

Feature scaling in Linear regression

Statistical Learning

Carranza-Alarcón Yonatan-Carlos¹

¹UMR CNRS 7253 Heudiasyc
Université de technologie de Compiègne

Outline

1 Regression

Feature scaling - Linear regression

Standardization (Z-score Normalization)

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

Given the regression model:

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

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

However, we want fit the following regression model

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

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

Feature scaling - Linear regression

Standardization (Z-score Normalization)

Applying the standardization (z-score)

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

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

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

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

$$x_{n+1} \xRightarrow{?} y_{n+1}$$

Feature scaling - Linear regression

Standardization (Z-score Normalization)

Thus, the fitted model with the optimal solution is:

$$\begin{aligned}\tilde{y}_{n+1} &= \hat{\tilde{\beta}}^T \tilde{\mathbf{x}}_{n+1} \\ \frac{y_{n+1} - \bar{y}}{\sigma_y} &= \hat{\tilde{\beta}}^T \left(\frac{\mathbf{x}_{n+1} - \bar{\mathbf{x}}}{\sigma_x} \right) \\ y_{n+1} &= \hat{\tilde{\beta}}^T \left(\frac{\mathbf{x}_{n+1} - \bar{\mathbf{x}}}{\sigma_x} \right) * \sigma_y + \bar{y}\end{aligned}$$

where $\mathbf{x} = \{\bar{x}_1, \dots, \bar{x}_p\}$, $\sigma_x = \{\sigma_{x_1}, \dots, \sigma_{x_p}\}$, and $\sigma_y = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y}_j)^2$

Prédicteurs centrées et réduites

Code in R

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

```

1 set.seed(36)
2 rm(list = ls())
3
4 # reading data
5 data <- read.table('data/TP4_a19_reg_app.txt')
6 idx.train <- sample(nrow(data), size=nrow(data)*0.9, replace=F)
7 validation <- data[-idx.train,]
8 training <- data[idx.train,]
9
10 # modeling with lm
11 features <- paste("+ scale(", names(training)[-101], ")", sep="", collapse
    = " ")
12 formule <- paste("y ~ 1 ", features)
13 model.fit <- lm(as.formula(formule), data=training)
14
15 # prediction
16 ypredi <- predict.lm(model.fit, newdata=validation)
17 mean((validation$y - ypredi)^2) # MSE = 727.1163

```

Prédicteurs et réponses centrées et réduites

Code in R

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

```

1 set.seed(36)
2 rm(list = ls())
3
4 # reading data
5 data <- read.table('data/TP4_a19_reg_app.txt')
6 idx.train <- sample(nrow(data), size=nrow(data)*0.9, replace=F)
7 validation <- data[-idx.train,]
8 training <- data[idx.train,]
9
10 # modeling with lm
11 features <- paste("+ scale(", names(training)[-101], ")", sep="", collapse
    = " ")
12 formule <- paste("scale(y) ~ -1 ", features)
13 model.fit <- lm(as.formula(formule), data=training)
14 y.mean <- mean(training[, 101])
15 y.sd <- sd(training[, 101])
16
17 # prediction
18 ypredi <- predict.lm(model.fit, newdata=validation)
19 ypredi <- y.sd*ypredi + y.mean
20 mean((validation$y - ypredi)^2) # MSE = 727.1163

```

Code in R

```

1 # reading data
2 data <- read.table('data/TP4_a19_reg_app.txt')
3 idx.train <- sample(nrow(data), size=nrow(data)*0.9, replace=F)
4 validation <- data[idx.train,]
5 training <- data[-idx.train,]
6
7 # scaled data
8 train.scaled <- scale(training)
9 X <- as.matrix(train.scaled[, -101])
10 y <- train.scaled[, 101]
11 beta.scaled <- solve(t(X)%*%X)%*%t(X)%*%y
12 x.means <- as.vector(attr(train.scaled, "scaled:center"))[-101]
13 x.scale <- as.vector(attr(train.scaled, "scaled:scale"))[-101]
14 y.mean <- as.vector(attr(train.scaled, "scaled:center"))[101]
15 y.scale <- as.vector(attr(train.scaled, "scaled:scale"))[101]
16
17 # prediction
18 x.mean.train <- matrix(rep(x.means, nrow(validation)), ncol=100, byrow = T)
19 Xtilde <- y.scale*(validation[, -101] - x.mean.train)
20 ypredi <- t(apply(Xtilde, 1, "/", x.scale)) %*% beta.scaled + y.mean
21 mean((validation$y - ypredi)^2) # output [1] 727.1163
22
23 # wrong
24 ypredi <- as.matrix(validation[, -101])%*%beta.scaled
25 mean((validation$y - ypredi)^2) # output [1] 10765.5

```


Conclusion

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

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