Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik

Bachelor Thesis Bioinformatics

Semi-supervised learning for nucleic acid cross-linking mass spectrometry

Emil Paulitz

14.08.2020

Reviewer

Prof. Oliver Kohlbacher
Department of Computer Science
University of Tübingen

Supervisor

Timo Sachsenberg
Address
University of Tübingen

Paulitz, Emilian Nicolaus Simons:
Semi-supervised learning for nucleic acid cross-linking mass spectrometry
Bachelor Thesis Bioinformatics
Eberhard Karls Universität Tübingen
Period: 14.04.2020-14.08.2020

Abstract

Write here your abstract.

Acknowledgements

Write here your acknowledgements.

Contents

List of Figures

List of Tables

List of Abbreviations

MS	Mass Spectrometry
LC	Liquid Chromatography
MS/MS	Tandem Mass Spectrometry
PSM	Peptide Spectrum Match
FDR	False Discovery Rate
ROC-curve	Receiver Operating Characteristic curve
SVM	Support Vector Machine
AUC	Area Under the Curve

Chapter 1

Introduction

- Motivation: Was zeichnet cross-links aus und warum probiere ich da Sachen?

1.1 Background

Proteomics is an interdisciplinary research field analyzing the composition, interaction and impacts of the proteome (the entirety of proteins) of single cells or up to a whole organism [6, 10]. In this thesis, research was done in a related field, focusing on peptides cross-linked with RNA. The chemical bond between cross-linked molecules has been artificially induced, for example using UV light [10]. Applying this to peptides and RNA could possibly give insight into their *in vivo* interactions, and may also allow conclusions about protein-DNA interaction.
For quantitatively characterizing the proteome of a sample, large scale measuring techniques are needed. Mostly, mass spectrometry (MS) is used, or more specifically, as for the data in this thesis, tandem mass spectrometry (MS/MS) combined with liquid chromatography (LC). In order to analyze the protein sample with MS, its complexity has to be reduced as much as possible, for example using LC [10]. As Han et al. [6] explains, the mass spectrometer then produces mass spectra, which have to be analyzed further. It does so by first ionizing the substrate, because it can only detect charged particles. Then, the sample is separated in the mass analyzer by the ratio $\frac{m}{z}$, mass of the particles to their charge. The detector then quantifies the amount of a particle in the sample. The result is a mass spectrum, as shown in figure 1.1.

Because mass alone does not give enough information about a peptide to determine its sequence, tandem mass spectrometry is often used to gather more detailed evidence. In this procedure, particles of similar $\frac{m}{z}$ ratio are selected for fragmentation in a collision cell after the first round of mass measurement [10]. In there, the substance collides with a gas to be broken down into smaller molecules. For proteins, fragmentation happens predominantly

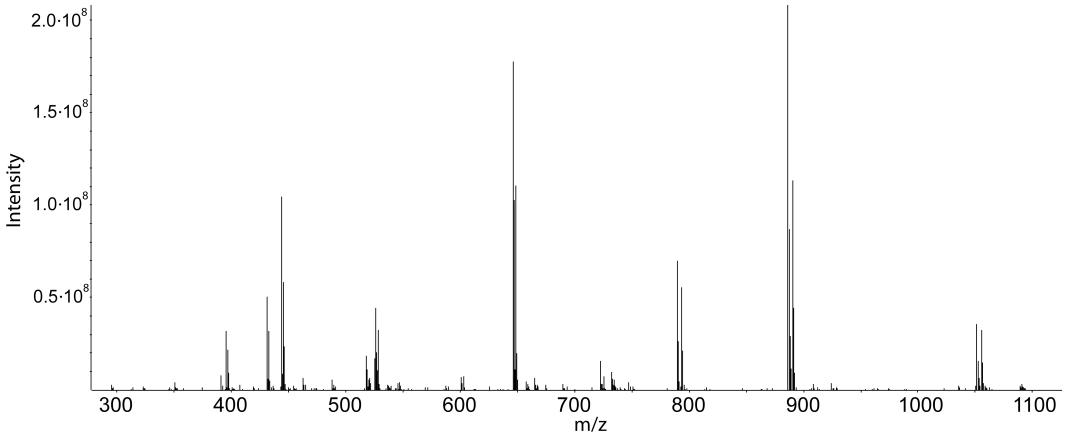


Figure 1.1: Example for a mass spectrum as recorded by a mass spectrometer. The ion intensity correlates with the amount of a molecule in the sample, $\frac{m}{z}$ is the mass-to-charge-ratio. From: Sachsenberg [10]

