Prof. Dr. Steffen Wagner
Angewandte Statistik
Fachbereich II
Berliner Hochschule für Technik

Exercise 04: K-Nearest Neighbour Regression

Machine Learning I – SoSe 24

1 Preparations **2**
1.1 RStudio Project 2
1.2 Required Packages 2

2 K-Nearest Neighbour Regression **3**
2.1 K-Nearest neighbours with one explanatory variable 3
2.2 K-Nearest neighbours with multiple explanatory variables 5

3 Mean Squared Error and cross validation **7**
3.1 Sneak preview 7

4 Written exercise **9**
4.1 KNN by hand 9

5 Congratulations **10**

This workshop covers hierarchical clustering and soft clustering. At the end of the worksheet there are a couple of written exercises for you to do at home, which should be good practice for the exam.

1 Preparations

1.1 RStudio Project

1. Open your Machine Learning 1 RStudio Project
2. Create an R Script file to perform this exercise

1.2 Required Packages

For this exercise you require the following additonal R packages. Please make sure that you have installed them on your computer before coming to the workshop session since Eduroam is still not working.

```r
# check if packages can be loaded, i.e. they are already installed
library(FNN)             # KNN regression
library(rgl)             # 3D visualisation
```

If you get an error at this stage, you need to install the packages.

2 K-Nearest Neighbour Regression

2.1 K-Nearest neighbours with one explanatory variable

Let's start with a simple (easy) simple (one variable) regression example. The predictor variable `x` is a vector of integers between 1 and 20. The outcome variable `y` is the vector `x` with normal random noise added.

```r
x <- 1:20
y <- x + rnorm(length(x), mean = 10)
xgrid <- data.frame(x)
```

`xgrid` is the set of predictor values, for which the predictor function will be evaluated. Initially this will just be the observed values `x`, i.e. the predicted values will be the fitted values. The KNN function requires these values to be stored in a `data.frame` object.

The function to fit a K-nearest neighbour regression model in the `FNN` library is called `knn.reg` and to obtain the correct fitted values we need to pass the data frame `xgrid` to the argument called `test`.

In Exercise 3 you will investigate what happens when the `test` argument is omitted.

Fit the KNN regression with $K=1$ and compare the fitted values with the outcome variable, first of all loading the `FNN` package

```r
library(FNN)
knnr_out_1 <- knn.reg(x, y = y, k = 1, test = xgrid)
round(cbind(x, y, fitted_1 = knnr_out_1$pred), 2)
identical(y, knnr_out_1$pred)
```

Repeat this with the value $K=3$. Make sure you understand the fitted values for the 3-nearest neighbour averages.

To plot the predicted values as a function, we will specify a much finer grid for the predicted values. Again start with $K=1$.

```r
xgrid <- data.frame(x = seq(0, 21, 0.05))
knnr_out_1 <- knn.reg(x, y = y, k = 1, test = xgrid)
plot(x, y)
lines(xgrid$x, knnr_out_1$pred)
```

Gradually increase the value of K and observe what happens to the predictor function.

```r
xgrid <- data.frame(x = seq(0, 21, 0.05))
plot(x, y)

k_seq <- seq(1, 7, 2)
for(i in seq(1, 11, 2)){
  knnr_out_i <- knn.reg(x, y = y, k = i, test = xgrid)
  lines(xgrid$x, knnr_out_i$pred, col = i)
}
legend("topleft", col = k_seq, legend = paste0("k=", k_seq), lty = 1)
```

Which values of K correspond to under fitting and which values to over fitting? An approximate answer will do.

For the sake of completeness and a bit of revision we will have a look at the **linear regression** for these data.

```
lm_out <- lm(y ~ x)
summary(lm_out)
plot(x, y)
abline(lm_out, col = 2)
knnr_out_19 <- knn.reg(x, test = xgrid, y = y, k = 19)
lines(xgrid[, 1], knnr_out_19$pred, col = 3)
```

Use the output of the `summary()` command to complete the formula for the regression line.

$$y = \underline{\hspace{3cm}} + \underline{\hspace{3cm}}x$$

Note that the linear regression model fits the data very well and requires only two parameters. Although for non-parametric models we do not use the concept of the number of "parameters" a comparison would be using just two constant functions i.e. $K=19$ which massively under fits these data. This comparison is somewhat unfair though, as the example data are ideal for a linear regression model.

