

Solution 1:

- a) multiclass classification (plate digits) (supervised learning)
- b) binary classification (supervised)
- c) outlier detection ((un)supervised)
- d) frequent pattern mining (unsupervised)
- e) classification (supervised) / clustering (unsupervised)
- f) classification (supervised)
- g) clustering / association rules (unsupervised)
- h) not a machine learning task
- i) not a machine learning task

Solution 2:

- a) We use the least squares-estimator introduced in the lecture: $\hat{\beta} = (X^T X)^{-1} X^T y$ with

$$X = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \dots & x_{1,m} \\ 2 & x_{2,1} & x_{2,2} & \dots & x_{2,m} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ n & x_{n,1} & x_{n,2} & \dots & x_{n,m} \end{bmatrix}$$

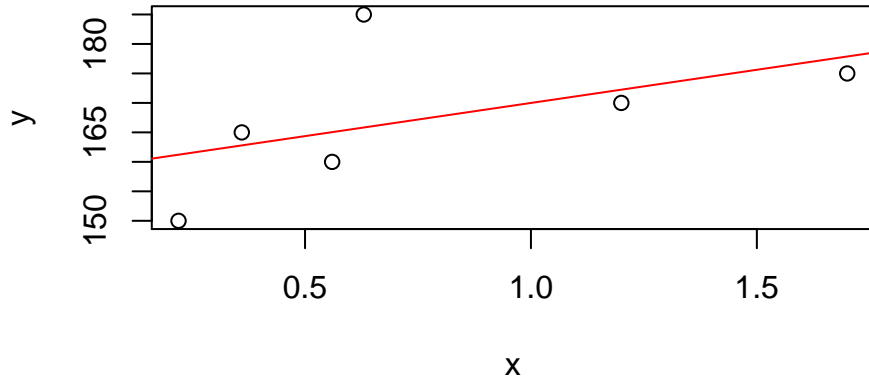
$$x = \begin{bmatrix} 0.56 \\ 0.22 \\ 1.7 \\ 0.63 \\ 0.36 \\ 1.2 \end{bmatrix}, X = \begin{bmatrix} 1 & 0.56 \\ 1 & 0.22 \\ 1 & 1.7 \\ 1 & 0.63 \\ 1 & 0.36 \\ 1 & 1.2 \end{bmatrix} \text{ and } y = \begin{bmatrix} 160 \\ 150 \\ 175 \\ 185 \\ 165 \\ 170 \end{bmatrix}$$

