COURSEWORK REPORT

Module title and code: Machine Learning – COIY065H7

Name: MARCO CATANIA

Email: mcatan01@dcs.bbk.ac.uk

Dataset: Detection of malignant regions in colonoscopy video frames

Academic Declaration

I have read and understood the sections of plagiarism in the College Policy on assessment offences and confirm that the work is my own, with the work of others clearly acknowledged. I give my permission to submit my report to the plagiarism testing database that the College is using and test it using plagiarism detection software, search engines or meta-searching software.

Marco Catania

I – Method and methodology

1) Neural network method

The machine learning solution applied to the classification problem is neural network.

“neuralnet” in R programming language is the library used to train the network. “neuralnet” is a back-propagation algorithm.
Back propagation is a supervised learning procedure for training feed-forward neural networks to learn from test samples. The training set is presented to the net. The errors between the actual and desired output of the net are propagated backward to the internal layer in order to adjust the connection weights in proportion to their contribution of the error. Backpropagation is as useful as computationally intensive.

RPROP+ is the backpropagation algorithm chosen. RPROP+ is a full-batch resilient backpropagation algorithm with weight backtracking, that requires less computation than the classic backpropagation algorithm. RPROP+ converges 4 times faster than classic backpropagation (1). Weight backtracking means that if there was a sign change of the partial derivative of the total error function, each weight is multiplied by a particular factor (update)to reflect that change. Full-batch means that the algorithm uses the training set as one batch to train the network at each iteration.

The number of hidden nodes varies from 5 to 25 by step of 5. Only one layer is used.
This is to study the impact of an increase in nodes number on the result of the network.

RPROP+ in “neuralnet” allows to not choose any initial learning rate. A learning rate is chosen by the algorithm randomly when the training is initiated. Then if the total error does not decrease at a certain point, the algorithm adjusts the learning by a default value (minus: 0.5, plus: 1.5) until the total error decreases again. Thus, running several tests to find the appropriate learning rate is avoided

The activation function is sigmoid (“logistic”). Sigmoid function is a S shape function. The function gives to the neuron output a value between 0 and 1 (and not 0 or 1) that is interpreted as a probability. When a threshold is reached, the neuron is activated and fires an output. Sigmoid function suits particularly to classification problems.

The error function chosen is Sum of Squared errors of prediction (SSE). SSE is the sum of square of residuals (deviation from the target value). SSE computes the error in the neural network capacity to predict the target value. The goal of the network training is to adjust the neuron weights until the SSE is minimised to reach a threshold.
The threshold chosen (default in "neuralnet") is 0.01, as the total percentage of error when the network makes a prediction.

The library has a default value of 1e+05 as the maximum of step allowed to the training (stepmax). The training stops automatically even if the network did not converge. To allow more time to the network to reach convergence, especially when the number of neurons increases, a stepmax value of 1e+09 has been chosen.

2) Data pre-processing and normalization

The data set tables are grouped in two data frames: train (that contains all the data frame with the training data) and test (that contains all the data frame with the test data).

The target values of both train and test dataset are converted to have only 0 and 1 (=255) as output.

The data points are scaled to [0,1] as to avoid any misclassification due to outlier. "Caret" library in R is used to scale the data points (preProcess function)

Principal Component Analysis (PCA) is used to reduce the dimensionality of the dataset. High dimensionality of the dataset gives a low training error rate but fails to generalize to new data (high test error rate).
PCA aims to reduce the number of variables in the study to a few that expresses most of the variation within the dataset. PCA uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components. This transformation is defined in such a way that the first principal component has the largest possible variance , and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components.
(2). The most relevant vectors obtained (principal components) are then used to train the network.
PCA is also applied on the test data set before prediction with the models obtaining after training of the network.

The advantages of PCA are:

- Less computation as the number of features decreases (in general two or three Principal Component vectors) that implies a training much faster.
- Accuracy of the prediction as the Principal Component vectors still explain a large part of the variability

PCA gives here the two first principal components of the dataset (train and test) as explaining more than 95% of the variability.

“prcomp” function in R is used to compute PCA.

II – Experiments, findings and discussion

1) Data pre-processing justification

The neural network is first trained on the dataset not pre-processed (PCA not computed, data points not scaled). The average success is more than 88% and seems high. But the result is biased as the network only predicts the output 0 for all observations (that is because the output 0 corresponds to 88% of the output in the test dataset. This explains the high average success. The network fails to identify even one malignant region in the videoframes. Therefore, the network obtains a poor result. The SSE is 102.

The training on the unprocessed dataset however still takes less time comparing to the training on the pre-processed dataset (0.3s vs 0.99s on 5 neurons one-layer network).

2) Network training on the pre-processed data

The network is trained on the pre-processed data using a one-layer neural network, then a two layers network. The number of neurons is gradually increased to study the impact of the neural complexity on the performance of the network. Then the performance (percentage of correct prediction) is computed using the test dataset.

