Support vector machines

Chapter 7-Part II Regression Models

Motivations

- SVM was developed in the computer science community in the 1990s, and it has been considered one of the best "out of the box" classifiers.
- In Chapter 6, we discussed linear regression model, which finds parameter estimates based on ordinary least squares (OLS) method. Given the training data, the OLS estimate the regression coefficients by minimizing

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where for a training data given by

$$\hat{y} = \beta_0 + \beta_1 x_1 + \cdots + \beta_P x_P$$

- Disadvantage: This method is very sensitive to outliers and influential observation.

SVM

- Within the framework of *robust* regression, we seek to finds parameter estimates while minimizing the effect of outliers on the regression equations.
- One way is to replace the squared error loss function with *ε-insensitive robust loss function, such that*

$$\sum_{i=1}^{n} L_{\varepsilon}(y_i - \hat{y}_i)$$

where Lε(·) is the epsilon-insensitive robust loss function.

SVM

- The SVM regression coefficients minimize (like ridge regression)

$$\text{Cost} \sum_{i=1}^{n} L_{\varepsilon}(y_i - \hat{y}_i) + \sum_{i=1}^{n} \beta_j^2$$

where *Cost* (C) parameter is the cost penalty that is set by the user, which penalizes large residuals.

- SVM model has two tuning parameters
  - The first tuning parameter, ε (scale), controls the training set samples that are within ± ε of the regression line (i.e., are within the "funnel" or "tube" around the regression line). A small value is often suggested.
  - The second tuning parameter is the cost parameter C.
- SVMs are a class of powerful, *highly flexible* modeling techniques.

The cost parameter

- C controls the bias-variance trade-off of the SVM
  - When the cost (C) is large, the model becomes very flexible.
  - When the cost (C) is small, the model becomes less likely to over-fit (but more likely to under-fit).
- In our experience, we have found that the cost parameter provides more flexibility for tuning the model. So, we suggest fixing a value for ε and tuning over the other kernel parameters.
- We recommend *centering and scaling* the predictors prior to building an SVM model.

SVM

- For a new sample, $u = (u_1, \ldots, u_P)$, the prediction equation is

$$\begin{aligned}\hat{y} &= \beta_0 + \beta_1 u_1 + \cdots + \beta_P u_P \\ &= \beta_0 + \sum_{j=1}^{P} \beta_j u_j \\ &= \beta_0 + \sum_{j=1}^{P} \sum_{i=1}^{n} \alpha_i x_{ij}\, u_j \\ &= \beta_0 + \sum_{i=1}^{n} \alpha_i \left(\sum_{j=1}^{P} x_{ij} u_j \right) \\ &= \beta_0 + \sum_{i=1}^{n} \alpha_i\, K(\boldsymbol{x_i}, \boldsymbol{u})\end{aligned}$$

where $K(\boldsymbol{x}_i, \boldsymbol{u})$ is called *kernel function*.

Types of kernel functions

- There are three popular kernel functions that can be used to generalize the regression model and encompass *nonlinear* functions of the predictors

$$\text{polynomial} = (\varphi(\boldsymbol{x}'\boldsymbol{u}) + 1)^{\text{degree}}$$

