Minimum levels of high energy particle bombardment on fusion reactor vessels:

Towards a computational multi-parameter scan for automated data reduction (”big data”) applications

Christof Backhaus

December 2017

Contents

Abstract

Chapter 1

Introduction

The design process for a facility like a fusion power plant takes into account a manifold of aspects. Thereunder a cost analysis for the fusion device. To make an estimate of the cost analysis one has to consider the lifetime of machine parts. Most prominently the divertor and first wall suffer from shortened life spans due to erosion. Which is partly due to neutral particle induced sputtering.
To include considerations like these in the design of a power plant one uses so called system codes. These codes focus on optimizing design parameters of

- System Code

-

1.1 System Codes

1.2 Reduced Model Approaches

Chapter 2

Model

2.1 EIRENE 1D

2.1.1 Kinetic Boltzmann Equations

2.2 Plasma Profiles

For the inputs of Eirene plasma profiles are needed, that can be dynamically provided by other algorithms like SOLPS[1] or EMC3[2]. Central piece of this work is to investigate if a substitute function can be found for the full range of possible plasma profiles by using big data methods. One can ascertain the physical limits of the parameters constituting the plasma profiles from table **??**. These limits are based on different phenomenons in plasma physics, which can be

2.3 Choice of sampling set

Since the parameter space is high dimensional, the training points have not been selected randomly. According to [3] randomly sampled points might form clusters, which could skew the training towards a subsection of the parameter space. To avoid this a low discrepancy sequence, namely the Sobol sequence, has been chosen from which the training points are sampled.

[1]Citation needed

[2]Citation needed

[3]Add reference

2.3.1 Sobol Sequence

The sobol sequence was first invented by the mathematician Ilya M. Sobol[4]

Low-discrepency sequences

[4] Add citation

Chapter 3

Methods

This chapter is concerned with introducing the methods used to investigate the data reduction of the previously introduced model. The focus of this work will be on artificial neural networks (ANN in short) and Gaussian processes. Before these two methods are described in greater detail a general overview over possible other approaches will be given.

- Support Vector Machines (SVM):
  Support Vector Machines (in short SVM) are used to classify data similar to ANNs. In contrast to ANNs SVMs are build up from theory and contain little Hyperparameters[1] making them easier to analyse and less prone to overfitting. On the other hand finding the correct choice of Hyperparameters for a SVM requires prior knowledge of the problem at hand while not allowing for the flexibility of neural networks.

- Random Forests:

Short description of Random Forests

- Boosting:

Short description of Boosting methods

3.1 Neural Networks

The following section is concerned with discussing neural networks as a means of investigating functional dependencies.[2]

[1]Please refer to section 3.1.4 for further information.

[2]To aid with understanding the terminology used there is a glossary in the appendix section B.1.

3.1.1 General Introduction To Neural Networks

An artificial neural network in the following called neural network, abbreviated to NN, is "a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs." [1] First concepts of learning algorithms based on neural plasticity have been introduced in the late 1940s by D. O. Hebb [3]. In 1975 backpropagation became possible via an algorithm by Werbo[4], this led to an increase in machine learning popularity. During the 1980s other methods like support vector machines and linear classifiers became the preferred and/or dominating machine learning approach. With the rise in computational power in recent years neural networks have gained back a lot of popularity.
The concept idea of neural networks is to replicate the ability of the human brain to learn and to adapt to new information. The structure and naming convention reflect this origin.
A neural network is made up of small processing units called neurons. These are grouped together into so called layers. Every network needs at least two layers, the input layer and the output layer. If a network has intermediary layers between input and output, they are called hidden layers. A network with at least two hidden layers is called a deep network. The amount of layers in a network is called the depth of the network. While the amount of neurons in a layer is called the layers width. As mentioned above the neurons are highly interconnected. The structure of connections determine the type of network. In the following we will discuss fully connected networks[5]. These feature a connection between each neuron of adjacent layers[6] From layer to layer the information stored in the neurons is transferred into the next layer by a weighted sum and a non-linear function called activation function. An example is provided in section 3.1.2.

Neural networks are usually used in two ways, optimization or classification. Well known examples are

[3]citation needed

[4]citation needed

[5]Fully connected networks are also called dense networks.

[6]The amount of connections between two layers in a dense network is therefore equal the the product of the width of adjacent layers.