Then

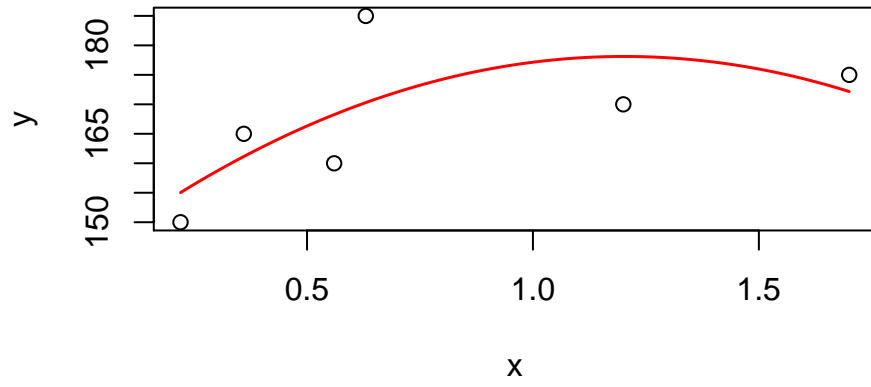
$$\hat{\beta} = (X^T X)^{-1} X^T y$$

$$\begin{aligned}
&= \left(\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ x_{1,1} & x_{2,1} & x_{3,1} & \dots & x_{n,1} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ x_{1,m} & x_{2,m} & x_{3,m} & \dots & x_{n,m} \end{bmatrix} \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \dots & x_{1,m} \\ 1 & x_{2,1} & x_{2,2} & \dots & x_{2,m} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & x_{n,1} & x_{n,2} & \dots & x_{n,m} \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ x_{1,1} & x_{2,1} & x_{3,1} & \dots & x_{n,1} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ x_{1,m} & x_{2,m} & x_{3,m} & \dots & x_{n,m} \end{bmatrix} \begin{bmatrix} 160 \\ 150 \\ 175 \\ 185 \\ 165 \\ 170 \end{bmatrix} \\
&= \left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0.56 & 0.22 & 1.7 & 0.63 & 0.36 & 1.2 \end{bmatrix} \begin{bmatrix} 1 & 0.56 \\ 1 & 0.22 \\ 1 & 1.7 \\ 1 & 0.63 \\ 1 & 0.36 \\ 1 & 1.2 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0.56 & 0.22 & 1.7 & 0.63 & 0.36 & 1.2 \end{bmatrix} \begin{bmatrix} 160 \\ 150 \\ 175 \\ 185 \\ 165 \\ 170 \end{bmatrix} \\
&= \begin{bmatrix} 6 & 4.67 \\ 4.67 & 5.2185 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0.56 & 0.22 & 1.7 & 0.63 & 0.36 & 1.2 \end{bmatrix} \begin{bmatrix} 160 \\ 150 \\ 175 \\ 185 \\ 165 \\ 170 \end{bmatrix} \\
&= \begin{bmatrix} 0.5491944 & -0.4914703 \\ -0.4914703 & 0.6314394 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0.56 & 0.22 & 1.7 & 0.63 & 0.36 & 1.2 \end{bmatrix} \begin{bmatrix} 160 \\ 150 \\ 175 \\ 185 \\ 165 \\ 170 \end{bmatrix} \\
&= \begin{bmatrix} 0.2739710 & 0.4410709 & -0.2863051 & 0.23956809 & 0.3722651 & -0.04056998 \\ -0.1378643 & -0.3525536 & 0.5819766 & -0.09366351 & -0.2641521 & 0.26625693 \end{bmatrix} \begin{bmatrix} 160 \\ 150 \\ 175 \\ 185 \\ 165 \\ 170 \end{bmatrix} \\
&= \begin{bmatrix} 158.73954 \\ 11.25541 \end{bmatrix}
\end{aligned}$$

Hence the linear model $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x = 158.73954 + 11.25541x$



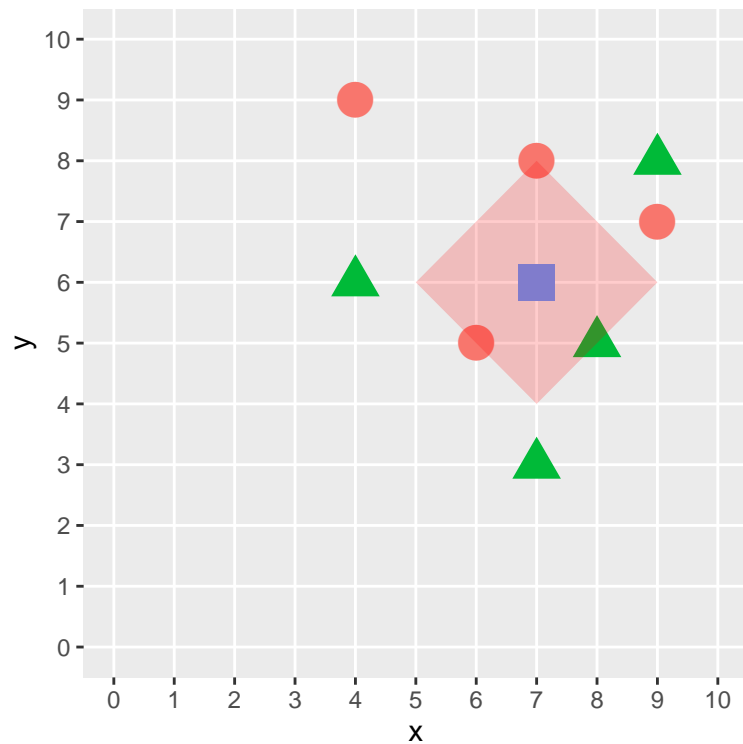
b) Here $X = \begin{bmatrix} 1 & 0.56 & 0.3136 \\ 1 & 0.22 & 0.0484 \\ 1 & 1.7 & 2.89 \\ 1 & 0.63 & 0.3969 \\ 1 & 0.36 & 0.1296 \\ 1 & 1.2 & 1.44 \end{bmatrix}$ and $\hat{\beta} = \begin{bmatrix} 143.51682 \\ 57.59155 \\ -23.96347 \end{bmatrix}$