2.2 K-Nearest neighbours with multiple explanatory variables

The following data were taken from a small study investigating the fitness rating for 10 participants. The measured explanatory variables are *weight* and *lung volume*. A small data set can be directly entered into R using the following code:

```
fitdata <- as.data.frame(
  matrix(c(1, 87, 42,
           6, 73, 43,
           7, 66, 44,
           15, 62, 54,
           12, 68, 45,
           4, 92, 46,
           12, 60, 50,
           13, 70, 46,
           14, 71, 54,
           10, 64, 47),
         byrow = TRUE, ncol = 3)
)
names(fitdata) <- c("fitness", "weight", "lungvol")
summary(fitdata)
```

The **linear model** for fitness dependent on weight and lung volume is fitted using:

```
lm_fitness <- lm(fitness ~ weight + lungvol, data = fitdata)
summary(lm_fitness)
```

Both explanatory variables are significant at the 5% level. Use this output to write the predictor function in terms of the two explanatory variables.

$$\textit{Fitness} = \underline{\hspace{3cm}} + \underline{\hspace{3cm}} \cdot \textit{weight} + \underline{\hspace{3cm}} \cdot \textit{lung volume}$$

We will plot the two variables using

i) a *perspective plot* and
ii) a *3-d plot*.

In both cases we need to compute the predicted values at points on a grid with *weight* taking values between 55 and 95 Kg and *lung volume* between 40 and 55 dl.

```
m1 <- seq(55, 95, length = 20)
m2 <- seq(40, 55, length = 20)
Xgrid <- expand.grid(weight = m1, lungvol = m2)
pred.grid <- predict(lm_fitness, newdata = Xgrid)
tt <- cbind(Xgrid, pred.grid)
res <- persp(m1,  m2,  matrix(pred.grid, nrow = length(m1)), border = grey(0.6),
             xlab = "Weight", ylab = "Lung Volume", zlab = "fitness",
             theta = 0, phi = 15)
points(trans3d(fitdata$weight, fitdata$lungvol, fitdata$fitness, pmat = res),
     pch = 16, col = (2:3)[1.5 + 0.5 * sign(lm_fitness$residuals)])
```

The points are coloured according to the sign of the residual, i.e. above or below the fitted plane. The function `persp` takes two arguments determining the position from which the data are viewed. `theta` controls the rotation of the x and y axes, and `phi` controls the height of the viewing point. Play about with these two parameters to get a better understanding of the data and the fitted plane.

Create a 3d plot for these data:

```
library(rgl)
with(tt, plot3d(lungvol, weight, pred.grid,
                col = "grey80", size = 3, zlab = "fitness"))
with(fitdata, plot3d(lungvol, weight, fitness, add = TRUE, size = 2,
                     col = (2:3)[1.5 + 0.5 * sign(lm_fitness$residuals)],
                     type = "s"))
```

You can rotate the plot by dragging it with the mouse.

We will now fit a K **nearest neighbour regression** model. We define a data frame `X` with the two variables `weight` and `lungvol` and start with K=1.

```
X <- fitdata[, c("weight", "lungvol")]
knnr_out_1 <- knn.reg(X, y = fitdata$fitness, k = 1, test = Xgrid)
res <- persp(m1, m2, matrix(knnr_out_1$pred, nrow = length(m1)),
             border = grey(0.6),
             xlab = "Weight", ylab = "Lungvol", zlab = "fitness",
             theta = 15, phi = 15)
points(trans3d(fitdata$weight, fitdata$lungvol, fitdata$fitness, pmat = res),
       pch = 16, col =  c(2:3)[1.5 + 0.5 * sign(lm_fitness$residuals)])
```

```
with(tt,  plot3d(lungvol, weight, knnr_out_1$pred,
                col = "grey80", size = 3, zlab = "fitness"))
with(fitdata,  plot3d(lungvol, weight, fitness, add = TRUE,
                      size = 2, col = "red", type = "s"))
```

Now increase the value of K each time obtaining the new plot.

3 Mean Squared Error and cross validation

3.1 Sneak preview

This exercise is just a sneak preview of an very important aspect of almost all supervised learning methods. Don't worry too much about the fine details this week, as they will be covered in more detail in the coming weeks.

What is the best value of K? Re-run the K nearest neighbour regression model with $K{=}1$ and obtain the fitted values. Calculate the mean squared error for this particular model. We would like the MSE to be small.

Ouch! The plot for $K{=}1$ showed obvious over-fitting of the data but the mean squared error suggests this is the best model, as the MSE can not be negative. Choosing a model based on fitted MSE alone is not a good idea.

