University of Applied Sciences Würzburg-Schweinfurt Faculty of Computer Science and Business Information Systems

Bachelor-Thesis

Evaluation and implementation of gradient descent algorithms on streaming data

submitted to the University of Applied Sciences Würzburg-Schweinfurt in the Faculty of Computer Science and Business Information Systems to achieve the Bachelor of Engineering degree in 'Information Systems'

Florian Hohn

Submitted on: 25.02.2020

First Reader: Prof. Dr. Frank-Michael Schleif
Second Reader: Moritz Heusinger

Summary

TODO

Abstract

TODO

Note of thanks

Contents

1 Introduction

2 Basics

The Goal of this Chapter is to give the Reader a basic understanding of how Machine Learning Algorithm work. Furthermore, it will describe what the difference is between unsupervised und supervised learning.

There will be also an introduction and description on how the Gradient Descent Algorithm, that is the basis for the Modified versions of the Algorithms that will be compared in this work, works and how it decides what to do. After this is done there will be also a short description on what Streaming Data is and on the framework that is used to realise the implementation of the Experiments.

Finally there will be a explanation on how the criteria for the comparission of the Algorithms where choosen and how theses criteria work.

2.1 Machine Learning

The term 'Machine Learning' stands for a collection of self-learning algorithms, that learn form input Data to make fast and correct decisions and predictions. It is importand to remember that it is not a real Artifical inteligence, but rather a sequence of statistical analyses on given data, with the goal to make gradualy improvments to the prediction models of the algorithm. A Machine can do this optimization alot faster and effectifly than a humans, especially if this optimization has to be done in real time on big data input sets. This give also a bigger flexability, as machine that can learn from a changing enviroment needs lesser redesigns, which leads to less time that gets wastet and can be used more productifly on other tasks.

All these are reasons why Machine Learning has an ever increasing importants in the field of computer science, but also an growing impact on everday life. It is thanks to Machine learning that there are intelligent assisstant softwares that can recognize voice commands and questions (Apples SIRI, Amazon Alexa or Microsofts Cortana), e-mail spam filter, nearly self-driving cars, reliable search-engines, faster and more precise weather forecasts and challenging game ai's nowadays

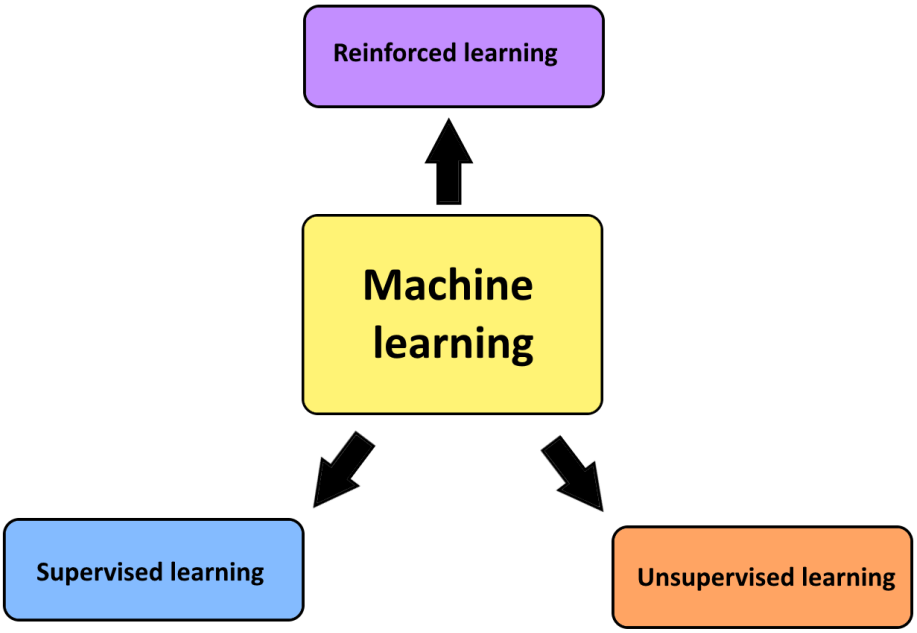

Figure 2.1: Overview of the Machine Learning Strategy's.