The algorithm also gives the time of training, the final SSE and the number of steps until convergence.

a- 5 neurons layer

The network obtained now predicts output 0 but also output 1. It takes 1425 steps to reach convergence, with a final SSE of 98.8 in 5 seconds. Performance obtained is 86.15%.

b- 10 neurons layer

The network takes 10437 steps to reach convergence, with a final SSE that decreases to 95.81 in 15.12 seconds. Performance obtained decreases to 83.01%.

c- 15 neurons layer

The network takes 23865 steps to reach convergence, with a final SSE that decreases to 93.27 in 48.06 seconds. Performance obtained stays stable to 83.56%.

d- 20 neurons layer

The network takes 23507 steps to reach convergence, with a final SSE that decreases to 90.89 in 59.44 seconds. Performance obtained stays stable to 83.80%.

e- 25 neurons layer

The network takes 82319 steps to reach convergence, with a final SSE that drops to 87.88 in 4.24 minutes. Performance obtained dives to 80.03%.

f- 3 – 2 neurons on two layers

The network takes 1249 steps to reach convergence, with a final SSE at 98.63 in 99 seconds. Performance obtained augments to 87.23%.

g- 5 – 5 neurons on two layers

The network takes 69396 steps to reach convergence, with a final SSE that decreases to 92.38 in 1.32 minutes. Performance obtained is stable to 87.57%.

h- 8 – 7 neurons on two layers

The network takes 1.598.818 steps to reach convergence, with a final SSE that drops to 81.55 in 50.28 minutes. Performance obtained dives to 79.92%.

3) Discussion

The increase of neuron number has as effect to continuously decrease the SSE (so the performance on the training dataset) and to improve the performance of the neural network (the performance of the 1 neuron layer is comparable to the performance on the unprocessed data). But when the number of neurons reaches a certain point of complexity, in the present case 5, the performance caps, is stabilised, then decreases for every 5 neurons added to the layer. But the SSE keep decreasing.
This phenomenon is called overfitting as the increasing complexity of the neuron overfit the network to the training data. The network obtained then fails to generalize to new data (test dataset).
Also, the increase of complexity of the network is computationally expensive as the network takes more time to converge (50 time for the 25 neurons layer compared to the 5 neurons layer network). Without increase in performance.
It seems that the maximum level of network complexity before drop in performance is correlated to the size of the training dataset. As in the commercial applications of neural

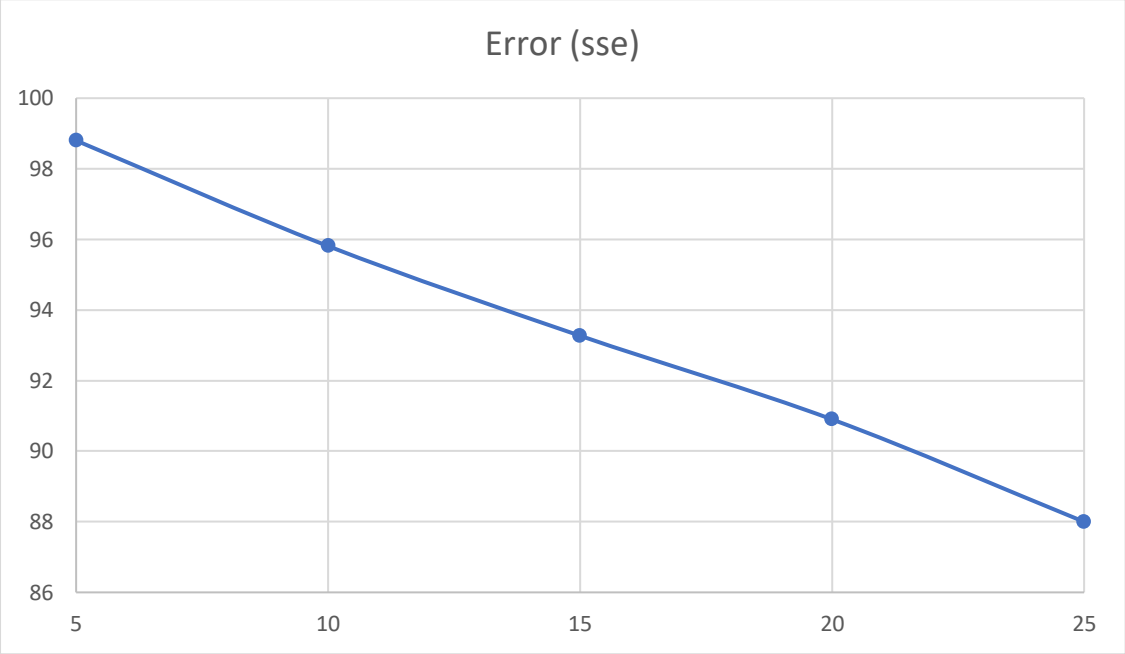
networks, high complexity (especially deep learning) is correlated to the density of the training data set (big data).

