# Project 1 - Feedforward Neural Networks
# Due Jan 31 20:00

January 21, 2022

In this project you will get to implement your own feedforward neural network for regression. Before starting the project, please keep in mind:

- A written report should be handed in by Jan 31 at 20:00 the latest.
- All code should be runable and provided together with the hand in.
- You are encouraged to discuss the problems. However, all reports are to be written individually.
- Any external source can be used to solve the tasks, provided the source is clearly cited and the idea taken from the source explained in your own words.
- You are free to choose any programming language of your own choice.
- Any programming package can be used, as long as you make clear what package is used, for what purpose the package is applied and what parameter values are set when calling a function from the package.

**Task 1:** Consider a fully connected feedforward neural network of $L$ layers, where layer 1 is the input layer and layer $L$ the output layer. The width of layer $l$ is equal to $n_l$. For $l = 2, \dots, L$, let $Z^{[l]} = W^{[l]}A^{[L-1]} + b^{[l]}$ be the linear output at layer $l$, where $W^{[l]}$ is the $n_l \times n_{l-1}$ matrix of weights, $b^{[l]}$ is the row-vector of biases and $A^{[l]} = g^{[l]}(Z^{[l]})$ is the row-vector containing the nonlinear activations of layer $l$. Note that $A^{[1]} = x$ is the input layer. Denote $\delta^{[l]} = \frac{\partial J}{\partial Z^{[l]}}$ for $l = 2, \dots, L$.

1. (25 p) Show that

$$\delta^{[l]} = g^{[l]\prime}(Z^{[l]}) \odot (W^{[l+1]} * \delta^{[l+1]}),\ l = 2, \dots, L-1$$

and

$$\delta^{[L]} = J'(A^{[L]}) \odot g^{[L]\prime}(Z^{[L]})$$

where $\odot$ denotes element-wise multiplication and $*$ denotes ordinary matrix multiplication.

2. (10 p) Show that

$$\frac{\partial J}{\partial W^{[l]}} = \delta^{[l]}(A^{[l-1]})^T$$

$$\frac{\partial J}{\partial b^{[l]}} = \delta^{[l]}$$

   for $l = 2, \ldots, L$.

3. (6 p) Derive the corresponding expression for the gradients in part 2 when mean squared error (MSE) cost function, sigmoid activation function and unit output function is used.

4. (6 p) How would you avoid an exponential blowup of computation when computing the gradients?

**Task 2:** (1 p) Download the data set from course page found under the Project 1 section and plot it.

**Task 3:** (35 p) It is now time to implement training of a network with the package of your choice. Motivate choice of depth and width, choice of activation function, cost function, output function, initialisation of parameters and training algorithm.

Before training, divide the data into a training and validation set with 90 % of the data in the training set. During training, monitor the training error as well as the validation error.

**Task 4:** Plot the validation error and training error curve where the x-axis indicate training epoch and y-axis error.

1. (5 p) Judging from the training error, does it appear that the training algorithm has converged?

2. (5 p) Does the validation curve look as you would expect? If not, can you explain why that is?

3. (5 p) Does the validation curve indicate overfitting? In case of overfitting, how would you proceed to prevent the overfitting?

**Task 5:** (2 p) Plot the function learned by your network. Discuss how good (or bad) the fit by the neural network is.