Machine Learning Algorithms can be categorisized depending on how they learn from the available training Data to create a model for the predictions. There are three main subdivisions, under wich these algorithms can fall. The first one is the Supervised learning strategy, then the Unsupervised learning strategy and the Reinforced learning strategy.

Only the Supervised Learning strategy will be explained in more detail on the following pages, as the focus of this thesis lays in the comparission of several different implementations of classification algortihms that are based on Supervised Learning.

[4]

2.1.1 Supervised Learning

The idea behind Supervised learning is to create a model from a already labeled training data set, that then can make prediction and decision on new or unknown data based on the learned model from the training set. The following example will help to understand this easier. Lets say a fruit farmer wants to automatically sort the harvested Fruits in either categorie A or B for the market. The sorting device is equiepped with 2 sensors to measure 2 features, one for colour and another one for the size of the fruit. It then decides to wich of the two classes the fruit belongs to. As this is a System that is capable of dividing features into a discrete number of classes the system can be called a *classifier*. This is a typical exampel of an *classifiecation task*. To configure the System in such a way that it correctly sorts the fruits into the right category, a specailist hand picks several fruits from a small test set and then sorts them depending on the already named features. In other words, the specailist classiefies them. Based on this by hand sortet training set the machine will then later decide how the other fruits should be sorted. It is used as a *model* for the classification. [5] Another type of Supervised learning is the *regression task*. The difference to the *classification task* is, that instead of a *discrete* number of classes that gets returned, a *continous* number of classes will be returned. An example of a *regressive task* is, for example, the prediction of the Stock market or the price of medications.

The Process as seen in figure 2.2 and as described as in the example of the farmer and his sorted fruits shows how the supervised learning works. The Algorithm (marked in green) learns with the help of the labeled training data. The result of this learning is then that the trained Algorithm, now called predictive model, can be used in a production enviroment. New, unlabeled, data can then be send into the predictiv model to get a predictied model as an output. [7]

To better understand how this predictiv model is formed, one has to understand how the classification task works. In the classification task the computer programm is asked to specify which of k catergories a input belongs to. To solve this task, the learning algorithm is usually asked to produce a function $f : \mathbb{R} \rightarrow 1, ..., k$. When $y = f(\boldsymbol{x})$, the model assigns an input described by vector $\boldsymbol{x}$ to a category identified by numeric code y. There are other variants of the classification task, for example, where f outputs a probability distribution over classes. An example of a classification task is object recognition, where the input is an image (usually described as a set of pixel brightness values), and the output is a numeric code identifying the object in the image. [3]