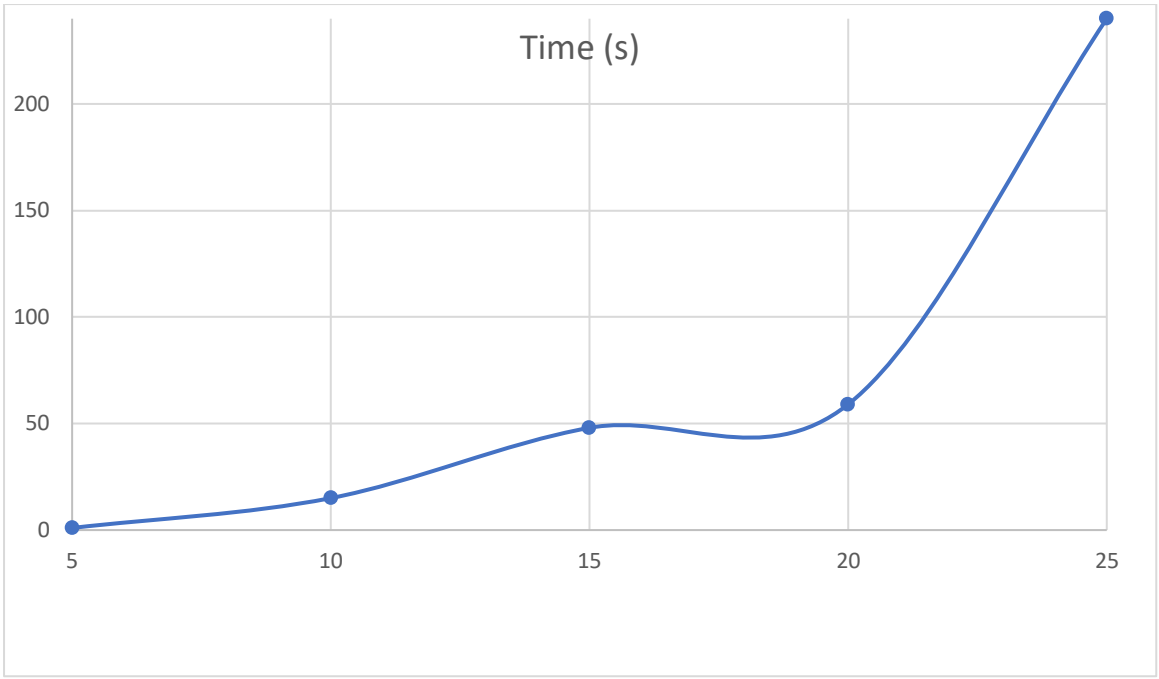
The long training time for layers of more than 25 neurons did not allow to continue experimenting. But the previous experiments let supposes that that SSE will continue to decrease but the network obtained will give even poorer generalization performance.

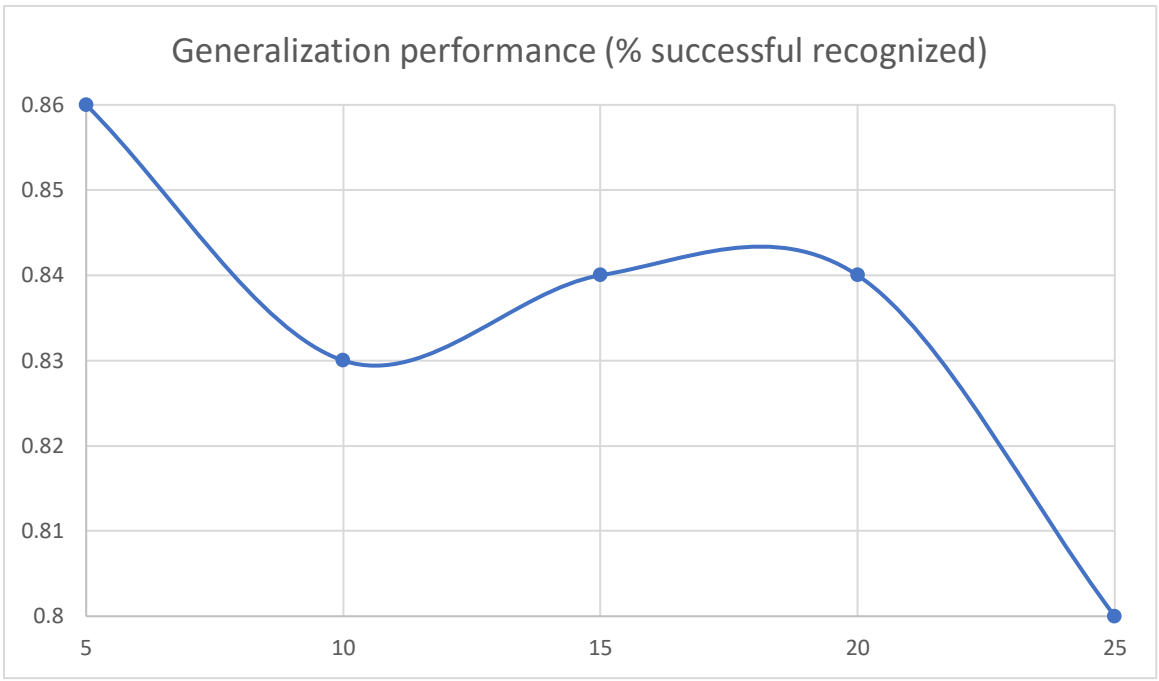
One question is to know if keeping the same number of neurons that gives the best performance on one-layer networks (5 neurons), but instead distributes the neurons on 2 layers, will increase the performance. Experiments have been extended to the other levels of complexity (10, 15, 20, 25).

