testing, reading and writing the model and a few general routines like searching the binaries and printing.

Note that the search for binaries might have a huge toll on HPC clusters, so it is advised to specify the paths directly instead of relying on the automatism.

## 3. Examples

To use the SVMBridge, one usually perform the following steps: Add the external wrapper to the SVMBridge, load the data, train a model, use the model for prediction.

Notice that the data must be written by the SVMBridge before calling the corresponding SVM Solver, as there is no way to pass data via memory to command line tools. Therefore

### 3.1. Adding the Wrapper to the SVMBridge

### 3.2. Training a model

### 3.3. Predicions

### 3.4. Optimization Values

At times, some of the training values are of interest, e.g. when comparing different solvers, the primal value of the SVM problem as well as the dual value might be of interest. These can be computed via the `optimizationValues()` routine.

### 3.5. Performance Considerations

We sum up the points to keep in mind when performance is of high priority: Do not use the automatism to find the binaries, specify the paths by hand. Do not load the data into memory, specify the path of the data when training instead. Do not re-read the model into memory, let it on disk.

## 4. Conclusion

We provided a simple framework to attach many SVM solvers to the R Subsystem. This allows for a systematic call of them. By providing a wrapper, linking any SVM solver to R become much easier.

## Acknowledgments

### Affiliation:

Aydın Demircioğlu  
Institut für Neuroinformatik  
Ruhr-Universität Bochum  
44790 Bochum E-mail: [aydin.demircioglu@ini.rub.de](mailto:aydin.demircioglu@ini.rub.de)  
URL: <http://www.ini.rub.de/PEOPLE/aydind/>