Artificial Neural Networks

Machine Learning for Data Science Homework #6

Vid Stropnik; 63200434

ABSTRACT

For the final homework in the Machine Learning for Data Science class, I worked with Artificial Neural Networks. Here, I present my own implementations of an ANN, fit for solving both regression and classification problems. I show results on the (now already well known) *housing* datasets, used throughout the present series of homeworks and compare my results to those of other machine learning models, implemented in the previous assignments. I compare the performance of the model using different hyper-parameters of the ANN. Finally, I also present the submitted results on a bigger data set, which are to be evaluated as the final part of this homework.

1 ARTIFICIAL NEURAL NETWORK

In the enclosed *Python* program, I implement the class *ANN*, which returns an object of type *predictor*. Both of these are generalizations of classification and regression solutions and are implemented in such a way that the Neural networks for the two types of problems share as much source code as possible. In fact, the main logic of prediction and self updating (back-propagation) is shared between the two classes. The ways in which the child classes differ include:

- the observed loss function (only for evaluation),
- the activation function in the prediction layer of the forward propagation (*softmax* in case of classification, no activation for regression),
- minor differences upon initialization (setting the correct layer dimensions, one-hot encoding the ground truth target variables in case of classification,
- QoL functions and dummy functions for more intuitive code.

The implemented approach is optimized manually, without the use of a peripheral library. All the input features are transformed by removing the mean and scaling to unit variance. The optimization is done using Polyak gradient descent with an adapting momentum term μ. The learning rate α in the function is hyperparametric and fixed, however the logic for its procedural contraction is also implemented. A fixed-size regularization term λ is also added. The final hyperparameter, `units`, defines the size and number of hidden layers in the neural network.

2 NUMERICAL VERIFICATION OF GRADIENT

One can confirm the gradient numerically by comparing the computed values to

$$g(\theta) \approx \frac{E(\theta + \epsilon) - E(\theta - \epsilon)}{2\epsilon},$$

where E denotes the loss function the network strives to minimize and θ the weights, with respect to which the error should be minimized. This is due to the direction in the gradient equalling $-\frac{\partial E}{\partial \theta_{ij}}$.

The proposed loss functions in our case can be, for example, (Root) Mean Square Error of Regression and Log Loss for classification.

The verification method is not implemented for continuous checking in the provided python code. However, I was able to verify example outputs by hand up to a tolerance of $1e^{-2}$.

3 LAYER SIZE AND COMPARISON TO OTHER METHODS

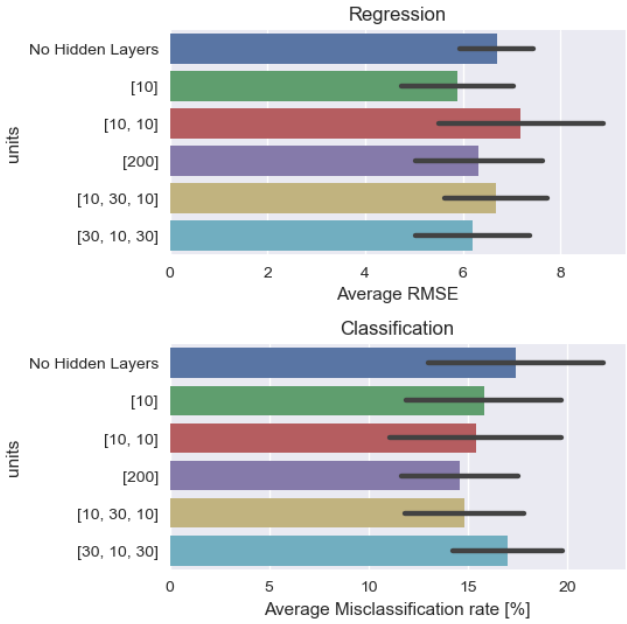

Figure 1: Comparison of Neural Network loss on different hidden layer arrangements. The used loss functions conform to those used in [1, 2]. All evaluation is done using 5-fold cross validation on housing datasets, with starting hyperparameters $\lambda = 1e^{-5}, \alpha = 1e^{-3}, \mu = 0.3$. Confidence intervals denote standard deviations of loss.

