KATHOLIEKE UNIVERSITEIT LEUVEN

Prof. Dr. ir. Johan A. K. Suykens

Artificial Neural Networks & Deep Learning

Exercise Reports

Marco Bischoff (R1012984)

May 26, 2024

Contents

1 Supervised Learning and Generalization

1.3 In the Wide Jungle of the Training

Task 1.3.1

What is the impact of the noise (parameter `noise` in the notebook) with respect to the optimization process?

The noise parameter controls the deviation of the training data from the true function. Figure 1 shows the impact of noise on the optimization process. For $noise = 0$, the data exactly matches the true function and the model will converge to the true function quickly. For $noise > 0$, the data is perturbed by noise and the model will take longer to converge. For $noise = 1$, the data is completely random and the model will only converge to the mean of the training data without being able to capture the underlying function.

Task 1.3.2

How does (vanilla) gradient descent compare with respect to its stochastic and accelerated versions?

Vanilla gradient descent is the slowest of the three methods. It computes the gradient of the loss function for the entire training set at each iteration. Stochastic gradient descent (SGD) is faster than vanilla gradient descent, because it computes the gradient for a random subset of the training data at each iteration. However it has a higher variance in the loss function. Accelerated gradient descent is the fastest of the three methods. It uses a momentum term to speed up convergence and reduce oscillations in the loss function.

Task 1.3.3

How does the size of the network impact the choice of the optimizer?

For small networks, vanilla gradient descent is sufficient, because the computation of the gradient is not very expensive. For larger networks, SGD is more appropriate due to its lower computational cost. Accelerated gradient descent is the best choice for very large networks, because it converges faster than the other two methods.

Task 1.3.4

Discuss the difference between epochs and time to assess the speed of the algorithms. What can it mean to converge fast?

The model is trained for 2500 epochs. In Figure 2, we can see that SGD with a learning rate of 0.05 and without momentum has an average training time, but very slow convergence. Using a learning rate of 0.1 already converges much faster, but also takes longer to compute. Momentum has a similar convergence rate and is much quicker to compute, but also has high variance. The Adam and the LBFGS optimizers converge the fastest, but LBFGS has the longest computational time. The Adam optimizer is the best choice for this problem, because it converges quickly and has low variance. All optimizers except for vanilla SGD with learning rate 0.05 can be considered to have converged after at most 1000 epochs.

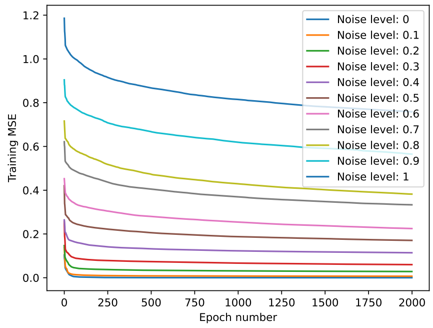


Figure 1: Impact of noise on the optimization process

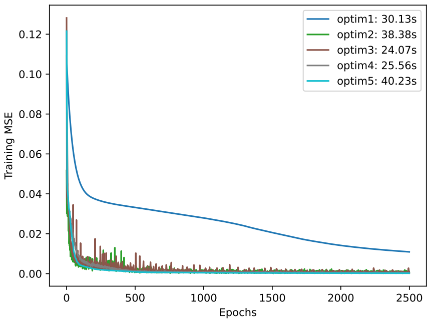


Figure 2: Comparison of optimizers

A bigger model

Task 1.3.5

How many parameters does the model have?

The model has 34826 parameters in total as shown in Table 1.

Layer Type	Shape	# Param
(Input)	(28, 28, 1)	0
Conv2D	(26, 26, 32)	320
MaxPooling2D	(13, 13, 32)	0
Conv2D	(11, 11, 64)	18496
MaxPooling2D	(5, 5, 64)	0
Flatten	(1600,)	0
Dropout	(1600,)	0
Dense	(10,)	16010
Total		**34826**

Table 1: Model parameters

Task 1.3.6

Replace the ADAM optimizer by a SGD one. Can you still achieve excellent performances? Try then the Adadelta optimizer. What is its particularity?

The SGD optimizer has a much slower convergence rate than the Adam optimizer as shown in Figure 3. The Adadelta optimizer achieves very good performance after the first epoch already and has a low variance. The Adam optimizer is in between the two in terms of convergence rate and variance. Compared to the Adam optimizer, the Adadelta optimizer does not require a learning rate to be set, because it uses the gradient and the average of the squared gradient over a window of time steps to adapt the learning rate.