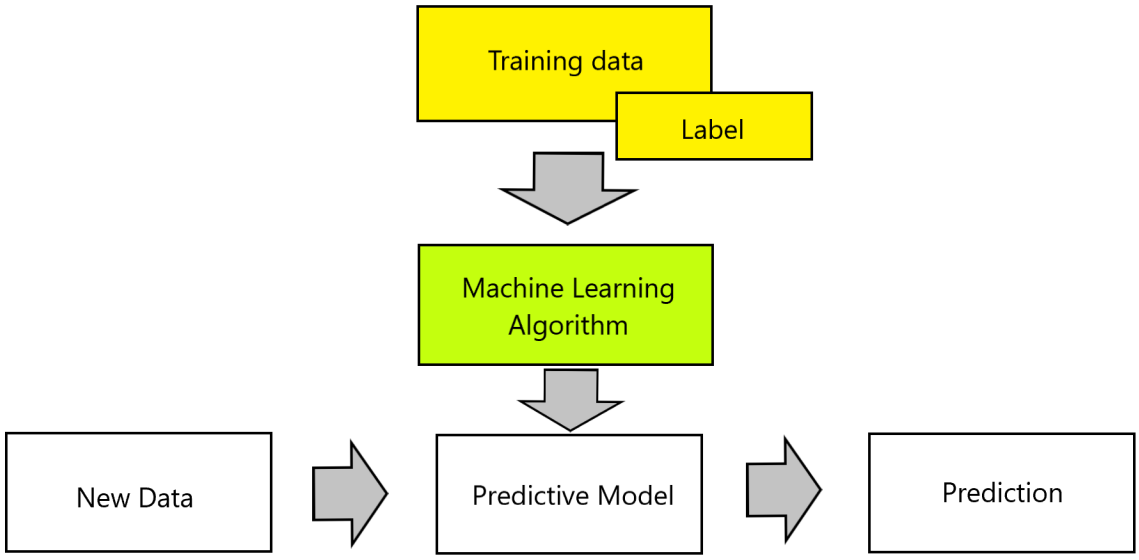

Figure 2.2: Process of the Supervised Learning. [7]

Then there must be also differentiated between the type of data that is put into the classification process. In batch or offine classification, a classifier-building algorithm is given a set of labeled examples. The algorithm creates a model, a classifier in this case. The model is then deployed, that is, used to predict the label for unlabeled instances that the classifier builder never saw. If we go into more detail, we know that it is good methodology in the first phase to split the dataset available into two parts, the training and the testing dataset, or to resort to cross-validation, to make sure that the classifier is reasonably accurate. But in any case, there is a first training phase, clearly separated in time from the prediction phase.

In the online setting, and in particular in streaming, like it is this thesis, this separation between training, evaluating, and testing is far less clear-cut, and is interleaved. We need to start making predictions before we have all the data, because the data may never end. We need to use the data whose label we predict to keep training the model, if possible. And probably we need to continuously evaluate the model in some way to decide if the model needs more or less aggressive retraining. [1]

2.1.2 Gradient Descent

The Gradient descent Algorithm is one of the most popular Algorithms to perform optimizations on and is by far the most commen way to optimize neural networks.This is also the reason why nearly every state-of-the-art Deep learning library has various forms of implementations to perform optimization on the gradient descent. The Problem, however is, that these Algorithms often perform as some form of black-box, because a practical explanation of what their stengths and weaknesses is are hard to find and explain. Gradient descent is a way to minimize an objective function $J(\theta)$ parameterizedby a model's parameters $\theta \in \mathbb{R}^d$ by updating the parameters in the opposite directionof the gradient of the objective function $\nabla_\theta \in J(\theta)$ with respect to the parameters. The learning rate η determines the size of the steps wetake to reach a local minimum. In other words, we follow the direction of the slopeof the surface created by the objective function downhill until we reach a valley. [8]

The normal gradient descent, also known as batch-gradient descent, computes the gradient of the functionwith respect to the parameters θ for the entire training set:

$$\theta = \theta - \eta * \nabla_\theta J(\theta) \tag{2.1}$$

Since we need to calculate the gradients for the complete dataset to get just one update, batch gradient descent can get very slow and is intractable for dataset that do not fit into the memory. For this reason the batch gradient descent implementation can not be used in this comparison, as it is not able to perform updates "on-the-fly", as this is a necessarity for use on streaming Data.

In contrast to that, the Stocastic Gradient Descent (short: SGD) can perform a update for *each* training example $x^{(i)}$ and label $y^{(i)}$:

$$\theta = \theta - \eta * \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{2.2}$$

While the bacht gradient descent performed redudant computations for large datasets, as it recomputes gradients for similar examples before each update, the SGD puts away this redundacy by performing one update at a time. Because of that it performce much faster and can be used on Streaming Data. The SGD performs frequent updates with a high variance, that causes the objective function to fluctuate heavily.

