MASTER'S THESIS | LUND UNIVERSITY 2016

Peak Detection in Data Independent Acquisition Analysis

Viktor Stymne

Department of Computer Science
Faculty of Engineering LTH

ISSN 1650-2884
LU-CS-EX 2016-20

Peak Detection in Data Independent Acquisition Analysis

Viktor Stymne
ada09vst@student.lu.se

June 17, 2016

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisors: Pierre Nugues, pierre.nugues@cs.lth.se
Johan Teleman, johan.teleman@immun.lth.se
Fredrik Levander, fredrik.levander@immun.lth.se

Examiner: Jacek Malec, jacek.malec@cs.lth.se

Abstract

The analysis of peptides using mass spectrometry produces signals, where intact and fragmented peptides are observable as specific signal peaks.

This report investigates how signal processing and supervised learning, along with specialized peak detection algorithms, can improve the analysis of such peptide signals in a mass spectrometry technique called Data Independent Acquisition. To make this improvement possible, 1120 different possible approaches were investigated and evaluated, and finally combined into an ensemble system.

In this ensemble system, the best fit approach for data currently being processed is chosen based on linear regression. The final system can identify all the correct peaks from a manually-annotated test set.

Keywords: proteomics, signal processing, machine learning, peak detection

Acknowledgements

We want to thank Johan Teleman, Pierre Nugues, Johan Malmström and Fredrik Levander for being understanding, for their support and for their help. A special thanks to Johan Teleman who has been an incredible source of help throughout the whole project.

Contents

Chapter 1
Introduction

1.1 Background

For its ability to repeatedly quantify 10,000s of peptides (parts of proteins) per sample injection, data independent acquisition (DIA) mass spectrometry is growing in popularity for highly resolved studies of the protein content of samples, a field commonly called proteomics (Hu et al., 2016).

In short, mass spectrometry proteomics using DIA works in the following way:

- Proteins are digested down into peptides using a specific endoprotease.
- These peptides are then separated, mainly based on their hydrophobicity through liquid chromatography. The sample is first injected onto the chromatographic column in a solution containing a high percentage of water. The percentage water decreases over time as the level of an organic solvent is gradually increased, and the peptides elute from the chromatographic column at certain times depending on the peptide properties, at their so called retention time. This value can help to identify which peptide that is being analyzed at a specific time point.
- The peptides are then ionized and analyzed by the mass spectrometer where the peptide ions are separated by their $\frac{m}{z}$ value, where m is the mass and z is the charge.

This gives us a list of spectra, where each spectrum shows the ion signal intensity as a function of $\frac{m}{z}$ in a certain retention time window.

Ideally, each peptide should give rise to one mass spectrometry signal peak. One mass spectrometry signal peak, defined by hydrophobicity (retention time) and $\frac{m}{z}$, is however not enough to uniquely determine the originating peptide since different peptides can give rise to peaks that are indistinguishably close to each other. To add more information, the peptides can be fragmented into fragments in the mass spectrometer, generating tandem

mass spectra (MS/MS). The fragment peaks generated in the MS/MS spectra can be coupled to the peaks from the intact peaks as they will appear at the same retention time as the peak from the intact peptide. The fragment ions are henceforth known as belonging to the peptide or being from the peptide. The collection of MS/MS thus gives a separate list of spectra containing the ion signal as a function of $\frac{m}{z}$ at certain retention time window. This information is enough to uniquely identify the peptides. For the analysis of DIA data, the spectrum format is transformed to the chromatogram format, where the ion signal intensities are stored as functions of retention time at certain $\frac{m}{z}$ windows.

1.2 Structure

The analysis tools of a DIA analysis are not yet fully developed and there are parts of the analysis pipeline that most likely can be optimized for better and more stable results. The pipeline of a DIA and its analysis consists of several steps quickly described below.

- Data acquisition
- Peak detection
- Scoring
- Semi supervised learning (Teleman et al., 2015)

The detection of certain wanted peaks is the first step of the analysis as they will be processed further down the pipeline. By having a better result in this step, the following steps will be faster and also more exact. Note that the supervised learning mentioned here is unrelated to the one discussed later in this report.

1.3 Problem definition

In order to quantify the peptides through mass spectrometry one matches the $\frac{m}{z}$ value where the ion signal peak occurred to a database where the peptide can be identified. This means that in the data obtained, described in Sect. 1.1, we want to detect a certain peak from each peptide in the solution. This certain peak is called the correct peak in this report. In order to match the peak to the database, we need to have unique $\frac{m}{z}$ values. Since the first list of spectra generated had overlapping peaks we need to look at the data generated by the fragments instead. The ion signal peak from the fragments, from the same peptide, should occur at the same retention time. The task is to detect each of these peaks from the different fragments. The problem is that the data contained noise and that the measurement tools are not exact. Figs. 1.1 and 1.2 show examples of the data. Note that both data examples are extracted from a low noise file.

The goal of this thesis is to find an algorithm which produces a result with a higher finding rate of peaks and fewer false peaks than the current solution, which is mentioned in Sect. 1.4. If this was not possible, we would try to identify the reason and address it.

