KAUNAS UNIVERSITY OF TECHNOLOGY

FACULTY OF INFORMATICS

INTRODUCTION TO ARTIFICIAL INTELLIGENCE

LABORATORY WORK 3

REPORT

Student: Gurban Shukurov

Lecturer: dr. Germanas Budnikas

Kaunas
2023

Table of Contents

Initial dataset description:

Selected Dataset: dataset of house sale prices for King County, including Seattle

Link: https://www.kaggle.com/code/tomasmantero/predicting-house-prices-keras-ann/input

Context: 21 columns (features). 21597 rows.

Format:

id: Unique ID for each home sold
date: Date of the home sale
price: Price of each home sold
bedrooms: Number of bedrooms
bathrooms: Number of bathrooms, where .5 accounts for a room with a toilet but no shower
sqft_living: Square footage of the apartments interior living space
sqft_lot: Square footage of the land space
floors: Number of floors
waterfront: A dummy variable for whether the apartment was overlooking the waterfront or not
view: An index from 0 to 4 of how good the view of the property was
condition: An index from 1 to 5 on the condition of the apartment
grade: An index from 1 to 13, where 1-3 falls short of building construction and design, 7 has an average level of construction and design, and 11-13 have a high quality level of construction and design.
sqft_above: The square footage of the interior housing space that is above ground level
sqft_basement: The square footage of the interior housing space that is below ground level
yr_built: The year the house was initially built
yr_renovated: The year of the house's last renovation
zipcode: What zipcode area the house is in
lat: Lattitude
long: Longitude
sqft_living15: The square footage of interior housing living space for the nearest 15 neighbors
sqft_lot15: The square footage of the land lots of the nearest 15 neighbors

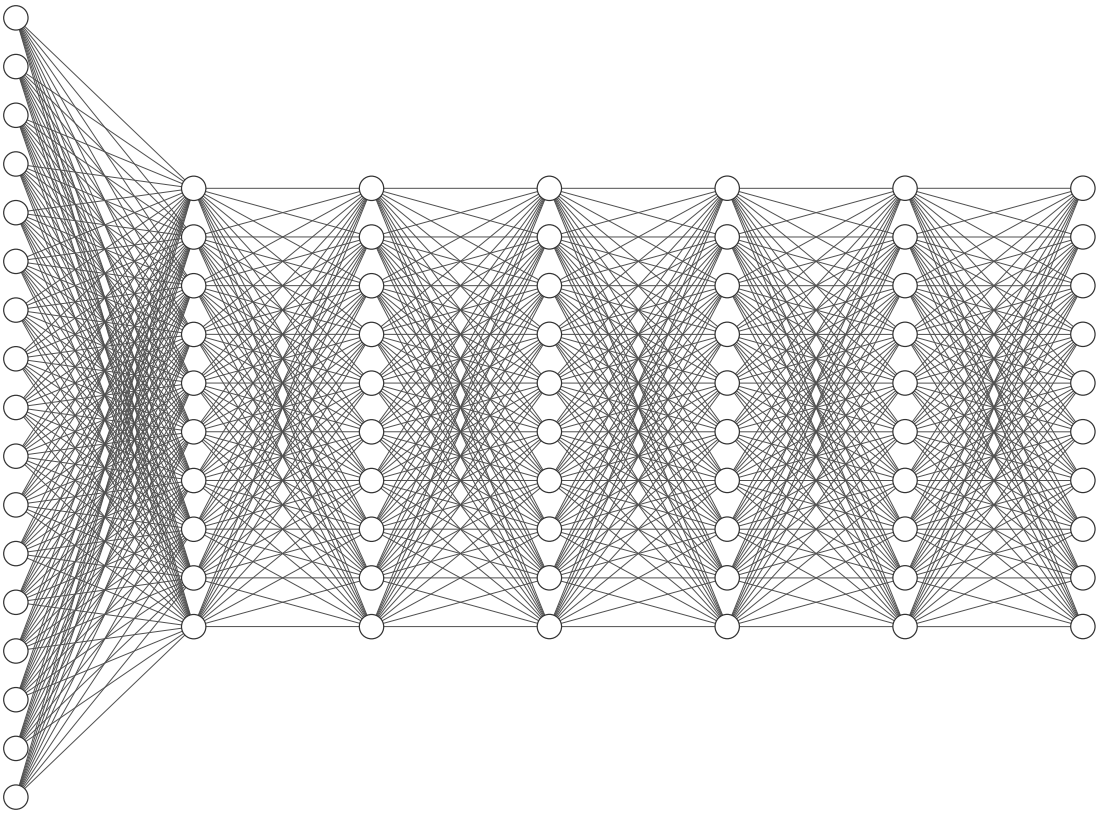
Rearrangements:

