Feature scaling in Linear regression

Statistical Learning

Carranza-Alarcón Yonatan-Carlos[1]

[1]UMR CNRS 7253 Heudiasyc
Université de technologie de Compiègne

Outline

Feature scaling - Linear regression

Standardization (Z-score Normalization)

Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} \subseteq \mathbb{R}^p \times \mathbb{R}\}_{i=1}^n$ be a training data.
Given the regression model:

$$y_i = \beta_0 + \beta^T \boldsymbol{x}_i + \epsilon_i \iff \boldsymbol{y} = \boldsymbol{1}\beta_0 + X\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

However, we want fit the following regression model

$$\tilde{y}_i = \tilde{\beta}_0 + \tilde{\beta}^T \tilde{\boldsymbol{x}}_i + \epsilon_i$$

with the normalized data $\mathcal{N} = (\tilde{y}_i, \tilde{x}_i)_{i=1}^n$.

Feature scaling - Linear regression

Standardization (Z-score Normalization)

Applying the standardization (z-score)

$$\tilde{x}_{ij} = \frac{x_{ij} - \overline{x}_j}{\sigma_{x_j}} \qquad \overline{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij} \qquad \sigma_{x_j} = \frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \overline{x}_j)^2$$

$$\begin{bmatrix} \tilde{y}_1 \\ \tilde{y}_2 \\ \vdots \\ \tilde{y}_n \end{bmatrix} = \begin{bmatrix} 0 & \tilde{x}_{11} & \tilde{x}_{12} & \cdots & \tilde{x}_{1p} \\ 0 & \tilde{x}_{21} & \tilde{x}_{22} & \cdots & \tilde{x}_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \tilde{x}_{n1} & \tilde{x}_{n2} & \cdots & \tilde{x}_{np} \end{bmatrix} \begin{bmatrix} \tilde{\beta}_0 \\ \tilde{\beta}_1 \\ \vdots \\ \tilde{\beta}_p \end{bmatrix}$$

The optimal solution of the previous regression model is: $\widehat{\tilde{\beta}} = \{\widehat{\tilde{\beta}}_1, \ldots, \widehat{\tilde{\beta}}_p\}^T$

How do we obtain the prediction of a new instance (not normalized)

$$x_{n+1} \underset{?}{\Longrightarrow} y_{n+1}$$

Feature scaling - Linear regression

Standardization (Z-score Normalization)

Thus, the fitted model with the optimal solution is:

$$
\begin{aligned}
\tilde{y}_{n+1} &= \widehat{\tilde{\beta}}^{T} \tilde{\boldsymbol{x}}_{n+1} \\
\frac{y_{n+1} - \overline{y}}{\boldsymbol{\sigma}_y} &= \widehat{\tilde{\beta}}^{T} \left(\frac{\boldsymbol{x}_{n+1} - \overline{\boldsymbol{x}}}{\boldsymbol{\sigma}_x} \right) \\
y_{n+1} &= \widehat{\tilde{\beta}}^{T} \left(\frac{\boldsymbol{x}_{n+1} - \overline{\boldsymbol{x}}}{\boldsymbol{\sigma}_x} \right) * \boldsymbol{\sigma}_y + \overline{y}
\end{aligned}
$$

where $\boldsymbol{x} = \{\overline{x}_1, \ldots, \overline{x}_p\}$, $\boldsymbol{\sigma}_x = \{\sigma_{x_1}, \ldots, \sigma_{x_p}\}$, and $\boldsymbol{\sigma}_y = \frac{1}{n} \sum_{i=1}^{n} (y_i - \overline{y}_j)^2$

Prédicteurs centrées et réduites

Code in R

$$y_i = \tilde{\beta}^T \tilde{\boldsymbol{x}}_i + \epsilon_i$$

```r
set.seed(36)
rm(list = ls())

# reading data
data <- read.table('data/TP4_a19_reg_app.txt')
idx.train <- sample(nrow(data), size=nrow(data)*0.9, replace=F)
validation <- data[-idx.train,]
training <- data[idx.train,]

# modeling with lm
features <- paste("+ scale(", names(training)[-101], ")", sep="", collapse
    = " ")
formule <- paste("y  1 ", features)
model.fit <- lm(as.formula(formule), data=training)

# prediction
ypredi <- predict.lm(model.fit, newdata=validation)
mean((validation$y -ypredi) # MSE = 727.1163
```

Prédicteurs et réponses centrées et réduites

Code in R

$$\tilde{y}_i = \tilde{\beta}^T \tilde{\mathbf{x}}_i + \epsilon_i$$

```r
set.seed(36)
rm(list = ls())

# reading data
data <- read.table('data/TP4_a19_reg_app.txt')
idx.train <- sample(nrow(data), size=nrow(data)*0.9, replace=F)
validation <- data[-idx.train,]
training <- data[idx.train,]

# modeling with lm
features <- paste("+ scale(", names(training)[-101], ")", sep="", collapse
    = " ")
formule <- paste("scale(y) -1 ", features)
model.fit <- lm(as.formula(formule), data=training)
y.mean <- mean(training[, 101])
y.sd <- sd(training[, 101])

# prediction
ypredi <- predict.lm(model.fit, newdata=validation)
ypredi <- y.sd*ypredi + y.mean
mean((validation$y - ypredi) # MSE = 727.1163
```

Code in R

```r
# reading data
data <- read.table('data/TP4_a19_reg_app.txt')
idx.train <- sample(nrow(data), size=nrow(data)*0.9, replace=F)
validation <- data[idx.train,]
training <- data[-idx.train,]

# scaled data
train.scaled <- scale(training)
X <- as.matrix(train.scaled[, -101])
y <- train.scaled[, 101]
beta.scaled <- solve(t(X)%*%X)%*%t(X)%*%y
x.means <- as.vector(attr(train.scaled,"scaled:center"))[-101]
x.scale <- as.vector(attr(train.scaled,"scaled:scale"))[-101]
y.mean <- as.vector(attr(train.scaled,"scaled:center"))[101]
y.scale <- as.vector(attr(train.scaled,"scaled:scale"))[101]

# prediction
x.mean.train <- matrix(rep(x.means, nrow(validation)), ncol=100, byrow = T)
Xtilde <- y.scale*(validation[, -101] - x.mean.train)
ypredi <- t(apply(Xtilde, 1, "/", x.scale)) %*% beta.scaled + y.mean
mean((validation$y -ypredi) # output [1] 727.1163

# wrong
ypredi <- as.matrix(validation[, -101])%*%beta.scaled
mean((validation$y -ypredi) # output [1] 10765.5
```

Conclusion

Linear regression model is scale-invariant to linear transformations of the data !![1].

[1]Formal proof: https://math.stackexchange.com/questions/2813060/linear-regression-proving-that-linear-regression-is-linear-invariant