Figure 1 shows the evaluation of loss of the ANN model with several different layer arrangements in both classification and regression applications. All evaluation was done on the provided Housing datasets. We can observe that in both scenarios, a single-layered arrangement performs best. We can compare these measures of loss to those achieved in previous assignments. We can recall that in [1], the achieved misclassification rate of a decision tree was $\sim$ 16%, while the best performing ensemble models (Bagging, Random forests) achieved as little as $\sim$ 13 % on average. We can observe that, while confidently beating the simple decision tree in most hidden-layer arrangements, our implementation of ANN was not able to beat the ensemble methods.

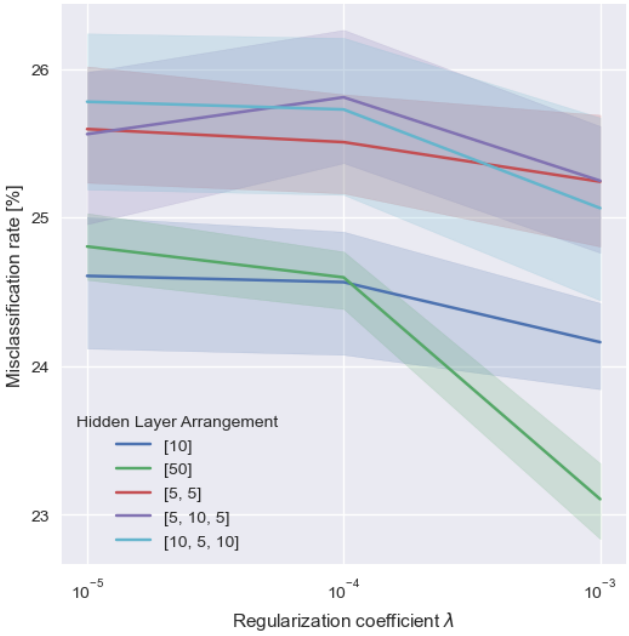

Figure 2: The results of internal cross validation show that the two models with a single hidden layer outperformed the more complex model, no matter the learning rate. The performance of the selected model is conveyed by the final point of the green line. All evaluations are done on 10k epochs with $\alpha = 0.1, \mu = 0.3$. The uncertainty is evaluated using bootstrap on the 5 results, output from internal CV.

Our regressor, however, was able to achieve much better results than those shown in [2], where the best performing models achieved a RMSE of ~ 10-11. We can observe that even a network with no hidden layers (analogous to simple linear regression) can outperform the best-performing SVR model. This is likely due to the different optimization techniques used in the two homeworks.

4 MULTINOMIAL CLASSIFICATION - BIG DATASET

Finally, we apply our ANN Classifier to a big dataset with 93 features and 9 classes. First, internal cross validation was performed on the `train dataset` to determine the best possible learning parameters. Due to computational limitations, the grid search was limited to 5 different hidden layer arrangements and 3 different regularization indices, while the starting [1]learning and momentum rate were fixed at $\alpha = 0.1, \mu = 0.3$. These decisions were made due to the computational intensity of working with the described data-set; 5-fold cross validation across the 15 combinations took several hours. The five folds over a single combination took between 7 and 15 minutes, depending on the hidden layer construction. In the longest of the observed layer constructions (a single layer of 50 neurons), over 5000 weights and 50 biases had to be accounted for in each learning iteration. Learning on this data set was bound at 10000 epochs, bar certain stopping criteria.

Of the observed combinations, the hyper parameters $\lambda = 1e^{-3}$, units = [50] yielded the best results in internal cross-validation, with a misclassification rate of ~ 23%. These results can be observed in Figure 2. The selected model, trained on all provided training data, achieved a log loss measure of 0.49 on it. It's predictions on the testing data are provided in the included `final.txt` file. The fitting of the model takes approximately 3 and a half minutes.

5 EVALUATION AND CONCLUSION

Given 9 possible classes, I deduce that a 23 % misclassification rate on unseen data is an acceptible error rate and that the proposed model is working well. Given that the dataset seems to be predicted best by networks with a single hidden layer and that performance is improved by its size, I hypothesise that this performance could be improved by a bigger hidden layer. These experiments were not attempted due to computational and time constraints.

Overall, this assignment has given me a great overview on the inner workings of neural networks. I firmly believe that the insights achieved here will aid me in my future work in deep learning.

REFERENCES

[1] Vid Stropnik. Decision trees. *Machine Learning for Data Science Homework*, #1, 2021.
[2] Vid Stropnik. Support vector regression. *Machine Learning for Data Science Homework*, #4, 2021.

[1]As noted before, both the learning rate and momentum coefficients are implemented dynamically so they might change upon running. However, in the internal experiments, they stayed fixed throughout.