1. Sometimes outliers can seriously hinder the overall performance of regression methods based on minimisation of mean square error. The following exercise explains how to detect that there is a problem with outliers and how to solve it using confidence intervals for predictions.

File `regression-benchmark.Rdata` contains data that roughly follows quadratic dependency $y = x^2$. However, there are some data points that seem to have arbitrary values

(a) Fit a model $y \sim \alpha x^2 + \beta$ on the data and draw a plot which contains the data points, prediction line. How good is the fitted model? Do you see a systematic bias? Also, use `qqplot` to test whether the residuals follow normal distribution, i.e., whether the assumptions of means square minimisation procedure are satisfied. (1p)

(b) Although the error distribution is far from normal distribution you can still try to find standard deviation of residuals $\sigma$ and approximate error distribution with $N(0, \sigma)$. Compute 95% confidence intervals for the error, i.e., solve the equations:

$$\Pr[\varepsilon \sim N(0, \sigma) : \varepsilon \leq q_0] = 2.5\%$$
$$\Pr[\varepsilon \sim N(0, \sigma) : \varepsilon \leq q_1] = 97.5\%$$

for $q_0$ and $q_1$. Draw two additional dashed lines $y = \hat{y}(x) + q_0$ and $y = \hat{y}(x) + q_1$ where $\hat{y}(x)$ is the prediction line obtained in the first part. We can label all points outside the region bounded by dashed lines as outliers. Write a function that detects all outliers. Now try to fit a model $y \sim \alpha x^2 + \beta$ only for the set of normal data points. Again, compute standard deviation for the residuals and corresponding 95% confidence intervals for the error and visualise 95% confidence intervals for the prediction, i.e., draw lines $y = \hat{y}(x) + q_0$ and $y = \hat{y}(x) + q_1$ for the prediction line $\hat{y}(x)$.

Compare the first and the second plot. Does this technique improve the quality of predictions. Are the assumptions of means square minimisation now satisfied? (2p)

(c) Write a function `robust.lm` that repeats outlier detection several times to fit a model. Describe the convergence criterion and demonstrate that it works also for multivariate linear regression. (2p)
2. When the number of hidden neurons is too big for the problem instance, training algorithms without regularisation commonly produce sub-optimal results, since many hidden layer neurons represent the same concept. Regularisation helps to alleviate this problem. The following exercise shows what is the main difference between $\ell_1$ and $\ell_2$ regularisation.

(a) Generate data by choosing 200 samples $x_1, \ldots, x_{200}$ uniformly from the range $[-2, 2]$. Let the target function $f(x)$ be a square hump

$$f(x) = \begin{cases} 1, & \text{if } |x| \leq 1, \\ 0, & \text{if } |x| > 1. \end{cases}$$

Compute the response $y_i = f(x_i) + \varepsilon_i$ where $\varepsilon_i \sim \mathcal{N}(0, 0.1)$. Use nnet package to train a neural network with 20 hidden neurons. Make sure that you train the network to predict and not to classify, i.e., set linout=TRUE. Calculate the output of the hidden layer on the data samples. You should get $200 \times 20$ matrix $H$ where the columns contain the outputs of hidden neurons on inputs $x_1, \ldots, x_{200}$. (1p)

Hint: You can and should reuse the code from Exercise Session VI where we considered neural networks. In particular, you should use functions as.neuralnetwork and CalculateNeuronOutputs. The latter allows you to inspect the outputs of the hidden layer on different inputs. Alternatively you could use compute function from the package neuralnet.

(b) Use lm and lm.ridge to find the weights of the output neuron. Recall that for prediction the output neuron is linear and thus we can determine the weights by linear regression of type $y \sim H + 1$ when the hidden layer neurons are fixed. Try different values for the regularisation coefficient $\lambda$. Start with near-zero and increase it till the regularisation completely destroys the shape of the prediction function. Choose 5 values for $\lambda$ between these two extremes (equally spaced on logarithmic scale) and visualise the effect. For that draw two graphs for each value of $\lambda$. The first graph should contain the target function, training data and the prediction function. The second graph should just output the values of regression coefficients for different neurons. Is the coefficient pattern stable when we alter $\lambda$, i.e., same coefficients are large compared to the others? (2p)

Hint: Unfortunately, predict function does not work with the model produced by lm.ridge. Hence, you have to use coef function to extract model coefficients and then matrix algebra to evaluate prediction on the data matrix.

(c) Use the lars package for $\ell_1$-regularisation. The function lars outputs values of regression coefficient on critical points. All other values can be reconstructed from these points via linear interpolation—it can be shown that the coefficient vector is a piecewise linear function.
of regularisation parameter. In practice, you should consider coefficient values only in critical points as potential weights for the output neuron. The lars function implements several sparse regression algorithms and only lasso implements $\ell_1$-regularisation.