This fluctuation enables the SGD to jump to new and potetially better local minima, while the batch gradient only converges to the minimum of the basin the parameters are placed in. But this also ultimatly complicates the convergensto the exact minimum, as

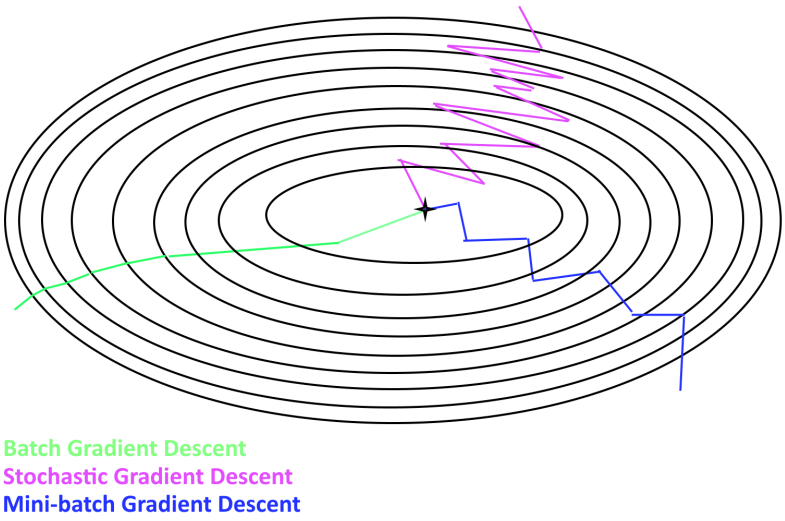

Figure 2.3: Visualisation differences of the three Gradient Descent Algorithms

the SGD keeps overshooting. This can be fixed by slowly decreasing the learning rate, as the SGD then shows the same convergeance behaviour as the batch gradient descent. It then almost certainly converges to a local or gloabl minimum for non-convex and convex optimization respectivly. [8] These advantages lead to the fact that the SGD is the most common Gradient descent algorithm.

Then there is also the Mini-batch gradient descent algorithm. It takes the best of both worlds and performs an update for every mini-batch on n training examples:

$$\theta = \theta - \eta * \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \tag{2.3}$$

This way it reduces the variance of the parameter updates, which can lead to a more stable convergence. It can also make use of highly optimized matrix optimizations common for state-of-the-art deep learning libraries, that makes computing the gradient with respect to a mini-batch very efficient. Common mini-batch sizes range between 50 and 256 , but can vary for different applications. Typically the Mini-batch gradient descent is the algorithm of choice when there is a neural network to train and the term SGD is usually also employed even when a mini-batches are used. [8]

2.2 Streaming Data

Data streams are an algorithmic abstraction to support real-time analytics. They are sequences of items, possibly infinite, each item having a timestamp, and so a temporal order. Data items arrive one by one, and we would like to build and maintain models, such as patterns or predictors, of these items in real time. There are two main algorithmic challenges when dealing with streaming data: the stream is large and fast, and we need to extract information in real time from it. That means that usually we need to accept approximate solutions in order to use less time and memory. Also, the data may be evolving, so our models have to adapt when there are changes in the data. [1]

2.2.1 Challenges with Streaming Data

2.3 Framework

2.3.1 SCIKIT Multiflow

2.4 Criteria for the Comparission

2.4.1 Performance

2.4.2 Accuracy

2.4.3 *kappa-errors

3 Used Algorithms

3.1 Theoretical Work

This chapter will try to give a short explanation about how the Robust Soft Learning Vector Quantization Algorithm (short: RSLVQ) works and will also explain the algorithm that it is based on. Furthermore will it explain what momentum based gradient descent algortihms there are and how they work. These are used in the different optimization implementation of the RSLVQ that will be compared in the later chapters.

3.2 Learning Vector Quantization

3.3 Robust Soft Learning Vector Quantization

3.4 Momentum Based Gradient Descent

The following section will outline three algorithms that are widely used in the Deep learning Community and that will later be used to optimize the RSLVQ Algorithm in hopes to improve its performence and accuracy.

3.4.1 Momentum

SGD has trouble navigating ravines, i.e. areas where the surface curves much more steeply in one dimension than in another [10, 9], which are common around local optima. In these scenarios, SGD oscillates across the slopes of the ravine while only making hesitant progress along the bottom towards the local optimum as in figure.

