Exercises Support Vector Machines

Marjolein Fokkema

Fit SVMs for predicting biodegradability of molecules

Get the QSAR (quantitative structure-activity relationship) dataset from Brightspace. These data comprise physical features (columns) from different chemical compounds (rows). Read in data and separate the data in a training and test set:

```
qsar <- readRDS("qsar.Rda")
set.seed(42)
train <- sample(1:nrow(qsar), size = 700)
test <- which(!(1:nrow(qsar) %in% train))
library("e1071")
```

The first 41 columns of the dataset comprise the following predictors. For this exercise, we are not going to interpret their effects, so no need to read carefully:

1) SpMax_L: Leading eigenvalue from Laplace matrix
2) J_Dz(e): Balaban-like index from Barysz matrix weighted by Sanderson electronegativity
3) nHM: Number of heavy atoms
4) F01[N-N]: Frequency of N-N at topological distance 1
5) F04[C-N]: Frequency of C-N at topological distance 4
6) NssssC: Number of atoms of type ssssC
7) nCb-: Number of substituted benzene C(sp2)
8) C%: Percentage of C atoms
9) nCp: Number of terminal primary C(sp3)
10) nO: Number of oxygen atoms
11) F03[C-N]: Frequency of C-N at topological distance 3
12) SdssC: Sum of dssC E-states
13) HyWi_B(m): Hyper-Wiener-like index (log function) from Burden matrix weighted by mass
14) LOC: Lopping centric index
15) SM6_L: Spectral moment of order 6 from Laplace matrix
16) F03[C-O]: Frequency of C - O at topological distance 3
17) Me: Mean atomic Sanderson electronegativity (scaled on Carbon atom)
18) Mi: Mean first ionization potential (scaled on Carbon atom)
19) nN-N: Number of N hydrazines
20) nArNO2: Number of nitro groups (aromatic)
21) nCRX3: Number of CRX3
22) SpPosA_B(p): Normalized spectral positive sum from Burden matrix weighted by polarizability
23) nCIR: Number of circuits
24) B01[C-Br]: Presence/absence of C - Br at topological distance 1
25) B03[C-Cl]: Presence/absence of C - Cl at topological distance 3
26) N-073: Ar2NH / Ar3N / Ar2N-Al / R..N..R
27) SpMax_A: Leading eigenvalue from adjacency matrix (Lovasz-Pelikan index)
28) Psi_i_1d: Intrinsic state pseudoconnectivity index - type 1d

29) B04[C-Br]: Presence/absence of C - Br at topological distance 4
30) SdO: Sum of dO E-states
31) TI2_L: Second Mohar index from Laplace matrix
32) nCrt: Number of ring tertiary C(sp3)
33) C-026: R–CX–R
34) F02[C-N]: Frequency of C - N at topological distance 2
35) nHDon: Number of donor atoms for H-bonds (N and O)
36) SpMax_B(m): Leading eigenvalue from Burden matrix weighted by mass
37) Psi_i_A: Intrinsic state pseudoconnectivity index - type S average
38) nN: Number of Nitrogen atoms
39) SM6_B(m): Spectral moment of order 6 from Burden matrix weighted by mass
40) nArCOOR: Number of esters (aromatic)
41) nX: Number of halogen atoms

The final column comprises the response:

42) experimental class: ready biodegradable (RB) and not ready biodegradable (NRB)

You will fit SVMs with three different kernels. Use package `e1071` to fit the SVMs. Specifically, use function `tune()` to tune the SVM parameters, use function `svm()` to fit the SVM.

```
library("e1071")
?tune
?svm
```

First, a side note: In practice, one should avoid fitting a range of SVMs with different kernels on the training data and evaluate performance on the test data. This would overuse the test data for choosing the optimal fitting parameters. The predictive performance for the best model on the test data is then more likely to provide an overly optimistic indication of future prediction error. Only for the sake of exercise do we do it here.

Part 1: Linear kernel

Tune the cost parameter using function `tune()`. Specify `kernel = "linear"` and `ranges = list(cost = c(0.001, 0.01, 0.1, 1, 10, 100))`.

a) Inspect the result by plotting the resulting object and printing the `$performances` slot of the resulting object. Evaluate the CV results: Did you obtain a convex curve? Was the resolution of the grid fine enough? Likely, you may want to expand the grid a little, by adding a few more extreme, or intermediate values to the grid. Expand the grid accordingly and reapply the tuning procedure.

With the optimal cost parameter, fit the SVM.

b) Make a plot of the decision boundary for the `SpMax_Bm` and `SpMax_L` variables as follows:

```
plot(svmfit, qsar[train, ], SpMax_Bm ~ SpMax_L)
```

Likely, you will not see a decision boundary, because this plot keeps other variables fixed at a value of zero (see also `slice` argument in `?plot.svm`). Many of the variables have a mean far from 0, so you need to adjust the `slice` argument until you can see the decision boundary. E.g.:

```
## Create a list of values close to the mean for the features
slice <- as.list(sapply(qsar[ , - which(names(qsar) == "class")], mean) - .2)
## Note you may have to adjust the -0.2 value in order to visualize the boundary well
plot(svmfit, qsar[train, ], SpMax_Bm ~ SpMax_L, slice = slice)
```

c) Using function `predict`, compute the test MCR.

Part 2: Polynomial kernel

Tune the budget parameter and degree of the polynomial kernel by passing the following grid to the `ranges` parameter of function `tune` (again make sure to also specify the `kernel` argument):

```
ranges = list(cost = cost, degree = c(2, 3, 4, 5))
```

a) Why was it not necessary to include a degree of 1?

b) Inspect the CV result by plotting and printing the `$performances` slot of the resulting object. Evaluate the CV results: Did you obtain a convex curve? Was the resolution of the grid fine enough?

The contour plot returned by `plot.tune` when tuning multiple SVM parameters might not be clear enough. It may be helpful to create a custom plot, with one of the tuning parameters on the x-axis, and separate lines for the levels of the other, e.g.:

```
library("ggplot2")
perf <- tune.out$performances
perf$cost <- factor(perf$cost)
ggplot(perf) + geom_line(aes(degree, error, group=cost, color=cost))
```

If necessary, expand the tuning grid and reapply the tuning procedure.

c) With the optimal cost and polynomial degree, fit the SVM with polynomial kernel. Again make a plot of the decision boundary, and compute MCR for the test observations.

d) How does predictive performance on test observations compare with the linear-kernel SVM?

Part 3: Radial basis kernel

Tune the budget parameter and degree of the polynomial kernel by using the following grid (again make sure to also specify the `kernel` argument):

```
ranges = list(cost = cost, gamma = c(.001, .01, .1, 10))
```

a) Inspect the CV result by plotting and printing the `$performances` slot of the resulting object. Evaluate the CV results: Did you obtain a convex curve? Was the resolution of the grid fine enough?

If necessary, expand the tuning grid and reapply the tuning procedure.

b) With the optimal cost and γ value, again fit an SVM. Again make a plot of the decision boundary, and compute MCR for the test observations.

c) Compare predictive performance on test observations with all earlier prediction models you fitted.