in their backbone, producing all possible sub-sequences of the peptide. This produces a spectrum that is almost unique for its protein, which allows for peptide identification using bioinformatics tools [2].
Algorithms like Sequest [4] or X! Tandem [3] compare the resulting spectra with theoretical spectra calculated from a list of possible peptides and compute a score based on their similarity. The peptides are generated by obtaining a list of proteins expected in the sample and calculating the peptides resulting from the, for example enzyme-based, degradation of the proteins. The best scoring peptide is then considered a peptide-spectrum-match (PSM). The scores produced by those algorithms often do not distinguish well enough between correct and incorrect matches [7], but they enable FDR estimation using decoy databases and serve as a basis for score re-calibration with the Percolator algorithm [7, 5].
Decoy databases are created from the target database, contain usually as many peptides [9, 8] in a reversed or shuffled order with respect to the amino acid sequence [1]. They are presented to the scoring algorithm either separately [5] or mixed with the target database [9]. It is assumed, that decoy and target peptides have similar features [8] and are not easily distinguishable by a scoring algorithm. When the actually fitting peptide for a given spectrum is not in the target database, and thus a wrong one will be chosen, the best scoring peptide will be a decoy approximately half of the time. This allows for an estimation of wrongly assigned targets, since the score distribution is assumed to be the same for decoys and false targets [1].
In practice, one estimates the probability of a PSM being a false target by counting the number of decoy-PSMs with the same or a higher score. It is then assumed, there are as many false targets and thus a false discovery rate

(FDR) can be estimated [5]. This leads to the following formula[1]:

$$FDR = \frac{\#\text{ false target PSMs}}{\#\text{ all target PSMs}} \approx \frac{\#\text{ decoy PSMs}}{\#\text{ all target PSMs}} \tag{1.1}$$

The q-value as a measure for a single PSM rather than a metric for a set of PSMs is then derived from this as the minimum FDR of all PSMs with a lower or equal score [5, 1]. It will be used for estimating the credibility for any one PSM.

As Käll et al. [7] say, separating correct from incorrect target PSMs with already mentioned algorithms works fine, but there is still room for improvement. This is because often not all information is used and considered jointly. Percolator [7, 5] tries to utilize as much information as possible by using scores from different algorithms, features of the peptide like its length, of the spectrum or the PSM itself. It joins them using a linear SVM and a semi-supervised approach with cross-validation to retain as many PSMs as possible. In every iteration, the top ranking, non-decoy PSMs up to a certain threshold of q-value are chosen as positive training examples, and the decoy PSMs are used as negative training set. The PSMs are then re-ranked using the SVM score, with the intend of getting a better separation of true and false PSMs. If that holds true, the positive training set of the next iteration better is of higher quality and the SVM can be trained even better. The algorithm usually converges within the first 10 iterations [7]. To avoid having to split the data into training and testing set and consequently losing possibly correct PSMs but also avoid overfitting, a nested cross-validation approach is being used [5].

- Nested CV näher erklären
- (Pseudo) ROC Curve noch erklären?
- AUC als Metrik
- SVM?

[1] In this thesis, the following approximation is used:
$FDR \approx \frac{\#\text{ decoy PSMs}}{\#\text{ all PSMs}} = \frac{\#\text{ decoy PSMs}}{\#\text{ decoy PSMs}+\#\text{ target PSMs}}$
It is faster to calculate and yields results differing by the FDR, so in the relevant range of FDRs of 0 to 5% up to 5%:
$\frac{\frac{\#\text{ decoys}}{\#\text{ targets}}}{\frac{\#\text{ decoys}}{\#\text{ decoys}+\#\text{ targets}}} = \frac{\#\text{ decoys}+\#\text{ targets}}{\#\text{ targets}} = 1 + \frac{\#\text{ decoys}}{\#\text{ targets}} \approx 1 + FDR$

