Support Vector Machines in R: a benchmark study

David Meyer

June 26, 2003

Overview

1. Support Vector Machines
2. `libsvm`
3. The R interface
4. Benchmark results
5. The EUNITE competition

SV ϵ-Regression

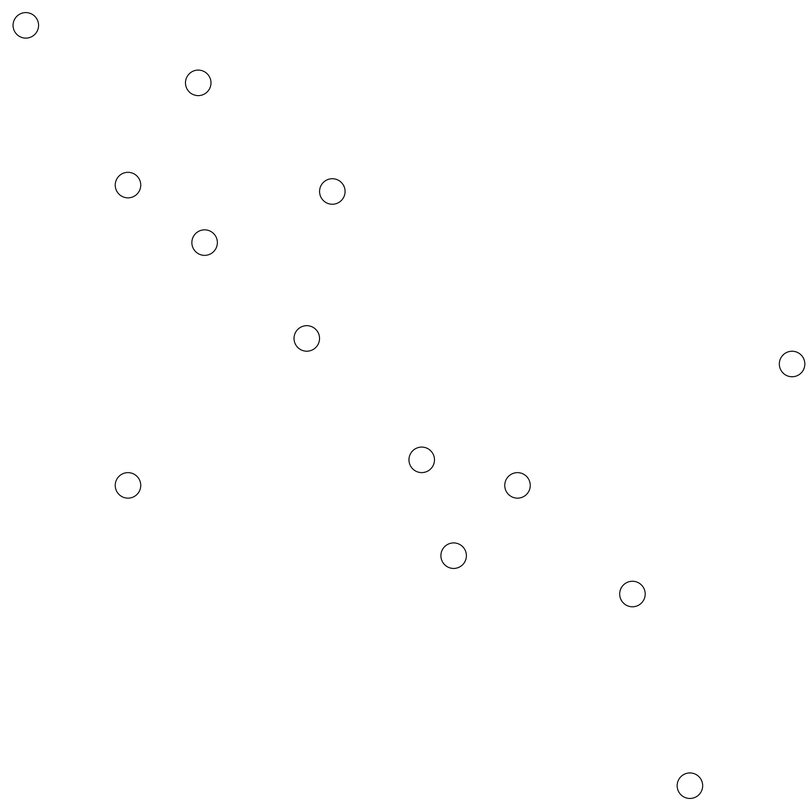
We are given training data:

$$\{(\mathbf{x}_i, y_i)\}_1^l, \mathbf{x}_i, y_i \in \mathbb{R}^n$$

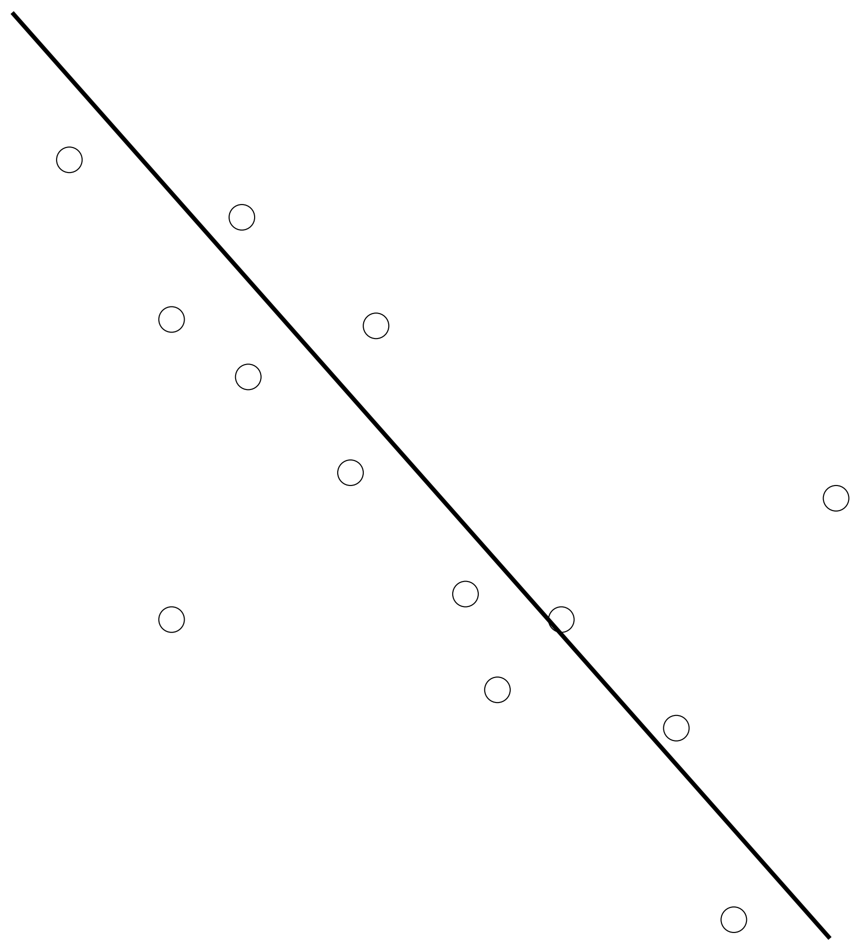
Suppose the data can be explained by a linear model.

Goal: find a fitting hyperplane $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$