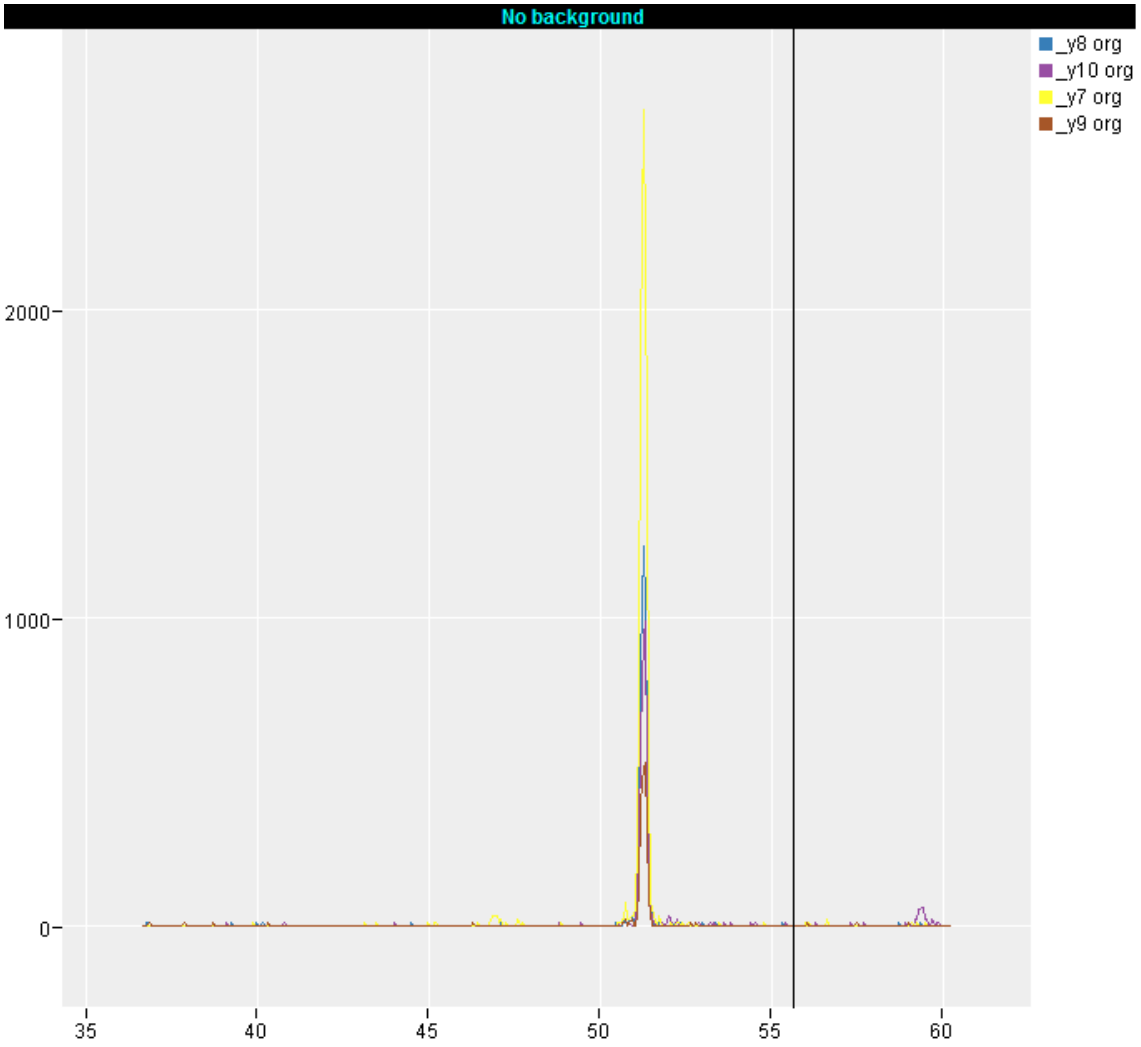

Figure 1.1: Example data. The picture shows intensities as a function of the retention time of four fragments belonging to one peptide. The retention time interval 37-60 min. The data contains low noise. The correct peak is clear at 52 min. because it is a high intensity spike occurring in all four fragments at the same retention time.