Chapter 2

Material and Methods

- Material: Was ich für ein Datensatz zum Testen benutzt habe und wo der herkommt

2.1 Implementation of the percolator algorithm

- Wichtige Punkte wären:
- Verwendete Scipy-Methoden
- Abbruch wenn es nicht besser wird und dass ich die AUC als Metrik nutze
- feature normalization
- Wichtige Hilfsfunktionen (pseudoROC zB)
- Heißt jetzt Pycolator

2.2 Adapting Percolator to Cross-link Identification

To be able to monitor the difference any experiment makes, especially with respect to the cross-linked or non-cross-linked PSMs, following features were implemented:
First, in addition to the q-value, which is calculated as described in 1.1, the calculation of a class-specific q-value was implemented. This is done by splitting the dataset according to the class affiliation and calculating the q-value separately for both splits.

Secondly, a ROC curve using the *pseudoROC* function is calculated after every iteration of Percolator, for the whole dataset, only for cross-linked and only for non-cross-linked PSMs. Accordingly, the respective class-specific q-value is used. Thus, three plots containing the corresponding class(es) and

every iteration are shown. This allows for fast visual detection of the impact a specific change to the algorithm has on certain classes, iterations or general sensitivity.

2.2.1 Different Ranks

As experience shows, cross-linked peptides can be harder to detect than linear peptides. This means, the possibly correct cross-linked peptide will frequently not get the highest score of all the peptides. It thus can be beneficial to not only consider the highest scoring peptide, but also the highest scoring cross-linked peptide or also some lower-scoring peptides and assign them ranks. Then, as experiments showed, Percolator can correct the scores of some lower-ranking PSMs, possibly detecting more cross-linked PSMs. Meanwhile, it is known to the experimenter, that only one of the peptides can be the correct match to a spectrum, and thus any PSM with a lower rank than 1 should be excluded at the end.
To tackle both constraints, Pycolator first trains the SVM with every PSM available and re-calculates the ranks based upon the newly assigned score, possibly correcting the ranks of some PSMs. When the used metric, normally the area under the curve of the pseudo ROC, does not improve beyond a certain threshold per iteration, every PSM with rank 2+ is dropped. The threshold is controlled by the parameter `cutOffImprove` with a default of 0.01 corresponding to a 1% increase of the used metric per iteration. Since also some of the best scoring PSMs will be dropped, the algorithm then runs some more iterations in order to properly integrate and score the new PSMs considered confident.
The performance of this feature was tested against dropping the lower ranking PSMs once at the very end of the algorithm and once at the very beginning. Pycolator was run on the given dataset 2 and pseudo ROCs were plotted as described in 2.2.

2.2.2 Characteristics of Cross-linked PSMs

Apart from being hard to detect, cross-linked peptides also have other characteristics, some of which pose problems to the computational detection of correct PSMs. As discussed in 2.2.2, the features of cross-linked and linear peptides are so dissimilar, splitting the dataset and training a linear SVM separately on cross-links and non-cross-links yields significantly better results than training one linear SVM. To reduce the impact of this heterogeneity, the following experiments were conducted:

Proportions of Different Classes

As discussed in 1.1, Percolator employs a nested cross-validation approach, splitting the dataset, training on all parts than one and testing/scoring on the remaining part. If the splitting was uneven by chance, the SVM would be trained badly and the scoring inaccurate. Having different classes with significant differences in the dataset, like in our case cross-linked and linear PSMs or targets and decoys, increases this problem. For example, if there was a testing split with many cross-linked PSMs, the SVM had to be trained on the remaining data with few cross-linked PSMs, resulting in probably poor scoring of the many cross-linked PSMs in the test set. In the average case, this should not be a problem, but it can produce on occasion worse or better results, following from overfitting.
To solve this problem, a mechanism of maintaining the proportion of the classes in the whole dataset for every inner and outer split was implemented. It can be toggled for targets and decoys or cross-linked and non-cross-linked PSMs as well as for inner and outer split independently. The impact has been measured by running the algorithm 10 times, plotting and recording the results of the best, worst and median run w.r.t. to the AUC of the pseudo ROC curve. Because this took approximately 90 minutes, the test was performed in a Google Colaboratory notebook[1].