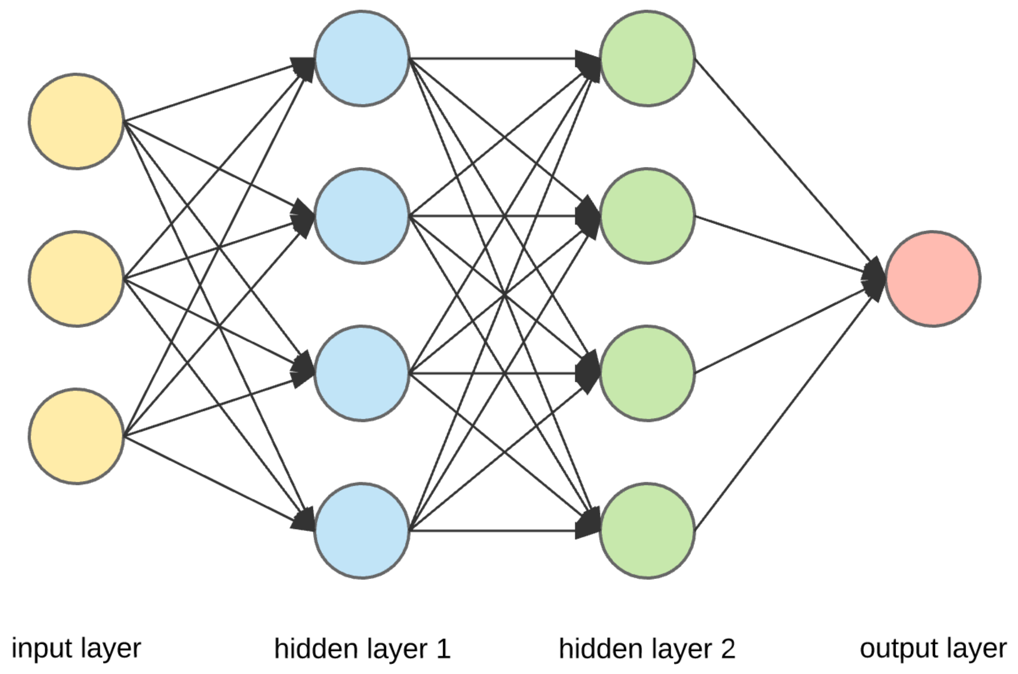

Figure 3.1: Schematic structure of the most basic fully connected deep neural network. Indicated are the input (yellow), output (red) and hidden layers (blue and green). Each neuron outputs to all neurons in the following layer, but there are no interconnection between neurons of the same layer. Note that while the network has the minimum depth (2 hidden layers) to qualify for a deep neural network, the width could be smaller.

3.1.2 Fully Connected

A fully connected or dense neural network like depicted in in figure 3.1 is characterized by connecting every neuron from the previous to the following layer with a weight. In contrast to other networks this allows for a very high flexibility but also lacks the spatial context of data. This kind of network is especially well suited for data that is given in form of vectors or drawn from an arbitrary parameter space.

The working principle is to form a weighted sum $\sum_{i=1}^{N} w_{i,j} * x_i$ over the values from neurons of the previous layer x_i weighted by the connecting weight $w_{i,j}$. The weighted sum is then evaluated by the activation function $\sigma()$ such that the new value $x_j = \sigma(\sum_{i=1}^{N} w_{i,j} * x_i)$. Here i refers to the index of the neuron in the previous layer and j to the index of the neuron in the current layer.
Since forming a weighted sum is a linear operation the activation function

must be non-linear to enable the network to learn non-linear behaviour.[7] The most common activation functions are the rectifier also called rectified linear unit (short ReLU) and exponential linear unit (ELU). Both are inspired by the asymmetrical behaviour of biological neural connections[8].

In contrast to fully connected networks convolutional networks apply a so called filter to the input. These filters consider the inputs of nearby neurons as well. They are usually used in image recognition and require data that has information stored in patterns, most commonly special patterns as in images. Fig. **??** depicts an exemplary convolutional network for image data and gives indication to it's working procedure.

3.1.3 Structures

3.1.4 Hyper parameters

A hyper parameter refers to a parameter of the network that is not changed during training. Since these can have substantial influence on the performance of the network they will be explained in the following

Depth

The depth of the network correlates directly to the amount of chained non-linear functions. Therefore it strongly influences the ability of a NN to learn abstract patterns. The more complicated a pattern is the more depth is "required" to learn the pattern. For example a simple classification between left and right only requires a shallow network, whereas recognizing different brands of car requires a deeper network.

Width

The width of a layer refers to the amount of neurons in that layer. Often networks are build of layers with the same width in each hidden layer. If that is the case one sometimes speaks of the width of the network which is then equivalent to width of a/each hidden layer.

Activation or Activation Function

As previously discussed the activation function introduces non-linearty to the network. Some activation function will be better suited to model a cer-

[7]ref needed

[8]reference needed

tain problem than others. If information about the model or pattern to be predicted is known, an activation function close to this will have better performance. For example predicting outcome of a sine function will perform better with exponential activations or uneven polynomial activations than even polynomials.

