MTAT.03.227 Machine Learning
Spring 2014 / Exercise session IV
Nominal score: 10p
Maximum score: 15p
Deadline: 9st of April 8:15 EET

1. Neural networks are often used in prediction of continuous variables. As such they can be viewed as a toolbox for interpolation and approximation. This exercises explains these concepts through one–dimensional examples, which are easier to understand and visualise. More precisely, we consider a specific target function which looks like a tooth of a chainsaw:

$$f(x) = \begin{cases} 0, & \text{if } x \notin [0,2] \ , \\ x, & \text{if } 0 \leq x \leq 1 \ , \\ 1, & \text{if } 1 \leq x \leq 2 \ . \end{cases}$$

   (a) Sample 500 random points $x_1, \ldots, x_{500}$ form the range $[-1, 3]$ and compute the corresponding responses for the function f. Train a neural network on this datasets and visualise outputs of hidden layer neurons and the final output. Repeat the experiment when the target function is $\sin x$. How does the neural network approximate $\sin x$ outside the range $[-1, 3]$? Explain this behaviour.(**1p**)

   Hint: The function f can be learnt with two hidden sigmoid neurons and $\sin x$ requires many hidden neurons.

   (b) Study what happens if the network contains significantly more than the minimal number of neurons needed to represent the function f, say 20 neurons. Explain what happens and describe why the outputs of the hidden layer are ill-positioned for fixing linear regression coefficients in the second layer. That is, why some of the points have high leverage and what does it do for the coefficients of the second layer. What consequences it has for the output. (**1p**)

   Hint: Plot neural network with minimal number of neurons and network with many neurons. Compare them. Think whether the outputs of the hidden layer are scattered uniformly over the 20-dimensional unit cube or not and link it with lecture slides.

   (c) Study how regularisation changes the outputs of hidden neurons for the function f. Experiment with minimal number of hidden neurons needed. Explain why this change occurs and how we can use it if the overall shape of the target function is known. (**1p**)

   Hint: `nnet` function has a parameter `decay`, which corresponds to regularisation coefficient.

   (d) Study what happens if the response of $f(x)$ is corrupted with an additive Gaussian noise. For clarity, let $y = f(x) + \epsilon$ where ϵ is drawn form a normal distribution with $\mu = 0$ and $\sigma = 0.1$. Again, try neural networks with 2 and 20 neurons and try different regularisation parameters. Contrast results by showing the response of these models. What do you think how easy is to determine a good number of neurons and a value of the regularisation parameter in general? (**1p**)

2. A standard neural network packages work with a neural network consisting of a single hidden layer of neurons with a sigmoid activation function. Although such networks are theoretically capable of interpolating any continuous function in a fixed range provided that there are enough neurons, there are some drawbacks illustrated by the next two-dimensional approximation task. Let us consider a square hump that is located in the origin of coordinates and is rotated by 45 degrees:

$$g(x_1, x_2) = \begin{cases} 1, & \text{if } |x_1| + |x_2| \leq 1 \ , \\ 0, & \text{if } |x_1| + |x_2| > 1 \ . \end{cases}$$

   (a) Study how easy it is to approximate a two-dimensional square hump $g(x_1, x_2)$ with neural networks. Sample 500 data points uniformly form the square $[-2, 2] \times [-2, 2]$. Visualise the outputs of the hidden layer and the final output. Does anything change if we would use regular measurement grid instead of randomly chosen points? (**1p**)

   Hint: What is the minimal number of neurons you need?

   (b) Now study how the solution behaves outside the training region by drawing the output of of the neural network in the range $[-8, 8] \times [-8, 8]$. Interpret result. What do you think how safe is to use the predictions outside the training range? Is it possible to represent the square hump with a neural network which finite number of sigmoid neurons in the hidden layer? Justify your answer. (**1p**)

   Hint: Each sigmoid neuron can be represented by the straight line where its value is 0.5 and a slope. Think how does the linear combination of two neurons look if their straight lines intersect.

3. The biggest drawback of sigmoid neurons is infinite support. The region where a neurons output is half-space. As a result, the output of a neural network with a single hidden layer can be quite arbitrary outside the training region. This might be rather dangerous is practical applications. An output of a neural network with radial base units is guaranteed to converge to constant default value when we depart from the training region.

   As standard neural network packages in R do not implement radial base neurons, we have to implement the hidden layer by ourselves. Recall that the output of RBF is defined as follows

$$f_{\mathrm{rbf}}(x) = \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

where μ is the location and σ is the width. To simplify the task, we consider one-dimensional approximation task where neurons with fixed centres located at positions $-20, -19, \ldots, 19, 20$ and fixed width σ and use functions $f_1(x) = sign(x)$ and $f_2(x) = x$ target functions.