1.4 A Personal Regression Exercise

Task 1.4.1

Define your training and testing dataset using respectively 2000 and 1000 samples drawn independently. Explain the point of having different datasets for training and testing. Plot the surface associated to the training set.

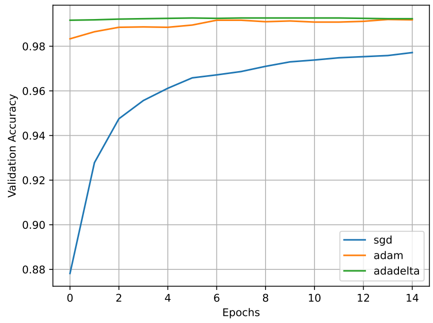


Figure 3: Validation accuracy

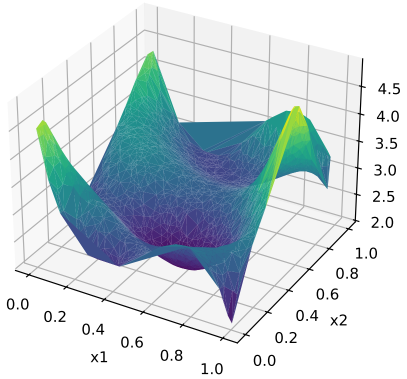


Figure 4: Function surface of the training data

If the same dataset is used for training and testing, the model will overfit the data and not generalize well to unseen data. Moreover, the evaluation of the model will be biased, because the model has already seen the test data during training. If two independent datasets are used, the performance on the test data corresponds more to real-world performance, where the model has not seen the data before. Also, the performance on the test data will be bad, if the model has overfit the training data. The surface associated to the training set is shown in Figure 4.

Task 1.4.2

Build and train your feedforward neural network. To that end, you must perform an adequate model selection on the training set. Investigate carefully the architecture of your model: number of layers, number of neurons, learning algorithm and transfer function. How do you validate your model?

The model architecture is shown in Table 2. The choice of the activation functions seemed to have the biggest impact on the performance. The `mish` activation function performed much better than the other functions I tried. Increasing the number of layers and neurons also improved the performance. However, the model was overfitting the training data when using too many neurons. The choice of the optimizer and the learning rate did not have a big impact on the performance. Here, I used the Adam optimizer with a learning rate of 0.05. All choices were validated by calculating the mean squared error on the test data.

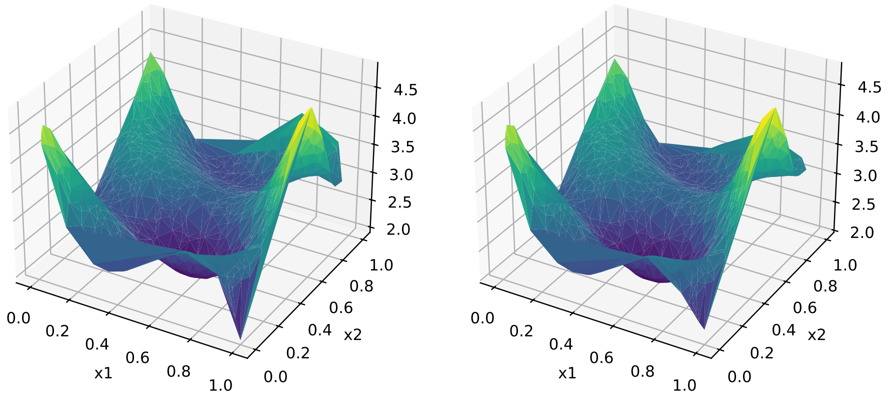
Layer Type	**Shape**	**Activation Function**	**# Param**
Dense	(16)	mish	48
Dense	(16)	mish	272
Dense	(16)	tanh	272
Dense	(1)	(None)	17
Total			**609**

Table 2: Regression model parameters

Task 1.4.3

Evaluate the performance of your selected network on the test set. Plot the surface of the test set and the approximation given by the network. Explain why you cannot train further. Give the final MSE on the test set.

The surfaces of the test data and the predicted data are shown in Figure 5. The model has already converged and further training would not improve the performance. The final mean squared error on the test data is 0.0017132.



(a) Test data

(b) Predicted

Figure 5: Function surface

Task 1.4.4

Describe the regularization strategy that you used to avoid overfitting. What other strategy can you think of?

I used early stopping to avoid overfitting. The training was stopped when the validation loss did not improve for 10 epochs. Another strategy to avoid overfitting is to use dropout layers, which sets a few neurons to zero during training.

[1]

References

[1] Adam Ries. *Rechenung auff der Linihen und Federn*. Hans Schönsperger, Annaberg, 1522.

A Benutzerdokumentation

B Introduction