# Bonus Lab06: Gradient Descent and Deep Learning

**Handed out:** Wednesday, April 26, 2023

**Firm return date:** Friday, May 5, by midnight at the eLearning link **Lab06Submit** in the **Lab06** folder.

**Objectives:**[a]You will program and explore a simple gradient descent problem to understand the difference between local and global minima.

[b] With deep learning you will estimate the median home values in Boston and identify with cross-validation the

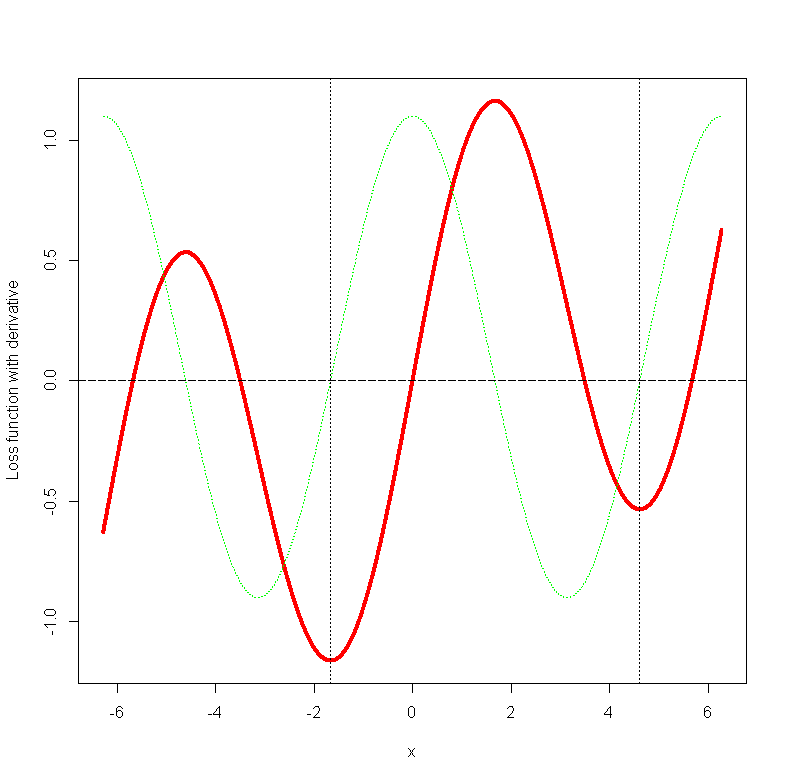
**Grades:** This lab counts 4 ***bonus points*** towards your final grade

**Format of answer:** Your answers (statistical figures and verbal description) should be submitted electronically as Word document. Add a running title with the following information: Lab06, your name and page numbers. Use this document as template: add your answers for each subtask, i.e., 1 (a) etc., in a red color as well as any requested statistical figures. Trial and error answers will lead to a deduction of points. You are expected to hand in professionally formatted answers: use a fixed pitch font, like **Courier New**, for any  code and output.

## Gradient Descent [2 points]

Find the minimum of the loss function in the interval by scripting your gradient algorithm in .

[a] Plot the loss function in the interval . For reference see the plot below with the loss function in red and the gradient in green.



[b] Find the gradient of the loss function with the  function **deriv(~sin(x) + x/10, "x")** .

[c] Program the gradient descent search for a minimum and show your  script.

[d] Evaluate the gradient descent search for the start values and . Discuss the difference based on the two start values.

## Deep Learning [2 points]

Follow the procedure outlined in the second part of the  script **ArtificalNeuralNetworks.r** to find with -fold cross validation the optimal number of epochs to predict the median home values **medv** in Boston’s 506 neighborhoods. Use **data(Boston, package="MASS")**. You can use all features to predict your target variable. Show your  script and the results.

[a] Split the data stratified into 70% training and 30% testing samples.

[b] Build a **keras** model with 2 hidden layers and one output layer. Use a 60 neurons in each hidden layer.

[c] Use -fold cross-validation with to find the optimal number of epochs using the training sample. Use the *mean-absolute-error* as evaluation criterium.

[d] Calibrate the model using all training samples to predict the home values in the test sample.

[e] Compare the predicted home values against the observed home values in the test sample and discuss the prediction quality.