ETP_Bachelor_2019-17

Karlsruher Institut für Technologie

Illustration of the Neural Network Learning Process during Training

Zur Erlangung des akademischen Grades eines

Bachelor of Science

von der Fakultät für Physik des
Karlsruher Instituts für Technologie (KIT)
genehmigte

Bachelorarbeit

von

Greta Heine

Tag der Abgabe: 04. 11. 2019

Referent: Dr. Roger Wolf
Korreferent: Prof. Dr. Günter Quast

Abstract

This thesis enhances the understanding of Neural Network (NN) trainings by investigation of the learning process especially focusing on the dependence of the NN output on the input space for given tasks. For this purpose, the NN function is decomposed into a Taylor expansion. The Taylor coefficients serve as a metric to illustrate the influence of input space features on the output at each step of the training. Both, the arithmetic mean values of the Taylor coefficients and their dependence on each point of the input space are investigated, giving new insights into the decision taking of NNs.

Abstract

Diese Arbeit verbessert das Verständnis der Trainingsprozesse von neuronalen Netzen, indem der Lernprozess untersucht wird. Hierbei liegt der Fokus auf der Abhängigkeit der Ausgabe neuronaler Netze vom Eingangsdatensatz für verschiedene Aufgabestellungen. Für diesen Zweck wird die Funktion des neuronalen Netzes in eine Taylorreihe entwickelt. Die Taylorkoeffizienten dienen zur Illustration des Einflusses von Eigenschaften des Eingangsdatensatzes auf die Ausgabe des Netzes zu jedem Zeitpunkt des Trainings. Sowohl die Mittelwerte der Taylorkoeffizienten, als auch deren Abhängigkeit von jedem Punkt des Eingangsdatensatzes werden untersucht. Dies liefert neue Einblicke in die Entscheidungsfindung von neuronalen Netzen.

Erklärung der selbständigen Anfertigung der Bachelorarbeit

Hiermit erkläre ich, dass ich die Bachelorarbeit mit dem Titel

»*Illustration of the Neural Network Learning Process during Training*«

selbständig und unter ausschließlicher Verwendung der angegebenen Hilfsmittel angefertigt habe.

Greta Heine
Karlsruhe, den 04. November 2019

Contents

1 Introduction **5**

2 Introduction to Artificial Neural Networks **7**
2.1 Model Architecture and Training of a NN 7
2.2 Model Architecture 8
2.2.1 Input Data 8
2.2.2 Neurons and Layers 9
2.3 Training 10
2.3.1 Loss Function 10
2.3.2 Optimizer 11
2.4 Building a Network 11
2.4.1 Keras and Tensorflow 11
2.4.2 Initialization and training procedure of NN 11
2.5 Taylor coefficients of the NN function 13

3 Analysis of the NN Learning Process **15**
3.1 Creating Test Samples 15
3.2 Analysis of mean Taylor Coefficients $\langle t_i \rangle$ 17
3.3 Analysis of $\langle t_i \rangle$ for different training seeds 20
3.4 Analysis on Input Space Dependence of Taylor Coefficients t_i 23
3.4.1 Input space Dependency: example 1 and 2 23
3.4.2 Input space Dependency: example 3 27
3.4.3 Input space Dependency: example 4, 5 and 6 29

4 Summary, Conclusion and Outlook **31**

A Appendix **33**
A.1 Input space Dependency: example 4, 5, 6 33

B Bibliography **37**

CHAPTER 1

Introduction

Artificial Neural Networks (NNs) as a part of machine learning methods are an increasingly important subject of current research and are becoming a key instrument for practical applications like automation tasks. Today, intelligent software enables near-human-level image classification, speech and handwriting recognition, autonomous driving, automated diagnoses in medicine and much more. Also, in high-energy particle physics, NNs have received considerable attention as they can be applied to distinguish a signal from one or more backgrounds and identify most influential input features. An advantage of the application of NNs on the discrimination of signal from background is that a NN can be sensitive to marginal distributions of the input variables but also to correlations across input variables.
However, a major problem with NNs is the so-called black box phenomenon: a NN can approximate any function but gives no insights on how the output is derived or which input characteristics are relevant. In general, NNs are non-identifiable models: two different NNs can generate the same results but with different intrinsic parameters. Although extensive research has been carried out on NNs, few authors have been able to draw on any systematic research into decision taking of NNs.
The main goal of this thesis is to develop a deeper understanding of NN learning processes by illustration of trainings and identification of the significance of single input characeristics on the NN decision taking. For this purpose, the NN model function is decomposed into a Taylor expansion for several toy datasets. The corresponding Taylor coefficients are associated with n-dimensional features of the input space variables and with correlations across these variables and can be used as metric that allows for the identification of the importance of the associated features for the decision taking of a NN.
This thesis begins by laying out the theoretical foundations of this research by describing the NN model architecture, the implementation of the NN model and how the Taylor coefficients are obtained. The following chapter covers the methods that are used in this thesis and presents the findings and corresponding analysis of these findings, focusing on the Taylor coefficients that are used to understand the input space influence on the NN output. Finally, the conclusion gives a brief summary of the findings and includes a short outlook on recommendations on potential future projects in this area.

CHAPTER 2

Introduction to Artificial Neural Networks

NNs are computing systems inspired by biological neural networks. During the training of NNs on given data samples, specific characteristics are identified and learned in order to execute dataprocessing tasks like classification to predefined classes. In this context, learning describes the automated search process for better performance on a desired task and the adjustment steps, made for this purpose. In general, NNs are stochastic minimization processes applied to a high-dimensional input space.
The following chapter is mainly based on information given by [1] and [2]. In this chapter the general structure and application of Neural Networks are described in order to understand the methods used in chapter 3. For this purpose, especially the model configuration used in this thesis is described in more detail. For comparibility reasons the configuration is chosen to be similar to the one used by [3].
The first section presents the general architecture of NNs. The second section describes the implementation and training of the used NN model. The final section is concerned with Taylor coefficients obtained from the NN model function gradients which are later on used to analyse the NN learning progress.

2.1 Model Architecture and Training of a NN

As shown in Fig.2.1, a NN consists of several mathematical building blocks, called modules. The core of the model, which is built by a combination of an arbitrary number of *layers* is run on the *input data* x to obtain the corresponding *predictions* y. Each layer is characterized by internal trainable parameters: *weights and biases*. Besides the trainable parameters weights and biases there are parameters that have to be specified before training and which determine the network structure and how the network is trained, so-called hyperparameters.
The iterative optimization of the trainable parameters for a given task is called training. For this purpose, a *loss function* is constructed to compare the predictions y with the true *targets* y_{true} corresponding to the input data x. The mismatch between y and y_{true} is quantified by the loss score obtained from the evaluation of the loss function on y_{true} and y. Finally, an optimizing algorithm (*optimizer*) using this loss score is run in order to update the weights and biases of all layers with the attempt to reduce the loss in the

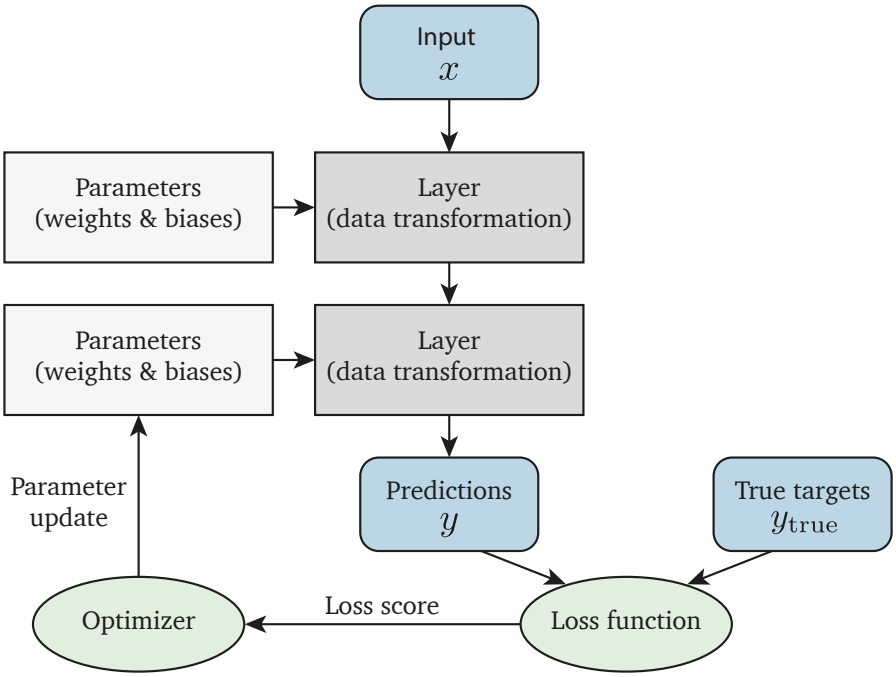

Figure 2.1: Schematic neural network model architecture (grey) and its training steps (green)s.

next iteration. This is called back-propagation. After completing a single training loop, the described process is repeated as long as specified by the training initialization. After a sufficiently large number of training loops, the loss score should be converged to a small value. At this point, the NN has learned the features of the dataset. The following sections will present the NN modules and the NN training more precisely.

2.2 Model Architecture

2.2.1 Input Data

Before NNs can be trained, an *input dataset* containing a collection of many training samples and corresponding targets e.g. labels is required. A sample, also called data point, is a collection of features that the NN has to process. In general, all NN systems use tensors as basic data structures. Four main types of data formats are common: 2D, 3D, 4D, and 5D tensors. The least complex format is the two-dimensional vector data format containing samples and corresponding features. Adding timesteps as an additional dimension, generates time-series data (3D). Another common format is the four-dimensional image format (samples, height, width, color channels) and based on this, the five-dimensional video format.
For simplicity, in this work, only two-dimensional sample vectors $\mathbf{x} \in \mathbb{R}^2$ are used for NN training and testing where each entry x_i is a feature of the sample. Later on, the used datasets are split into two equal but independent halves. The first half is used as

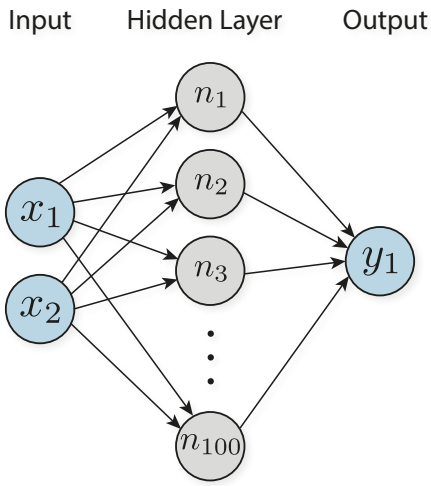

Figure 2.2: Sketch of neural network layer architecture of a two-layer model with two-dimensional input x_i, 100 neurons n in the hidden layer and a one-dimensional output y_i.

training dataset in order to train the NN and to validate the predictions. The second half is used to test the trained model on unknown data.