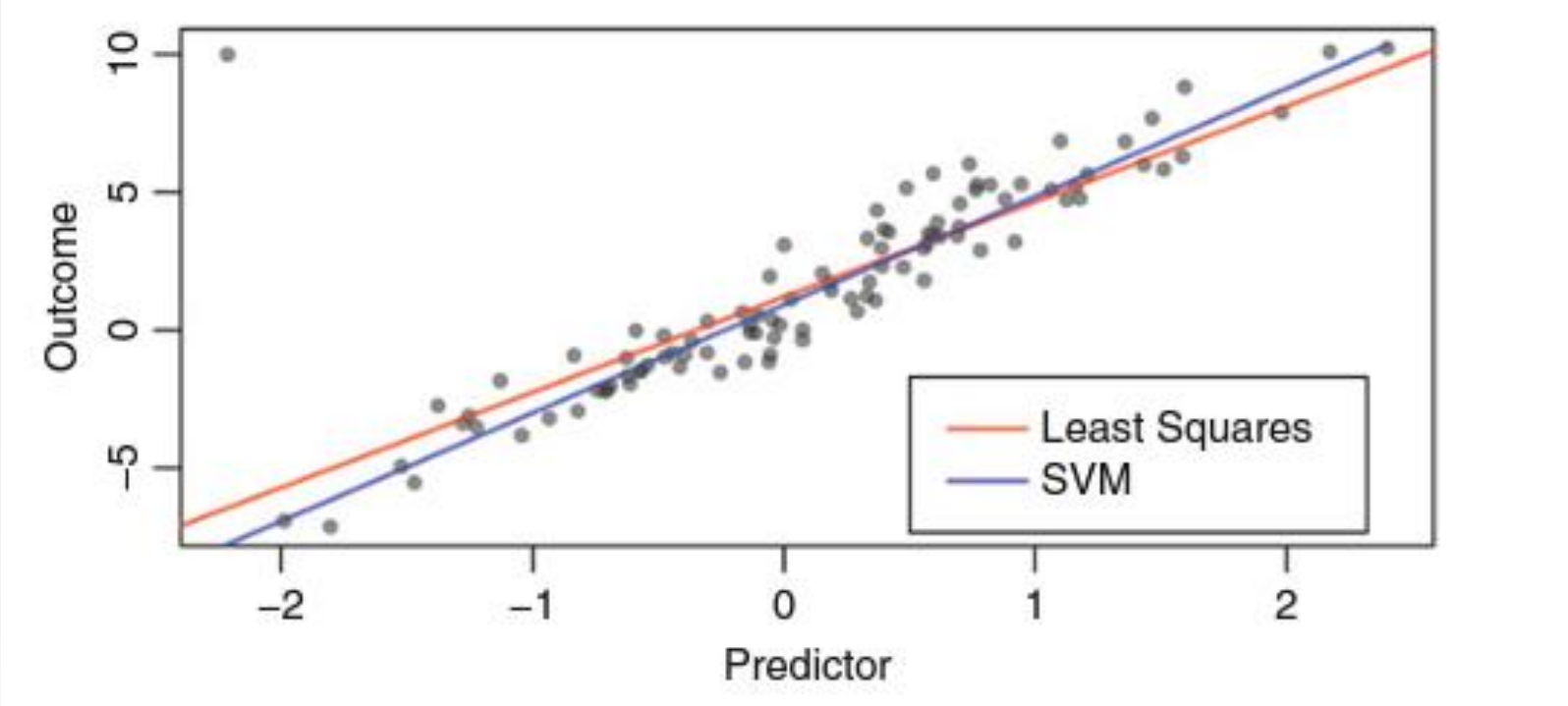
$$\text{radial basis function} = \exp(-\sigma||\boldsymbol{x} - \boldsymbol{u}||^2)$$

$$\text{Hyperbolic tangent} = \tanh(\varphi(\boldsymbol{x}'\boldsymbol{u}) + 1)$$

where φ and σ are scaling parameters.