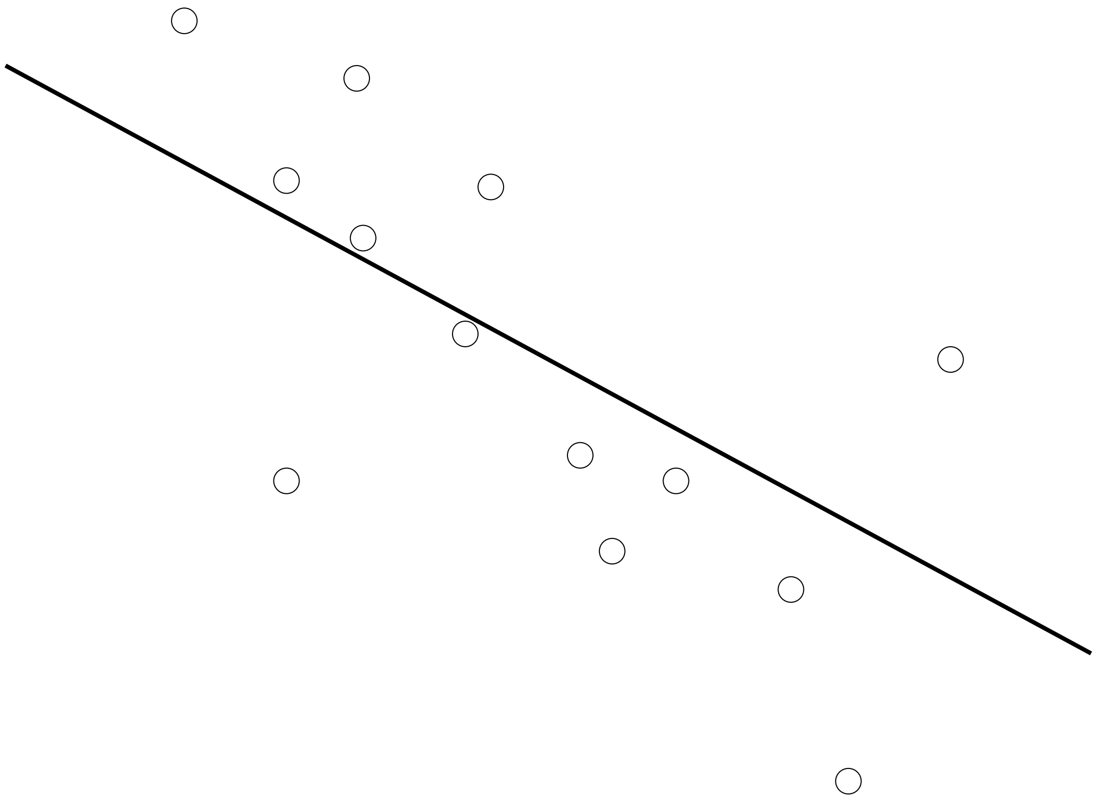
SV ϵ-Regression



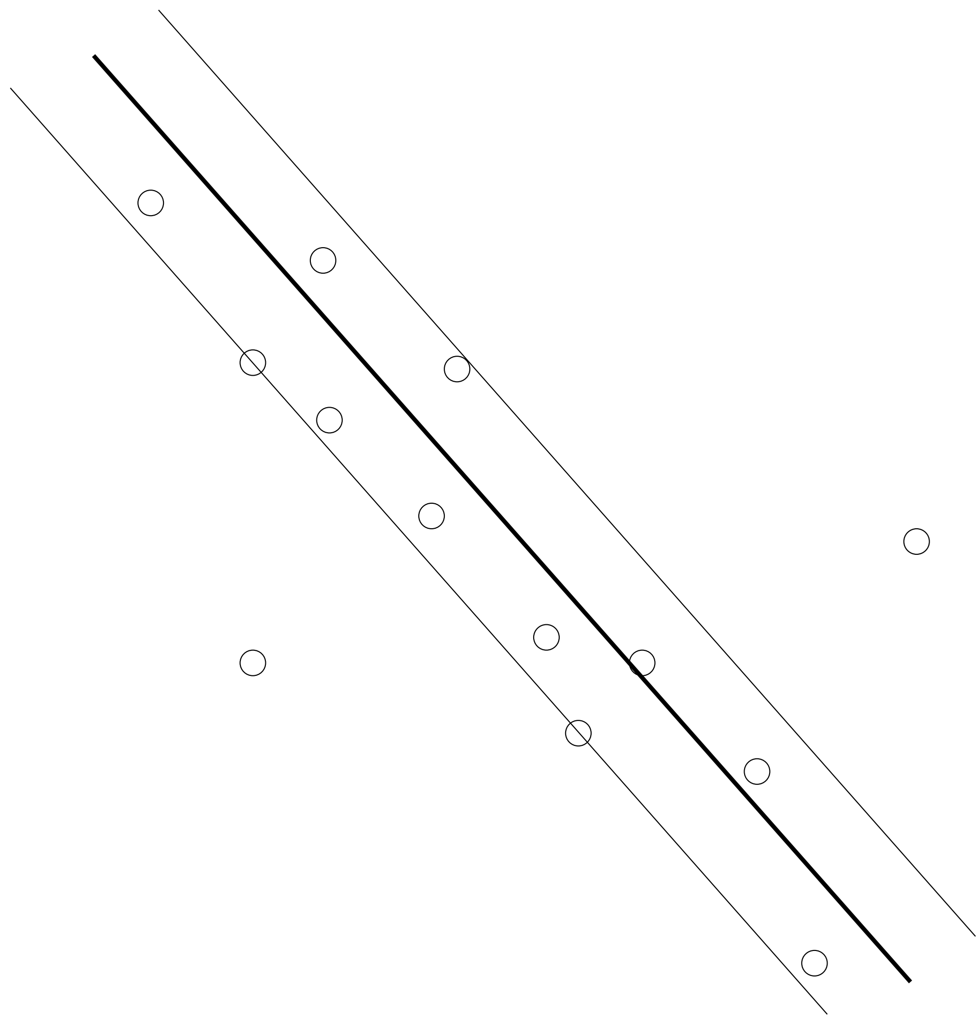
SV ϵ-Regression



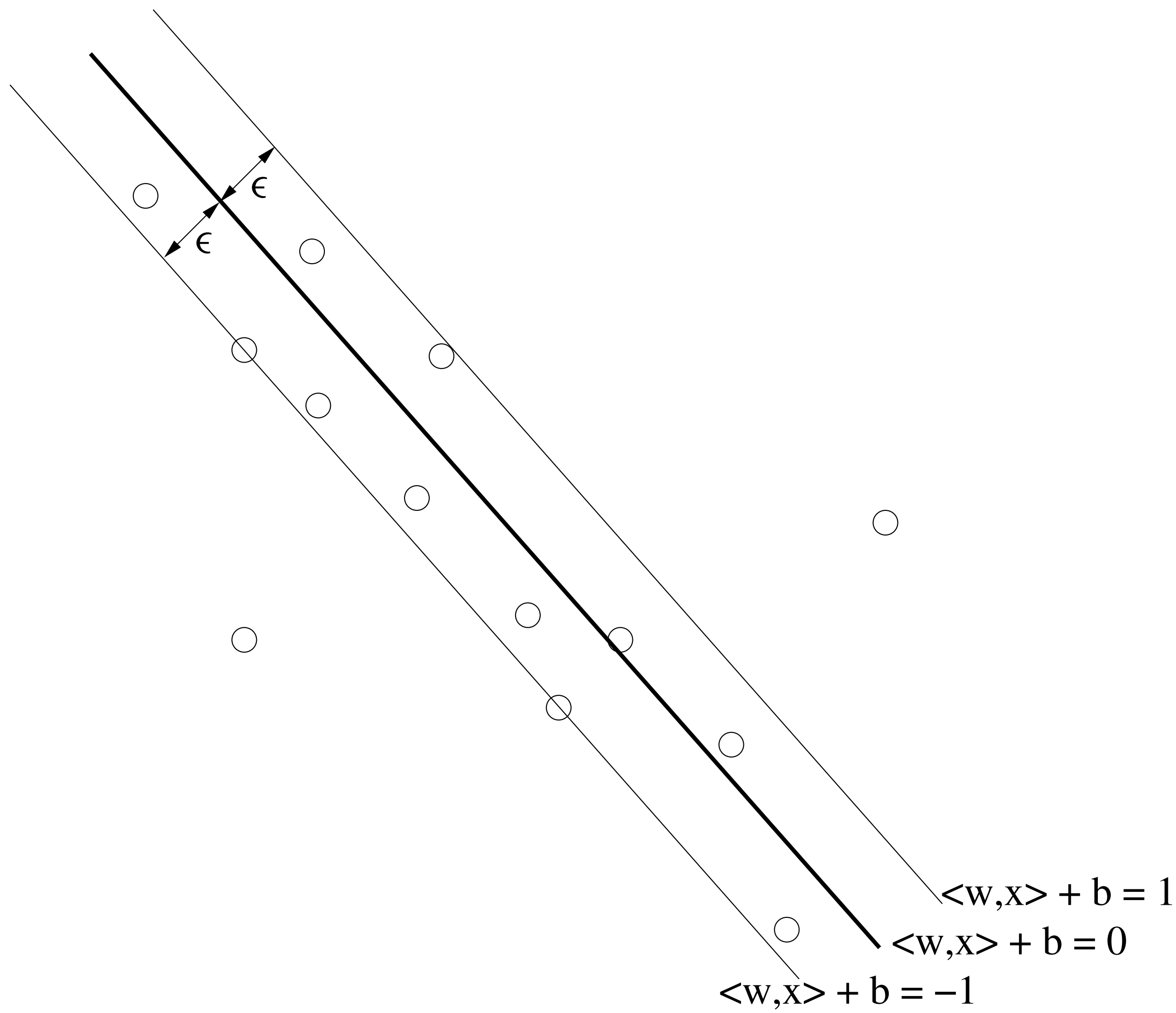
SV ϵ-Regression



SV ϵ-Regression



SV ϵ-Regression



SV ϵ-Regression

The distance from the hyperplane to a border is:

$$\frac{1}{\|\mathbf{w}\|}$$

Thus, the margin is given by:

$$\frac{2}{\|\mathbf{w}\|}$$

Goal:

Maximize margin such it will contain all points with errors smaller than an ϵ to be specified.