2.2.2 Neurons and Layers

The core of a typical NN model consists of several data-processing modules, so-called *layers* which perform transformations on n-dimensional data. A model is built by chaining several layers: one input layer, an arbitrary number of hidden layers and one output layer. The input layer receives the inputs and propagates these to the subsequent hidden layer. The calculations and outputs of hidden layers are not visible from outside of the NN. For most problems one hidden layer is sufficient but the more hidden layers are used in a NN the more complex problems can be solved. Finally, an output layer is used to calculate the final results. Its dimension is equivalent to the dimension of the targets associated to the given data, where the targets contain information about the desired output for a given input. In this case, the targets are 1-dimensional and have the value 0 or 1. The layer structure used in this work is shown in Fig.2.2. There is only one hidden layer and the information moves directly from the input layer via the hidden layer to the output layer, where only fully connected layers (also called dense layers) are used. This simple kind of network is called fully connected feed-forward network or multilayer perceptron. However, there are many other different types of networks like Recurrent Neural Networks, Markov Chains, Deep Convolutional Networks or Generative Adversarial Networks.
The layers of a NN model are composed of multiple nodes represented by mathematical functions: *neurons*. As shown in Fig.2.3, the output n of a single neuron k for a total

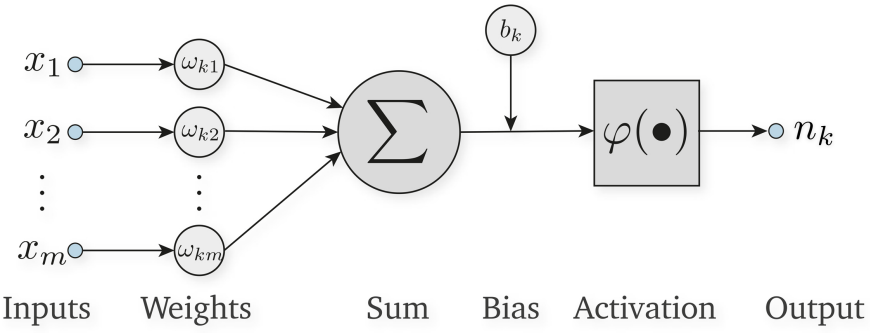

Figure 2.3: Graphic representation of an artificial neuron function as described by Eq. 2.1.

number N of neurons per layer is

$$n_k(v_k) = \varphi(v_k) = \varphi\left(\sum_{j=1}^{m} w_{kj}x_j + b_k\right) \qquad k \in \{1,\dots,n\},\ j \in \{1,\dots,m\} \tag{2.1}$$

where v_k is the weighted sum of the inputs x_j plus bias b_k with an m-dimensional input vector $x = (x_0,\dots,x_m)$. Since in this work, a two-dimensional input vector x is used, m equals 2. The corresponding weights w_{kj} with $k \in \{1,\dots,n\},\ j \in \{1,\dots,m\}$ represent the relative importance of each connection between the neurons and parameterize the data transformation implemented by the layers. The bias $b = \{b_0,\dots,b_m\}$ is added to additionally shift the weighted inputs.
Finally, the output of a neuron is characterized by the activation function φ. There are several different function types used as activation functions (also called transfer functions). Commonly used functions are step, sigmoid (logistic), ReLu (Rectified Linear Unit) or hyperbolic tangent function. In this thesis, the hypberbolic tangent and the sigmoid function

$$n_k(v_k) = \tanh(v_k) \quad \text{and} \quad n_k(v_k) = (1 + e^{-v_k})^{-1} \tag{2.2}$$

are used. As before, n_k is the output of the k-th neuron and v_k the weighted sum of the inputs x_j plus bias b_k as in equation 2.1.

2.3 Training

Learning takes place by finding a set of parameters w and b for all layers in order to map an example input correctly to associated targets. The iterative optimization of the weights and biases is achieved by the loss function and the optimizer algorithm.

2.3.1 Loss Function

Following the information flow through the network, the *loss function* is used to compare the generated predictions y of the trained network with the true targets y_{true}. It represents

a measure of success and provides the loss score that will be minimized during training. Depending on the given task, an optimal loss function should be chosen: For example, categorical cross-entropy is suitable for classification problems and mean-squared error for regression problems. In this thesis, binary cross-entropy [4] as a special two-dimensional case of categorical cross-entropy is used as given by

$$H(x) = -\frac{1}{N}\sum_{i=1}^{N} y_{\text{true},i} \cdot \log(y(x_i)) + (1 - y_{\text{true},i}) \cdot \log(1 - y(x_i)) \tag{2.3}$$

where N is the number of samples of the input dataset, $y_{\text{true},i}$ are the targets associated to the samples x_i and y the associated predictions computed by the NN. The quantity $H(x)$ has to be minimized to obtain optimal predictions.

2.3.2 Optimizer

The minimization of the loss score throughout the training is performed by the *optimizer*. Initially, the parameters w and b are assigned to random values. The optimizer updates the weights and biases of the neurons after each training step in order to reduce the value of the loss score by a small amount. This minimization can be achieved by using the *gradient* of the NN function, which points to the minimum of the loss function dependent on all trainable parameters. Since the loss value is a function of the parameters w, b and the input x, the model is optimized by altering the weights and biases in small steps with opposite sign of the gradient based on the current loss value. This method is called *gradient descent* [5]. In this thesis, the gradient is applied by the complex optimization algorithm adaptive momentum estimation (Adam) optimizer [6], which is a popular deep-learning gradient-based optimizer.

2.4 Building a Network

2.4.1 Keras and Tensorflow

In this thesis, TensorFlow [7] and Keras [8] are used for the implementation of the NN. TensorFlow is a commonly used open-source library to develop and train machine learning models in Python [9]. Keras is also an open-source model-level library written in Python. It uses Tensorflow but also any other tensor library as backend. In this case, Tensorflow is needed for construction of the model structure and Keras for model building.

2.4.2 Initialization and training procedure of NN

For building of a NN model, the model itself has to be implemented by defining the single layers as shown in Listing 2.1.

```
1 from keras.models import Sequential
2 from keras.layers import Dense
3
4 model = Sequential()
```

```
model.add(Dense(100, activation= 'tanh', input_dim=2))
model.add(Dense(1, activation='sigmoid'))
```

Listing 2.1: Two-layer NN model defined using the `Sequential` class with a hidden layer activated by a hyperbolic tangent containing 100 nodes and a one-dimensional output activated by a sigmoid function.

A two-layer model is defined using a sequence of two layers from the `Sequential` class. The input is not specified as layer in this example but it is characterized as two-dimensional in the attributes of the first layer. This is the only hidden layer consisting of 100 neurons with a hyperbolic tangent as activation function. It is implemented with the Keras built-in `Dense` class. The output layer including 1 neuron is activated by a sigmoid function and gives an output between 0 and 1. Here, 1 corresponds to a classification of a given data point to signal class and 0 refers to the background class. An output of 0.5 indicates that the classification whether the given sample point is from the background or signal class is not possible. All values in between can be interpreted in terms of classification certainty.
Once the model architecture is defined, its compilation parameters have to be configured. This is shown in Listing 2.2.

```
model.compile(optimizer='adam', loss = 'binary-crossentropy', metrics =
    ['acc'])
```

Listing 2.2: Network compilation using Adam optimizer and binary-cross entropy as loss function. The classification accuracy is used as metric to display the classification success.

Here, the optimizer Adam [6] and the binary cross-entropy as loss function are used. In this case, the prediction accuracy (`'acc'`) which displays the proportion of correctly classified output is the metric to monitor the training performance.
Finally, after defining the network it has to be trained. For this purpose, the `model.fit`-method is used as can be seen in Listing 2.3.

```
early_stop = keras.callbacks.EarlyStopping(patience=30)
history = model.fit(input_dataset, input_labels, batch_size = input_shape
    , validation_split=0.2, epochs = 2000, shuffle=True, callbacks=
    early_stop)
```

Listing 2.3: Network training for 2000 epochs using an early stopping condition with a patience of 30 gradient steps, a batch size equivalent to the number of samples and a validation rate of 20 percent. The samples are shuffled after every epoch to prevent overfitting.

Also in this case, several parameters can be specified. For example, the input-data, batch size, number of training loops (*epochs*) or the ratio of input-data that is not used for training but for validation of the trained network can be defined.
Like the number of epochs, the batch size is a hyperparameter that has to be set before training. The batch size defines the number of samples that are worked through before the the weights and biases are updated. In contrast to the number of epochs that can be arbitrarily high, possible batch sizes range from 1 to the total number of samples. The training dataset can be separated according to three main kinds of batch sizes [10]: (1) all training samples are assigned to a single batch. So the batch size equals the size of the training dataset which is called batch gradient descent. (2) The batch size equals the size

of a single entry in the dataset (sample point), which is called stochastic gradient descent. (3) a batch size in between one and the total training data size is called mini-batch gradient descent. Commonly used batch sizes are equal to 32, 64 or 128. However, in this thesis, the batch size is equal to the number of samples within the entire dataset. Despite the advantages of small batch sizes (faster training, less memory usage), a large batch size generates a more accurate estimate of the gradient. This approach is adequate for this thesis which is focused on accuracy. Nevertheless, it might be interesting to investigate the behavior of the NN with smaller batch sizes.
The number of epochs which determines the number of complete passes through the whole dataset can be set to an arbitrary value. In this case, each epoch corresponds to exactly one gradient step due to the choice of the batch size being equal to the number of samples. In this thesis, a training epoch size of 2000 is used. After that period, the training is mainly completed for all given example tasks which can be seen by converged Taylor coefficient values. In addition, the value of the `early-stop`-function (`patience`) is set to 30 epochs. Early stopping terminates the training before the last training epoch is reached if the accuracy value on the validation dataset has not improved over the last 30 epochs.
Twenty percent of the training dataset are chosen randomly in order to validate the NN model. The validation loss is a measure of training performance: the smaller the validation loss value, the closer the predictions of the NN are to the targets. Finally, after completion of a whole training epoch, the dataset is shuffled to reduce the risk of overfitting on the given dataset. Overfitting and underfitting are main problems that can occur during NN trainings. Overfitting occurs when the model has memorized the training dataset to well leading to a bad performance on unknown data. Underfitting occurs when the model has not trained long enough on the training dataset, hence, has not learned all features of the input leading, also, to poor predictions on new data.
Now, the NN is fully defined and can be applied to any two-dimensional input datasets.