At 5 neurons distributed on 2 layers, performance is better than the one-layer network with an increase of approximately 1% (87.23 % vs 86.15%). The time to reach convergence is multiplied by 12, but still reasonable as in seconds (5 seconds to 1 minute). The trade-off time/performance is in favour of the 2 layers network. However, a larger training dataset would increase the convergence time dramatically and would not justify the use of a more complex network if the improvement in performance is small.
But as observed during the experiment with the one-layer networks, further increases in complexity gives worst performance result for a convergence time that dramatically increases by 50.

The best performance overall is obtained by the 5 – 5 two layers network with a performance of 87.57% in 1.32 minutes. Here the computation time increase is still reasonable compared to the previous best network (3 – 2 two layers in 1 minute), even compared to the best network in one layer (5 neurons) if we take in consideration the trade-off time/performance.




Time (s)
200
150
100
50
0
5
10
15
20
25


Generalization performance (% successful recognized)
0.86
0.85
0.84
0.83
0.82
0.81
0.8
5
10
15
20
25

One-layer networks performance (%)

Number of neurons	Successful 0	Successful 1	Overall Performance
5	97.37	4.51	86
10	92.70	12.55	83
15	94.44	4.35	84
20	94.65	4.86	84
25	89.92	8.04	80

Two-layer networks performance (%)

Number of neurons	Successful 0	Successful 1	Overall Performance
3-2	98.96	1.91	87
5-5	99.44	1.14	88
8-7	89.69	8.77	80

III – Conclusions

The general idea of the study was to verify that the increasing complexity of a neural network is correlated positively to the performance of the network on a test dataset (generalization performance). Our experiments show that the level of complexity of the network has to be correlated to the density of the training dataset to avoid overfitting. In fact, if the training dataset is not large enough, a network excessively complex overfits the training dataset and fails to generalize on the test dataset. Even if the Sum of Square error continuously decreases as the complexity increases, the network generalization performance increases then stays stable until a certain point, then fall dramatically. Therefore, when training a network on a dataset, multiple tests have to be performed to find the appropriate level of complexity for this particular dataset.
Also, a multi-layer network can be considered if it increases significantly the generalization performance (especially for more complex problems with dense dataset). Even in this case, a trade-off between performance increase and computation time has to be weighted as multi-layer networks are computationally intensive, for an increase in performance that is not always significant for less complex problems.

Furthermore, the importance of dataset pre-processing has been demonstrated as data scaling and use of Principal Component Analysis on the dataset have increased significantly the accuracy of our predictions on the test dataset. However, we could not show that PCA decreases the training time.

Neural networks, and statistical learning techniques in general, are a case-by-case techniques that have to be fine-tuned to the problem.

IV – Bibliography

(1) Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The Rprop algorithm. Proceedings of the IEEE International Conference on Neural Networks, 586-591, IEEE Press, 1993.

(2) Principal Component Analysis, definition available online at https://en.wikipedia.org/wiki/Principal_component_analysis.

INSTRUCTION FOR USING THE CODE

The code is written in R markdown, exported with encoding ASCII.
The code and output can be viewed in any web browser. But to run the code on a particular computer, R has to be installed on the machine, with RMarkdown.

Also, the code uploads the dataset from a directory that is particular to the programmers' computer. To properly run the code and upload the dataset related ("endoscopy-video-frames-data" folder), the user should change the path in the code to the location on his/her computer, where the dataset is stored. If failed, the code will fail to load the data and not run properly.

The dataset is available at on Moodle.

APPENDIX (files info)

(1) Markdown.Rmd
File that contains the notebook in format Rmd
The notebook has been exported with ASCII encoding

(2) Markdown.nb.html
Notebook encoded in ASCII
Can be opened by any web browser