Figure 3.1: Differences of a SGD with and without momentum.[1]

Momentum [6, 9] is a method that helps accelerate SGD in the relevant direction and dampens oscillations as can be seen in figure . It does this by adding a fraction γ of the update vector of the past time step to the current update vector

$$\begin{aligned} v_t &= \gamma v_{t\text{-}1} + \eta\theta - \eta\nabla_\theta J(\theta) \\ \theta &= \theta - v_t \end{aligned} \tag{3.1}$$

The momentum term γ is usualy set to 0.9 or a similar value.

In his Article, Ruder [8] describs it like if we would push a ball down a hill. The ball accumulates momentum as long as it rolls downhill, becoming faster and faster on the way (until it reaches its terminal velocity, if there is air resistance, i.e. γ ¡ 1). The same thing happens to our parameter updates: The momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. As a result, we gain faster convergence and reduced oscillation.

3.4.2 Adadelta

One of the more often used momentum-based Algorithm is the Adadelta approach. It is an extension of the Adagrad [11] that seeks to reduce its aggressive, monotonically decreasing learning rate. Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients to some fixed size ω.

Rather than inefficently storeing ω previous squared gradients, the sum of gradients is recursively defined as a decaying average of all past squared gradients. The running average $E[g^2]_t$ at time step t then depends (as a fraction γ similarly to the Momentum

[1] Genevieve B. Orr

term) only on the previous average and the current gradient:

$$E[g^2]_t = \gamma E[g^2]_t - 1 + (1-\gamma) g_t^2 \tag{3.2}$$

We then set γ to a simliar value as the momentum, around 0.9. The SGD update will then be rewriten in terms of the parameter update vector $\Delta\theta_t$:

$$\begin{aligned} \Delta\theta_t &= -\eta \cdot g_{t,i} \\ \theta_{t+1} &= \theta_t - \Delta\theta_t \end{aligned} \tag{3.3}$$

Bacuase of this, the parameter update vector, that is base of the parameter update vecot of the Adagrad, takes the form of:

$$\Delta\theta_t = -\frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t \tag{3.4}$$

In the next step the diagnoal Matrix G_t gets replaced with the decaying average over past squared gradients $E[g^2]_t$:

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{3.5}$$

As the denominator is just the root mean squared (RMS) error criterion of the gradient, we can replace it with the criterion short-hand:

$$\Delta\theta_t = -\frac{\eta}{RMS[g]_t} g_t \tag{3.6}$$

As in [2] noted, the units in this update (as well as in the SGD, Momentum and Adagrad) do not match, i.e. the update should have the same hypothetical units as the parameter. To realize this, they first define another ecaying average, this time not of squared gradients but of squared parameter updates:

$$E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_t - 1 + (1-\gamma)\Delta\theta_t^2 \tag{3.7}$$

This leads to that the root mean squared error of the parameter updates is:

$$RMS[\Delta\theta]_t = \sqrt{E[\Delta\theta^2]_t + \epsilon} \tag{3.8}$$

Because the $RMS[\Delta\theta]_t$ value is unknown to us, we try to approximate it with the RMS of parameter updates until the previous time step. Replacing the learning rate η in the previous update rule with $RMS[\nabla\theta]_t - 1$ leads to the final update rule for the Adadelta:

$$\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t\text{-}1}}{RMS[g]_t} g_t \theta_{t+1} = \theta_t + \Delta\theta_t \tag{3.9}$$

In the Adadelta we do not need to set a learning rate η, eliminated from the update rule. [8]