2.5 Taylor coefficients of the NN function

In order to obtain the Taylor coefficients of the NN model functions, the gradients which are the multidimensional derivatives of the NN functions have to be calculated. In order to generate the gradients of a NN function, for technical reasons, the function must at first be converted from Keras to Tensorflow. Tensorflow provides the function `tf.gradients`, that determines the gradients of a given network function. These multidimensional derivatives can then be used to obtain the Taylor coefficients of the NN model functions. Here, the Taylor series expansion T [11] of a two-dimensional function $y(x)$ that is infinitely differentiable at the point a is given where $y^n(a)$ is the n-th derivative of $y(x)$ at expansion point a:

$$T(x_1, x_2) = \sum_{n_1=0}^{\infty} \sum_{n_2=0}^{\infty} \frac{\partial^{n_1+n_2} y(a_1, a_2)}{\partial x_1^{n_1} \partial x_2^{n_2}} \frac{(x_1 - a_1)^{n_1} \cdot (x_2 - a_2)^{n_2}}{n_1! n_2!}. \tag{2.4}$$

For practical reasons, in this case, only the first two orders are considered as shown in Eq.2.5. This equation shows the first and second order of a two-dimensional Taylor expansion with a two-dimensional input vector x and an expansion point a. Here, the function y is a composition of the functions of each model layer: $y(x) = y_2(y_1(x))$ where y_2 and y_1 are defined by Eq.2.1. In the following chapter, the expansion points a will equal the given samples i.e. datapoints.

$$\begin{aligned} Tf(x;a) =& y(a) + \frac{\partial y(a)}{\partial x_1}(x_1 - a_1) + \frac{\partial y(a)}{\partial x_2}(x_2 - a_2) \\ &+ \frac{1}{2}\frac{\partial^2 y(a)}{\partial x_1 \partial x_1}(x_1 - a_1)(x_1 - a_1) + \frac{1}{2}\frac{\partial^2 y(a)}{\partial x_1 \partial x_2}(x_1 - a_1)(x_2 - a_2) \\ &+ \frac{1}{2}\frac{\partial^2 y(a)}{\partial x_2 \partial x_1}(x_2 - a_2)(x_1 - a_1) + \frac{1}{2}\frac{\partial^2 y(a)}{\partial x_2 \partial x_2}(x_2 - a_2)(x_2 - a_2) \qquad (2.5) \\ =& t_0 + t_{x_1}(x_1 - a_1) + t_{x_2}(x_2 - a_2) + t_{x_1 x_1}(x_1 - a_1)^2 \\ &+ t_{x_1 x_2}(x_1 - a_1)(x_2 - a_2) + t_{x_2 x_2}(x_2 - a_2)^2 \qquad (2.6) \end{aligned}$$

Comparison of the coefficients leads to the following set of equations

$$\begin{aligned} t_0 = y(a), \quad t_{x_1} = \frac{\partial y(a)}{\partial x_1}, \quad t_{x_2} = \frac{\partial y(a)}{\partial x_2}, \quad t_{x_1 x_1} = \frac{1}{2}\frac{\partial^2 y(a)}{\partial x_1 \partial x_1} \\ t_{x_1 x_2} = \frac{1}{2}\left(\frac{\partial^2 y(a)}{\partial x_1 \partial x_2} + \frac{\partial^2 y(a)}{\partial x_2 \partial x_1}\right) = \frac{\partial^2 y(a)}{\partial x_1 \partial x_2}, \quad t_{x_2 x_2} = \frac{1}{2}\frac{\partial^2 y(a)}{\partial x_2 \partial x_2} \end{aligned} \qquad (2.7)$$

These Taylor coefficients are used to analyze the NN response sensitivity to the input space. As proposed in [3], the features t_i are interpreted as characteristics of single elements or pair-wise relations between two input space elements and can be used to quantify the influence of the corresponding features on the NN output. The first-order features corresponding to first-order expansion coefficients display the influence of single input elements on the NN output. Whereas, second-order features capture the influence of pair-wise and self-correlations across the input space variables. In the next chapter, Taylor coefficients are determined and studied in order to understand sensitivity of the NN to certain input features.

CHAPTER 3

Analysis of the NN Learning Process

This chapter is divided into four sections. The first section describes the datasets that are used throughout this thesis and the creation of these test samples. The second section deals with the analysis of the arithmetic mean Taylor coefficients $\langle t_i \rangle$ of the NN functions according to the datasets defined in the first section. The third section presents the analysis method from section two but applied to 100 NN functions, each of them trained with a different seed in order to consolidate the findings of section two. The final section analyses the Taylor coefficients with respect to the input space features providing a deeper insight into the NN training.

3.1 Creating Test Samples

In order to analyze and understand the training process of a NN learning, fundamental feature characteristics of inputs, toy datasets are produced. As this work focuses on fundamental neural network characteristics, simple classification tasks are chosen. Gaussian distributed signal and background classes with two input variables x_1 and x_2 for different parameter sets as seen in Tab. 3.1 are used for binary classification. The resulting distributions are shown in Fig.3.1. The signal class is represented by red contours, the background class by blue contours where darker colors indicate a higher sample density. For each of the six distributions, 400000 pseudo data points are generated: 200000 each for the signal and the background class.

The first example distribution at the top left of Fig.3.1 contains two Gaussian distributions with different mean values but equal spread. The second example differs from example 1 only in switching positions of the signal and the background class centers. For the third example, both signal and background class share the same mean value but are distributed asymmetrically with a shift in orthogonal directions relative to each other. The signal class is stretched along the diagonal axis, the background class along the off-diagonal axis. This is achieved by introducing off-diagonal elements into the covariance matrix. Thus, signal and background have different correlations between x_1 and x_2. The fourth example combines the first and third example by two distributions that do not share the same center and are stretched orthogonal relative to each other. The fifth example shows two distributions that share the same center but have different

Table 3.1: Mean values and covariance matrices used to initialize the Gaussian distributed datasets for the signal and background classes that can be seen in figure 3.1.

Example	Mean value		Covariance matrix	
	Signal (x_1, x_2)	Background (x_1, x_2)	Signal	Background
1)	(0.5, 0.5)	(-0.5, -0.5)	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
2)	(-0.5, -0.5)	(0.5, 0.5)	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
3)	(0, 0)	(0, 0)	$\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix}$
4)	(0.5, 0.5)	(-0.5, -0.5)	$\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix}$
5)	(0, 0)	(0, 0)	$\begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}$	$\begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix}$
6)	(0.5, 0.5)	(-0.5, -0.5)	$\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$	$\begin{pmatrix} 3 & -1.5 \\ -1.5 & 3 \end{pmatrix}$

spreads. Finally, the last example distribution implements a combination of the last two introduced distributions which is the most difficult and realistic example in this thesis. In summary, the differences of the following key characteristics between signal and background are tested: position of the center, spread and correlation.
For the training of the NN, the samples are split into two equal halves. The first half (training dataset) is used to train and validate the NN, the second independent half (test dataset) is used to calculate the taylor coefficients t_i that are later on used to interpret the NN training process. Having generated the toy datasets, a NN as described in the previous chapter is initialized and trained on these datasets. The next section of this thesis will discuss the analysis of the trained NN models.

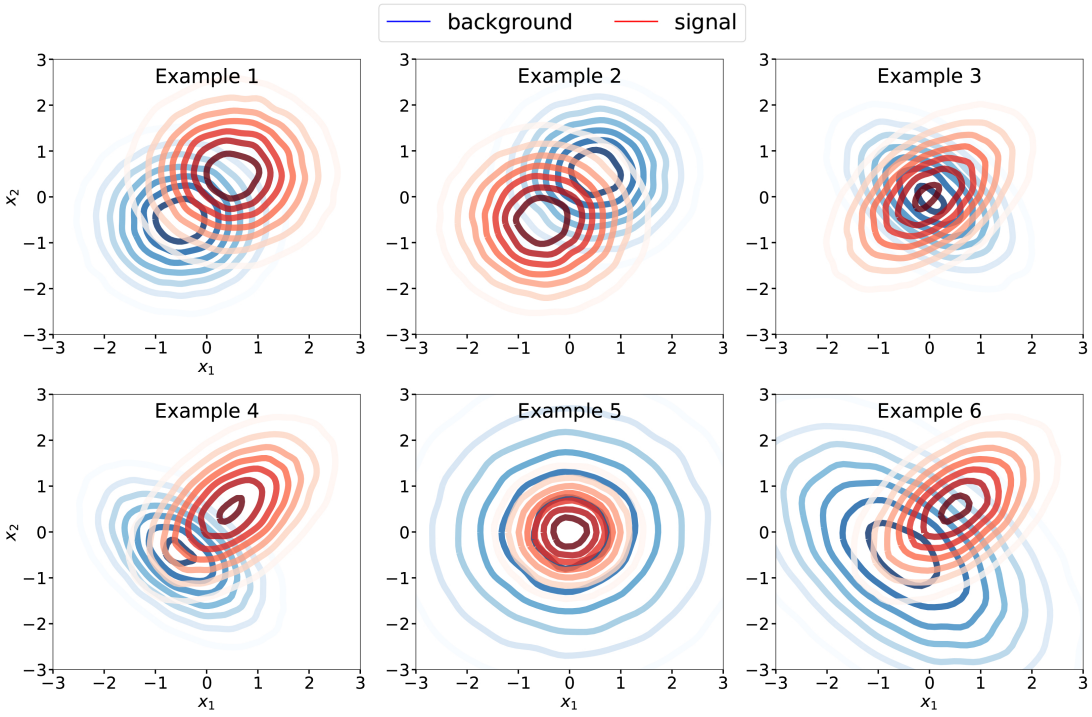

Figure 3.1: Contours of the distributions used as NN example training tasks as discussed in section 3.1 and specified in Tab. 3.1. The red contours represent the signal class, blue contours represent the background class.

3.2 Analysis of mean Taylor Coefficients $\langle t_i \rangle$

Now, after the NN is trained, the Taylor coefficients t_i can be obtained. The first and second-order gradients are used to determine the first and second order Taylor coefficients t_i as described in section 2.5. In this case, the so far unused test dataset is used to calculate the gradients. Besides, saving all $t_i(x_1, x_2)$ for later analysis in 3.4, the arithmetic means of the absolute values of the Taylor coefficients

$$\langle t_i \rangle = \frac{1}{N} \sum_{k=1}^{N} |t_i(\{x_j\})|_k| \qquad i, j, k \in \mathbb{N}, \tag{3.1}$$

are calculated to serve as a general metric for the sensitivity of the NN on the input space after training as proposed by [3]. Here, $\{x_j\}$ are the input space variables x_1, x_2 and N is the number of samples in the test dataset. The $\langle t_i \rangle$ are calculated for each training epoch. The average of the absolut values gives a better measure of the influence of inputs than the average of the signed values as the average of the signed values can produce averages near to zero despite both large positive and large negative values. The first order coefficients $\langle t_{x_1} \rangle$ and $\langle t_{x_2} \rangle$ display the influence of marginal distributions of single input space variables on the NN output and the second-order coefficients are related to pairwise and self-correlations between x_1 and x_2, where, high values indicate a high

influence on decision finding of the NN. The resulting $\langle t_i \rangle$ for the examples given in Eq. 3.1 using a maximum of 2000 gradient steps are shown in Fig.3.2. In addition, the loss score of the validation dataset is displayed in order to give information on the training performance.

The resulting Taylor coefficients and the loss function of the first two classification tasks are similar since the underlying datasets are equal except for exchanging mean values of signal and background class and are, therefore, discussed at once. Compared to the other tasks the training is completed fast after about 160 training epochs. Here, the early stopping condition leads to an end of training as can be seen by the constant validation loss at the end of training. In this case, the validation loss has converged to the value of about 0.5.

The first order features, as well as the self-correlation features converge to pairwise equal values due to the symmetry of the given example dataset. Here, $\langle t_{x_1} \rangle$ and $\langle t_{x_2} \rangle$ reach an end value of 0.16, whereas $\langle t_{x_1 x_2} \rangle$ reaches an end value of 0.07 and the self-correlations $\langle t_{x_1 x_1} \rangle$ and $\langle t_{x_2 x_2} \rangle$ reach only 0.04 in the end. Large values of $\langle t_{x_1} \rangle$ and $\langle t_{x_2} \rangle$ at the end of the training process can be interpreted in the following way: First-order features are more important for the NN decision finding compared to second-order features. The explanation of giving the first-order features a much higher influence than the correlations between input space variables might be that the classification of samples at the margins is relatively clear. For example, a point at the upper right corner of the plot is most likely from the signal class, a point at the lower-left corner from the background class. Large $\langle t_{x1} \rangle$ and $\langle t_{x2} \rangle$ indicate, therefore, a good spacial separation of signal and background which is of more importance than the correlation between the input variables.