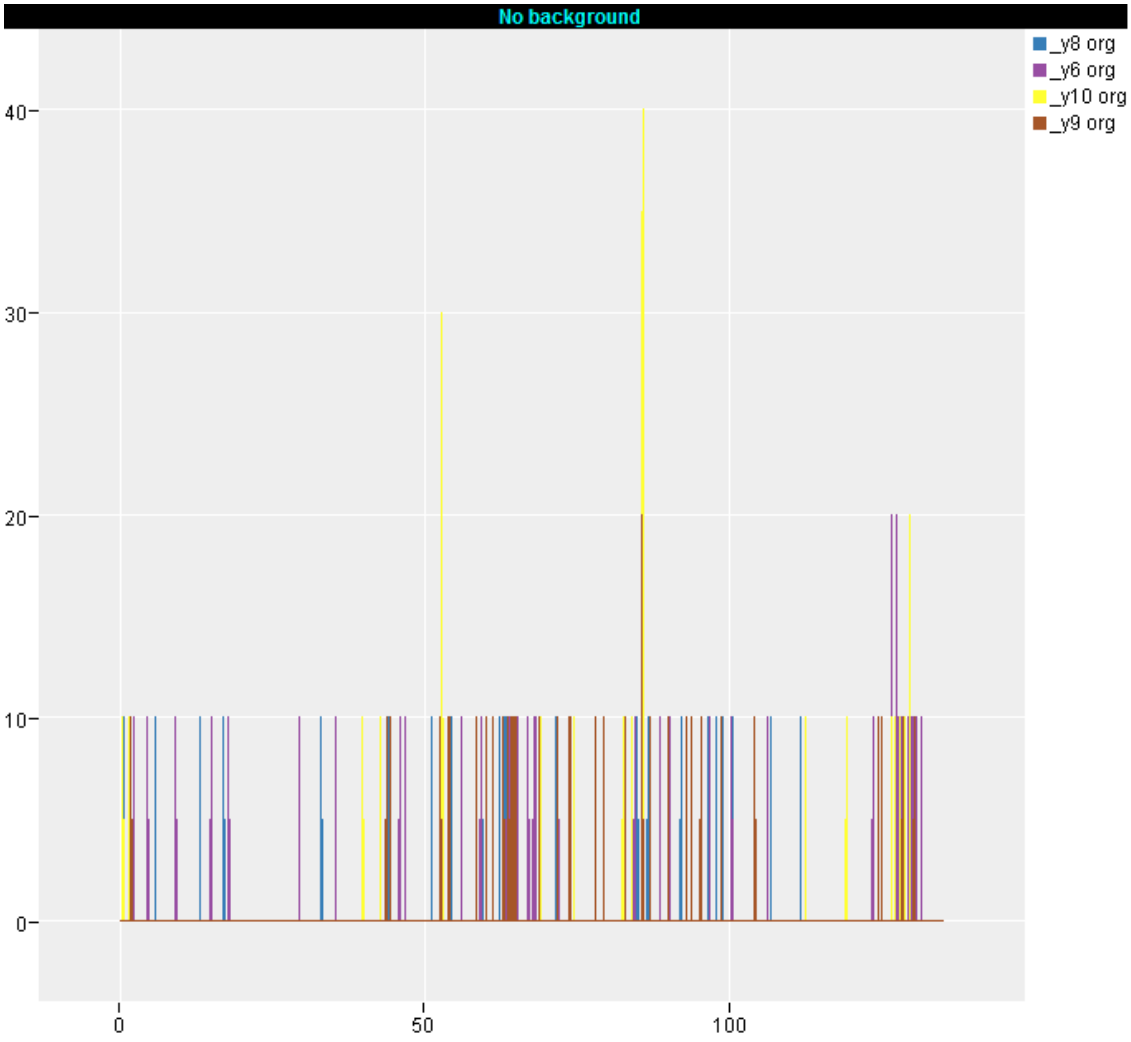

Figure 1.2: Example data. The picture shows intensities as a function of the retention time of four fragments belonging to one peptide. The retention time interval is 0-140 min. The data contains some noise. The correct peak is unclear as no high intensity spike occurs at the same retention time for all four fragments.

1.4 Related work

A related project called DIANA (Teleman et al., 2015) implemented all the steps of the DIA analysis including the peak detection step. The method used consists of a simplified derivative calculation on a wavelet filtered signal to detect the peaks through minimums and maximums. This method is similar to one of the implementations described in this thesis. DIANAs focus lies on the whole process and is likely therefore too broad to maximize the performance of the peak detection step.

There are more articles related to the subject of data processing for DIA mass spectrometry, for example:

- OpenSWATH (Röst et al., 2014), which is a software for automated DIA analysis.
- mProphet (Reiter et al., 2011), which is a software for automated selected reaction monitoring. Selected reaction monitoring is another method for identifying peptides.
- Anubis (Teleman et al., 2012), which is an algorithm for selected reaction monitoring analysis.
- DIA-umpire (Tsou et al., 2015) is an open-source software for DIA data that detects precursor and fragment chromatographic features and assembles them into pseudo–tandem MS spectra.

But none of these tools is specifically dedicated to optimizing peak detection. This means that, to the best of my knowledge, the focus of this thesis has not been addressed by any previous work on the subject.

Chapter 2
Approach

To identify the peaks, we split the problem up into several smaller steps. The first step was processing the signal to make it smoother. The second was implementing an algorithm for detecting data peaks in the signal. The third was figuring out a method to group the detected peaks of different fragments together. This chapter explains what methods we used implemented and how we later evaluated our results. The last section is about supervised learning, which is used to learn how to best combine the three methods.

2.1 Data set

To develop our algorithms we had a data set consisting of three files. Each file contained data from the same sample but in different backgrounds. This means that the files contained the same peptides but the amount of noise, and possibly phase shift, varied in all three files.

In each of the three files we tried to find the same peptides in order to see how the same parameters performed on different levels of noise. For each of the three files we had manually annotated results which we used to evaluate our results.

2.2 Signal processing

Since the data obtained by the mass spectrometry contained noise as mentioned in Sect. 1.3, we concluded that some signal processing method would be needed. Normal Fourier transforms were not an option because our data varies in frequency depending on time. Windowed Fourier transforms could have been used but seemed unnecessarily complex. To solve the problem, we followed the approach of Vidakovic and Mueller (1991) and used wavelet filters since they can be localized in both frequency and time. As the data did not have any clear structure, it was hard to decide for one single approach, and therefore we

tried multiple different approaches to both see which had the best performance but also to evaluate them against each other.

Furthermore we included a simple sliding window smoothing to start with a simple solution to see if signal processing had any impact. The sliding window filter we chose to include was a Savitzky-Golay filter. We also ran the filtered data through the same filter different number of repetitions for stronger smoothing. Finally we also included unfiltered data as a solution, both for comparison to the filtered data and because it possibly could be a good solution in the data with less noise.

2.2.1 The Savitzky-Golay Filter

A Savitzky-Golay filter (Savitzky and Golay, 1964) is a simple filter used to increase the signal-to-noise ratio without greatly distorting the signal. To use it, one creates successive subsets of adjacent data points and then fits them to a low-degree polynomial by the method of linear least squares. Since the data points are equally spaced, there is a single set of polynomial coefficients that can be applied to all data subsets to calculate the smoothed signal.

We chose to use the subset size of nine as it seemed enough to properly reduce the short noise spikes in the data. There are different windows where one can not create the full set. These windows occur in the $\frac{n}{2} - 1$ first points and $\frac{n}{2} - 1$ last points where n is the size of the subset (in our case nine). The reason the windows can not be created in these points is because the window goes out of bounds of the data set. There are different approaches to handle the data points where one can not create the subset to calculate the averages. In these start and end points, we chose to simply keep the data points as they are. Coefficients were chosen according to the quartic/quintic filter in the original article. See Savitzky and Golay (1964).