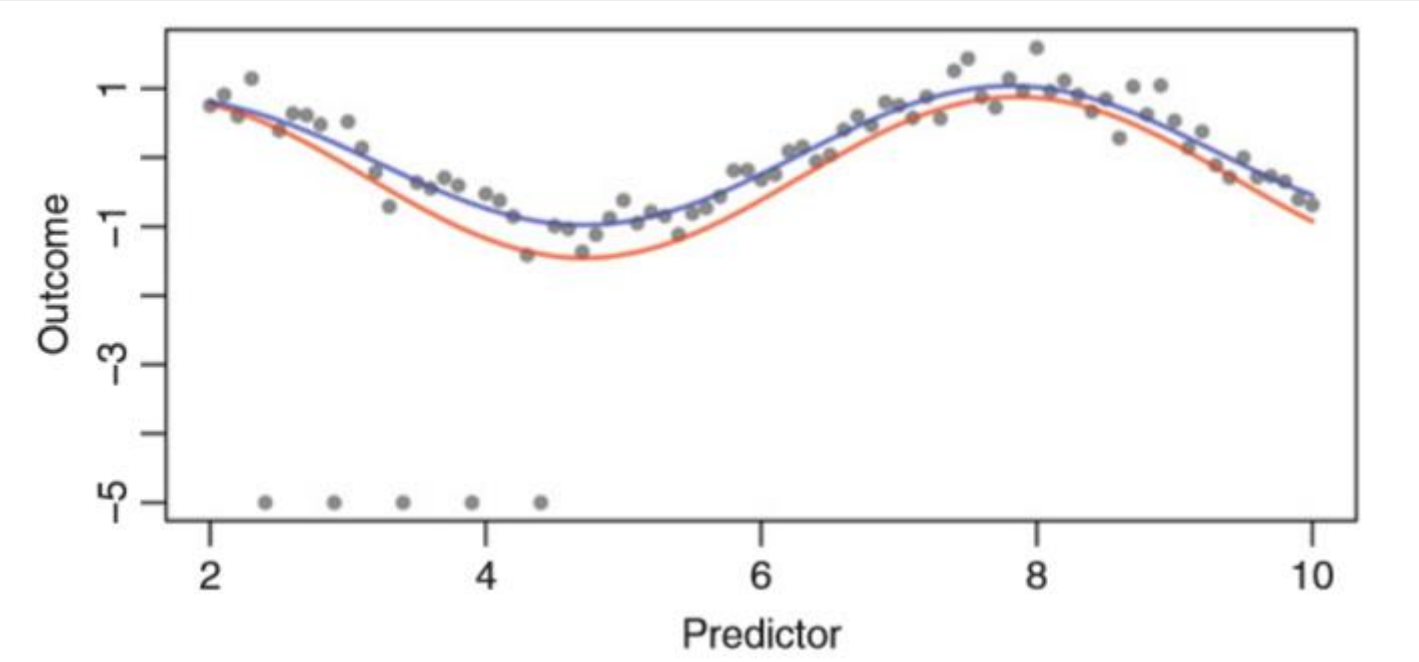
- Since these functions of the predictors lead to nonlinear models, this generalization is often called the "kernel trick."

The robustness of the SVM model



A simulated data with a single large outlier. The red line is an ordinary regression line, and the blue line is a radial basis function SVM model.

Adaptability to nonlinear relationship



A simulated sin wave with several outliers. The red line is an ordinary regression line (intercept and a term for sin(x)) and the blue line is a radial basis function SVM model.

Remarks

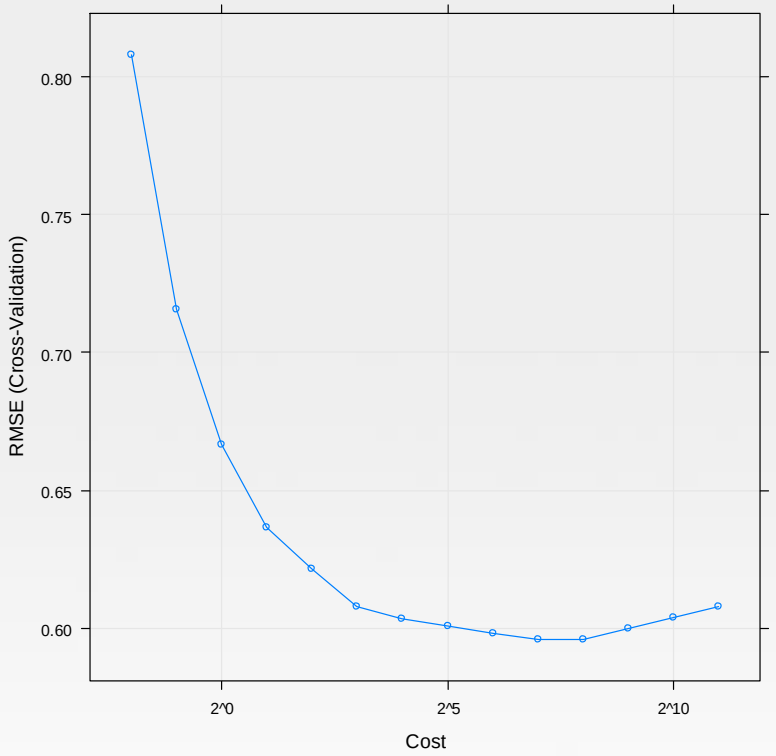
- Which kernel function should be used? This depends on the problem.
  - The radial basis function has been shown to be very effective.
  - When the regression line is truly linear, the linear kernel function will be a better.