The small differences between the plots of example 1 and 2 except for the switched colors can be explained by fluctuations at the beginning of the trainings as the NNs are initialized randomly.

Looking at example 3, a fluctuation of the $\langle t_{x_i} \rangle$ can be seen at the beginning. This suggests a high NN uncertainty resulting in various adjustments on the model weights and biases. Then, after about 50 gradient steps, all values begin to increase intensively until step 300. After vast improvements in the early training steps from this point on the validation loss does barely improve. After 300 gradient steps, the highest value (0.21) is reached by the correlation feature $\langle t_{x_1 x_2} \rangle$ which still increases after this point to an end value of 0.22. Thus, the difference between signal and background in correlation is picked up by the NN as most important feature. The first order features approach a value below $\langle t_{x_1 x_2} \rangle$: $\langle t_{x_2} \rangle$ has converged immediately at about 0.17 and $\langle t_{x_1} \rangle$ decreases slightly from epoch 300 on until it converges to a similar value as $\langle t_{x_2} \rangle$. The least important features, in this case, are the self-correlations. Also, here $\langle t_{x_1 x_1} \rangle$ has approached its value immediately and $\langle t_{x_2 x_2} \rangle$ decreases until both values converge to 0.05. On closer inspection, the self-correlation values decrease slightly until the end. But in combination with the increase of $\langle t_{x_1 x_2} \rangle$ the validation loss stays constant even though the metric values change. In contrast to the first two examples, the signal and background class have different covariance matrices leading to different correlations between x_1 and x_2. This characteristic is identified by the NN as the most influencial feature of the input space. Also the first order features are taken into account as separation by the marginal distributions is a promising approach.

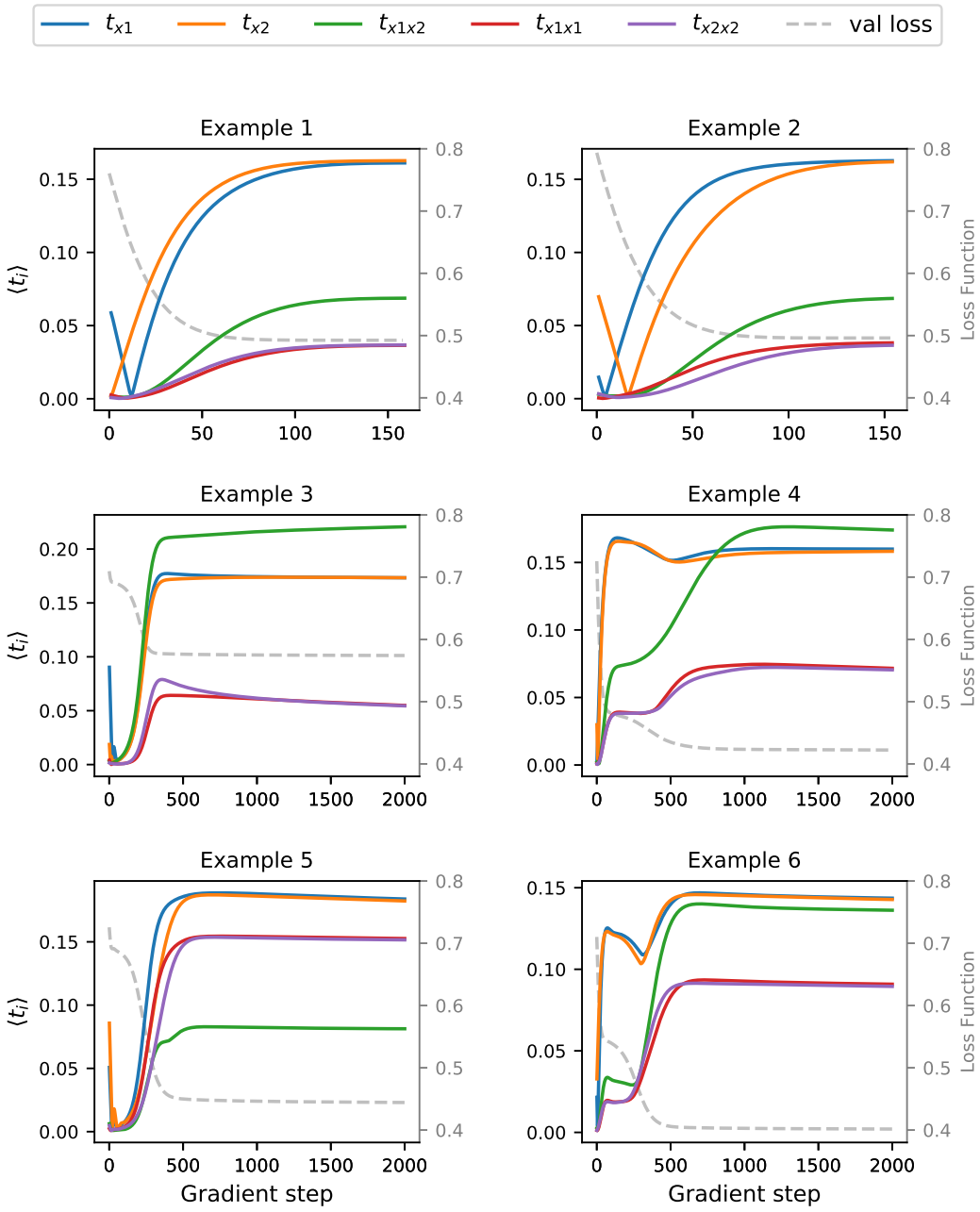

Figure 3.2: Values of the metrics $\langle t_i \rangle$ defined by equation 3.1 over the training process with a maximum of 2000 training steps (gradient steps) and behaviour of the validation loss showing the performance success of the NN.

The fourth example of Fig.3.2 is the most interesting case so far. At the beginning, all metrics start to increase steeply until they reach local maxima after about 100 training epochs. Again, the first order features gain the highest influence followed by $\langle t_{x_1x_2} \rangle$ and lastly the self-correlations. Like in the first example, the NN focuses on the spacial differences. To be more precise, example 4 and 1 are numerically identical for the first 100 epochs. But after this point, the first-order features decrease while the second-order features stay nearly constant. About 150 steps later there is a steep rise in $\langle t_{x_1x_2} \rangle$ that does not stop until the thousandth training step. The value gets even higher than the values of the first-order features. The simultaneous rise in $\langle t_{x_1x_1} \rangle$ and $\langle t_{x_2x_2} \rangle$ stops a lot earlier. The training is mainly completed after 1000 epochs. The observations for this example can be interpreted as follows: up to the 100th epoch the classification made by the NN is made without taking account of the difference in spreads of signal and background. The distributions are interpreted like in example 1. After about 100 gradient steps, also the difference in the covariances of signal and background class is recognized. The influence of the additionally learned feature leads to a result more similar to the third example task. Moreover,an additional performance gain corresponding to a decreased validation loss can be observed.
The training on example 5 does not lead to new insights. All values rise until they reach stable values at 500 gradient steps. The value of $\langle t_{x_1x_2} \rangle$ is small as the distributions are symmetrically in x_1 and x_2 and, thus, the correlation between x_1 and x_2 has no great importance. Compared to the examples before the values of $\langle t_{x_1x_1} \rangle$ and $\langle t_{x_2x_2} \rangle$ are high but still smaller than $\langle t_{x_1} \rangle$ and $\langle t_{x_2} \rangle$. Again, the marginal distributions are the most important feature of the input space in order to classify the dataset. Looking at the plot, this is reasonable as in regions near the origin the propability for samples from the signal class is much higher than for background class and vice versa in outer regions. But also the self-correlation features are a characteristic that is used to distinguish the signal from the background class as the spreads are different.
Lastly, in the sixth example the combination of example 4 and 5 is shown in Fig.3.2. As expected, the resulting Taylor coefficient progressions look similiar to the ones of example 4. The most important difference is that $\langle t_{x_1x_2} \rangle$ does not reach a higher value than the first order features $\langle t_{x_1} \rangle$ and $\langle t_{x_2} \rangle$.
In this section, the findings of [3] could be reproduced for example 4. Furthermore, with this method, five new examples (1,2,3,5,6) could be analyzed.

3.3 Analysis of $\langle t_i \rangle$ for different training seeds

In the last section, the metrics $\langle t_i \rangle$ have been studied as functions of the training epochs. But each training has only been performed once. So it is not obvious wether the NN decides similarily if the calculation is repeated several times. In order to consolidate the results of the previous section and quantify reliability of the converged NN models, the metrics $\langle t_i \rangle$ are evaluated for 100 NN models trained with different seeds. Each time the seed used to initialize the random seed function is increased by 1. This leads to different network initializations for each training and simultaneously enables the possibility of

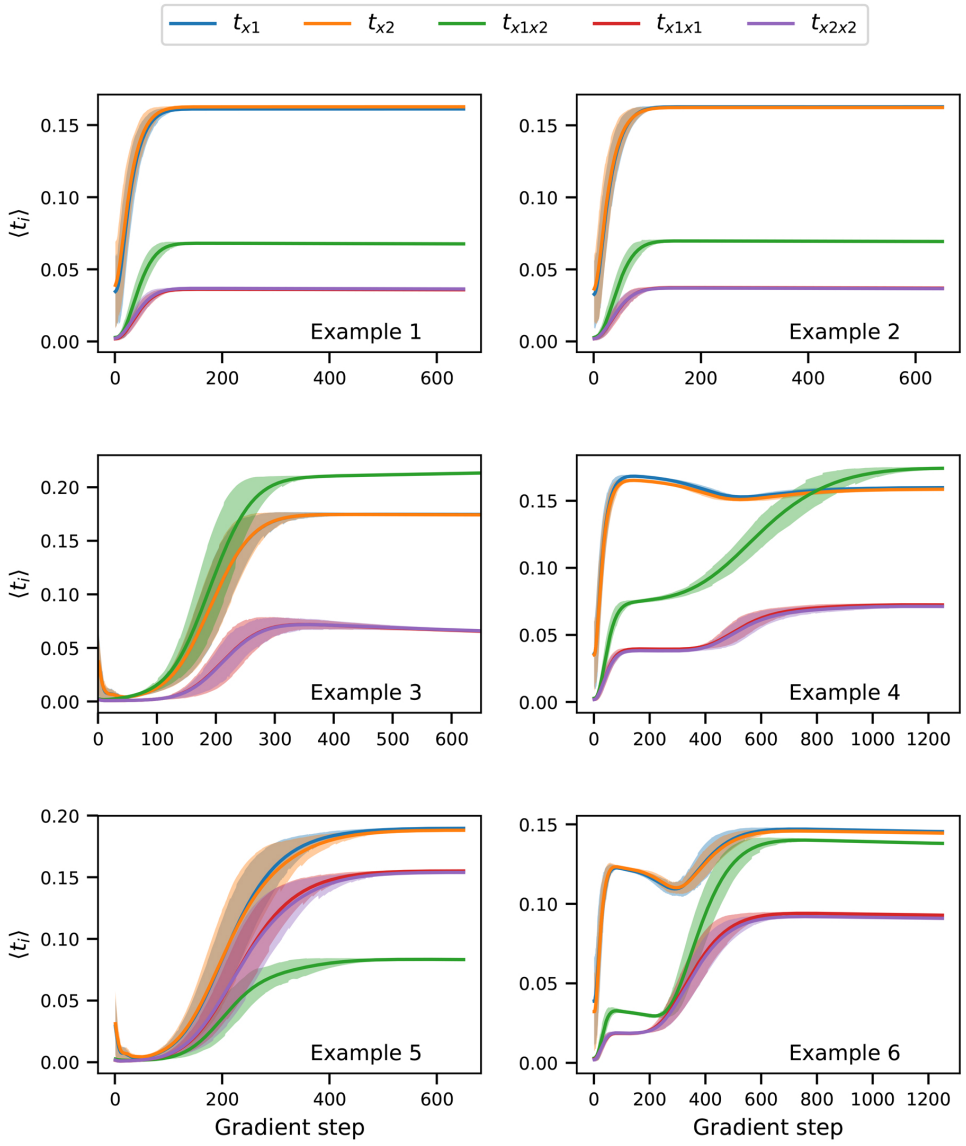

Figure 3.3: Taylor coefficients evaluated for 100 trainings with different seeds.