Use lars package to plot predictions on first five critical points together with coefficient values as in the previous part of the exercise. Explain how you could use $\ell_1$-regression to prune redundant neurons in the hidden layer. Confer the results with $\ell_2$-regularisation (2p)

**Hint:** Unfortunately, the package is seriously under-documented. Fortunately, functions predict.lars and coef.lars allow you to make predictions and extract coefficients for each critical value.

(* ) Implement a neural network pruning strategy that uses $\ell_1$-regularisation to remove unnecessary neurons from the hidden layer. The algorithm should work as an add-on for nnet. Test is on standard target functions (square hump, saw-tooth and staircase) and few more complex one-dimensional functions. (2p)

3. This exercise illustrates the concepts of leverage and how regularisation changes the leverage of individual data points. We use here a non-standard definition of leverage that is easy to interpret. Let $(x_1, y_1), \ldots, (x_n, y_n)$ be the set of training points. Let $a$ be the linear regression coefficients returned by the regression algorithm $R$. We define the leverage vector $\Delta a(x_i, \delta)$ of point $x_i$ at error level $\delta$ the change of linear regression coefficients if we replace $y_i$ by $y_i + \delta$:

$$\Delta a(x_i, \delta) = R(\ldots, (x_i, y_i + \delta), \ldots) - R(\ldots, (x_i, y_i), \ldots).$$

(a) For the comparison generate two data sets. The first should consist of 100 points that are sampled uniformly from the range $[-1, 1] \times [-1, 1]$. The second consists of 100 data points such that $x_i$ is sampled uniformly from $[-1, 1]$ and $x_2 = x_1 + \epsilon$ where $\epsilon$ is uniformly sampled in $[-0.05, 0.05]$. For clarity assume that $y \equiv 0$ over the region. Fix

$$\text{lm}(y \sim x_1 + x_2 + 0, \ldots)$$

as the regression algorithm. Compute leverage of all data points for the error level $\delta = 0.1$. Draw data and highlight the points with high leverage. Since leverage consists of two components $\Delta a_1$ and $\Delta a_2$ order points first by the length of leverage vector

$$\| \Delta a(x_i, \delta) \| = \sqrt{\Delta a_1^2(x_i, \delta) + \Delta a_2^2(x_i, \delta)}$$

and choose 10 points with the highest length. Colour them red on the plot. Secondly, compare how the length of leverage vectors depends on the dataset. For that draw the boxplots of the lengths for both data sets. What can you conclude? (2p)
(b) Now use `lm.ridge` with different regularisation values $\lambda$ and study how the length of leverage vectors changes. Draw boxplots for 5 different values of $\lambda$ which are equally spaced on logarithmic scale. Interpret results. Does it make sense to consider $\ell_1$-regression? (2p)

4. Let us consider maximum-likelihood classifier for the following model. If $y_i = 0$ then $x_i$ is chosen from the normal distribution with parameters $\mu_0$ and $\Sigma_0$ and if $y_i = 1$ then $x_i$ is chosen from the normal distribution with parameters $\mu_1$ and $\Sigma_1$.

(a) Derive a density of all observations given model parameters

$$ p([x_1,y_1],\ldots,[x_1,y_1]|\mu_0,\Sigma_0,\mu_1,\Sigma_1) $$

and show that it decomposes into a product of two functions $f_0(\mu_0,\Sigma_0)$ and $f_1(\mu_1,\Sigma_1)$ such that $f_j$ depends only on the data points with label $y_i = j$. (1p)

(b) Derive or Google a maximum likelihood estimate for $f_0$ and $f_1$. Implement the corresponding parameter inference algorithm and test the algorithm on the Iris dataset. (2p).

**Hint:** The Matrix Cookbook should contain the formula.

5. In the lecture, we argued that the problem of outliers can be solved using Laplace distribution as an error distribution in the linear regression. The latter leads to the minimisation of sum of absolute deviations (LAD). The latter is implemented as `rq` function in `quantreg` package. The package implements quantile regression. However, the quantile regression with parameter $\tau = 0.5$ and method `br` is equivalent to the LAD.

(a) Use the data in the file `regression-benchmark.Rdata` as benchmark. Fit a model $y \sim ax^2 + \beta$ on the data and draw a plot which contains the data points, prediction line. (1p)

(b) To plot 95% confidence intervals, we need to estimate the $\mu$ and $\beta$ parameter from the residuals and then compute 95% confidence intervals for the Laplace distribution. For that note that maximum likelihood estimates for these parameters are:

$$ \mu = \text{median}(\epsilon_1,\ldots,\epsilon_n) $$

$$ \beta = \frac{1}{n} \sum_{i=1}^{n} |\epsilon_i - \mu| . $$

Use these parameters and the relation between Laplace and exponential distribution to compute confidence intervals. Plot them as dashed lines. Does the residuals follow Laplacian distribution? Compare the result with the first exercise. Interpret the result. (1p)