2.2.2 The Haar Wavelet

As mentioned earlier, we decided that the wavelet filters probably were the best approach. The Haar wavelet is the most simple one, and a natural starting point. The Haar wavelet takes two subsequent numbers from the data and creates two other numbers where the first is the average and the second is the difference between the average and the original numbers (Haar, 1910). The Haar wavelet can be used as noise reduction method as one can nullify the second number if it is under a certain threshold, thus reducing lower amplitudes of noise but preserving the original structure of the data.

The two numbers created by the Haar wavelet can be used to recreate the original signal, with or without the noise reduction, the first by subtracting the difference from the average and the second by adding the difference to the average. The Haar wavelet is not continuous and therefore not differentiable but in our application this does not make any difference. The Haar wavelet could be considered a Daubechies wavelet but we chose to treat it as a separate wavelet since there were different creators.

2.2.3 The Daubechies Wavelets

The Daubechies wavelets are discrete wavelet transforms. Each wavelet consists of a scaling function and a wavelet function (Daubechies, 1992). The principle is the same as in the Haar wavelet and one of the reasons the Haar wavelet can be treated as a Daubechies wavelet. The scaling function will contain the overall structure of the data while the wavelet function will contain the trends and the noise of the data. The noise can then be nullified by removing any data in the wavelet function under a certain threshold.

One of the main differences from the Haar wavelet is that the Daubechies wavelet uses more vanishing points and are more complex, and might therefore give a better smoothing. The Daubechies wavelets are furthermore continuous and therefore differentiable, but once again this makes no difference in our use of it. There are different Daubechies wavelets. The wavelets we used were Daubechies with two vanishing moments (db2), Daubechies with three vanishing moments (db3), and Daubechies with eight vanishing moments (db8). Coefficients used were obtained from Getreuer (2006).

2.2.4 Multiwavelet

By using multiple wavelets, one can create another effective smoothing method (Brittain et al., 2007). By using more than one wavelet, one can get more freedom in the construction of the wavelet, and thus possibly more advantages than the scalar wavelets. Features such as short support, orthogonality, symmetry, and vanishing moments are known to be important in signal processing, and a scalar wavelet can not possess all these properties at the same time (Strela et al., 1999).

We have chosen to try two different multiwavelet filters. The first filter is a combination of the db2 and db8 wavelets where we use the smoothing of both wavelets by cascading the data from the scalar function from the first wavelet forward to the second wavelet. The reconstruction is then generated in the same way as in the single wavelet case. The second multiwavelet filter we created uses Haar and db2 and works in the same manner as the one previously described (Keinert, 2004; Balambigai Subramanian and Rangasamy, 2014).

2.3 Peak detection

In order to choose only one optimal algorithm to detect the peaks in the data, the different data series of the different fragments would have to be consistent (i.e. the data would have to have peaks of similar structure, peaks of similar amplitude, and similar level of noise). Since the series are not consistent, but varies in different peptides and fragments we chose several different approaches.

Even if we construct an algorithm that is optimal for noise-free data, it is not guaranteed to have the best result overall because of the noise in the data. This means that it is impossible to theoretically construct an optimal algorithm even for the data with similar structure, and it is therefore needed to use different approaches and test which algorithm that is the best. Another problem in this step is also having an algorithm which finds as many of the correct peaks as possible but minimizes the amount of false peaks. This is to avoid expensive calculations further down the pipeline of the DIA-analysis.

2.3.1 Single derivative

One of the algorithms constructed uses simplified derivative calculations to find minimum and maximum points in the data series. A minimum point or a maximum point will be found if the value of the derivative of the data series changes sign from one point to the next. To decide if the point is a minimum or a maximum, we look at the second derivatives sign.

A peak will be constructed once the combination minimum-maximum-minimum has been detected. This peak will be deemed suitable if the peaks amplitude is high enough (i.e. if the difference between the maximum and the higher minimum point exceeds a threshold) and if the second minimum point's value is only slightly higher than the first minimum point's value (i.e. if the difference between the second minimum and the first minimum is lower than another threshold). The reasons for these threshold checks are to avoid choosing noise as peaks and also to adapt if the data changes baseline.

2.3.2 Template matching

We figured that if the peaks we are looking for follow the same shape in the different data series, a template matching might be effective at detecting the considered peaks (Shin et al., 2008). The real advantage would be if only the correct peaks followed this shape as this method then would have a very high precision.

First we sampled a few correct peaks manually and we computed the average of these samples to create a template. This template is then slided over the data and a correlation for each frame is calculated. If this correlation is over a threshold the peak will be counted, and otherwise discarded (Krasteva and Jekova, 2007). The correlation limits we used were: 0.49, 0.74, 0.95 and 0.99.

2.3.3 Multidetection

We combined the peak detection and the grouping of peaks (described in Sect. 2.4) to try to reach high precision.

In this approach, we used data series from multiple fragments belonging to one peptide in order to simultaneously detect peaks in the considered series using the above described derivative algorithm but only keep the result if several data series had overlapping peaks. Otherwise we discarded the peaks.

2.4 Grouping

The peaks we are looking for should occur in multiple fragment data series belonging to one peptide at the same retention time. One way to filter out noise is to only use peaks that actually occur in multiple fragments at the same time.

As data might be distorted by noise and also because the data-generation is not completely consistent, peaks detected in data series belonging together might not be obvious to match together. There could exist some phase shift or distorted shape of the peak which

led us to try a couple of different approaches. We discarded detected peaks that could not be grouped as false peaks.