replication by controlling randomization. In Fig.3.3, the resulting metrics are shown. The darker lines illustrate the arithmetic mean values of the corresponding $\langle t_i \rangle$ and the transparent error bands around the mean values present the 68 nearest values to the mean for each gradient step. At the end of the training, all 100 models converge to the same metric values. It can be seen that no matter how the network is initialized at the beginning for each training, it produces the same end results. Thus, it is important to stop the training not at an early stage as there can be vast differences in dependencies on the input space during the training process.

It can also be seen, that the resulting plots for example 1 and 2 look similar. This confirms that the differences of these plots in the previous section are due to stochastical fluctuations.

3.4 Analysis on Input Space Dependence of Taylor Coefficients t_i

In the previous sections, the mean values of the Taylor coefficients were discussed as metrics that display the influence of input space characteristics on the NN output. But the importance of input features does not only depend on the size of the Taylor coefficients, but also on the location in the input space. This section will now take a closer look at the dependence on the input space by discussing the Taylor coefficients as functions of the input space. For this purpose, the previously obtained Taylor coefficients are plotted as colormaps with respect to x_1 (horizontal axis) and x_2 (vertical axis). Therefore, the Taylor coefficients are filled into bins which are normalized by the input space samples before being plotted as there are many samples in the plot center and few near the margins regarding the input data as shown in Fig.3.1. A bin number of 41 is used in x_1 and x_2 direction as an odd number is more suitable for displaying symmetries around the origin. For the purpose of illustrating positive and negative values, a diverging colormap is chosen. High absolute values are indicated by dark colors where blue represents negative and red positive values. To improve comparability, a constant color scale is used for as many epochs as long as recognizability is not affected. The color black indicates that a bin is empty. This is likely to happen near the plot margins as there the probability for occuring samples is small due to the used distributions. In addition to first and second-order Taylor coefficients, now, also the output function f is shown in the following figures. Another color range from 0 to 1 is assigned to f. Here, dark red corresponds to an output value 1 and dark blue 0. In the following, the colormaps for the six example toy datasets are shown in Fig.3.4 to Fig.3.11 for different training epochs. For practical reasons, only a few but representative training epochs are chosen.

3.4.1 Input space Dependency: example 1 and 2

Beginning with Fig.3.4, colormaps corresponding to example 1 as defined in Tab. 3.1 in section 3.1 are shown for epochs ranging from 1 to 100. At the beginning, the upper half of the input space is interpreted as background and the lower half as signal class due to random initialization of the NN parameters. This interpretation changes represented by a point-symmetric rotation of the output. In the end, a rotation of 135 degrees is completed. Now, the off-diagonal is colored white, the upper right half red and the bottom left half blue. This evaluation made by the NN seems to be reasonable, since a sample in the upper right half is more likely to be from the signal class and a sample in the lower left half from thebackground class. Only in the area around the off-diagonal axis the NN is not able to find a clear classification represented by the white-colored off-diagonal axis. This is reasonable, since the propabilities for signal and background class from the original dataset are equal along this axis. In summary, the final output matches well to the original example dataset which indicates a reasonable decision-making of the NN. Having discussed the output f of the NN, the following discussion will address the resulting Taylor coefficients and the relationship between output and Taylor coefficients. In order to give a better understanding of the results, different slices of the colormaps

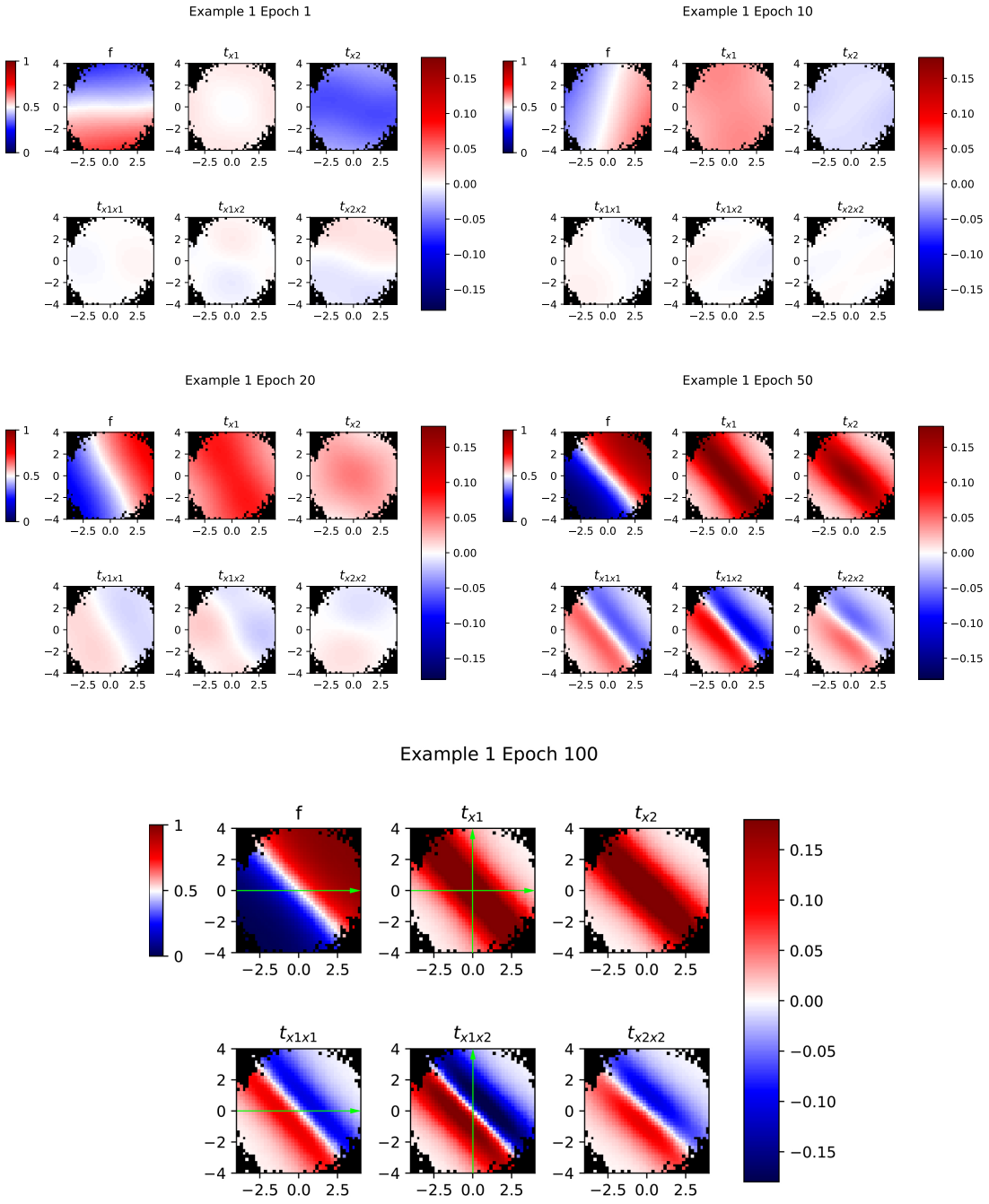

Figure 3.4: Illustration of the dependence on the input space of the first and second order Taylor coefficients for different training epochs of example distribution 1. x_1 is displayed on the horizontal axis and x_2 on the vertical axis. Moreover the output function f is displayed with a separate colorbar ranging from 0 to 1. The green arrows indicate the path of the two-dimensional slices of the three-dimensional colormaps shown in Fig.3.5.

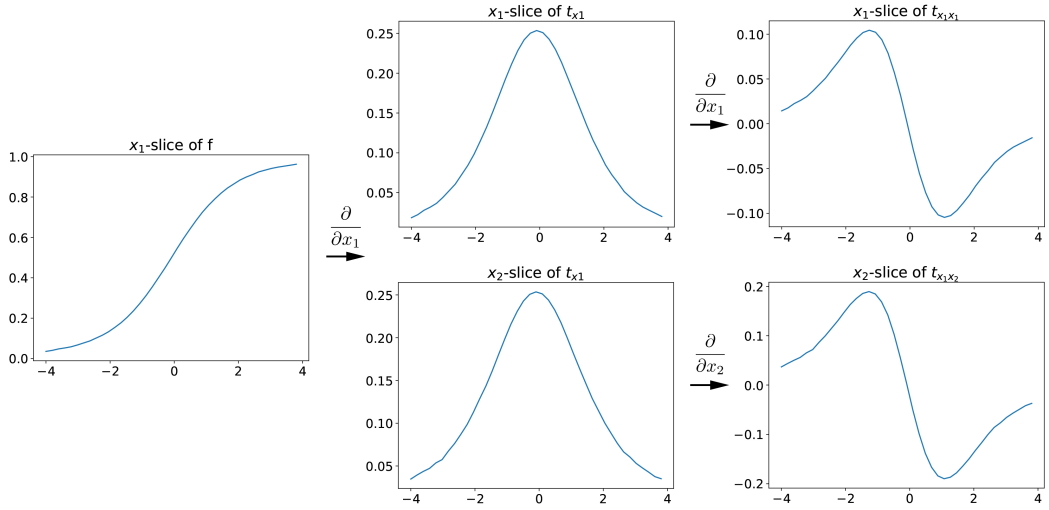

Figure 3.5: Two-dimensional slices of the t_i in x_1 and x_2 as marked in Fig.3.4.