Imputation

Cross-linked PSMs naturally have features linear PSMs do not have, which can however be used for training the SVM. An example would be the nucleotide it was linked to. BESSERES BEISPIEL S. MAIL. In the dataset given 2, 16 of 61 features were only given for cross-linked PSMs. Optimally, these should not influence the score a non-cross-linked PSM gets. However, 0 was filled in for the missing values and because that is a valid value for the linear SVM, it biases the decision made. For example, if a high value in a feature leads the SVM to a decision against the PSM, 0 as the lowest value possible after feature normalization 2.1, will tell the SVM to give the PSM a higher score, based on an actually missing feature. However, the scikit-learn package provides solutions for this problem[2], and one of these, the `IterativeImputer`, was tested.

[1]The Google Colaboratory notebook used:
https://colab.research.google.com/drive/1VqZAmdta57YhgobA0WkQMqe_U9YIUnDl?usp=sharing
[2]https://scikit-learn.org/stable/modules/impute.html

Splitting the Dataset

- Trennung von Datensatz nach XL/nXL oder sogar cross-linking target falls Datensatz groß genug

2.2.3 Small datasets

- Ratio Testing (nicht-random aus ganzem Datensatz und random aus Top 10%. Liefert Erkenntnisse über die mögliche Größe des Datensatzes und eventuell die Sinnhaftigkeit, wann man die Datensätze einfach trennen kann → Für den Leser relevant)
- Einbau von Identifikationen bei 1% FDR als Metrik (Sinnhaftigkeit kann man ja diskutieren)

(- Performance auf anderem Datensatz
- Vergleich mit Entrapment FDR)

Chapter 3

Results

3.1 Implementation of the percolator algorithm

- Reimplementierung funktioniert wie Original
- feature normalization war wichtiger boost
- ROC nach jeder Iteration zeigen

3.2 Adapting Percolator to Cross-link Identification

- Beispiel für ROCs nach jeder Iteration und wie die zu lesen sind (für alle, XLs und nXLs)

3.2.1 Different Ranks

Figure 3.1 shows the pseudo ROCs of Pycolator before the implementation of the new mechanism. 3.1a was plotted when Pycolator used all PSMs available, meaning all ranks, and only at the end lower ranking PSMs than 1 were excluded. 3.1b is the result of running Pycolator only with rank 1 PSMs of the given dataset 2 and 3.1c shows the final result with the new feature.
As one can see, without the new mechanism Pycolator gradually improves the area under the curve and takes 5 and 4 iterations to roughly converge when given every PSM or only rank 1 PSMs to train respectively. This can be seen by the AUC results after every iteration or the ROC curves themselves. The end result, however, is better when removing the lower ranking PSMs right at the beginning (AUC of 345.79) instead of in the end (AUC of 343.21 after dropping the lower ranking PSMs). Since in the latter case Pycolator ended

with an AUC of 343.79 as one can see in 3.1a, dropping the PSMs slightly worsened the end result. With the new feature, the algorithm takes about 5 iterations to converge, then performs a jump and again improves over two iterations. The end result, an AUC of 348.13, is higher than both alternatives. Multiple runs yielded very similar results.

3.2.2 Characteristics of Cross-linked PSMs

Proportions of Different Classes

- Verhältnis Targets:Decoys und XL:non-XL verringt die Streuung: MinMax-Median Auswertungen

Imputation

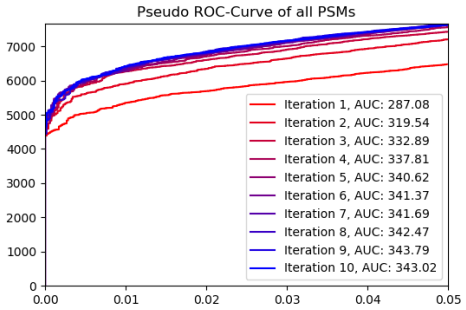
- Bei Imputation kam nichts heraus

Splitting the Dataset

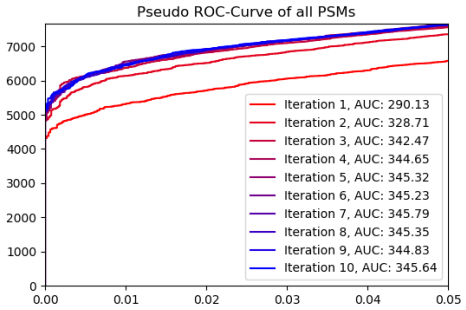
- Großer Unterschied wenn man den (großen) Datensatz nach XL/nXL oder sogar cross-linking target aufteilt

