Data Mining

Contents

From Azzalini and Scarpa, Chapter 3

A simple prototype problem

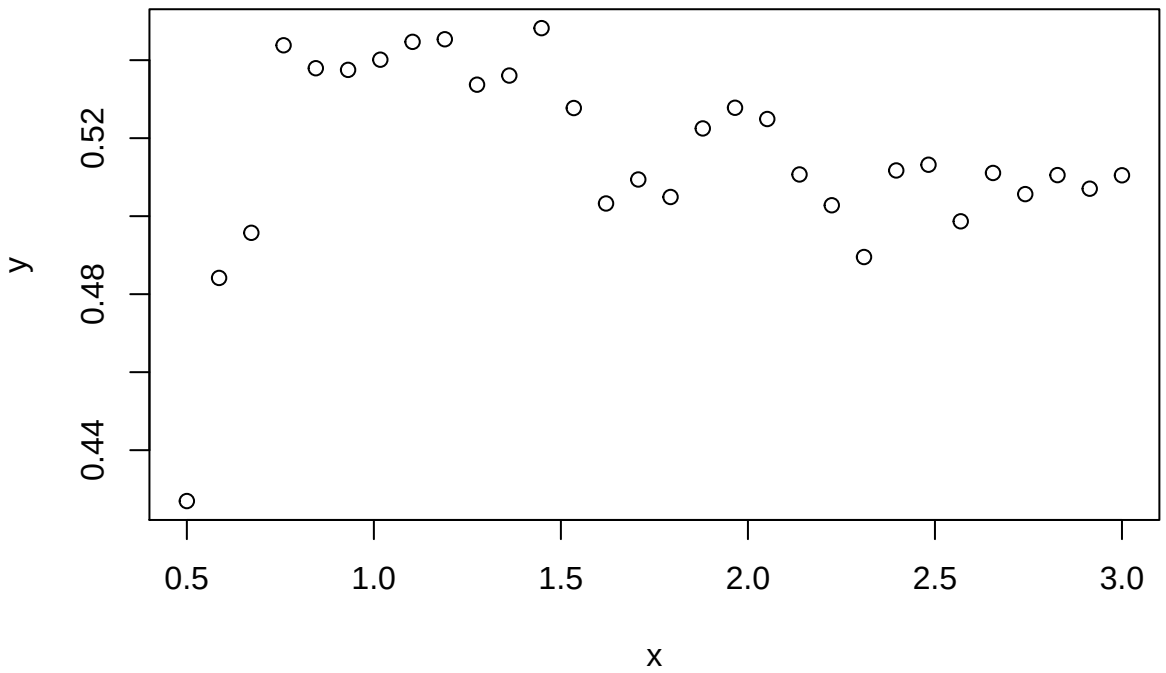
We consider here a very simple example serving as the prototype for much more complex and realistic circumstances. Let us presume that yesterday we observed $n = 30$ pairs of data, the **training set**

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

```
# import data from the web
library(readr)
df <- read_table2("http://azzalini.stat.unipd.it/Book-DM/yesterday.dat")[-31,]
# create training set
train <- data.frame(x=df$x, y=df$y.yesterday)
```

The data are shown in the scatterplot:

```
plot( y ~ x , train)
```



The data were generated artificially by an equation such as

$$y = f(x) + \varepsilon$$

where ε is an error component with distribution $N(0, \sigma^2)$ and $\sigma = 10^{-2}$; $f(x)$ is a function which we leave unspecified - the only requirement is that this function should follow an essentially regular trend.

Clearly, to generate the data, we had to choose a specific function (not a polynomial), but we do not disclose our choice.

Say we wish to obtain an estimate of $f(x)$ today that allows us to predict y as new observations of x become available, i.e. we need to fit a predictive model

$$\hat{y} = \hat{f}(x)$$

Tomorrow data, or the **test set**, will be like

$$(x_1, y_1^*), (x_2, y_2^*), \ldots, (x_n, y_n^*)$$

Here the new observations

- are n as in the training set
- have new y but old (train) x

This is a particular case which we will call **fixed-x setting** .