are shown in Fig.3.5. Such a slice can be obtained by fixing one of the variables or their combination to a certain value. These slices are marked in Fig.3.4 by green arrows. For example, following the horizontal arrow in the output plot, the output increases monotonically from zero (blue) to one (red). The Taylor coefficient t_{x_1} is a derivative of f in x_1 direction and $t_{x_1x_1}$ is again a derivative of t_{x_1}in x_1 direction and, thus, a second-order derivative of f in x_1 direction. So, the first-order Taylor coefficients can be interpreted as measure of the gradient and the second-order features as measure of curvature.
Now, the slice plots are easier to understand. The point with the highest gradient of f corresponds to a maximum of t_{x_1} and a zero-point of $t_{x_1x_1}$. The curvature at this point changes from concave up to concave down. As the output function f is monotonically increasing in x_1 direction, all t_{x_1} values are positive. Near the upper right and lower left margins of the plot the output function reaches near constant values, therefore, the corresponding t_{x_1} values approach zero. Interesting is, also, the correlation feature $t_{x_1x_2}$ that can be explained as a sum of the derivative of t_{x_1} in x_2 direction and the derivative of t_{x_2} in x_1 direction that both look similar due to the symmetry of the data. Also because of the symmetry, $t_{x_1x_2}$ looks similar to $t_{x_1x_1}$ but with higher absolute values. Also for t_{x_2} and $t_{x_2x_2}$, a similiar explanation can be applied. The two-dimensional slices are a good method to improve understanding of the three-dimensional plots in Fig.3.4.
Focusing on the first-order feature colormaps for different epochs it can be seen that both are classified differently at the beginning: t_{x_2} is colored completely blue and t_{x_1} completely red. Then, the values of both metrics increase until both look equal showing a dark red band across the off-diagonal axis and white bins at the remaining margins. As classification in these white-colored areas should be relatively simple, this suggests the assumption that the color white indicates a high level of certainty of the NN about the classification of samples in so-colored bins. Consequently, the optimal metric would look completely white in this illustration. According to this interpretation, the dark red

in the middle might represent a high level of uncertainty of the NN. In dark colored regions, a small difference in the input space can lead to a big change in classification of the NN. So, regions with high first order Taylor coefficients have a strong impact on the NN output. On the other hand, in the beginning the most bins of the t_i plots are light-colored and become darker during training. So, the absolute values get higher which again corroborates the earlier asumption stating that high absolute values correspond to a high influence on the NN output. These results confirm the findings in [3] and regions are identified where uncertainties on variables have a high influence on the NN output.
In the end, the second-order feature plots look similiar to the plot of the output function f except for switched colors blue and red and white colored upper right and lower left corners. Here, the darkest colors occur in the $t_{x_1x_2}$ plot.
Overall, the colormaps are a useful instrument in order to characterize the output f. In addition, the colormaps provide a good illustration of the first and second-order Taylor coefficients as derivatives of the output function f.

The results corresponding to example distribution 2 are similar to the results of example distribution 1. The only differences are, that the bin colors are interchanged from red

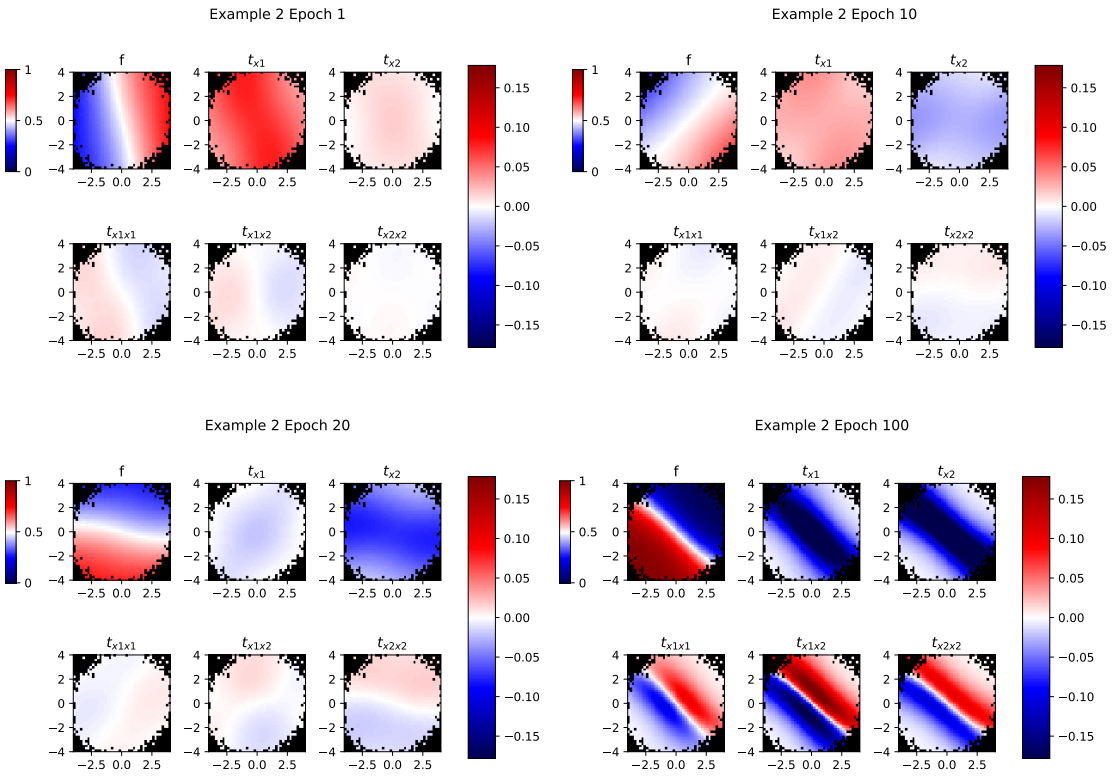

Figure 3.6: Illustration of input space dependence on first and second order Taylor coefficients for different training epochs of example distribution 2.

to blue and blue to red. This result is consistent with the dataset where also the colors are switched and the results from the previous sections. Considering an interpretation as used for example 1, a horizontal slice of the output f would show a monotonically decreasing function. Thus, the first-order gradients and the first-order Taylor coefficients are overall negative. The results of the second-order coefficients can be explained in a similar way.

3.4.2 Input space Dependency: example 3

In the case of Fig.3.7, the training progress according to example distribution 3 can be seen. The output f evolves from two differently colored halves separated by a narrow white area to four quadrants separated by a narrow point-symmetric white cross centered at the origin. The upper right and lower left quadrant are colored dark red, the other two quadrants dark blue. This geometry resembles the symmetry of the analyzed dataset. From the beginning on, the network realizes the symmetry of the classification but it takes about 100 training epochs until the NN identifies the four quadrants dominating the final plots. In each of the first-order feature plots after 1999 epochs a broad bar covering the axes can be seen changing color from blue to red; for t_{x_1} horizontally and for t_{x_2} vertically.
As for example 1, slices are shown in 3.8. This time, the slices are not taken from the center of the plots but shifted away from the center, as through the center the path is along a white area and, thus, nothing remarkable can be seen. For example, looking at the horizontal center, also the t_{x_1} and $t_{x_1x_1}$ plots display white lines. The same applies to t_{x_2} and $t_{x_2x_2}$ in vertical direction. Now, looking at the corresponding slice of f in x_1 direction, a monotonically decreasing function similar to the one from example 1 can be seen that results in negative t_{x_1} values. Moving the slice above a x_2 value of 0, would result in a monotonically increasing function and positive t_{x_1} values. The curvature of the diagonal slice leads to the plot that can be seen for $t_{x_1x_2}$. The corners of the t_i plots are colored white. Looking at the corresponding data sample in Fig.3.1, these regions should be easy to classify by the marginal distributions. For example, the possibility for a signal class like data point in the upper right corner is relatively high. This confirms the theory that white regions of the metrics indicate certainty of the NN classification. The bins at $x_1 = x_2 = 0$ are colored white which would correspond to a straightforward classification as stated in 3.4.1. But in this case, the color white does not correspond to a maximum or minimum but to a saddle point as can be seen by the second order features and, therefore, this white-colored area does not indicate high certainty of the NN in classification.
The self-correlation metrics $t_{x_1x_1}$ and $t_{x_2x_2}$ form a frame around a centered white cross similar to the one that can be seen in the output plot but with contrary colors. As for the first-order metrics, either the horizontal or vertical framing is more pronounced: horizontal for $t_{x_1x_1}$ and vertical for $t_{x_2x_2}$. The findings suggest unambiguous classification at the corners. Interesting is also the plot of $t_{x_1x_2}$ that shows a star-like geometry which can again can be understood better by looking at the slices.

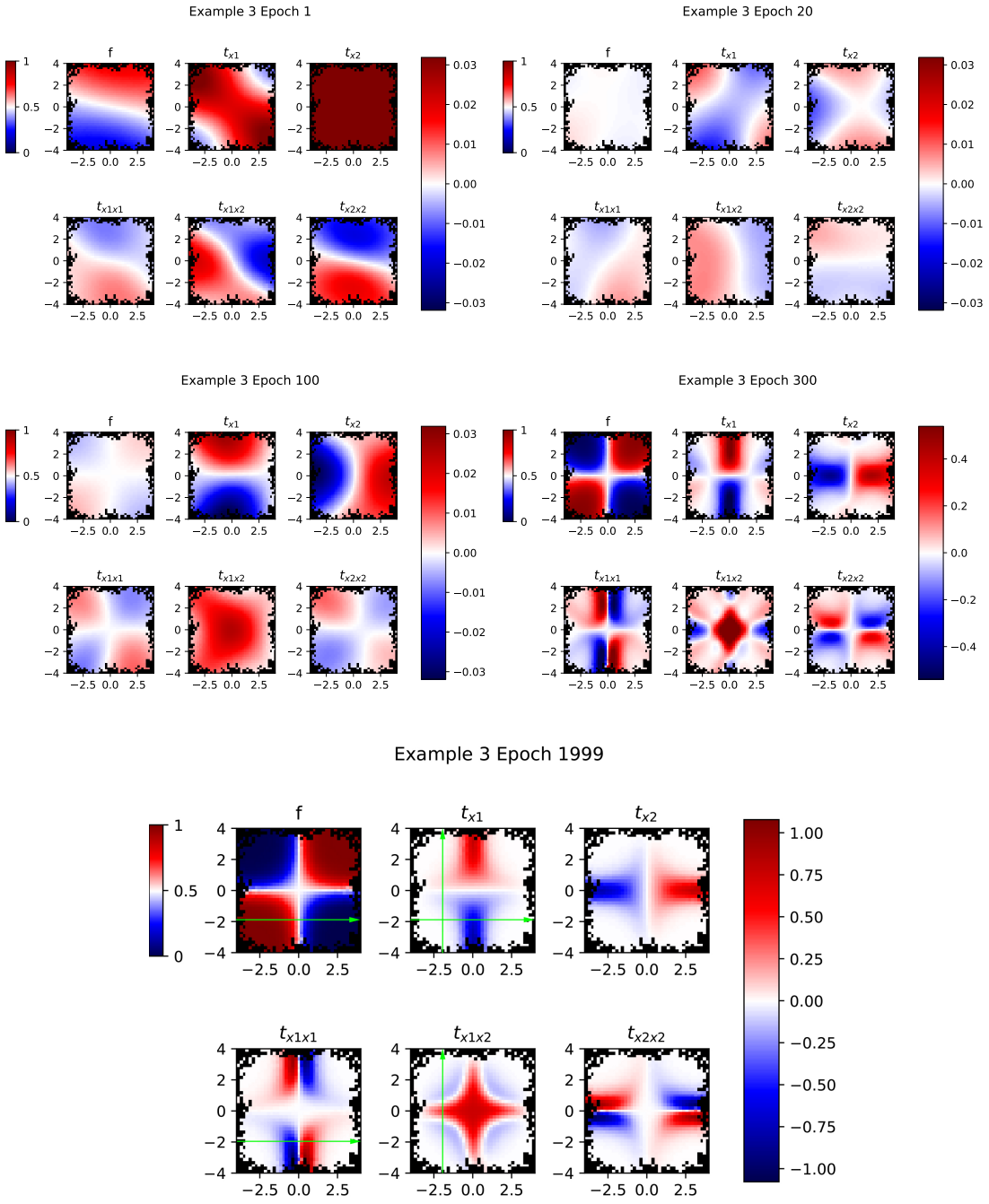

Figure 3.7: Illustration of input space dependency on first and second order Taylor coefficients for different training epochs of example distribution 3.