SV ϵ-Regression

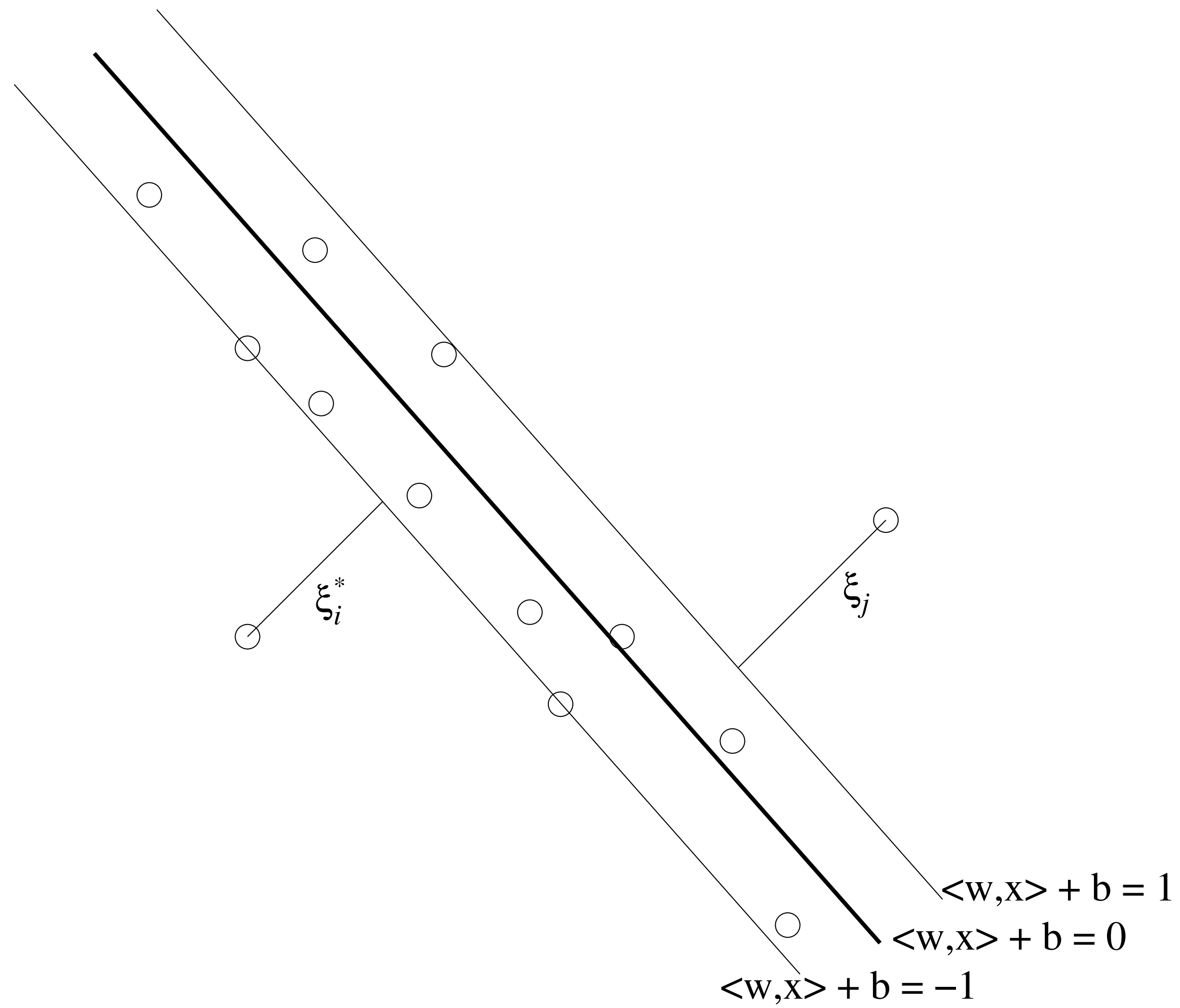
Formally:

$$\min_{\mathbf{w},b} \frac{\|\mathbf{w}\|^2}{2}$$

under constraints:

$$\begin{aligned} y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b &\leq \epsilon \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i &\leq \epsilon \end{aligned}$$

SV ϵ-Regression



SV ϵ-Regression

Handling outliers:

Allow points to be "outside" the margin by using weighths ξ_i:

$$\begin{aligned} y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b &\leq \epsilon + \xi_i \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i &\leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0, i = 1, \ldots, l & \end{aligned}$$

but introduce penalty weight C:

$$\min_{\mathbf{w}, b, \xi, \xi^*} \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^{l} (\xi_i + \xi_i^*)$$

SV ϵ-Regression

Handling non-linearity:

Map the input data $\mathbf{x}_i$ into a (possibly higher dimensional) *feature space*:

$$\mathbf{x}_i \mapsto \phi(\mathbf{x}_i)$$

Thus, the complete optimization problem becomes:

$$\min_{\mathbf{w},b,\xi,\xi^*} \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^{l} (\xi_i + \xi_i^*)$$

$$\begin{aligned} \text{s.t.} \quad y_i - \langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle - b &\leq \epsilon + \xi_i \\ \langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b - y_i &\leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* &\geq 0, i = 1, \ldots, l \end{aligned}$$

SV ϵ-Regression

The Kernel trick:

Find solution using Lagrangian method in the *dual* form:

$$\min_{\alpha,\alpha*} \quad \frac{1}{2} \sum_{i,\ j=1}^{l} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$$

$$+ \epsilon \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i (\alpha_i + \alpha_i^*)$$

$$\text{s.t.} \quad \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) = 0$$

$$0 \le \alpha_i, \alpha_i^* \le C, \ \ i = 1, \ldots, l$$

The inner product $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ is called a *Kernel*.

SV ϵ-Regression

Kernels:

For certain functions ϕ, the inner product can be computed by $K(\mathbf{x}_i, \mathbf{x}_j)$ without explicitely knowing $\phi(\cdot)$.

Some popular kernels:

Polynomial	$\left(\gamma\langle\mathbf{x}_i, \mathbf{x}_j\rangle + \beta\right)^d$
Radial Basis Function	$\exp\left(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$
Sigmoid	$\tanh\left(\gamma\langle\mathbf{x}_i, \mathbf{x}_j\rangle + \beta\right)$

We do have to specify (in addition to C and ϵ) the hyperparameters d, γ, and β: this usually requires *tuning*.

SV ϵ-Regression

Predicting:

The approximate function is simply:

$$f(x) = \sum_{i=1}^{l} (-\alpha_i + \alpha_i^*) K(x_i, x) + b$$

. . . a weighted sum (i.e., superposition) of kernel functions.

Chih-Chen Lin's `libsvm`

- Starting point: svm^{light} (Joachims, 1998) and Pseudo code for SMO (Platt, 1998)

- Motivation: bad performance of svm^{light} on complex cases $\Rightarrow$ development of `bsvm`

- Drawbacks of `bsvm`: very complex, uses third-party optimization solvers $\Rightarrow$ simple, usable code: `libsvm`

- Meanwhile: several extensions and optimizations $\Rightarrow$ very competitive

- Future development: increase usability ("black box")

Features

- C and ν classification
- multiclass-classification
- ϵ and ν regression
- novelty detection
- unbalanced data
- internal sparse data format
- many interfaces (C, Python, Java, Perl, R, MATLAB, . . .)