2.4.1 Brute force grouping

The first approach we chose was the most simple, brute force comparing all the peaks in the fragments belonging to the same peptide. We compared a peak from one fragment to all the peaks of the other fragments belonging to the same peptide. We grouped them together if they overlapped in any of the four following ways:

$$\begin{aligned} P1min &< P2min < P1max \\ P2min &< P1min < P2max \\ P1min &< P2max < P1max \\ P2min &< P1max < P2max. \end{aligned}$$

Where min is the starting point's retention time and max is the ending point's retention time.

A group is only chosen if it consists of at least two different fragments, but can consist of the same number as there are fragments belonging to the peptide, otherwise we discard the peak.

2.4.2 Sorted grouping

The second approach was sorting all the peaks from all the fragments belonging to the same peptide together. We sorted the peaks on how low their starting points retention time was. We then checked if the peak following the previous one overlapped (if the starting point's retention time of the second point was lower than the ending point's retention time of the first). Only one peak per fragment was allowed in this method. Each grouping stopped when either of these condition were not met.

We expected this to yield the same result as the brute force approach but since everything in sorted grouping is checked in chronological order and that peaks in this are only allowed in one group, the groups generated actually differ. We discarded peaks that did not overlap with any other.

2.4.3 Multi-grouping

As explained in the peak detection section, one of the approaches was to use peak detection at multiple data series belonging to different fragments at the same time. The obvious way to group peaks together was then used, which is to group the peaks which were detected simultaneously together.

2.5 Supervised learning

The smoothing, peak detection, and grouping steps have different parameters. Choosing and combining these parameters is essential to the performance of the analysis. In order

to choose the best parameter combination we had a couple of different ideas. The first was to use binary predictors trained on a data set consisting of generated results compared to manually annotated results, which then would choose which parameter combination that was most likely to give the correct peak.

The second idea was to use regression to calculate which parameter combination choice yielded the highest precision. And the last idea was to combine both, to use both a predictor to choose with parameter combination that had a high chance of finding the correct peak and then using the regression to get as high precision as possible (Caruana and Niculescu-Mizil, 2006; Loh, 2011; Weisberg, 2005).

We chose to extract features for both single fragments and for the peptides. We chose features based on what we thought might be relevant variations in the data. The different features are listed below;

- Average value
- Variance
- Median value
- Maximum value
- Number of sign changes in the derivative
- Low-pass filtered average
- Average value ignoring values under a threshold

First we focused on using features from whole peptides as this were the results we were interested in from the start, and therefore did not need to alter the focus of the peak detection. Second we would alter the peak detections focus in order to provide us with data from single fragments in order to further improve the supervised learning if necessary. For reasons explained in Sect. 3.4 we only implemented the regression.

2.6 Implementation

The system is designed as a client-server architecture, where the server hosts the data collected from the mass spectrometry. We implemented the client that reads the data from the server and runs the analysis described in this chapter. We run the whole system through a bash script in order to try all the 1,120 parameter combinations on each of the three files. We implemented this client in Scala and the supervised learning in a Python script. This script calculates the regression parameters, which are then used by the Scala implementation.

The bash script and the Python script were run on Ubuntu 14.04.2, while the Scala implementation was run on Ubuntu 14.04.2 and Windows 8. The Ubuntu computer had 165 GB RAM while the Windows computer had 12 GB.

Code of implementation available at: `https://github.com/ViktorSt/panther/tree/master/panther/src/main/scala/se/lth/immun`

Chapter 3
Evaluation

To evaluate the data, we compared our results to the previously mentioned manually annotated results consisting of the inital data set and and an unseen test set. The unseen test set consisted of three data files with different noise levels in each. Our detected peaks were considered to be a correct peak if the retention time of our peaks apex's retention time occurred ± 1 minutes of the manually annotated results apex's retention time and if the area under the detected peak were within a factor of ten of the manually annotated results area.

The reason for this interval is both to handle human error in the annotated results and to accept slight distortions of the peak from the signal processing step. We evaluated our implementations against the three initial data sets with different level of noise. In our visualization, we kept the three different data sets separated to see if we could find differentiating trends in the different noise levels. In the analysis process, we focus on one parameter and we vary it. The other parameters may also have an influence on the result that can be greater than the varied parameter. This means that our conclusions may not be 100 percent accurate in each of the following sections.

In the results, we evaluated on two parameters, a percentage of how many of the manually annotated peaks of the test data we found (recall) and a percentage of how many of the automatically detected peaks that were correct peaks (precision). As mentioned previously, we define a correct peak as the peak that can help identify the peptide. The focus was on getting the recall as high as possible, preferably 100 percent, while getting as high precision as possible as well. In this evaluation, we always valued recall before precision.

Every parameter combination was run in an attempt to isolate single parameter choices that performed better than others. This resulted in three different tables, one for each level of noise in the data files, each consisting of 1120 rows. To visualize our results we used a R-script to produce graphs showing how different parameter-combinations performed in both recall and precision, and how well they performed compared to each other. We were able to obtain multiple parameter-combinations that yielded 100 percent recall with varying precision. These are the ones we consider the best results. Table 3.1 shows examples of the data obtained.

Table 3.1: Parameter settings and corresponding results. The File column marks which of the three test files it was tested on. The Min and Max columns are the peak detection boundaries for the derivative detection method. The Group method column describes what grouping method that was used. The Detection method column describes what peak detection method that was used. The Corr limit column describes what correlation limit that was used for the template method. The Filter column describes what filter that were used. The Filter reps column describes how many times the filter was repeated (0 corresponds to the filter being used once). The Recall column describes how many of the manual annotated peaks that were detected in percentage (1.0 corresponds to all). The Recall column describes how many of the found peaks that were correct peaks in percentage (1.0 corresponds to all). The Run time column describes how long the running time for the program was.