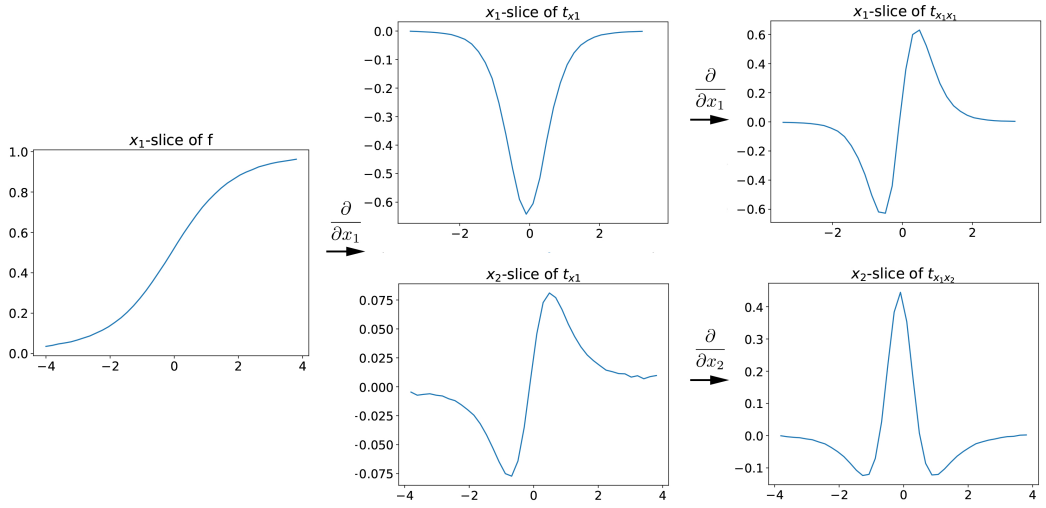

Figure 3.8: Two-dimensional slices of the t_i in x_1 and x_2 as marked in Fig.3.7.

3.4.3 Input space Dependency: example 4, 5 and 6

For examples 4, 5 and 6 only few plots are shown. The corresponding plots during training are shown in the appendix section A.
Looking at Fig.3.9 a plot displaying the geometry of the input data emerges for f and the metrics t_i. As discussed in section 3.2 the plots look similar to example 1 up to the 100th epoch. After this point, the additional features are recognized, leading to the final plot after 1999 epochs. The reddish bottom left corner of f can be explained by samples from signal class of the input dataset that are located in this corner even though the possibility for signal class is very small in this area. The colormaps of the Taylor coefficients are too complex to discuss in detail in this thesis.

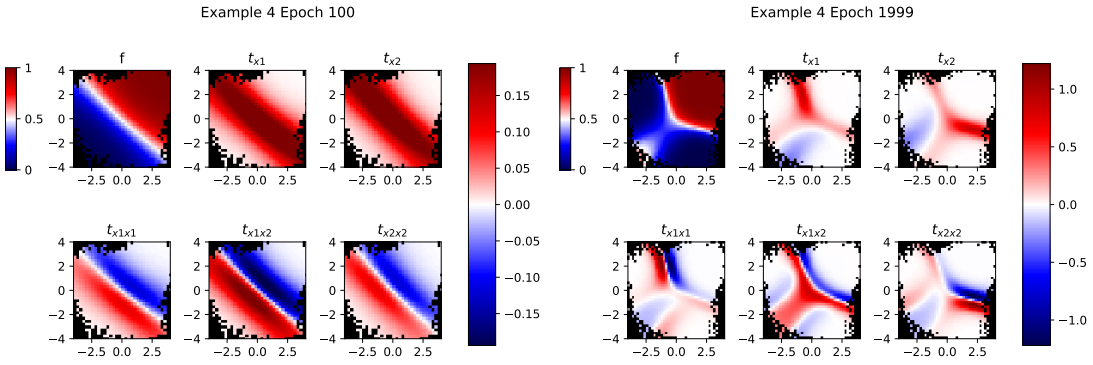

Figure 3.9: Illustration of input space dependency on first and second order Taylor coefficients for training epoch 1999 of example distribution 4.

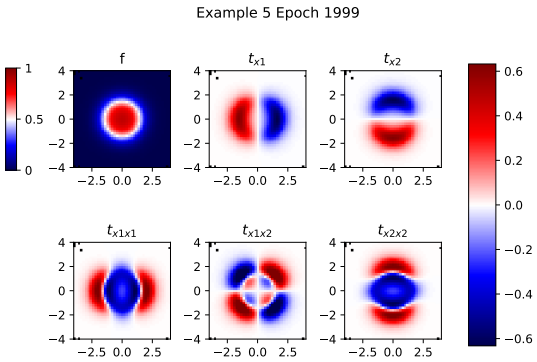

Figure 3.10: Illustration of input space dependence on first and second order Taylor coefficients for training epoch 1999 of example distribution 5.

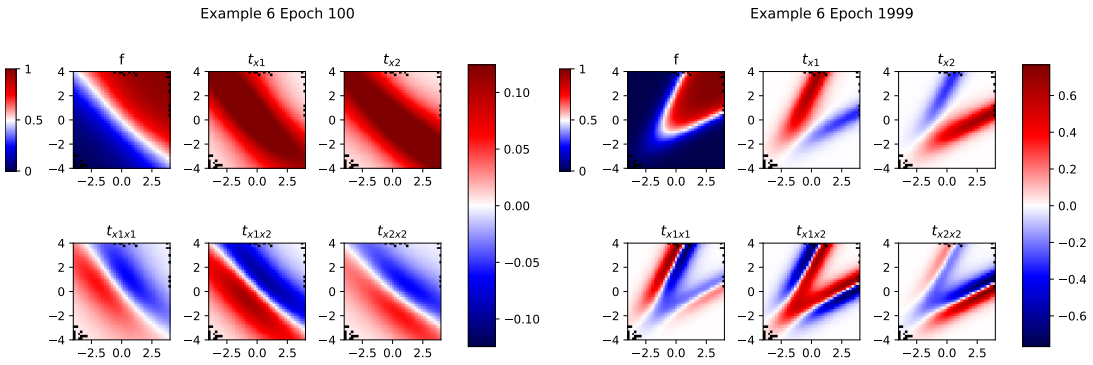

Figure 3.11: Illustration of input space dependence on first and second order Taylor coefficients for different training epochs of example distribution 6.

Also in the cases of example 5 and 6 (Fig.3.10, Fig.3.11), the output function f and its Taylor coefficients resemble the geometries of the associated datasets. Also in example 6, the NN decides similarily to example 1 up to the 100th epoch but then recognizes the additional features of the dataset. The NN adapts to the dataset by emerging a geometry looking like a Y similar to example 4.

CHAPTER 4

Summary, Conclusion and Outlook

The purpose of this thesis is to develop a deeper understanding of Neural Network learning processes by illustration of trainings and examination of the significance of single input characeristics on Neural Network decision taking. The investigation of Artificial Neural Network trainings has shown that it is in fact possible to visualize the training process of NNs by analysis of the Taylor coefficients of the NN model function. This research serves as a base for future studies and the methods used for this thesis can be applied to other NN studies.
Within this work, the findings of [3] could be reproduced and applied to further examples. This thesis has shown, that NN trainings with different seeds can be used to analyze the convergence and stability of a NN model. If the training is performed long enough, NNs that are initialized differently at the beginning can produce similar outputs in the end. The illustration of the training process using colormaps is a valuable method in order to gain a deeper understanding of the training process and especially of the final NN output. This method has facilitated a comparison between the NN outputs and the six toy datasets, which has shown that the decision taking of the NNs is indeed reasonable in these cases. By analysis of the Taylor coefficients corresponding to the NN output functions, regions of the input space could be determined in which small variations of the input variables for example caused by systematical uncertainties lead to great variations of the NN output. So, especially in these regions, it is important to gain control over uncertainties of the input space. Furthermore, those Taylor coefficients allow for general predictions on the influence of variations of the input variables x_i on the NN output. The findings suggest that areas of the first-order taylor coefficient plots with values near to zero correspond to areas with high classification confidence of the NN or to saddle points.
Although the current study is based on a small number of datasets, this thesis contributes in enhancing the understanding of Neural Network training. Further studies can investigate on applying these finding to other classification problems or even tasks besides classification in order to confirm and generalize the results of this thesis. As this thesis only investigated on NN trainings of feedforward network with one hidden layer, it would be interesting to investigate on the behaviour of different and particularly more complex NN models. For example, the number of layers and nodes or the batch sizes are attributes

of NN models that can be easily changed and can lead to variations of the training performance. The methods used in this thesis can be applied to find a most suitable NN model for a given task. Also, further research might investigate on Taylor coefficients of higher than second-order. Another interesting approach would be to investigate on classification of an input space with more than two independent variables. In this case, the colormap method presented in section 3.4 can be very helpful as the Taylor coefficients can be used to display the dependence of the NN output on all given input variables.