Solution 3:

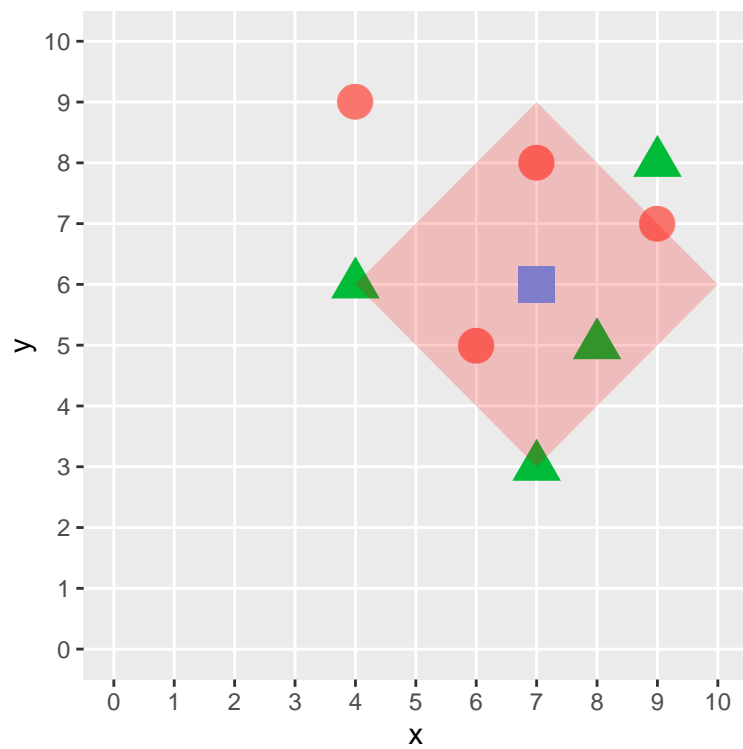
a) $k = 3$

2 circles and 1 triangle, so our point is also a circle



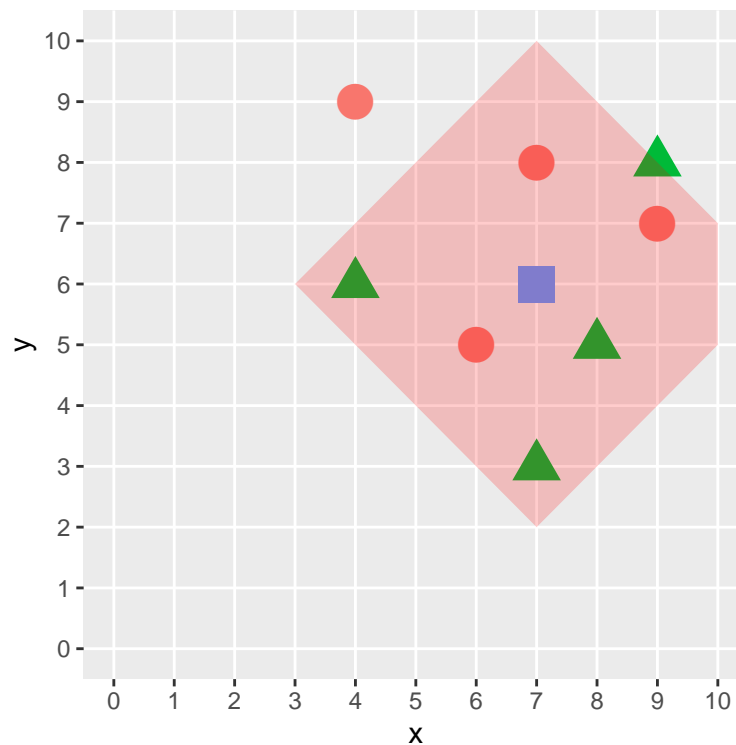
b) $k = 5$

3 circles and 3 triangles, we have to specify beforehand what to do in case of a tie



c) $k = 7$

3 circles and 4 triangles, so our point is also a triangle



Solution 4:

a) Learning consists of *representation*, *evaluation* and *optimization*.