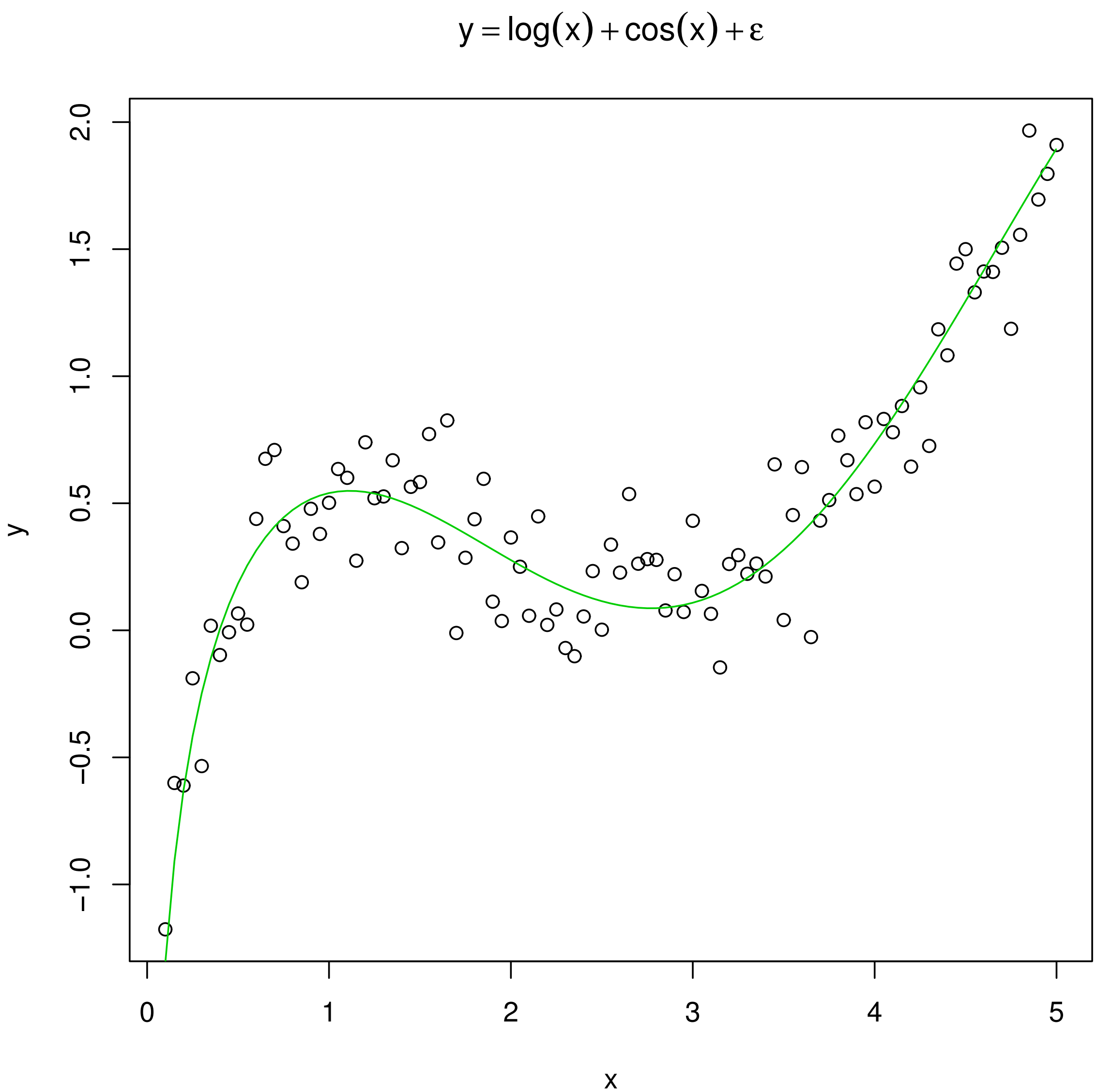
Performance

- `libsvm` won several competitions, e.g.:
  - EUNITE 2001 (competition on electricity load prediction)
  - IJCNN Challenge 2001 (two of three competitions)
- very good (fast) quadratic solver (decomposition method), high standard in numerical methods.

The R interface

- `libsvm` is a C++ library with standard C interface
- R interface in package `e1071` on CRAN (`http://cran.R-project.org/`)
- main functions:
  - training (returning an '`svm`' object)
  - predicting
  - tuning (for hyperparameter selection)

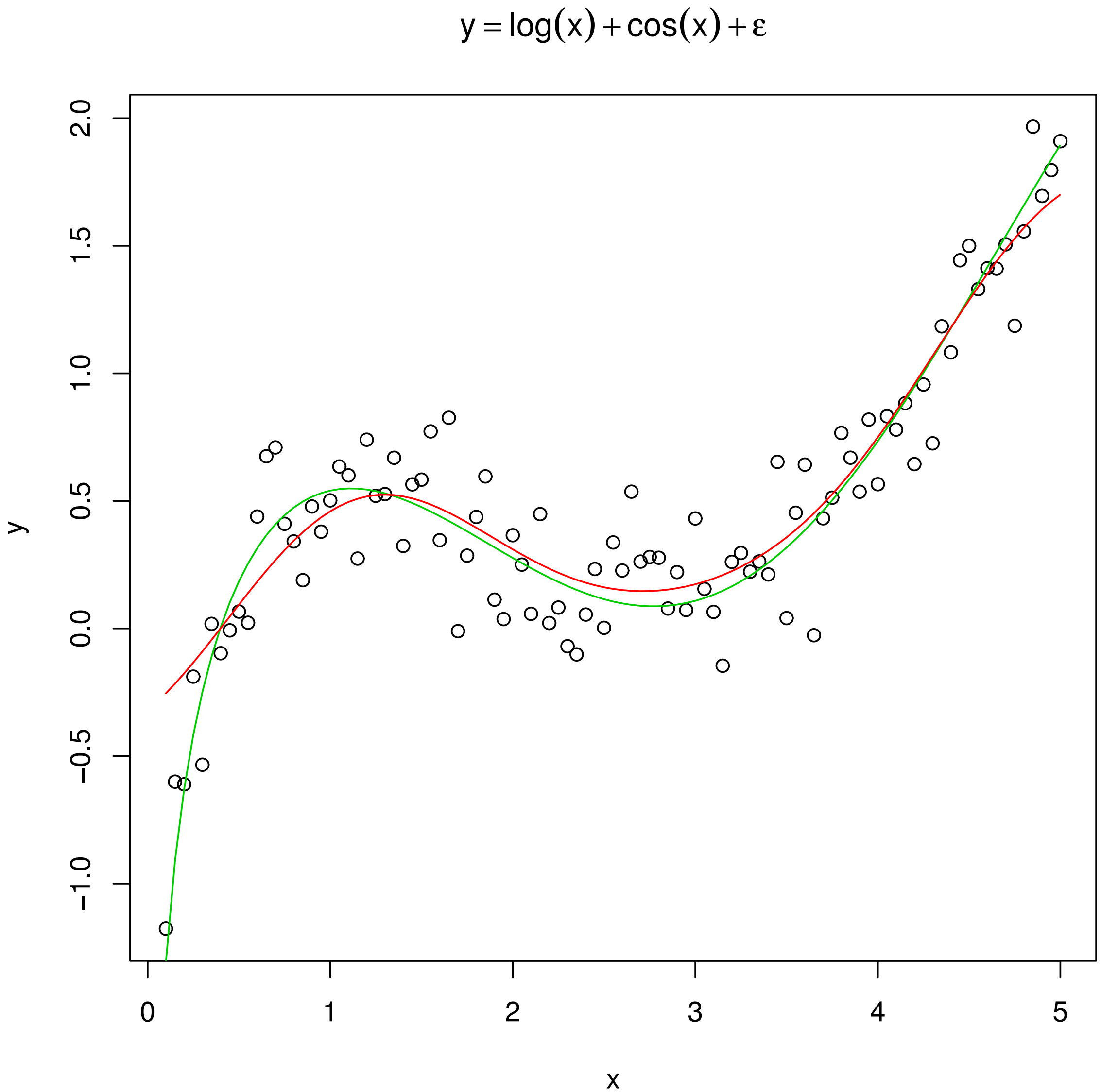
Training



Training

```
R> library(e1071)
R> f <- function(x) log(x) + cos(x)
R> x <- seq(0.1, 5, by = 0.05)
R> y <- f(x) + rnorm(x, sd = 0.2)
R> print(svmmodel <- svm(x, y))

Call:
 svm.default(x = x, y = y)


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  1
    epsilon:  0.1
```