APPENDIX A

Appendix

A.1 Input space Dependency: example 4, 5, 6

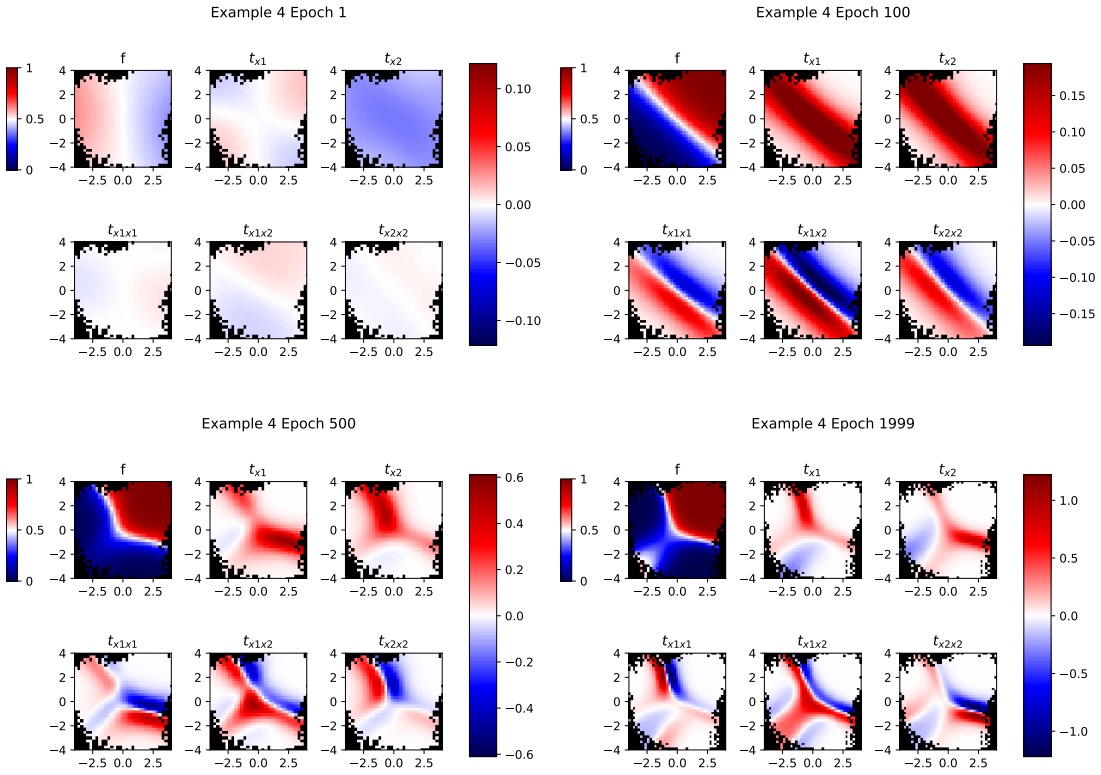

Figure A.1: Illustration of input space dependency on first and second order taylor coefficients for different training epochs of example distribution 4.

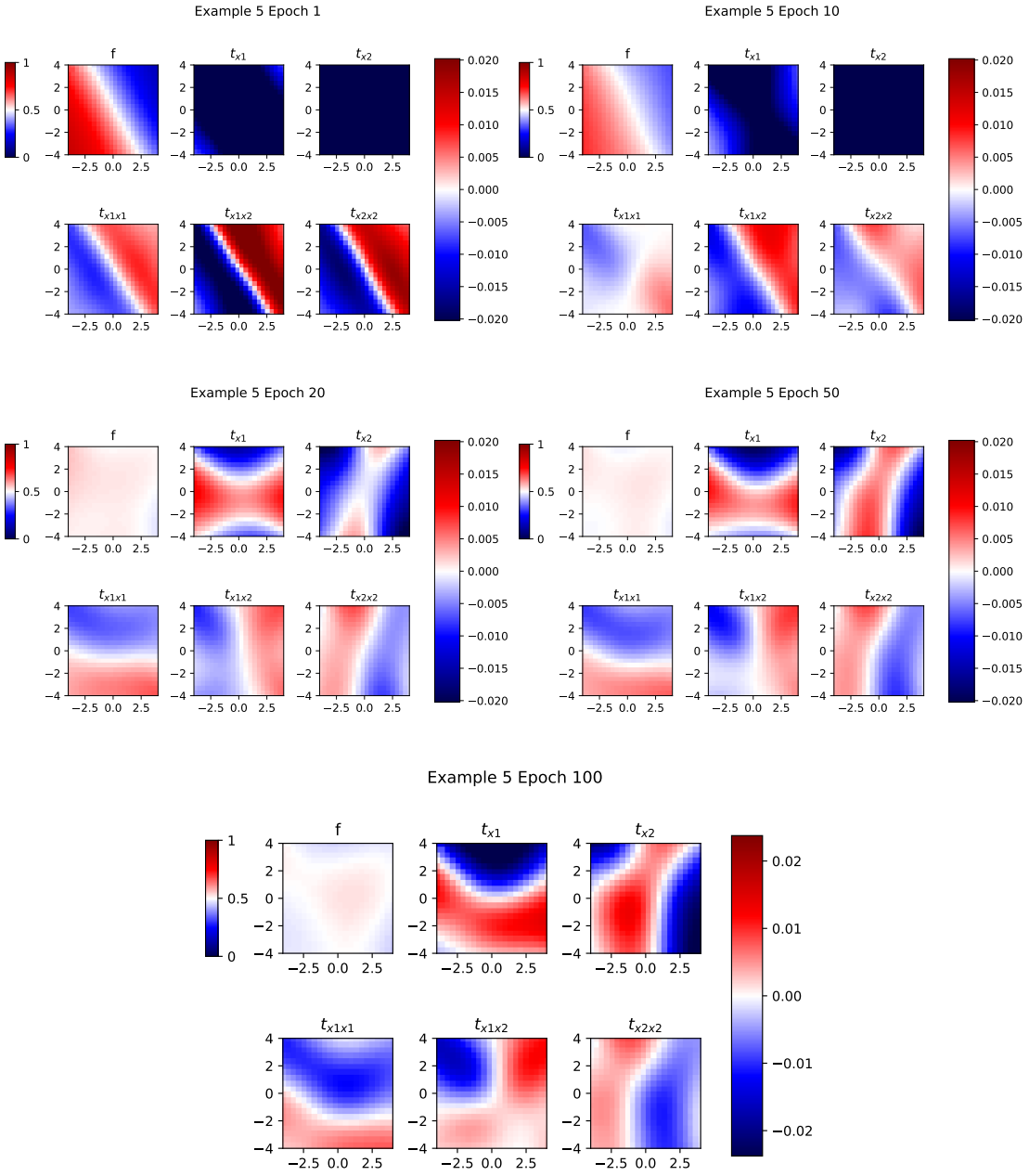

Figure A.2: Illustration of input space dependency on first and second order taylor coefficients for different training epochs of example distribution 4.

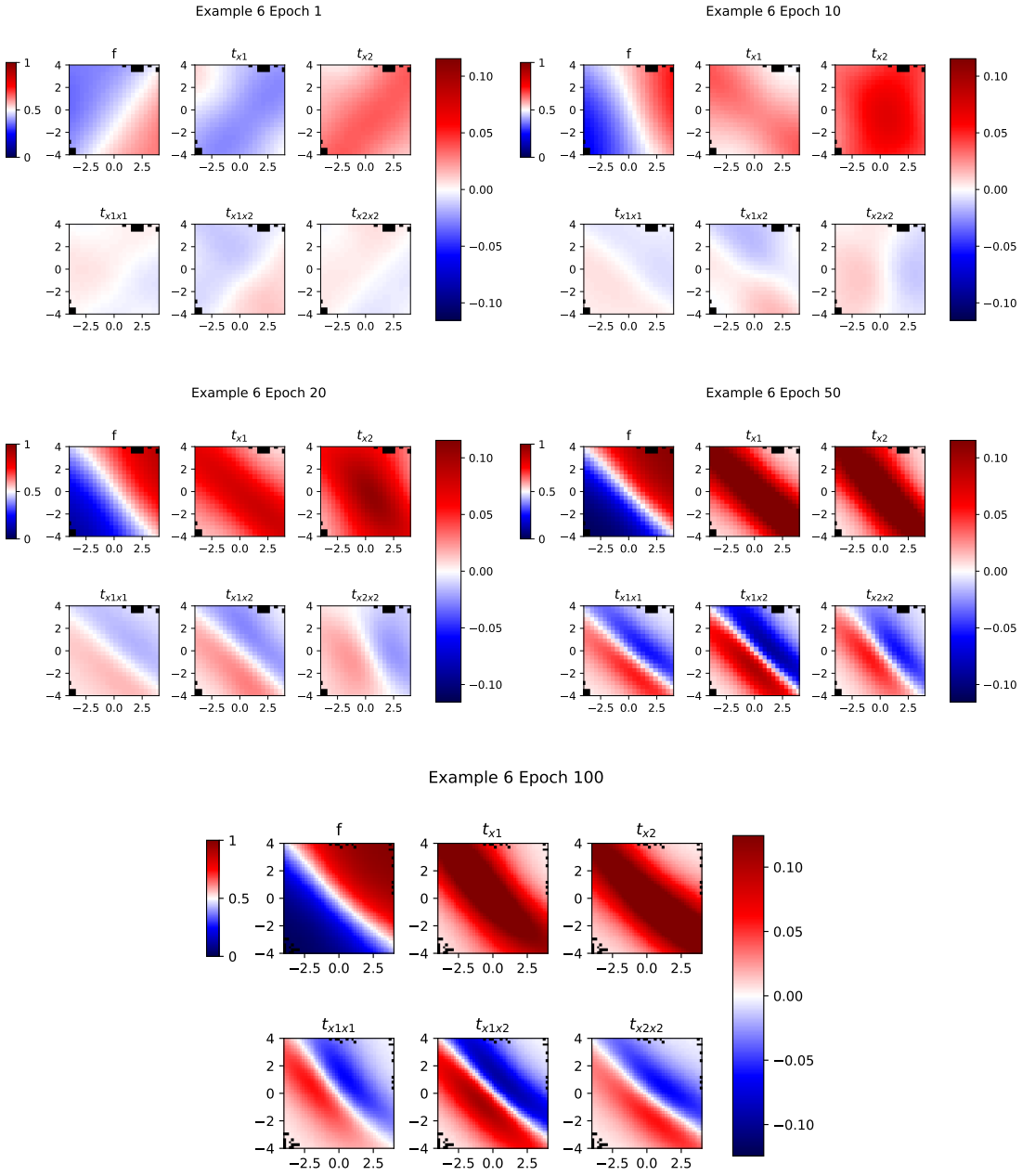

Figure A.3: Illustration of input space dependency on first and second order taylor coefficients for different training epochs of example distribution 4.

APPENDIX B

Bibliography

[1] François Chollet. "Deep Learning with Python". Manning, Nov. 2017. ISBN: 9781617294433.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. "Deep Learning". The MIT Press, 2016. ISBN: 0262035618, 9780262035613.

[3] Stefan Wunsch et al. "Identifying the Relevant Dependencies of the Neural Network Response on Characteristics of the Input Space". *Computing and Software for Big Science* 2.1 (Sept. 2018). ISSN: 2510-2044.
DOI: 10.1007/s41781-018-0012-1. URL: http://dx.doi.org/10.1007/s41781-018-0012-1.

[4] Shie Mannor, Dori Peleg, and Reuven Rubinstein. "The Cross Entropy Method for Classification". *Proceedings of the 22Nd International Conference on Machine Learning*. ICML '05. Bonn, Germany: ACM, 2005, pp. 561–568. ISBN: 1-59593-180-5.
DOI: 10.1145/1102351.1102422. URL: http://doi.acm.org/10.1145/1102351.1102422.

[5] A. Cauchy. "Méthode générale pour la résolution de systèmes déquations simultanées". *Compte rendu des séances de lacadémie des sciences* (1847), pp. 536–538.

[6] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980* (2014).

[7] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[8] François Chollet et al. *Keras*. https://keras.io. 2015.

[9] Guido Rossum. "Python Reference Manual". Tech. rep. Amsterdam, The Netherlands, The Netherlands, 1995.

[10] Machine Learning Mastery. *What is the Difference Between a Batch and an Epoch in a Neural Network?* https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/. Accessed: 2019-09-25.

[11] "Taylor expansions and applications". Milano: Springer Milan, 2008, pp. 223–255. ISBN: 978-88-470-0876-2.
DOI: 10.1007/978-88-470-0876-2_7. URL: https://doi.org/10.1007/978-88-470-0876-2_7.