One way to avoid this problem is to use **cross validation**, which can be used in many supervised learning methods. We will return to the *one dimensional data* from the first section, as it is easier to investigate what is going on. Compare the output of `knn.reg` with and without the argument `test`, using only one nearest neighbour $K{=}1$.

```r
xgrid <- data.frame(x)
knnr_out_a_1 <- knn.reg(x, y = y, k = 1)  # what is the default value for test?
knnr_out_b_1 <- knn.reg(x, test = xgrid, y = y, k = 1)
round(cbind(x, y,
            pred_wo_test = knnr_out_a_1$pred,
            pred_with_test = knnr_out_b_1$pred), 2)
```

Look closely at the two sets of predicted values, what do you notice?

When the `test` argument is not specified, then algorithm excludes the current x value when determining the K neighbours nearest to x.

This is called *cross validation*. For each observation in turn, the model is fitted without that observation point, and then the predicted value obtained at that point.

Now compute the MSE for the cross validated model for different values of K. Once the best value of K is chosen, the non cross validated model needs to be run to obtain the predictions.

```r
k_vec <- 1:10
mse_vec <- rep(NA, length(k_vec))
for(k in k_vec){
  knnr_out_k <- knn.reg(x, y = y, k = k)  # important: test = NULL for CV
  mse_vec[k] <- mean((y - knnr_out_k$pred)^2)
}
plot(k_vec, mse_vec)
```

Increase the value of `k` to find, which value of K gives the smallest cross validation MSE.

Now do the same for the fitness data.

```r
k_vec <- 1:(nrow(fitdata) - 1)  # for CV there is a maximum on n-1 neighbours
mse_vec <- rep(NA, length(k_vec))
for(k in k_vec){
  knnr_out_k <- knn.reg(X, y = fitdata$fitness, k = k)
  mse_vec[k] <- mean((fitdata$fitness - knnr_out_k$pred)^2)
}
plot(k_vec, mse_vec)
```

What is the best number of neighbours using this method?

Plot the optimal fitted KNN regression in a 3D-plot.

```
knnr_out_2 <- knn.reg(X, y = fitdata$fitness, k = 2, test = Xgrid)
res <- persp(m1, m2, matrix(knnr_out_2$pred, nrow = length(m1)),
             border = grey(0.6),
             xlab = "Weight", ylab = "Lungvol", zlab = "fitness",
             theta = 15, phi = 15)
points(trans3d(fitdata$weight, fitdata$lungvol, fitdata$fitness, pmat = res),
       pch = 16, col =  c(2:3)[1.5 + 0.5 * sign(lm_fitness$residuals)])
```

```
with(tt,  plot3d(lungvol, weight, knnr_out_2$pred,
                 col = "grey80", size = 3, zlab = "fitness"))
with(fitdata,  plot3d(lungvol, weight, fitness, add = TRUE,
                      size = 2, col = "red", type = "s"))
```

Finally we can calculate the MSE for the multiple regression model.

The MSE of the linear regression is considerably smaller, as these data fit the linear regression model well. This is not always the case.

Another possible approach is to set aside a proportion of the data calling this the *test data* set and the remaining part the *training data* set. For our two data sets today we only have 20 and 10 observations, so splitting the observations into training and test data sets was not a sensible option.

You will learn more about these concepts in the lecture next week.

4 Written exercise

4.1 KNN by hand

If you have understood the KNN regression method this should be a quick and easy exercise for you. If you are unsure, then the practice will be good.

In the table below are values of three variables x_1, x_2 and y.

x_1	x_2	y
3	7	1
4	3	4
7	9	9
8	6	10
9	4	8
11	12	3
13	5	0
13	8	1

Using x_1 and x_2 as the explanatory variables and y as the outcome variable, find the KNN regression estimate for the point $\boldsymbol{x}' = (10, 5)$ with values of k=2 and k=4.

Solution:

The following steps are required:

1. Calculate the (squared) euclidean distances between the point $\boldsymbol{x}' = (10, 5)$ and the given data.
2. Determine the order of the (squared) distances. You should have obtained:

x_1	x_2	y	$d_{P(10,5)}$	order
3	7	1	53	8
4	3	4	40	6
7	9	9	25	5
8	6	10	5	2
9	4	8	2	1
11	12	3	50	7
13	5	0	9	3
13	8	1	18	4

3. Now calculate the mean $\bar{y}_{i \in \mathcal{N}(\boldsymbol{x}_\cdot, k)}$ based on the k=2 and k=4 nearest observations:

- k=2: $f(x_\cdot) = \frac{1}{2}(y_5 + y_4) = \frac{1}{2}(8 + 10) = 9$
- k=4: $f(x_\cdot) = \frac{1}{4}(y_5 + y_4 + y_7 + y_8) = \frac{1}{4}(8 + 10 + 0 + 1) = 4.75$

5 Congratulations

You have learnt a lot today!