FYS-STK4155: Project 2

Simen Nyhus Bastnes

13. November 2020

Abstract

In this project, we want to study a few methods commonly used in data science for both normal regression problems, as well as classification problems. The methods we will be employing are the stochastic gradient descent (SGD) and feed-forward neural networks (FFNN). For regression, we will be looking at is the Franke function studied in the previous project [1], comparing our results to the ones found there. For classification the MNIST database of handwritten digits will be studied. Cross-validation will be used for the hyperparameter search for both problems in order to make sure the model is the optimal one. For the Franke function, we found that both SGD and FFNN performed worse than both OLS and Ridge, however their R^2 scores, 0.84 and 0.93 respectively, indicate that the results are not too terrible, at least not for the neural network. For the MNIST data set, both the SGD and FFNN performed very well, with an accuracy of $\sim 95\%$, surpassing the results gotten with Scikit-Learn. The FFNN results are a bit weird as it seems to converge extremely fast, with very few nodes in the network.

1 Introduction

In the last two decades, computation power and availability of said machines has increased drastically, making it possible to employ more and more complex methods of solving both regression and classification problems.

In this project, we will be looking at two different methods of performing regression and classification, and see how they compare against each other. For regression, we can also compare to the results we found in Project 1 [1]. The methods we will be looking at is the stochastic gradient descent and a feed-forward neural network. These methods contain a few hyperparameters that will need to be adjusted in order to find the best model. In order to remove some dependency on the exact training/test data split, we will employ k-fold cross-validation during the hyperparameter search.

The data sets we will be looking at is the Franke function from [2], as well as a reduced MNIST data set [3] consisting of approximately 1800 handwritten digits. First, in Chapter 2 we will briefly introduce the theory of logistic regression, and then go more in depth on stochastic gradient descent and artificial neural networks. A more in-depth description of the data sets can be found in Chapter 3. Then, in Chapter 4 we go through the results of both the Franke function and the MNIST data set, while discussing them. Finally, we conclude our findings in Chapter 5.

2 Theory

2.1 Decision trees

No feature normalization needed

you can fit it extremely well to train data, asking many questions and including all data, but that leads to overfitting, which is why random forests are good

Algorithm 1 Stochastic Gradient descent

```
for i = 0, N_epochs − 1 do
    Shuffle X_train and y_train
    Split into N_mb minibatches
    for j = 0, N_mb do
        Compute gradient using minibatch j
        Update β
    end for
end for
```

3 Data sets

In this project, we will be using two different data sets. The first, is the Franke function we looked at in the previous project [1]. The equation for the Franke function can be found there and in [2]. This function is a two-dimensional function that has been widely used for testing implementations for regression and interpolation.

The second data set is the so-called MNIST data set [3], which is a large data set consisting of handwritten digits from 0 to 9, and is commonly used for testing various types of classification methods. The original data set consists of 70 000 images of size 28×28 pixels. However, as the scope of this project is fairly limited, we will be looking at a reduced version of the data set both in size and resolution in order to have time to produce results. The specific version of the data set we will be using is the one available within the scikit-learn python package, and consists of 1797 images of size 8×8 pixels. Figure **??** shows an example of some of the images in the data set. For the full resolution MNIST data set, human accuracy has been noted to be roughly 0.2%, according to [4]. However, given the reduction in resolution, it is not unlikely that it would be somewhat lower for the data set we will be looking at. **PCA: Since our data set is fairly small, there is no problem fitting the entire data set into memory, meaning that we can easily use the full PCA, and not have to do stuff like minibatch PCA or random PCA, kernel PCA, check lecture notes week 43, slide 73ish**

4 Results and discussion

The code used to generate the results presented in this section can be found in the Github repository [5]. For both data sets, we employ cross-validation when searching for the hyperparameters. The results section will be split into, the first one pertaining to the Franke function, while the second contains the analysis of the MNIST data set

For simplicity, for all the neural network results we will deal with hidden layers consisting of

identical amount of nodes, using the same activation functions for all layers. Strictly speaking there is nothing stopping us from having arbitrary choices of amount of nodes for the layers in the network, and some neural net structures (like for example autoencoders) rely on that. Taking this into account drastically increases the potential things to test, and is thus outside of the scope of this report.