3.2.3 Small datasets

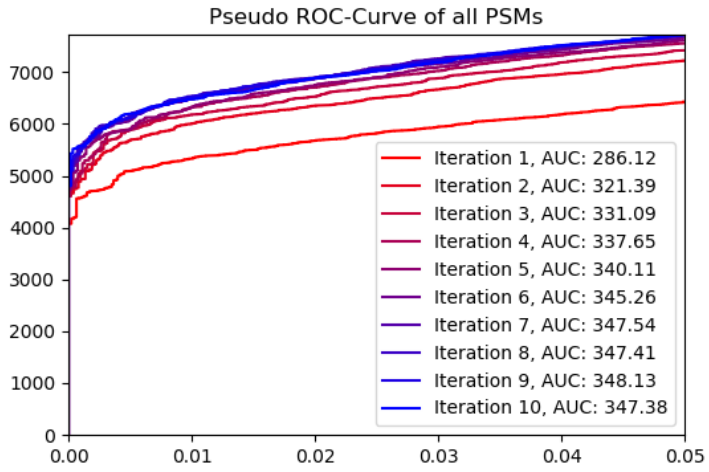
- Sinnvolle Plots zu Ratio Testing
- Neue Metrik erlaubt es der Implementierung, auch auf kleineren Datensätzen zu funktionieren



(a) The resulting pseudo ROCs (explained in 3.2) if Pycolator is given every PSM regardless of its rank. Lower ranking PSMs are only dropped in the end, which results in a final AUC of 343.21.



(b) The resulting pseudo ROCs (explained in 3.2) if Pycolator is given only the top scoring peptide for every spectrum. Lower ranking PSMs are dropped before running the algorithm.



(c) The resulting pseudo ROCs (explained in 3.2) if Pycolator is run with the newly implemented feature. It first trains with every PSM available, and after the results converge, lower ranking PSMs are dropped. Then, the algorithm runs for some more iterations.

Figure 3.1: Results of running Pycolator with different strategies as to how lower ranking PSMs are dealt with.

Chapter 4

Discussion and Outlook

As already explained in 2.2.1, cross-linked peptides often are harder to score than linear ones. Therefore, they can get a lower score than appropriate and it can be useful to also include the best scoring cross-linked peptide when running the Percolator algorithm. It may be able to revise the PSMs scores and the actually correct cross-linked peptide may become the best scoring PSM. This thesis is also supported by the findings in 3.2.1. If only including the best scoring PSM, of which 3.1b shows the results, the end result is worse than when giving Pycolator some iterations to re-rank the found PSMs. However, it was better than when giving Pycolator all of the PSMs available, which is unexpected, since giving a machine learning model more information should generally improve its learning. Apparently, the lower ranking PSMs, even when having such a high score a q-value of $\leq 5\%$ is estimated, contain misleading information and thus the SVM learns patterns not valid for correct PSMs. This also explains why the algorithm converges faster when only given rank 1 PSMs. The higher quality of data lets the SVM learn the correct patterns after fewer iterations.
The pseudo ROC generated when using the newly implemented mechanism (3.1c) shows a convergence after 5 iterations, just like when using every PSM (3.1a). Then, as the log shows, lower ranking PSMs are dropped and the next iteration has a much higher AUC, probably as a result of the better quality of the PSMs. Letting the algorithm run on the new dataset again improves the AUC even beyond that of Pycolator when only using rank 1 PSMs. This suggests, that indeed a re-ranking takes place in the first half of iterations.
Comparing the AUC of Pycolator when run with the new mechanism (3.1c) after iteration 6 (345.26) with the end result of running Pycolator with every PSM available (3.1a, 343.21), yields the following insight: Dropping all PSMs with rank 2+ yields a worse result when the Percolator algorithm has been running for 10 than in the case of 6 iterations. This implies an overfitting onto the PSMs with lower quality and thus a worse scoring.