File	Min	Max	Group method	Detection method	Corr limit	Filter	Filter reps	Recall	Precision	Run time
Low noise	0	0	Brute Force	Single Derivative	n/a	No filter	0	1.0	0.04454	7348.0
Low noise	30	30	Brute Force	Single Derivative	n/a	NoFilter	1	0.91150	0.35100	1574.0
Low noise	30	0	Multi	Multi	n/a	Daubchies8	3	1.0	0.01643	8850.0
Low noise	n/a	n/a	Sorted	Template	0.95	Daubchies3	2	0.44837	0.02218	4296.0
Medium noise	60	0	Sorted	Single Derivative	n/a	MultiWavelet	3	1.0	0.01164	24917.0
Medium noise	30	60	Sorted	Single Derivative	n/a	Daubchies8	0	0.88268	0.03125	7543.0
Medium noise	60	60	Brute Force	Single Derivative	n/a	Daubchies2	3	0.94413	0.03863	5638.0
Medium noise	n/a	n/a	Sorted	Template	0.99	MultiWavelet	2	0.04469	0.13235	18675.0
Medium noise	n/a	n/a	Brute Force	Template	0.49	Daubchies2	2	1.0	0.02073	16295.0
High noise	30	0	Brute Force	Single Derivative	n/a	Daubchies2	1	0.98850	0.01407	9228.0
High noise	30	60	Sorted	Single Derivative	n/a	Daubchies2	2	0.83908	0.03392	4283.0
High noise	60	60	Sorted	Single Derivative	n/a	MultiWavelet	3	0.94252	0.03144	22625.0
High noise	0	0	Multi	Multi	n/a	NoFilter	0	0.59770	0.00920	4294.0
High noise	60	0	Multi	Multi	n/a	SavitzkyGolay9	3	0.96551	0.01316	5672.0
High noise	60	60	Multi	Multi	n/a	MultiWavelet	3	1.0	0.02120	23049.0
High noise	n/a	n/a	Brute Force	Template	0.49	NoFilter	0	1.0	0.02034	8336.0
High noise	n/a	n/a	Sorted	Template	0.74	SavitzkyGolay9	3	1.0	0.01519	4834.0

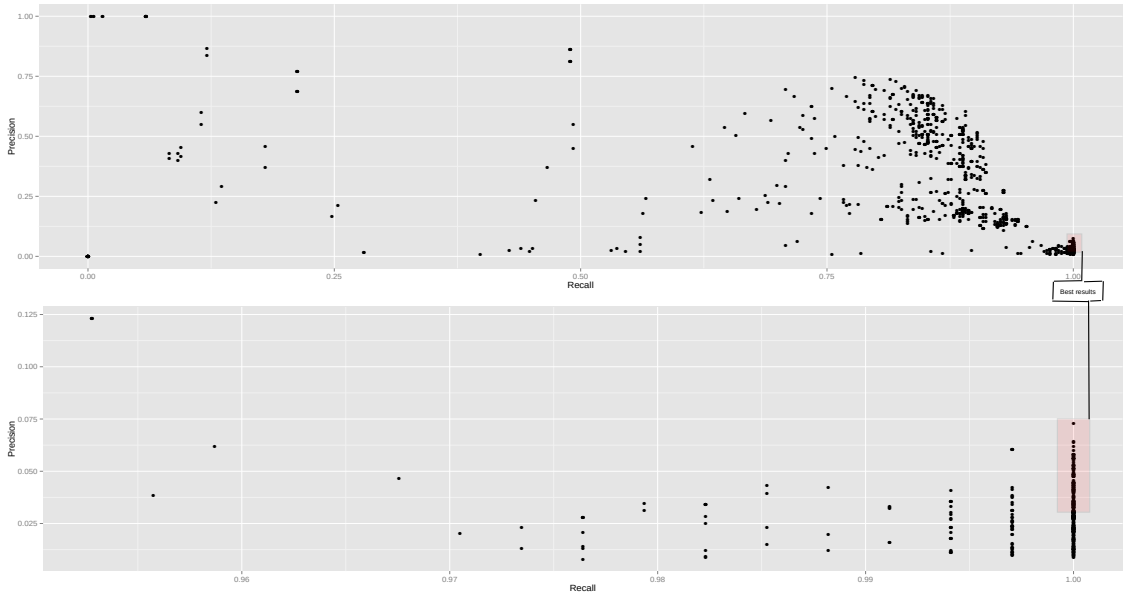

Figure 3.1: Performance on the low noise file. The first graph is a overview and the second is zoomed in on high recall. Each dot represents a parameter combination and show the performance in recall and precision.

Result data is available at: `http://quantitativeproteomics.org/DIA_peak_detection/`

Figs. 3.1, 3.2, and 3.3 show the results from the three different files.

3.1 Signal processing

There was no single solution in the signal processing that performed absolutely better than the other, instead their results varied depending on the data they processed and with which other parameters they were combined. In Figs. 3.4, 3.5, and 3.6 one can see how well the different filters perform against each other in recall and precision. There were also varying results depending on how many times the data were run through the filter, also visualized in the picture.

3.2 Peak detection

There were no single peak detection method that outperformed the others either. As in the signal processing results, the results varied depending on the data they processed and with what other parameters that they were combined. We expected the template method to perform worse in data with higher noise, but it turned out to be able to find all the correct peaks in the data with the highest noise we had.