3.4.3 RMSprop

3.4.4 Adam

3.5 Implementation of the RSLVQ variants

3.5.1 RSLVQ SGD

3.5.2 RSLVQ prop

3.5.3 RSLVQ adadelta

3.5.4 RSLVQ adam

4 Test realisation

4.1 Used Streaming Datat sets

4.1.1 Real world data

4.2 Implementation of the Comparission and Results

4.2.1 Experiment begrenzung

4.2.2 Experiment aufbau

4.2.3 Experiment durchlauf

Listing 4.1: Beispiel für einen Quelltext

```
1
2 public void foo() {
3     // Kommentar
4 }
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut vehicula felis lectus, nec aliquet arcu aliquam vitae. Quisque laoreet consequat ante, eget pretium quam hendrerit at. Pellentesque nec purus eget erat mattis varius. Nullam ut vulputate velit. Suspendisse in dui in eros iaculis tempus. Phasellus vel est arcu. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Integer elementum, nulla eu faucibus dignissim, orci justo imperdiet lorem, luctus consectetur orci orci a nunc.

Praesent at nunc nec tortor viverra viverra. Morbi in feugiat lectus. Vestibulum iaculis ipsum at eros viverra volutpat in id ipsum. Donec condimentum, ligula viverra pharetra tincidunt, nunc dui malesuada nisi, vitae mollis lacus massa quis velit. Integer feugiat

ipsum a volutpat scelerisque. Nulla facilisis augue nunc. Curabitur eget consectetur nulla. Integer accumsan sem non nisi tristique dictum.

Sed lacinia eu dolor sed congue. Ut dui orci, venenatis id interdum rhoncus, mattis elementum massa. Proin venenatis elementum purus ut rutrum. Phasellus sit amet enim porta, commodo mauris a, bibendum tortor. Nulla ut lobortis justo. Aenean auctor mi nec velit fermentum, quis ultricies odio viverra. Maecenas ultrices urna vel erat ornare, quis suscipit odio molestie. Donec vel dapibus orci, vel tincidunt orci.

Etiam vitae eros erat. Praesent nec accumsan turpis, et mollis eros. Praesent lacinia nulla at neque porta aliquam. Quisque elementum neque ac porta suscipit. Nulla volutpat luctus venenatis. Aliquam imperdiet suscipit pretium. Nunc feugiat lacinia aliquet. Mauris ut sapien nec risus porttitor bibendum. Aenean feugiat bibendum lectus, id mattis elit adipiscing at. Pellentesque interdum felis non risus iaculis euismod fermentum nec urna. Nullam lacinia suscipit erat ac ullamcorper. Sed vitae nulla posuere, posuere sem id, ultricies urna. Maecenas eros lorem, tempus non nulla vitae, ullamcorper egestas nibh. Vestibulum facilisis ante vel purus accumsan mattis. Donec molestie tempor eros, a gravida odio congue posuere.

Sed in tempus elit, sit amet suscipit quam. Ut suscipit dictum molestie. Etiam quis porta mauris. Cras dapibus sapien eget sem porta, ut congue sapien accumsan. Maecenas hendrerit lobortis mauris ut hendrerit. Suspendisse at aliquet est. Quisque eros est, scelerisque ac orci quis, placerat suscipit lorem. Phasellus rutrum enim non odio ullamcorper, sit amet auctor nulla fringilla. Nunc eleifend vulputate dui, a sollicitudin tellus venenatis non. Cras condimentum lorem at ultricies vestibulum. Vestibulum interdum lobortis commodo. Nullam rhoncus interdum massa, ut varius nisi scelerisque id. Nunc interdum quam in enim bibendum vulputate.

5 Result Evaluation

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut vehicula felis lectus, nec aliquet arcu aliquam vitae. Quisque laoreet consequat ante, eget pretium quam hendrerit at. Pellentesque nec purus eget erat mattis varius. Nullam ut vulputate velit. Suspendisse in dui in eros iaculis tempus. Phasellus vel est arcu. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Integer elementum, nulla eu faucibus dignissim, orci justo imperdiet lorem, luctus consectetur orci orci a nunc.