In project 2 [5] we wrote our own code for grid-search, noting that it was something that would have benefited greatly from parallelization, as one of our runs took 15.5 hours to complete. **We use SKLEARNS GridSearchCV in order to test more combinations (maybe try RandomSearchCV?) as it allows for parallelizing the search, allowing us to look at a larger part of the hyperparameter space. In our case, the authors CPU[1] has 12 threads, allowing us to potentially run 12 times as many combinations in the same amount of time.**

SEEMS TO PERFORM BETTER AT PREDICTING CAT SOUNDS THAN DOG WOOFS, WHICH MAY EITHER BE A RESULT OF THE LOW AMOUNT OF SAMPLES, AND THE FACT THAT THE NUMBER OF DOG SAMPLES ARE LOWER. OR POTENTIALLY THE FACT THAT THERE IS MORE VARIATION IN THE ACTUAL SOUNDS THAT DOGS MAKE, AND MAKING JUST A DIRECT FREQUENCY ANALYSIS INSUFICCIENT TO PREDICT REALLY GOOD RESULTS, AND THAT A WAVELET APPROACH COULD POTENTIALLY SOLVE THIS ISSUE.

DATA SET QUALITY IS KINDA SPOTTY THOUGH, WITH SOME SAMPLES CONSISTING OF MANY DOGS, MANY CATS, OTHER NOISE, LOW VOLUME MEOWS, SYNTHETIC MEOW/BARKS

COULD LOOK AT THE MISCLASSIFIED DOG SOUNDS AND SEE THEIR FOURIER SPECTRUM TO SEE IF THERE IS SOME CONSISTENCY.

4.1 Future work

5 Conclusion

In this project, we set out to investigate how stochastic gradient descent and feed-forward neural networks work on both regression and classification problems. We used cross-validation in order to assess which of the hyperparameter combinations gave the best fit.

For the Franke function, we found that both SGD and FFNN performed worse than OLS and Ridge, though the neural net got much closer, with an R^2 score of 0.93, while SGD got 0.84, none of which are bad. For both, we found that the largest number of epochs $N_{\text{epochs}} = 100$ and the largest learning rate $\eta = 0.1$ gave the best results. SGD preferred a batch size of 5, with no regularization for $p = 8$, and $\lambda = 0.001$ at $p = 15$, showing similar behavior with how OLS and Ridge performed for those polynomial degrees. The neural net yielded much better results as the batch size is increased. Based on tests, it seems that using either one of the ReLu family activation functions would have improved the results.

[1] A Ryzen 5 3600X **maybe remove?**

For the classification case, where we studied a reduced version of the MNIST data set, both the SGD and Neural net performed exceedingly well, outperforming the equivalent Scikit-Learn implementation. The SGD implementation gave an accuracy of 96%, which starts to approach what you would expect from human accuracy. SKL was within a percent point away from it. The NN code yielded an accuracy of 95%, but doing so with only 10 epochs and 10 nodes in the singular hidden layer.

References

[1] GitHub repository, Project 1. `https://github.com/simennb/fysstk4155-project1`.

[2] Richard Franke. A critical comparison of some methods for interpolation of scattered data, 1979.

[3] MNIST handwritten digit database, Yann LeCun Corinna Cortes and Chris Burges. `http://yann.lecun.com/exdb/mnist/`.

[4] Dan Ciresan, Ueli Meier, and Juergen Schmidhuber. *Multi-column Deep Neural Networks for Image Classification*, 2012.

[5] GitHub repository, Project 2. `https://github.com/simennb/fysstk4155-project2`.