Figs. 3.7, 3.8, and 3.9 show the results from the different peak detection methods along with the threshold values for the template method (called correlationLimit). Note that the

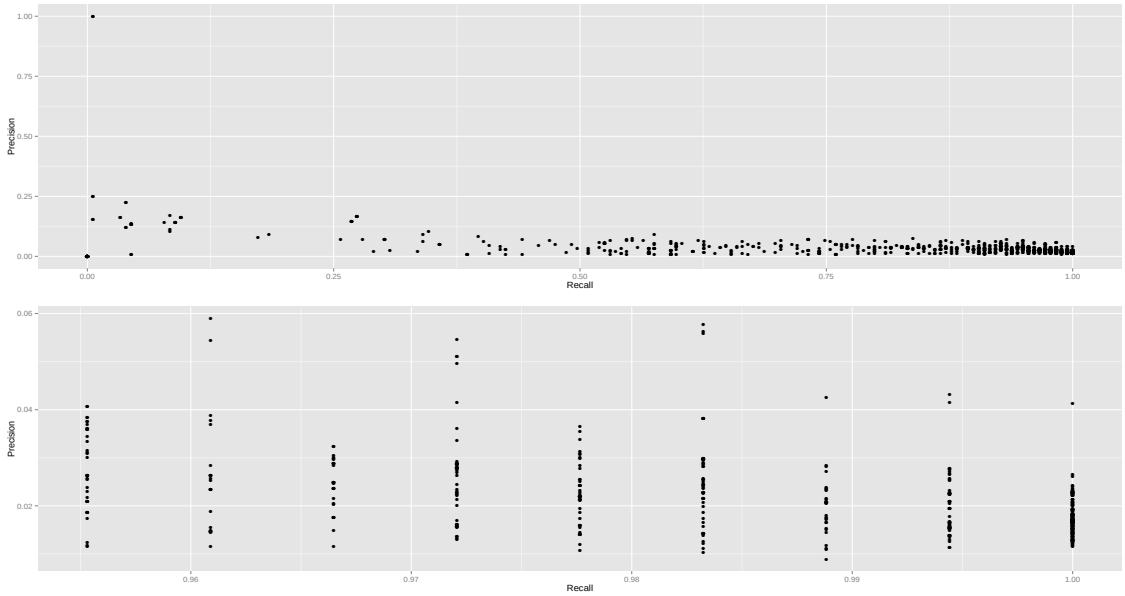

Figure 3.2: Performance on the medium noise file. The first graph is a overview and the second is zoomed in on high recall. Each dot represents a parameter combination and show the performance in recall and precision.

N/A value from the threshold means that the template method was not used. Figs. 3.10, 3.11, and 3.12 show the results from the three different files for the different boundary values used for the derivative methods. Note that the N/A value means that the template method was used.

3.3 Grouping

Just as in the two previous sections, the grouping methods results varied depending on both the data they processed and the other parameter choices and therefore no solution was clearly superior. However we consider the brute force approach the best approach overall since it, along with certain parameter choices, had the highest precision with a 100 percent recall in all files with different noise levels. However, it is not the best approach for all cases and because with certain parameter combinations other grouping methods yielded better results. Figs. 3.13, 3.14, and 3.15 show the results from the three different files.

3.4 Supervised learning

There were several parameter choices in the peak detection algorithm that yielded a 100 percent recall and therefore we deemed the binary predictor unnecessary and it was therefore not implemented and tested. We did not try changing the feature extraction either as we were pleased with the results in this section.

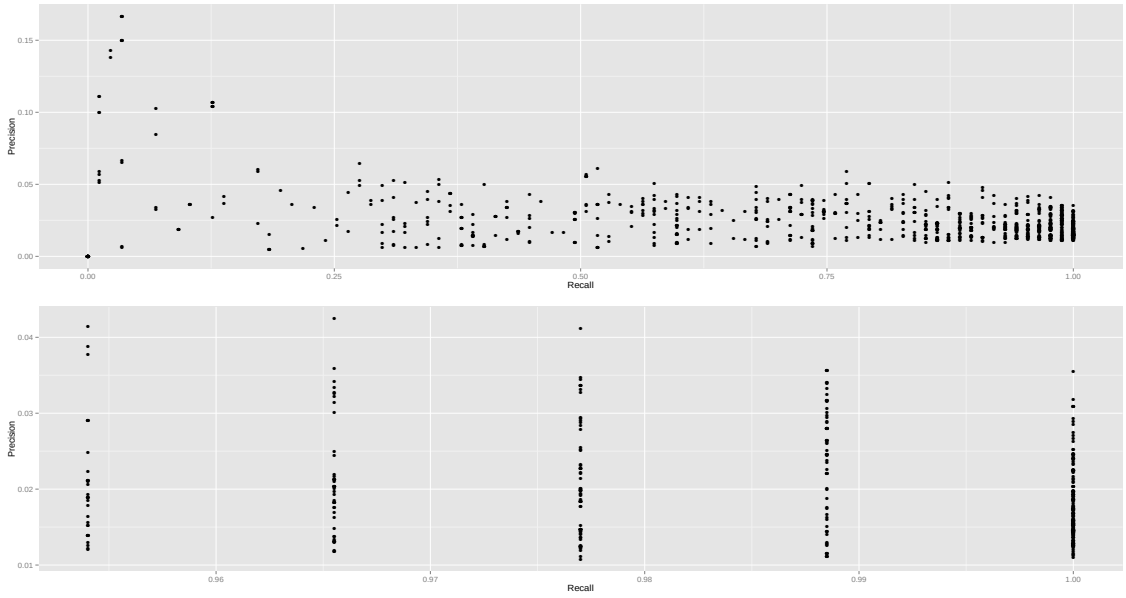

Figure 3.3: Performance on the high noise file. The first graph is a overview and the second is zoomed in on high recall. Each dot represents a parameter combination and show the performance in recall and precision.

We used linear regressions for the regressions. The regressions did not give a single best parameter combination, instead results varied depending on the data's structure. Each linear regression follows the form of:

$$\hat{y} = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 + \beta_7 x_7,$$

where the β parameters are the regression parameters and the x parameters are the features from Sect. 2.5.