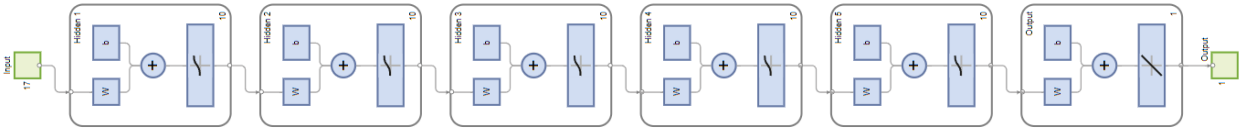
Following rearrangements were done before building a prediction model:

1) Deletion of columns **yr_renovated** (most of the values were zeros), **long** (negative numbers), **date** (had no possibility to read the date values).
2) Deletion of 19859 rows (to reduce the time spent for training the model).
3) Normalization performed (to significantly decrease MSE values).
4) Changing the location of the column "price", moving it to make it the last column (this column was selected as a target attribute).

Initial ANN architecture:

The ANN architecture was constructed with 5 hidden layers that use *logsig* (Logarithmic sigmoid transfer) activation function, 1 output layer with *purelin* (Linear transfer) activation function and 17 inputs (target attribute was excluded from inputs as it is the one to be predicted). All layers (except input and output) consisted of 10 neurons. Levenberg-Marquardt backpropagation algorithm was used to train the model:





```
% Load the dataset
data = readtable('normalized.csv');

% Set the target attribute
target = 'price';

% Split the data into inputs and targets
varNames = data.Properties.VariableNames;
varNames = varNames(~ismember(varNames, target));
inputs = data(:, varNames);
inputs = table2array(inputs);
targets = table2array(data(:, target));

% Create the neural network
net = feedforwardnet([10,10,10,10,10]);
net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'logsig';
net.layers{3}.transferFcn = 'logsig';
net.layers{4}.transferFcn = 'logsig';
net.layers{5}.transferFcn = 'logsig';
net.layers{6}.transferFcn = 'purelin';
net.trainFcn = 'trainlm';                       % use Levenberg-Marquardt algorithm for training
net.trainParam.epochs = 1000;                   % set the number of training epochs
net.trainParam.lr = 0.00024670416823848107;     % set the learning rate
```

Application of 10-fold cross validation:

Non exhaustive K-Fold cross validation method was used to evaluate the performance of the prediction model. Across all 10 folds, at each fold the MSE and accuracy values were calculated to measure the error and see how exact the model predicts the target attribute values:

```
% Configure 10 fold cross-validation
cv = cvpartition(size(inputs, 1), 'KFold', 10);

mse = zeros(1, cv.NumTestSets);
accuracy = zeros(1, cv.NumTestSets);

% Train and test the neural network for each fold
for i = 1:cv.NumTestSets
    trainIdx = cv.training(i);
    testIdx = cv.test(i);
    xTrain = inputs(trainIdx,:);
    tTrain = targets(trainIdx);
    xTest = inputs(testIdx,:);
    tTest = targets(testIdx);

    % Train the neural network
    [net, tr] = train(net, xTrain', tTrain');

    % Test the neural network
    yPred = net(xTest')';

    % Compute the accuracy of the prediction model
    diff = abs(yPred - tTest);
    acc = 100 * (1 - mean(diff./tTest));
    accuracy(i) = acc;
    disp(['Fold ', num2str(i), ' accuracy: ', num2str(acc), '%']);

    % Compute mse
    mse(i) = mean((tTest - yPred).^2);
    disp(mse);
end
```

```
Fold 1 accuracy: 64.6076%
    0.0012         0         0         0         0         0         0         0         0         0

Fold 2 accuracy: 71.8292%
    0.0012    0.0007         0         0         0         0         0         0         0         0

Fold 3 accuracy: 77.2042%
    0.0012    0.0007    0.0006         0         0         0         0         0         0         0

Fold 4 accuracy: 77.8195%
    0.0012    0.0007    0.0006    0.0005         0         0         0         0         0         0

Fold 5 accuracy: 76.8135%
    0.0012    0.0007    0.0006    0.0005    0.0011         0         0         0         0         0

Fold 6 accuracy: 74.9983%
    0.0012    0.0007    0.0006    0.0005    0.0011    0.0009         0         0         0         0

Fold 7 accuracy: 71.7795%
    0.0012    0.0007    0.0006    0.0005    0.0011    0.0009    0.0009         0         0         0

Fold 8 accuracy: 76.977%
    0.0012    0.0007    0.0006    0.0005    0.0011    0.0009    0.0009    0.0007         0         0

Fold 9 accuracy: 59.3015%
    0.0012    0.0007    0.0006    0.0005    0.0011    0.0009    0.0009    0.0007    0.0006         0

Fold 10 accuracy: 73.8729%
    0.0012    0.0007    0.0006    0.0005    0.0011    0.0009    0.0009    0.0007    0.0006    0.0012
```