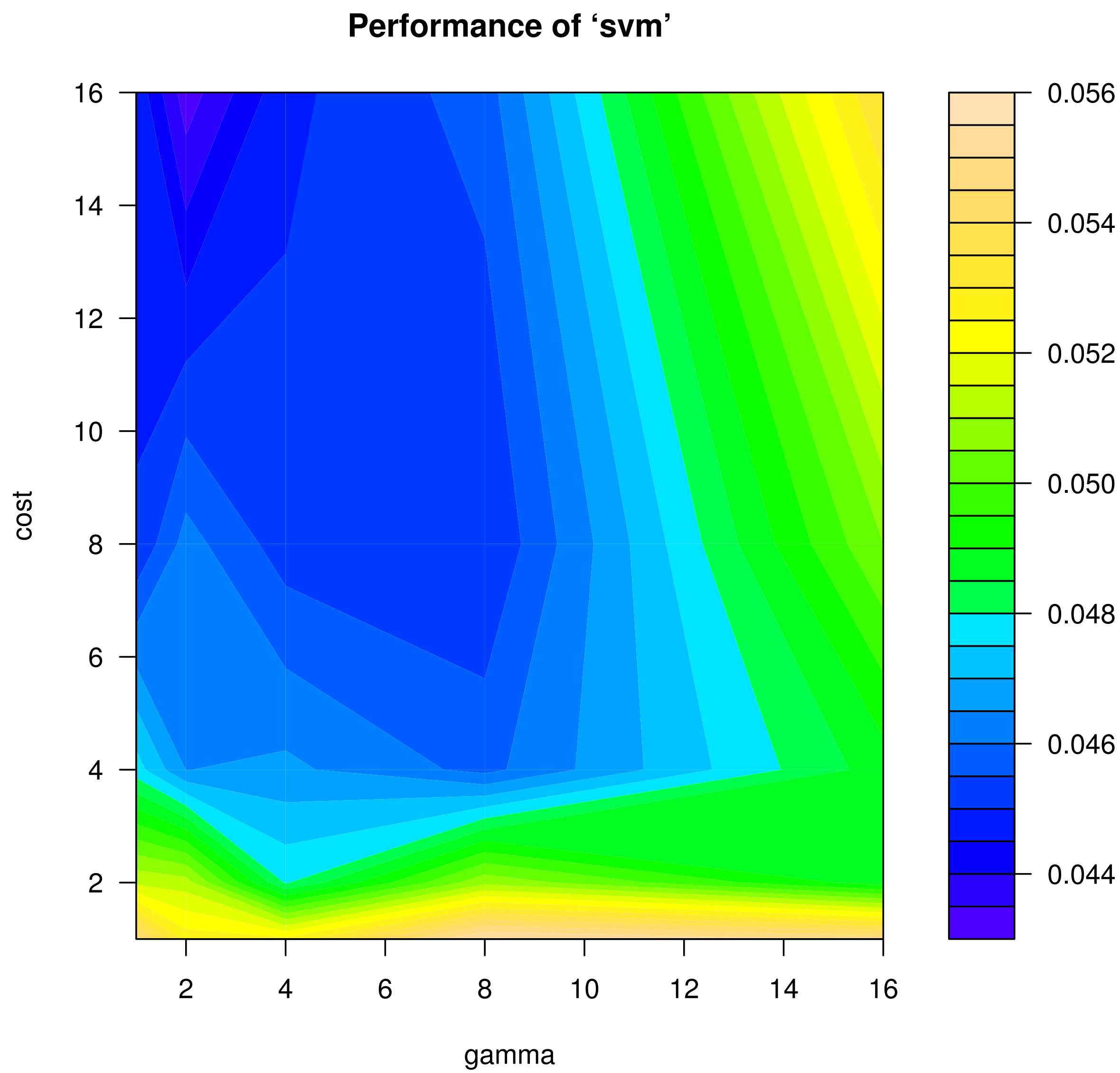
Training



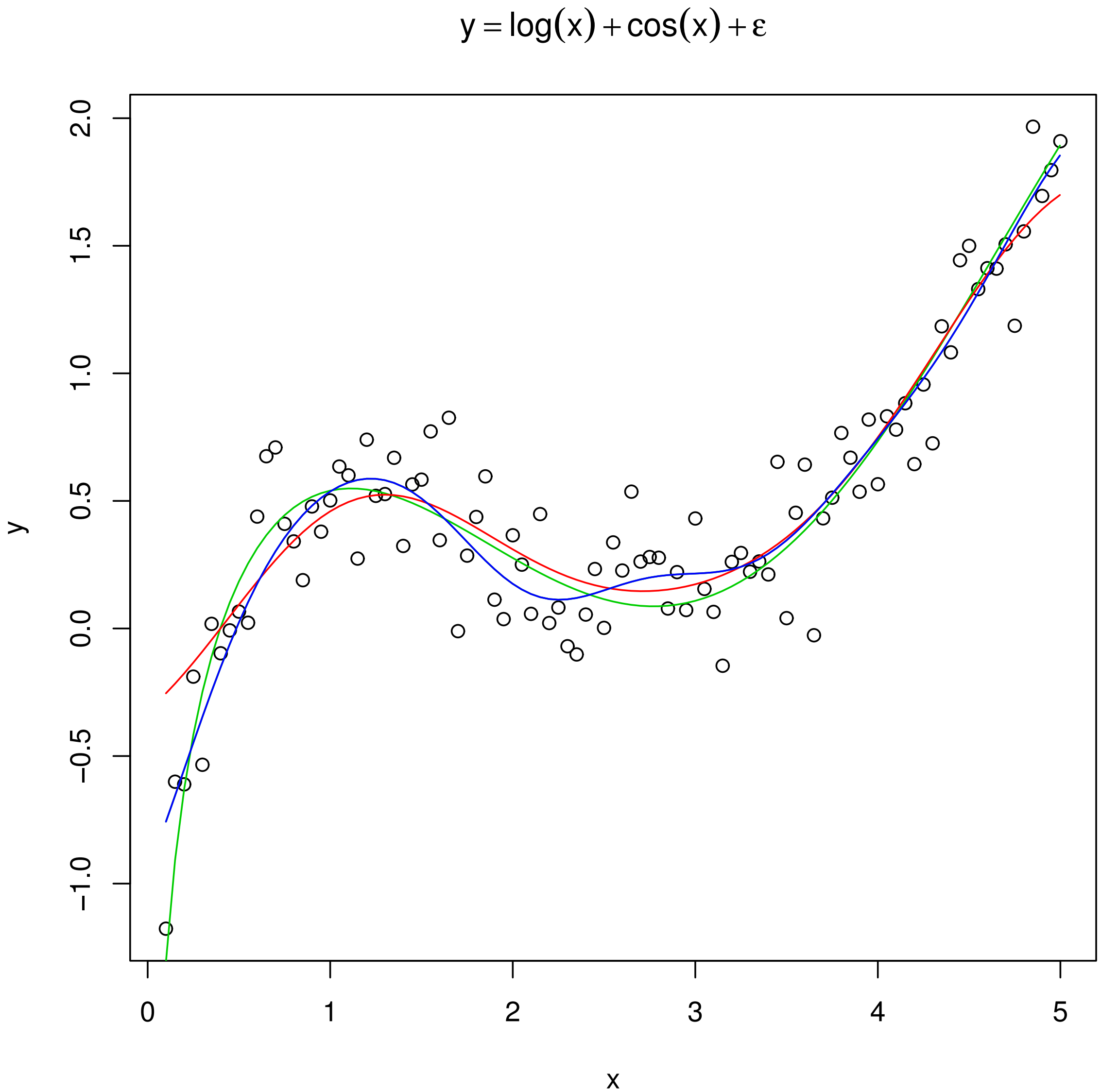
Parameter Tuning

```
R> tunemodel <- tune.svm(x, y, gamma = 2^(-4:0), cost = 2^(-2:2))
R> tunemodel

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
     2   16

- best performance: 0.05016086

R> plot(tunemodel)
```

Parameter Tuning



Parameter Tuning



`libsvm` under test

- No extensive benchmark study available comparing SVMs to many methods on many data sets
- Often only compared to other machine learning techniques (e.g., neural networks)
- How do SVMs compare to 'traditional' methods (e.g., linear models)?
- R offers a wide variety of methods, all accessible with a similar interface $\Rightarrow$ easy to use in a benchmark study

Methods

9 regression methods:

Machine Learning Techniques: Neural Networks, Trees

Resample & Combine Methods: Bagged Trees, Random Forests, MART

Spline Models: MARS, BRUTO, Projection Pursuit

'Traditional' Methods: Linear Models

Data sets

Most data sets were taken from the UCI machine learning data base:

- 12 data sets
- artifical and real data
- small (167) to large (8192) data sets
- Mix of binary, categorical, and metric input variables

Simulation Setup

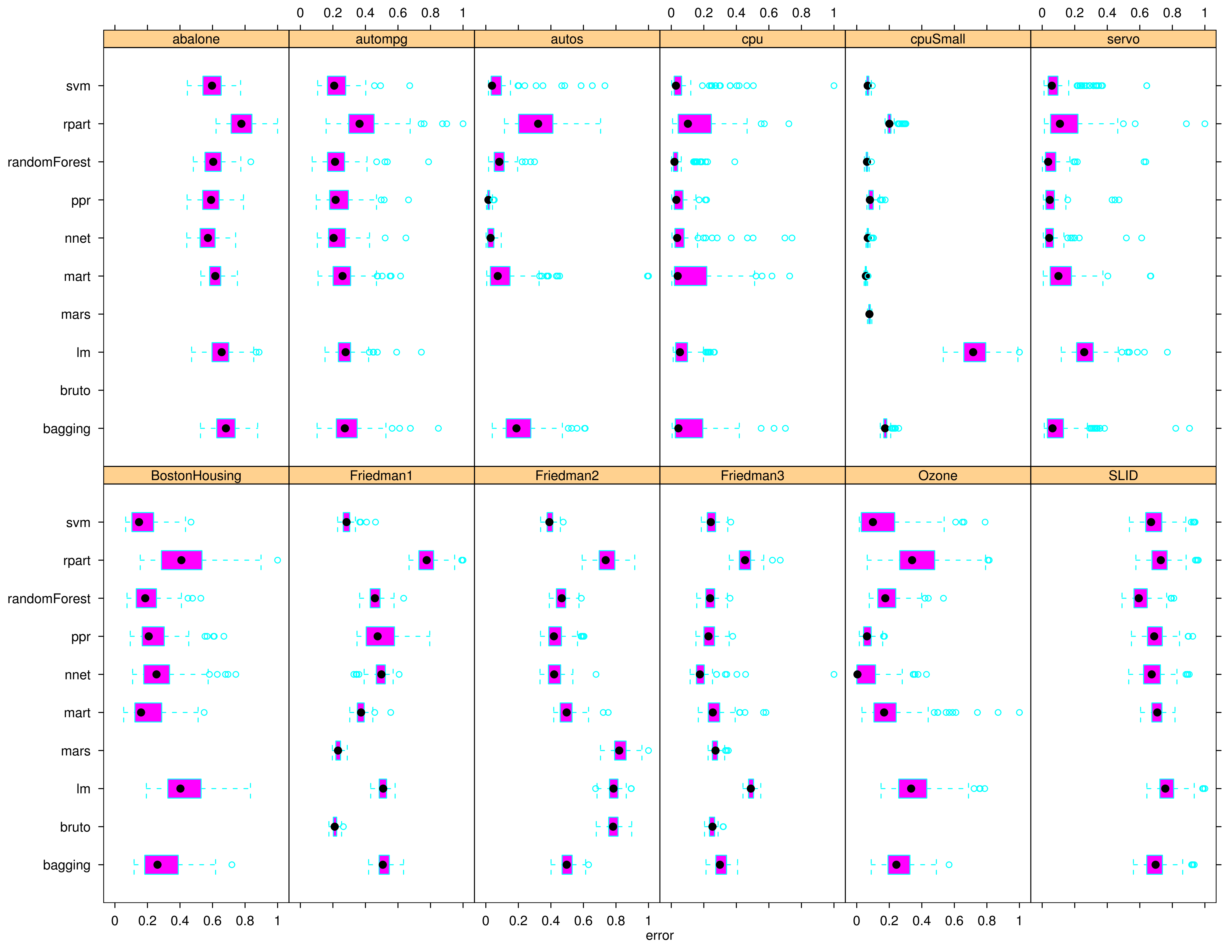
- Generation of 100 training and 100 test sets
  (for real data, using 10 times 10-fold cross validation)

- Scaling of data

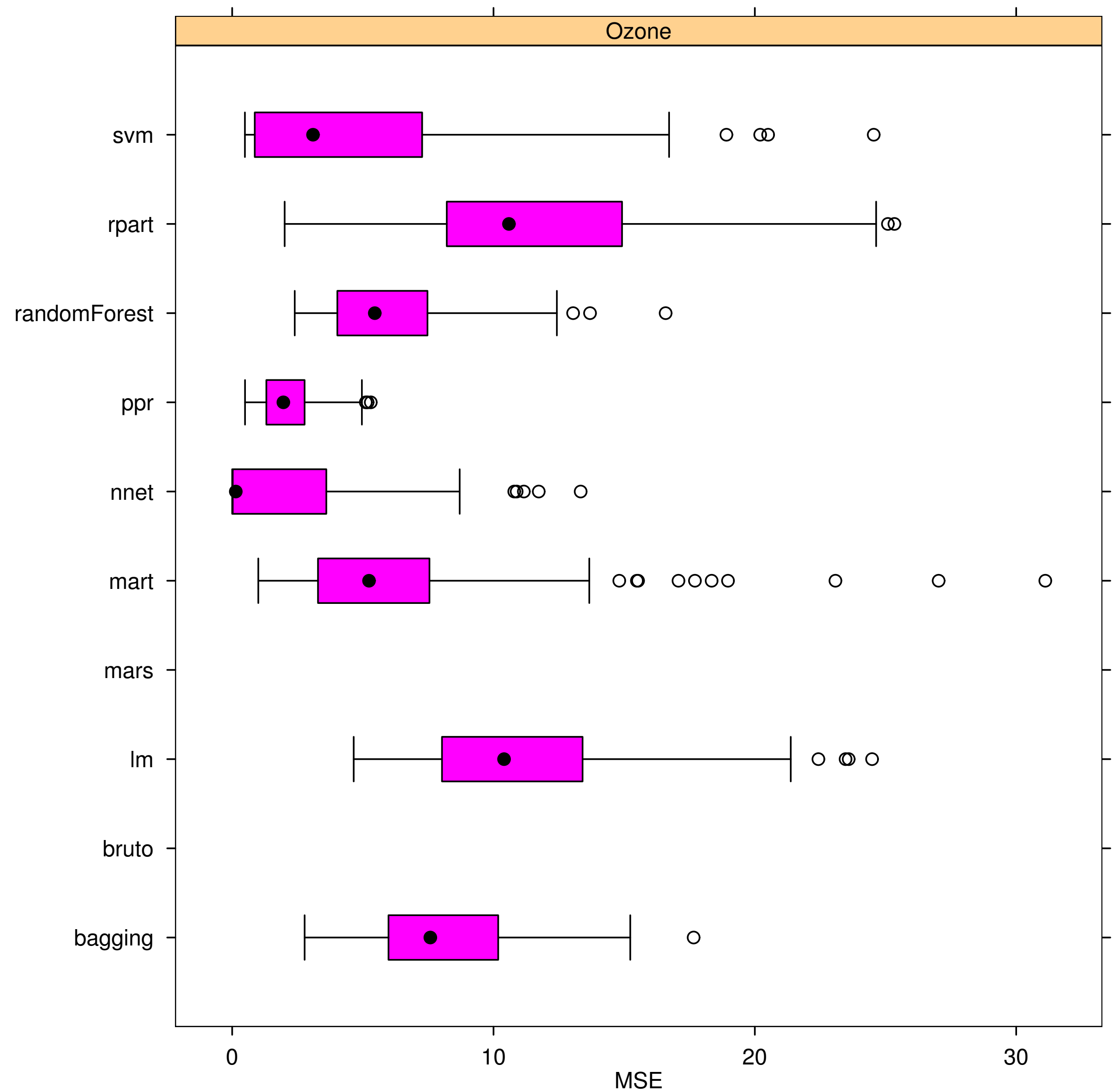
- Tuning on 2/3 of training set, other 1/3 used for validation

- Training on complete training set

- Performance measure computed on test set
  (Mean Squared Error)


abalone
autompg
autos
cpu
cpuSmall
servo
BostonHousing
Friedman1
Friedman2
Friedman3
Ozone
SLID
svm
rpart
randomForest
ppr
nnet
mart
mars
lm
bruto
bagging
0
0.2
0.4
0.6
0.8
1
error

Results



Ozone
svm
rpart
randomForest
ppr
nnet
mart
mars
lm
bruto
bagging
0
10
20
30
MSE

Results

- Generally good performance of SVMs (almost always ranked in top 3)
- However, only ranked first on two data sets
- Average rankings for dispersion
- Good performances by neural networks and random forests

The EUNITE competition

World-wide competition on electricty load prediction (Eastern Slovakia)