In the final implementation of the software we added all parameter combinations that yielded 100 percent recall and their linear regressions and then chose parameters for each peptide processed. The selection was done by calculating the linear regressions and choosing the parameter combination that yielded the least estimated peaks, i.e. the highest precision.

In this final implementation the precision on less noisy data was suboptimal compared to other parameter choices that were discarded. The reason behind this is probably that the parameter combination that could increase the precision did not have a 100 percent recall on the data with higher noise, and therefore was not included in the calculations.

Table 3.2 shows the final implementations result on the unseen test data set.

Figure 3.4: Performance of the different filters on the low noise file. The first graph is a overview and the second is zoomed in on high recall. Each dot represents a parameter combination and show the performance in recall and precision.

Figure 3.5: Performance of the different filters on the medium noise file. The first graph is a overview and the second is zoomed in on high recall. Each dot represents a parameter combination and show the performance in recall and precision.

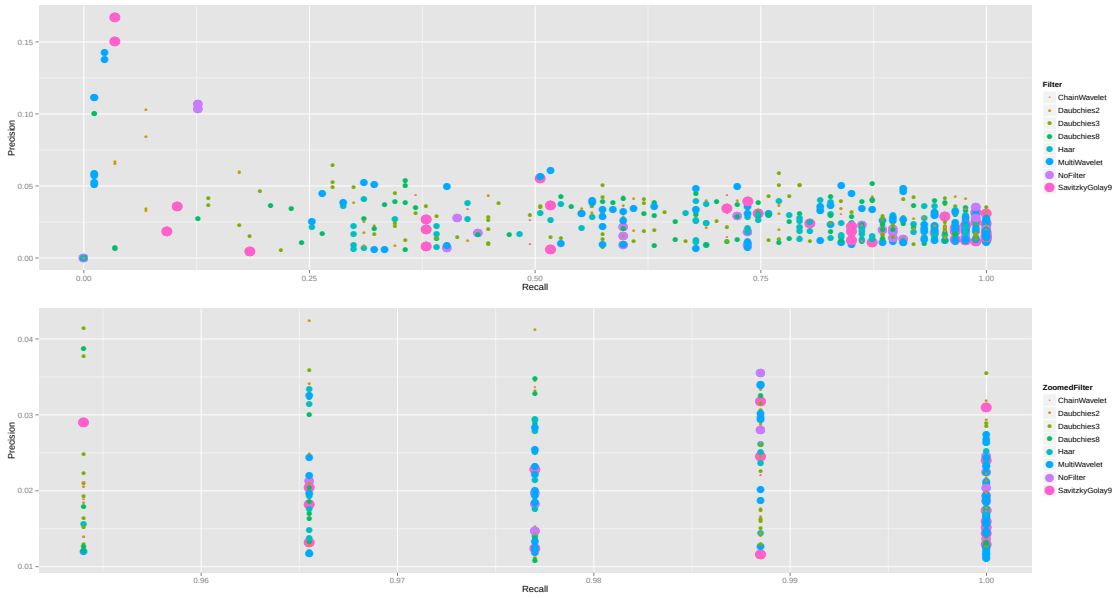

Figure 3.6: Performance of the different filters on the high noise file. The first graph is a overview and the second is zoomed in on high recall. Each dot represents a parameter combination and show the performance in recall and precision.

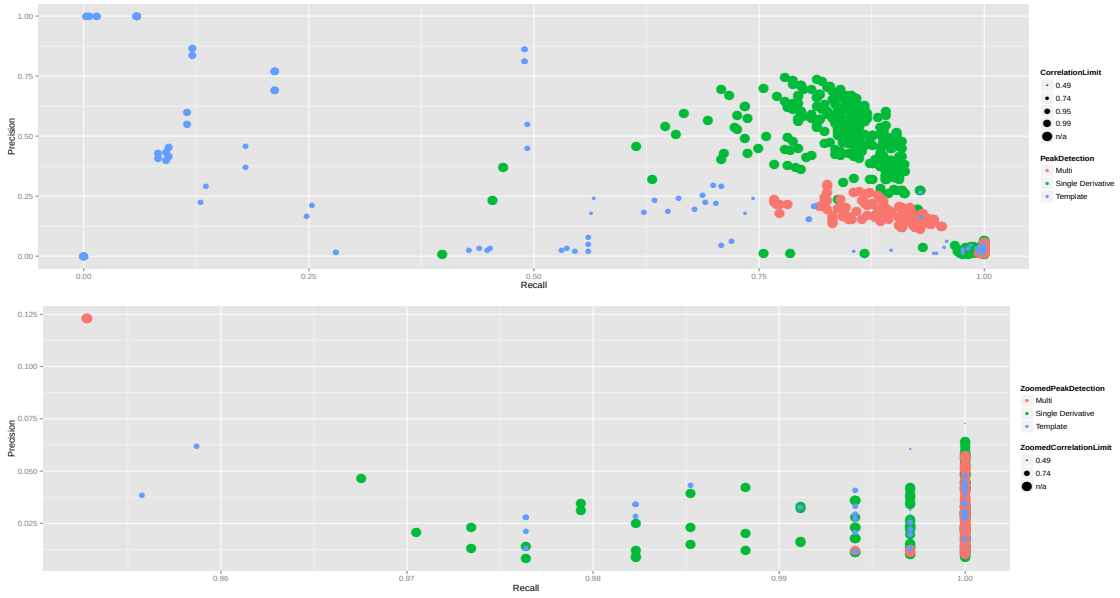

Figure 3.7: Performance of the different peak detection methods along with the threshold value of the template method on the low noise file. The first graph is a overview and the second is zoomed in on high recall. Each dot represents a parameter combination and show the performance in recall and precision.

Figure 3.8: Performance of the different peak detection methods along with the threshold value of the template method on the medium noise file. The first graph is a overview and the second is zoomed in on high recall. