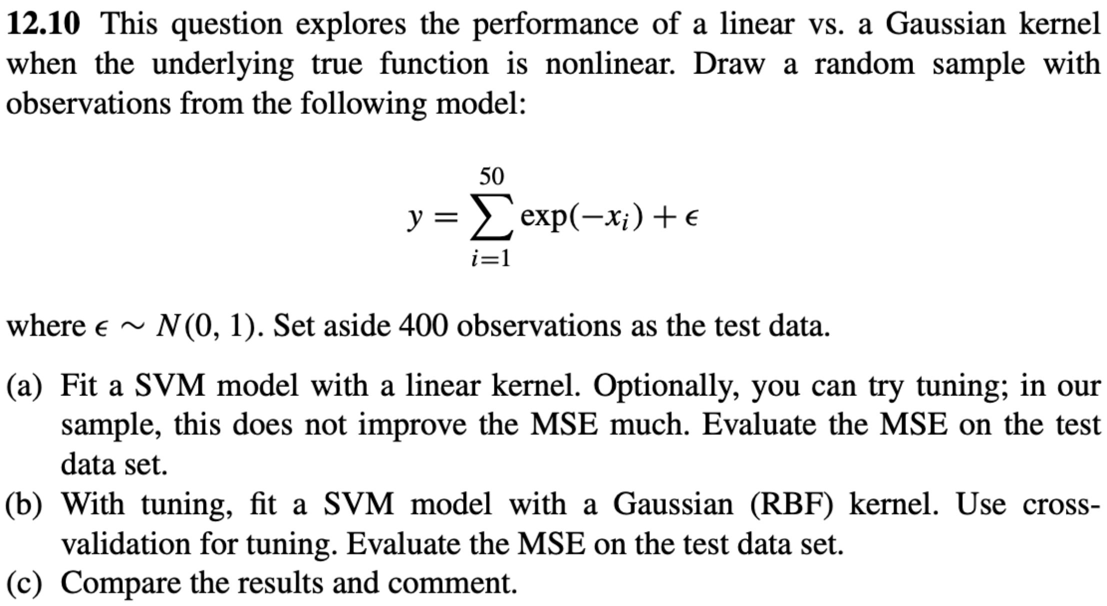
Q12.10

March 12, 2024



12.10 This question explores the performance of a linear vs. a Gaussian kernel when the underlying true function is nonlinear. Draw a random sample with observations from the following model:

$$y = \sum_{i=1}^{50} \exp(-x_i) + \epsilon$$

where $\epsilon \sim N(0, 1)$. Set aside 400 observations as the test data.

(a) Fit a SVM model with a linear kernel. Optionally, you can try tuning; in our sample, this does not improve the MSE much. Evaluate the MSE on the test data set.

(b) With tuning, fit a SVM model with a Gaussian (RBF) kernel. Use cross-validation for tuning. Evaluate the MSE on the test data set.

(c) Compare the results and comment.

[ ]:

```
# load in e1071
library(e1071)
```

[ ]:

```
# set the seed
set.seed(32908)
# sample and feature sizes
n_samples = 1000
n_features = 50
# draw the samples
X = matrix(rnorm(n_samples * n_features), n_samples, n_features)
y = rowSums(exp(-X)) + rnorm(n_samples)
# set the train and test data
trainidx = 1:600
X_train = X[trainidx, ]
y_train = y[trainidx]
X_test = X[-trainidx,]
y_test = y[-trainidx]
```

0.0.1 Part a)

SVM with a linear kernel.

[ ]:

```
# use the svm (courtesy of e1071)
svm_linear = svm(X_train, y_train, type = 'eps-regression', kernel = 'linear')
# predict on the test data
linear_pred = predict(svm_linear, X_test)
# calculate the MSE
linear_mse = mean((y_test - linear_pred)^2)
```

$MSE_{linear} = 96.7694439467381 $

0.0.2 Part b)

SVM with a radial kernel, and tuning.

[ ]:

```
# set the seed again
set.seed(4930248)
tune_result = tune(svm, train.x = X_train, train.y = y_train,
                   type = 'eps-regression', kernel = 'radial',
                   ranges = list(cost = 10^(-1:2),
                                 gamma = 10^(-3:0),
                                 epsilon = seq(0, 0.5, by = 0.025)))
# get the best model from CV
best_rbf_svm = tune_result$best.model
# predict the best model on the test data
rbf_pred = predict(best_rbf_svm, X_test)
# calculate the mse
rbf_mse = mean((y_test - rbf_pred)^2)
list(linear_mse = linear_mse, rbf_mse = rbf_mse, best_parameters = tune_result$best.parameters)
```

$linear_mse 96.7694439467381

$rbf_mse 35.3926631679522

$best_parameters A data.frame: 1 × 3

	cost <dbl>	gamma <dbl>	epsilon <dbl>
84	100	0.001	0.125

We report the best model as having $c = 100, \gamma = 0.001, \epsilon = 0.125$.

This leads to $MSE_{\text{radial}} = 35.3926631679522$

0.0.3 Part c)

Comparing results…

To recall, we have: - $MSE_{linear} = 96.7694439467381 $, and - $MSE_{\text{radial}} = 35.3926631679522$

There is a **significantly** lower MSE with the radial kernel, being about 37% of the linear MSE.

This is probably due to the obvious non-linearity of the data. Since the linear kernel is... well... linear, there are sure to be shortcomings when the data is not so ideal. So, the radial basis function as a kernel can better handle the non-linear data.