Loss Function

Choosing a loss function determines what criteria the network optimizes for which directly corresponds to which patterns it learns. For prediction one typically chooses a root mean square function. For classification cross entropy loss functions are most common.

Batch size

The Dataset is divided into subsets called batches which are fed to the current network during training. After each batch the weights are adjusted. Splitting the dataset in this way is advantageous to the computational performance during training. Less memory is used during training and the number of epochs trained is reduced. The flip side of using a batch size smaller than the number of training data is that the gradient for optimization will be worse in comparison to the gradient calculated with the full data set.

Epochs

An epoch describes a full training cycle of training, validating and adjusting weights for the entire training data set. If the batch size is smaller than the number of training points then multiple[9] adjustments are made.

Metrics

Metrics are additional information gained from the network during training and evaluation. Metrics are not hyperparameters since they do not influence the resulting network but are an important source of information for further improving the network structure. For example a secondary loss function can be implemented as a metric to evaluate general optimization of the network in contrast to only the chosen loss quantity.

[9]Number of batches in one epoch = rounded up $\left(\frac{AmountofTrainingData}{BatchSize}\right)$

Regularization

Regularization describes methods used to reduce the generalization error but not the training error. Commonly used regularization methods include L1, L2, Dropout and Early Stopping regularization. L1 and L2 regularization is applied by adding a penalty term to the loss function. This requires initial knowledge of input influences. For example an image with bad resolution might have a larger penalty term applied than an image with high resolution. Dropout regularization and early stopping are used to prevent overfitting. Since the amount of parameters in the network is often on the same order of magnitude as the amount of training data, neural networks are prone to overfitting. Early stopping interrupts the training process as soon as the validation loss stops improving by a user set minimum delta.

3.2 Gaussian Processes

3.2.1 General Introduction

3.2.2 GOGAP algorithm

Chapter 4

Results

4.1 Physical Results

4.2 Neural Networks

4.2.1 Hyper Parameters

The following subsets of hyper parameters have been investigated: Beginning with investigating the effect of the data amount. Using same network size per

Number	Width	Depth (hidden)	Amount of Data	Activation	Droprate
1	100	10	2^17	Relu	0.25
2	X	X	2^16	X	X
3	X	X	2^15	X	X
4	X	X	2^14	X	X
5	80	X	2^17	X	X
6	X	X	2^16	X	X
7	X	X	2^15	X	X
8	X	X	2^14	X	X

Depth & Width

Depth and Width together determine the total number parameters and complexity of the network. The question is how complex does the network need to be to accurately learn the model. Ideally we'd like to trim the network as much as possible without loosing too much accuracy.

Activation

Elu, Relu, Sigmoid

Loss Function

SGD

4.2.2 Derivatives

4.3 Gaussian Processes

4.3.1 Subdivion of parameter space

4.3.2 Derivatives

4.4 Comparison

4.4.1 Accuracy

4.5 NNGP - Maybe

Chapter 5

Conclusion

5.1 Summary

5.2 Outlook

Appendix A

More Data probably

Appendix B

Background Neural Network

B.1 Introduction to Neural Networks

Glossary:

- Network: A series of layers. The first layer of a network is called the input layer, the last layer is called the output layer. Any layers in between are called hidden layers. The amount of hidden layers is called the **depth** of the network.

- Layer: A collection of neurons. The amount of neurons in a layer is called the **width** of a layer.

- Neuron: A single node in a layer. It contains a single number formed by a weighted sum of it's inputs evaluated by the activation function.

- Activation function: A non-function applied to the weighted sum of a neuron. Used to introduce non linearity into the network in order to enable non linear model "fitting".
- Weight: Each connection between layers has it's own weight factor. These are adjusted during training to fit the data. Weights are often referred to as parameters.

- Regularization: Methods used to suppress overfitting.

- Metric: Additional information gathered during training/testing.

- Loss function: Function that dictates the optimization, e.g. Root Mean Squared.

- Training Data: Set of Data used during training phase. Weights are adjusted to these data.

- Validation Data: Set of Data used during training phase to evaluate training results intermediary.

- Test Data: Data set not seen during training to evaluate trained network performance.

- Input: Data fed to the network for training or evaluation.
- Output: Prediction of the network.
- Label: True output value for input data.
- Hyperparameter: A parameter not changed during training e.g. width and depth of the network.

B.2 Examples of neural networks in different contexts

Appendix C

Background Gaussian Processes

C.1 Baysian Statistics

Bibliography

[1] Maureen Caudill. Neural networks primer, part i. *AI Expert*, 2(12):46–52, December 1987.