- Methoden hinterfragen oder begründen, Ergebnisse interpretieren, Anwendbarkeit diskutieren, z.B.:
- Falsche Formel für q-value
- C Parameter für jeden split neu optimieren führt zu overfitting? → Original-Algorithmus macht es auch so
- Wie sinnvoll ist die neue Metrik (idents bei 1%)?
- ScanNr Versuche: Gleiche Spektren (identifiziert anhand der ScanNr.) auf verschiedene splits verteilen verändert nichts, d.h. vermutlich sind die niedrigeren Ränge dann so schlecht, dass es nichts bringt die schonmal gesehen zu haben.
- Peptide Versuche: Schlechtere Ergebnisse, aber vllt ehrlicher?
- Mögliche weiterführende Experimente: mächtigere Klassifikatoren + monotonic constraints (wie von Timo ausprobiert), Ada-Boosting, feature selection

Bibliography

[1] Suruchi Aggarwal and Amit Kumar Yadav. False discovery rate estimation in proteomics. In *Methods in Molecular Biology*, pages 119–128. Springer New York, 2016. doi: 10.1007/978-1-4939-3106-4_7. URL https://doi.org/10.1007/978-1-4939-3106-4_7.

[2] Thomas E. Angel, Uma K. Aryal, Shawna M. Hengel, Erin S. Baker, Ryan T. Kelly, Errol W. Robinson, and Richard D. Smith. Mass spectrometry-based proteomics: existing capabilities and future directions. *Chemical Society Reviews*, 41(10):3912, 2012. doi: 10.1039/c2cs15331a. URL https://doi.org/10.1039/c2cs15331a.

[3] R. Craig and R. C. Beavis. TANDEM: matching proteins with tandem mass spectra. *Bioinformatics*, 20(9):1466–1467, February 2004. doi: 10.1093/bioinformatics/bth092. URL https://doi.org/10.1093/bioinformatics/bth092.

[4] Jimmy K. Eng, Ashley L. McCormack, and John R. Yates. An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *Journal of the American Society for Mass Spectrometry*, 1994.

[5] Viktor Granholm, William Noble, and Lukas Käll. A cross-validation scheme for machine learning algorithms in shotgun proteomics. *BMC Bioinformatics*, 13(Suppl 16):S3, 2012. doi: 10.1186/1471-2105-13-s16-s3. URL https://doi.org/10.1186/1471-2105-13-s16-s3.

[6] Xuemei Han, Aaron Aslanian, and John R Yates. Mass spectrometry for proteomics. *Current Opinion in Chemical Biology*, 12(5):483–490, October 2008. doi: 10.1016/j.cbpa.2008.07.024. URL https://doi.org/10.1016/j.cbpa.2008.07.024.

[7] Lukas Käll, Jesse D Canterbury, Jason Weston, William Stafford Noble, and Michael J MacCoss. Semi-supervised learning for peptide identification from shotgun proteomics datasets. *Nature Methods*, 4(11):923–925, October 2007. doi: 10.1038/nmeth1113. URL https://doi.org/10.1038/nmeth1113.

[8] Roger E. Moore, Mary K. Young, and Terry D. Lee. Qscore: An algorithm for evaluating SEQUEST database search results. *Journal of the American Society for Mass Spectrometry*, 13(4):378–386, April 2002. doi: 10.1016/s1044-0305(02)00352-5. URL `https://doi.org/10.1016/s1044-0305(02)00352-5`.

[9] Junmin Peng, Joshua E. Elias, Carson C. Thoreen, Larry J. Licklider, and Steven P. Gygi. Evaluation of multidimensional chromatography coupled with tandem mass spectrometry (LC/LC-MS/MS) for large-scale protein analysis: the yeast proteome. *Journal of Proteome Research*, 2(1):43–50, February 2003. doi: 10.1021/pr025556v. URL `https://doi.org/10.1021/pr025556v`.

[10] Timo Sachsenberg. Computational methods for mass spectrometry-based study of protein-rna or protein-dna complexes and quantitative metaproteomics. 2017. URL `http://hdl.handle.net/10900/83311`.

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Bachelorarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Ort, Datum Unterschrift