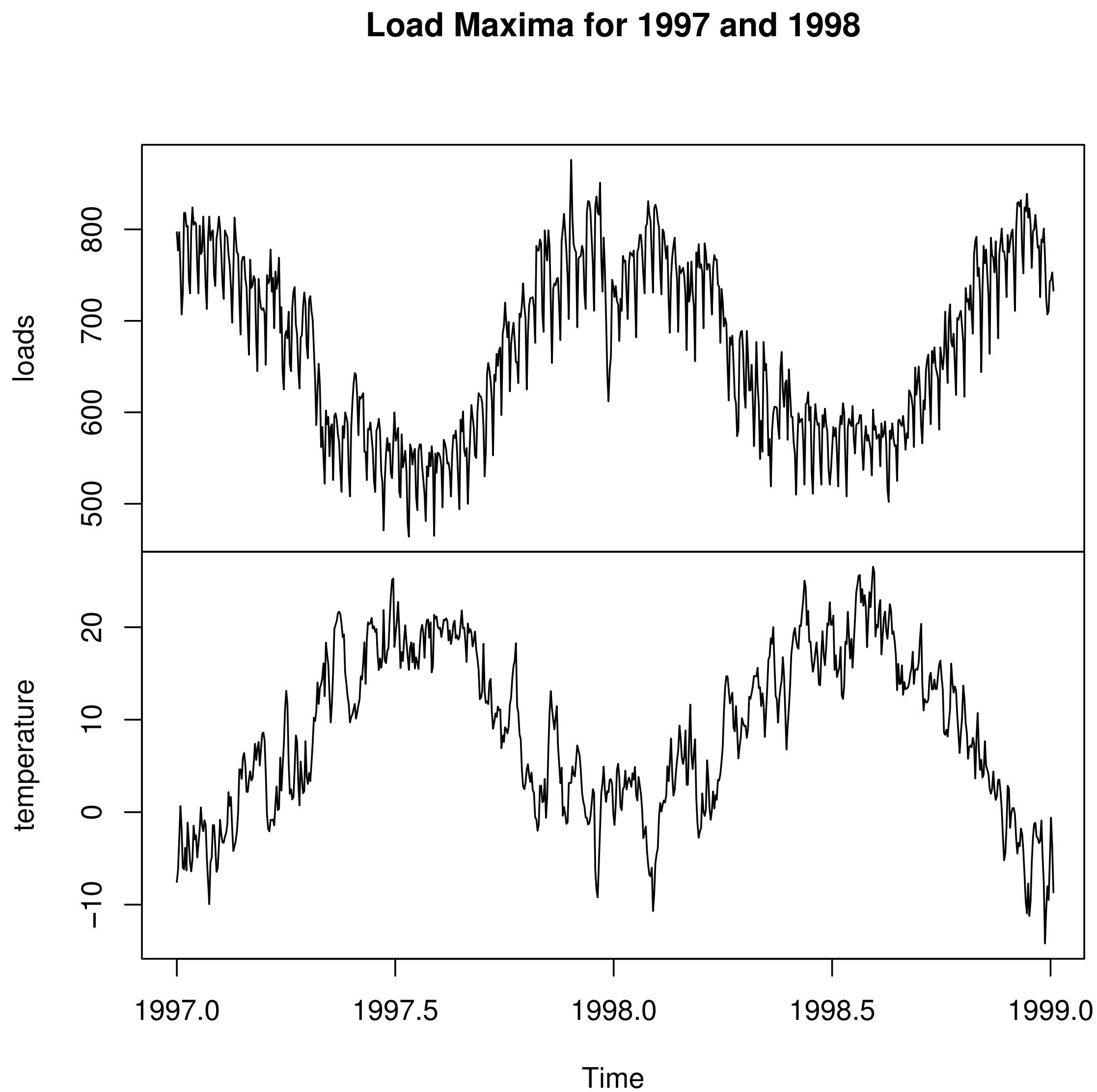
Data: loads (every 30min) and temperatures for 2 years (1997 and 1998)

Task: predict maximum load for a whole month (January 1999)

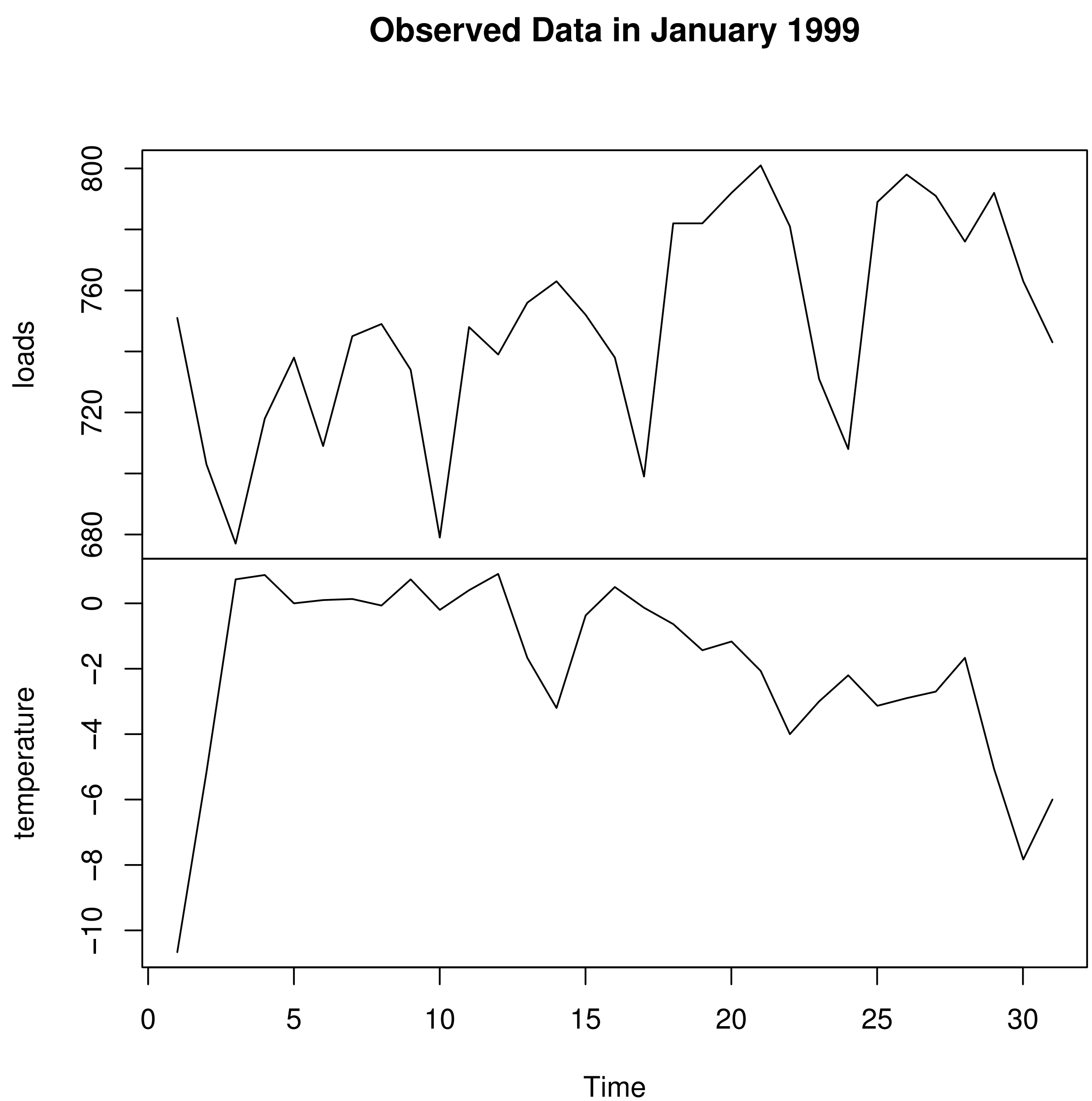
Performance: Mean Average Prediction Error

$$\text{MAPE\%} = 100 \times \sum_{t=1}^{T} \left| \frac{y_t - \widehat{y}_t}{y_t} \right|$$

The EUNITE competition



The EUNITE competition



The EUNITE competition

Data preparation:

- Maximum loads of seven past days
- Seven binary attributes for day of week
- One binary attribute for indicating a holiday
- One attribute for dayly average temperature

After some experiments, some information has been discarded:
– both holiday and temperature
– data in summer time (April–September)

The EUNITE competition

Training:

1. Use January, 1998, as test set, rest as training set

2. Hyperparameters:

   - use RBF kernel ($K(x_i, x_j) = \exp -\gamma \|x_i - x_j\|^2$)
   - use $\epsilon = 0.5$ (default)
   - tune C and γ using 5-fold cross-validation