A learner in mlr3 can be thought of as the implementation of these components, since

- a representation of the associated model learnt from the data by using the implemented optimization is stored in such a learner object,
- its performance measures can be accessed afterwards.

```
b) library(mlr3)
library(mlr3learners)

# show all available learners
mlr_learners$keys()

## [1] "classif.debug"      "classif.featureless" "classif.glmnet"
## [4] "classif.kknn"       "classif.lda"         "classif.log_reg"
## [7] "classif.naive_bayes" "classif.qda"         "classif.ranger"
## [10] "classif.rpart"      "classif.svm"         "classif.xgboost"
## [13] "regr.featureless"   "regr.glmnet"         "regr.kknn"
## [16] "regr.km"            "regr.lm"             "regr.ranger"
## [19] "regr.rpart"         "regr.svm"            "regr.xgboost"

# see settings for a specific learner, e.g., for a regression tree
rpart_learner <- lrn("regr.rpart")
print(rpart_learner)

## <LearnerRegrRpart:regr.rpart>
```

```
## * Model: -
## * Parameters: xval=0
## * Packages: rpart
## * Predict Type: response
## * Feature types: logical, integer, numeric, factor, ordered
## * Properties: importance, missings, selected_features, weights
```

Solution 5:

See R code `lm_knn.1.1a.R`

Solution 6*:

Firstly note that for $n = 1$ the median $y_{\text{med}} = y^{(1)}$ obviously minimizes the empirical risk \mathcal{R}_{emp} associated to the L1 loss L . Hence let $n > 1$ in the following:

Since for $a, b \in \mathbb{R}$, $S_{a,b} : \mathbb{R} \rightarrow \mathbb{R}_0^+$, $c \mapsto |a - c| + |b - c|$ it holds that

$$S_{a,b}(c) = \begin{cases} |a - b|, & \text{for } c \in [a, b] \\ |a - b| + 2 \cdot \min\{|a - c|, |b - c|\}, & \text{otherwise} \end{cases}$$

$c^* \in [a, b]$ minimizes $S_{a,b}$.

With this \mathcal{R}_{emp} can be expressed for a constant $c \in \mathbb{R}$, s.t.

$$n \cdot \mathcal{R}_{\text{emp}}(c) = \sum_{i=1}^n L(y^{(i)}, c) = \sum_{i=1}^n |y^{(i)} - c| = \begin{cases} \sum_{i=1}^{\overbrace{n/2}^{=:i_{\max}}} \underbrace{S_{y^{(i)}, y^{(n+1-i)}}(c)}_{=:S_i(c)}, & \text{for } n \text{ is even} \\ (\sum_{i=1}^{\overbrace{(n-1)/2}^{=:i_{\max}}} \underbrace{S_{y^{(i)}, y^{(n+1-i)}}(c)}_{=:S_i(c)}) + \underbrace{|y^{(n+1)/2} - c|}_{=:S_0(c)}, & \text{for } n \text{ is odd} \end{cases}.$$

Now we define for $i \in \{1, \dots, i_{\max}\}$ $\mathcal{I}_i := [y^{(i)}, y^{(n+1-i)}]$.

From construction it follows that for $j \in \{1, \dots, i_{\max} - 1\}$

$$\mathcal{I}_{j+1} \subseteq \mathcal{I}_j \Rightarrow \forall i \in \{1, \dots, i_{\max}\} : \mathcal{I}_{i_{\max}} \subseteq \mathcal{I}_i.$$

From this it follows, that

- for "n is even": $c^* \in \mathcal{I}_{i_{\max}} = [y^{(n/2)}, y^{(n/2+1)}]$ minimizes S_i for all $i \in \{1, \dots, i_{\max}\} \Rightarrow \mathcal{R}_{\text{emp}}$ reaches its global minimum at c^* ,
- for "n is odd": $c^* = y^{(n+1)/2} \in \mathcal{I}_{i_{\max}}$ minimizes S_i for all $i \in \{0, 1, \dots, i_{\max}\} \Rightarrow \mathcal{R}_{\text{emp}}$ reaches its global minimum at c^* .

Since the median fulfills these conditions, we can conclude that it minimizes the L1 loss.