Increase of the averaged performance. Change of ANN architecture:

Initially the performance of the model was as follows:

```
Mean MSE:
0.00084
Mean accuracy:
72.5203%
```

```
% Compute the mean MSE over all folds
mean_mse = mean(mse);
disp('Mean MSE:');
fprintf('%.5f\n', mean_mse);

% Compute the mean accuracy over all folds
mean_acc = mean(accuracy);
disp('Mean accuracy:');
disp([num2str(mean_acc), '%']);
```

Results were good enough, but it was required to increase the averaged performance by 5% and test the model again. For that 5 additional hidden layers were added, number of neurons in each layer changed to 12 and built-in *maxlinlr* function was used to know the exact learning rate suitable for this model:

```
% Create the neural network
net = feedforwardnet([12,12,12,12,12,12,12,12,12,12]); % 10 hidden layers with Logarithmic sigmoid transfer activation function:
net.layers{1}.transferFcn = 'logsig';                  % use logsig activation function for hidden layer
net.layers{2}.transferFcn = 'logsig';
net.layers{3}.transferFcn = 'logsig';
net.layers{4}.transferFcn = 'logsig';
net.layers{5}.transferFcn = 'logsig';
net.layers{6}.transferFcn = 'logsig';
net.layers{7}.transferFcn = 'logsig';
net.layers{8}.transferFcn = 'logsig';
net.layers{9}.transferFcn = 'logsig';
net.layers{10}.transferFcn = 'logsig';
net.layers{11}.transferFcn = 'purelin';
net.trainFcn = 'trainlm';                      % use Levenberg-Marquardt algorithm for training
net.trainParam.epochs = 1000;                  % set the number of training epochs
net.trainParam.lr = 0.000247100964773;         % set the learning rate
```

```
% Get the most optimal learning rate
b = net.b{1};
maxlr = maxlinlr(inputs, b);
fprintf('MaxlinLr: %.15f\n', maxlr);
```

And the results were obvious, MSE decreased and accuracy increased by exactly 5%:

```
Mean MSE:
0.00067
Mean accuracy:
77.8534%
MaxlinLr: 0.000247100964773
```

Also, replacing the activation function with any other function could help to change the result for better, but it was not necessary for this model because the used functions already performed the best.

Conclusion

The goal of this experiment was to build a prediction model using the artificial neural network that will manage to forecast the target values with at least 70% of accuracy. This goal was achieved. While doing the experiment it was understood that writing a code is not enough to build a prediction model, and a model cannot be correct for all the datasets. Each model should be designed for a particular problem (dataset) in order to make proper predictions. Event if the model is specific to a problem, some changes should be performed to the dataset for reducing mean squared error and/or increasing accuracy of the model. In addition, usage of suitable activation functions is mandatory for a concrete prediction model and its layers (hidden, output); generally, without any activation function it will be a simple linear regression model, not a neural network; without suitable activation function the model will be inaccurate and ineffective. Selecting appropriate cross validation method is also necessary for evaluating the performance of the model; K-Fold was used because it is the most efficient for the given problem, it is less biased and more accurate.

References

https://www.mathworks.com/matlabcentral/answers/421958-where-can-i-find-a-neural-network-transfer-functions-list
https://www.mathworks.com/help/stats/cvpartition.html#mw_463e6431-ba17-47c8-a91f-9d195cf094f7
https://www.nickmccullum.com/python-deep-learning/artificial-neural-network-tutorial/#building-the-artificial-neural-network
https://towardsai.net/p/machine-learning/how-to-build-and-train-your-first-neural-network-9a07d020c4bb
https://blogs.mathworks.com/loren/2015/08/04/artificial-neural-networks-for-beginners/
https://www.turing.com/kb/how-to-choose-an-activation-function-for-deep-learning
https://www.mathworks.com/help/deeplearning/ref/trainlm.html
https://medium.com/the-rise-of-unbelievable/what-is-cross-validation-and-when-to-use-which-cross-validation-327d25bbb3f3