(a) Generate a training set by sampling 200 values uniformly from the range $[-20, 20]$. Define a function `RBFLayer` that given an input x outputs RBF-s with centres $\mu_1, \ldots, \mu_k$ and the width $\sigma = 0.5$. Next, write a wrapper around this function that maps inputs $x_1, \ldots, x_{200}$ to outputs of the hidden layer. The result should be a data frame where columns are neurons outputs. Add corresponding target values for the function $f_1(x)$. Use the linear regression `lm` routine to find the weights of the aggregating neuron. (**1p**)

(b) The approximation quality of RBF network is tightly linked to the placement of centres compared to width. If centres are too far apart compared with their withs then the RBF network creates jagged output. To observe this effect, use the same set of inputs $x_1, \ldots, x_{200}$ and try to approximate target functions with four RBF networks with different widths $\sigma \in \{0.1, 0.4, 0.5, 2\}$. For that you should first package the solution obtained in the first subtask as a function `TrainRBFNetwork` that takes in inputs $\boldsymbol{x}$, outputs $\boldsymbol{y}$, locations of centres $\boldsymbol{\mu}$ and the width σ and outputs a linear model corresponding to output neuron. Secondly, define a function `PredictRBF` that takes in the linear model specification, inputs $\boldsymbol{x}$, locations of centres $\boldsymbol{\mu}$ and the width σ and computes the output of RBF network. Draw illustrative plots in the range $[-25, 25]$ for both target functions and for all kernel widths. Interpret the result. (**1p**)

(c) Modify the function `TrainRBFNetwork` so that it would use regularised linear regression function `lm.ridge` from MASS package. Next, study how regularisation parameter in `lm.ridge` influences the smoothness of the RBF network. Consider the RBF network with the same placement of centres as above and with width $\sigma = 2$. Use $f_1(x)$ as the target function and but try different λ values, e.g. $\lambda \in \{0, 0.1, 10, 1000\}$. Interpret results. (**1p**)

(d) Study how the amount of noise influences the approximate size of the best regularisation parameter λ. For that consider the target function $f_1(x)$ together with Gaussian noise $\mathcal{N}(0, \rho)$ for $\rho \in \{0.01, 0.1, 0.5\}$. Again, try different values of $\lambda \in \{0, 0.1, 10, 1000\}$. Draw some illustrative plots. Is there a single λ value that is universal for all noise levels? What else can you conclude about reconstruct ability of a function under different noise levels. (**1p**)

4. The aim of this exercise is illustrate what happens during the training phase. Let us consider a simple neural network with a single input x, two hidden sigmoid neurons and linear output neuron.

(a) Express the output of the neural network as an analytical function $f(\boldsymbol{w}, x)$ where the vector $\boldsymbol{w}$ contains all wights in of the neural network. Document which weight corresponds to which neuron and which input wire. Next compute partial derivatives $\frac{\partial f}{\partial w_i}$ with respect to each weight w_i (**1p**)

(b) Look up the description of stochastic gradient descent form lecture materials or from exercise session materials. Try to write the corresponding method that minimises mean square error on the training set. For that you should use the chain rule for derivatives and the results obtained from the first subtask. (**1p**)

(c) Write a program in GNU R that computes the derivative $\frac{\partial f(x,\boldsymbol{w})}{\partial w_i}$ at any given point. Use this to implement stochastic gradient minimisation algorithm. (**1p**)

(d) Show that the resulting learning algorithm can be indeed used to train the neural network to approximate the target function from the first exercise. (**1p**)

5. Recall that $\sin(wx)$ is a pathological neuron that can recall all input output pairs (x_i, y_i) provided that $|y_i| \leq 1$. Let us verify this in a simplified case where $x_i = i$ and $y_i \in \{0, 1\}$ and the output is binarised with a threshold neuron, i.e., the output of a neural network is $sign(\sin(wx))$.

   (a) As the output is binarised, we must find w such that the signs of $sin(w), \ldots, \sin wk$ would be the same as for $y_1, \ldots, y_k$. Use a graphical method to solve this for possible labelings for $k = 3$. (**1p**)

   (b) Write a corresponding algorithm that solves this problem for $k = 8$ and present a solution for labelling $-1, -1, -1, 1, -1, -1, 1, -1$ (**2p**)

6. Try to predict the house prices (MEDV) in the Housing Data Set using neural networks. First, use cross validation or bootstrapping to select the right number of hidden layer neurons. Second, try to tune the weight decay parameter in the same manner. Third, run the basic diagnostics to detect any anomalies:

   - Is the discrepancy between training and test error reasonable?
   - Is residues (prediction errors) strongly correlated with any of the inputs? For binary and categorical variables, you can do t-test or see whether a linear model can predict these residues. For continuous variables, you can compute correlation and use permutation test for determining significance threshold. That is, permute residue vector randomly 100 times. If the absolute value of computed correlation is large than 95 correlation values obtained in the simulations, then the correlation is still strong between residues and variable. The latter means that model is still sub-optimal.

- Plot the histogram of residues and estimate visually whether it is reasonably close to normal distribution. Long tails and several humps are clear indication that it is far form normal distribution and thus the mean square error based training methodology might have been sub-optimal choice.

To investigate whether the use of neural networks was justified by comparing it with a simple linear regression model. To the same diagnostics and report whether cross validation errors differ or not and whether neural networks perform better wrt diagnostic measures. (**4p**)

7. Another way to reduce the number of data samples needed to train the network is to reduce the number of connections, i.e., set some weights to zero. To do that, you should somehow estimate the how much the weights fluctuate over all data sets. After that you can set those weights to zero that fluctuate around zero and train the network again. There are only two problems with this approach. You do not have many datasets to try and due to random starts in gradient descent resulting end configuration can be completely different. For instance, the outputs of hidden layer are just randomly permuted. To resolve these issues train the network over the entire data. Then store weights and draw 100 bootstrap datasets for training. Use stored weights as initial weights for gradient search. Record the distribution of all weights and compute mean and standard deviation for each weight. Set all weights to zero if they are less than 1.5σ away from zero. Train the network again. Document procedure and report whether such approach improves results. (**5p**).