SVM with radial basis function

```
### Support Vector Machines
## In a recent update to caret, the method to estimate the
## sigma parameter was slightly changed. These results will
## slightly differ from the text for that reason.

#SVM with the radial basis function function
ptm <- proc.time() # Takes 72.23 seconds in my computer
set.seed(100)
svmRTune <- train(x = solTrainXtrans, y = solTrainY,
                  method = "svmRadial",
                  preProc = c("center", "scale"),
                  tuneLength = 14,
                  trControl = ctrl)
svmRTune
proc.time() – ptm
plot(svmRTune, scales = list(x = list(log = 2)))
##save the predicted values into testResults
testResults$SVMr <- predict(svmRTune, solTestXtrans)
```

```
> svmRTune
Support Vector Machines with Radial Basis Function Kernel

951 samples
228 predictors

Pre-processing: centered (228), scaled (228)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 856, 855, 857, 856, 856, 855, ...
Resampling results across tuning parameters:

  C        RMSE       Rsquared   MAE
    0.25   0.8082294  0.8636665  0.6028126
    0.50   0.7158045  0.8849736  0.5315914
    1.00   0.6666914  0.8966605  0.4957241
    2.00   0.6363044  0.9037211  0.4692192
    4.00   0.6216719  0.9070653  0.4543005
    8.00   0.6077567  0.9109752  0.4430939
   16.00   0.6031571  0.9123541  0.4393460
   32.00   0.6007413  0.9132360  0.4383832
   64.00   0.5981252  0.9142755  0.4362848
  128.00   0.5958107  0.9151748  0.4353230
  256.00   0.5957323  0.9155000  0.4377823
  512.00   0.5997785  0.9145462  0.4415741
 1024.00   0.6038872  0.9136810  0.4440797
 2048.00   0.6079735  0.9127263  0.4476578

Tuning parameter 'sigma' was held constant at a value of 0.00265531
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 0.00265531 and C = 256.
```

The cost parameter



- The cross-validation profiles for a radial basis function SVM model applied to the solubility data.

SVM with polynomial

```
ptm <- proc.time() # takes 313.55 second to run
#SVM with the polynomial basis function
svmGrid <- expand.grid(degree = 1:2,
                       scale = c(0.01, 0.005, 0.001),
                       C = 2^(-2:5))
set.seed(100)
svmPTune <- train(x = solTrainXtrans, y = solTrainY,
                  method = "svmPoly",
                  preProc = c("center", "scale"),
                  tuneGrid = svmGrid,
                  trControl = ctrl)

svmPTune
proc.time() - ptm
plot(svmPTune, scales = list(x = list(log = 2),  between = list(x = .5, y = 1)))

##save the predicted values into testResults
testResults$SVMp <- predict(svmPTune, solTestXtrans)
```

Tuning parameters

```
> svmPTune
Support Vector Machines with Polynomial Kernel

951 samples
228 predictors

Pre-processing: centered (228), scaled (228)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 856, 855, 857, 856, 856, 855, ...
Resampling results across tuning parameters:

  degree  scale  C      RMSE       Rsquared   MAE
  1       0.001   0.25  1.0523656  0.7929664  0.7860298
  1       0.001   0.50  0.8983142  0.8327514  0.6759869
  1       0.001   1.00  0.7985992  0.8561235  0.6049115
```

.

.

.

.

```
  2       0.010   8.00  0.6207772  0.9088222  0.4563315
  2       0.010  16.00  0.6228678  0.9086508  0.4579929
  2       0.010  32.00  0.6282422  0.9072207  0.4635977

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were degree = 2, scale = 0.01 and C = 2.
```

The cost parameter