Praesent at nunc nec tortor viverra viverra. Morbi in feugiat lectus. Vestibulum iaculis ipsum at eros viverra volutpat in id ipsum. Donec condimentum, ligula viverra pharetra tincidunt, nunc dui malesuada nisi, vitae mollis lacus massa quis velit. Integer feugiat ipsum a volutpat scelerisque. Nulla facilisis augue nunc. Curabitur eget consectetur nulla. Integer accumsan sem non nisi tristique dictum.

Sed lacinia eu dolor sed congue. Ut dui orci, venenatis id interdum rhoncus, mattis elementum massa. Proin venenatis elementum purus ut rutrum. Phasellus sit amet enim porta, commodo mauris a, bibendum tortor. Nulla ut lobortis justo. Aenean auctor mi nec velit fermentum, quis ultricies odio viverra. Maecenas ultrices urna vel erat ornare, quis suscipit odio molestie. Donec vel dapibus orci, vel tincidunt orci.

Etiam vitae eros erat. Praesent nec accumsan turpis, et mollis eros. Praesent lacinia nulla at neque porta aliquam. Quisque elementum neque ac porta suscipit. Nulla volutpat luctus venenatis. Aliquam imperdiet suscipit pretium. Nunc feugiat lacinia aliquet. Mauris ut sapien nec risus porttitor bibendum. Aenean feugiat bibendum lectus, id mattis elit adipiscing at. Pellentesque interdum felis non risus iaculis euismod fermentum nec urna. Nullam lacinia suscipit erat ac ullamcorper. Sed vitae nulla posuere, posuere sem id, ultricies urna. Maecenas eros lorem, tempus non nulla vitae, ullamcorper egestas nibh. Vestibulum facilisis ante vel purus accumsan mattis. Donec molestie tempor eros, a gravida odio congue posuere.

Sed in tempus elit, sit amet suscipit quam. Ut suscipit dictum molestie. Etiam quis porta mauris. Cras dapibus sapien eget sem porta, ut congue sapien accumsan. Maecenas hendrerit lobortis mauris ut hendrerit. Suspendisse at aliquet est. Quisque eros est, scelerisque ac orci quis, placerat suscipit lorem. Phasellus rutrum enim non odio ullamcorper, sit amet auctor nulla fringilla. Nunc eleifend vulputate dui, a sollicitudin tellus venenatis non. Cras condimentum lorem at ultricies vestibulum. Vestibulum interdum

lobortis commodo. Nullam rhoncus interdum massa, ut varius nisi scelerisque id. Nunc interdum quam in enim bibendum vulputate.

6 Summary

7 Sources

List of Figures

List of Tables

Bibliography

[1] Albert Bifet et al. *Machine Learning for Data Streams with Practical Examples in MOA*. https://moa.cms.waikato.ac.nz/book/. MIT Press, 2018.

[2] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *J. Mach. Learn. Res.* 12.null (July 2011), pp. 2121–2159. ISSN: 1532-4435.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[4] Nils J. Nilsson. *Introduction to Machine Learning*. 1998.

[5] placeholder. *Introduction to Artifical Ineligence*. Springer, 2003.

[6] Ning Qian. "On the momentum term in gradient descent learning algorithms". In: *Neural Networks* 12.1 (1999), pp. 145–151. ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(98)00116-6. URL: http://www.sciencedirect.com/science/article/pii/S0893608098001166.

[7] Sebastian Raschka. *Python MAchine Learning*. Packt Publishing, 2015. ISBN: 1783555130.

[8] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2016.

[9] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *ArXiv* abs/1609.04747 (2016).

[10] Richard S. Sutton. *Two problems with backpropagation and other steepest-descent learning procedures for networks*. 1986.

[11] Matthew D. Zeiler. "ADADELTA: An Adaptive Learning Rate Method". In: *ArXiv* abs/1212.5701 (2012).

Statutory Declaration

Hiermit versichere ich, dass ich die vorgelegte Bachelorarbeit selbstständig verfasst und noch nicht anderweitig zu Prüfungszwecken vorgelegt habe. Alle benutzten Quellen und Hilfsmittel sind angegeben, wörtliche und sinngemäße Zitate wurden als solche gekennzeichnet.

Florian Hohn, am January 29, 2020