The EUNITE competition

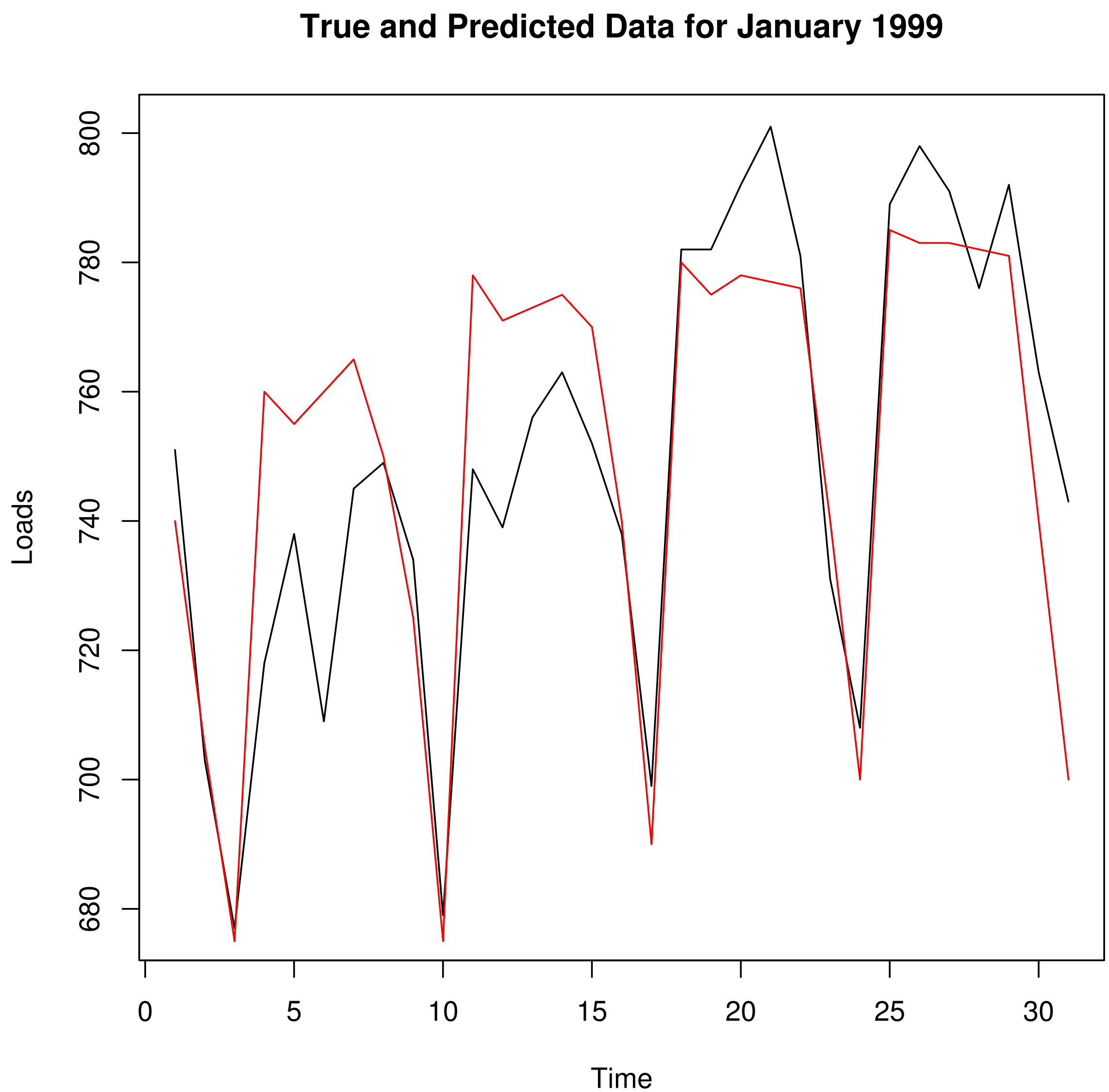
Predicting:

1. Start with loads of last seven days of December, 1998

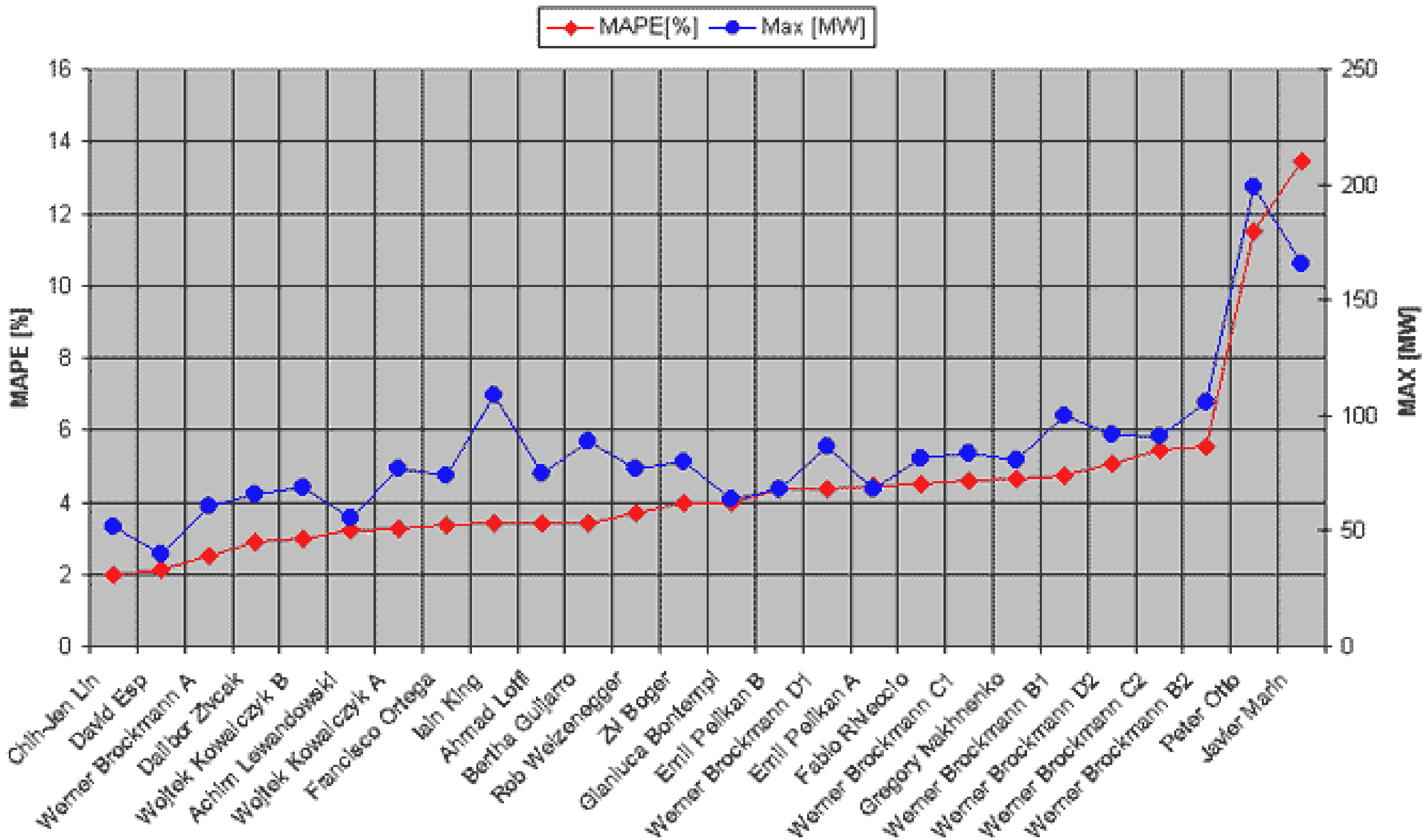
2. Remove first value, add predicted value for next day

3. Repeat until all values are predicted

The EUNITE competition



The EUNITE competition



Resources

- CRAN:
  http://cran.R-project.org/
- `libsvm`:
  http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- EUNITE:
  http://neuron.tuke.sk/competition/
- Benchmarks:
  http://www.wu-wien.ac.at/am/Download/report78.pdf
- This talk & data sets:
  http://www.ci.tuwien.ac.at/~meyer/svm/