In general the test set is of size m and have both new x and new y

$$(x_1^*, y_1^*), (x_2^*, y_2^*), \ldots, (x_m^*, y_m^*)$$

Mean squared error

For the yesterday data (training data), we can compute the **mean squared error** (MSE)

$$\text{MSE}_{\text{Tr}} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$$

However, we would like to have a good performance on the test MSE

$$\text{MSE}_{\text{Te}} = \frac{1}{n} \sum_{i=1}^{n} (y_i^* - \hat{f}(x_i))^2$$

Polynomial regression

A reasonable choice consists of using polynomial regression (of degree d)

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_d x^d$$

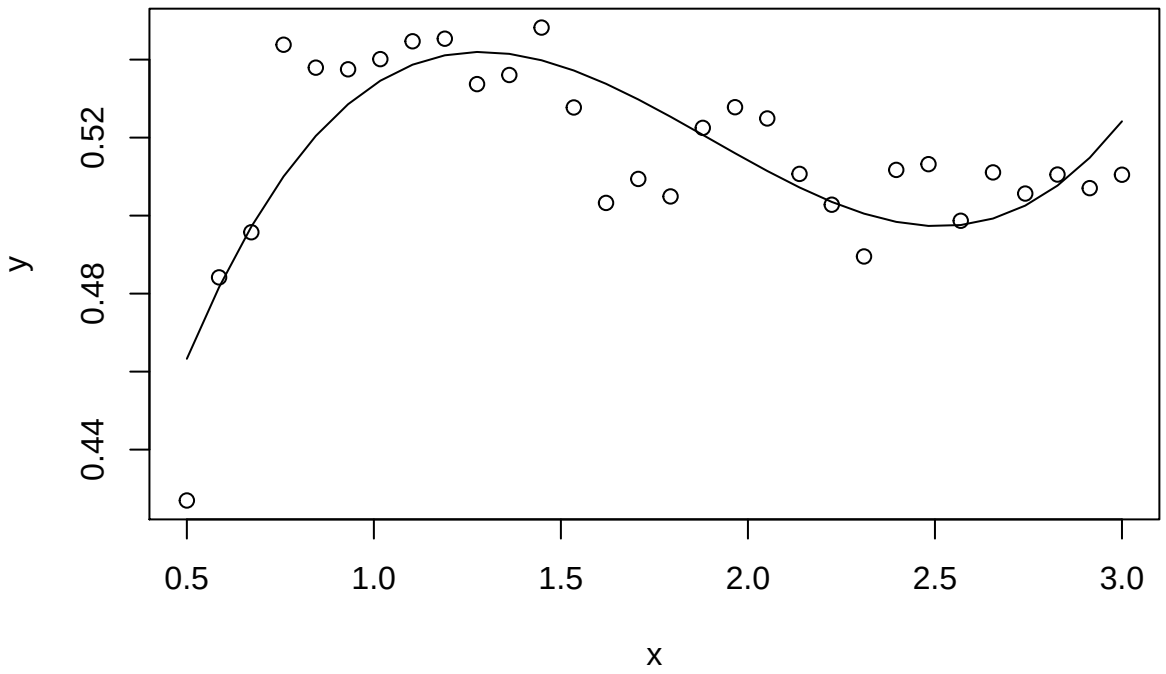
If we have no information to guide us in choosing the degree d of the polynomial, we first consider all possible degrees from 0 to $n - 1$, thereby introducing $p = d + 1$ parameters ranging from 1 to n, in addition to σ.

Let's try a 3rd degree polynomial model ($d = 3$)

```r
# create test set
test = data.frame(x=train$x)

# 3rd degree polynomial regression fit
fit <- lm( y ~ poly(x, degree=3), train)
yhat <- predict(fit, newdata=test)

# plot
plot( y ~ x , train)
lines( yhat ~ x, train)
```



Is it a good fit? We can compute the mean squared error for the training data

```r
# compute mean squared error for the training data
MSE.tr <- mean( (train$y - yhat)^2 )
```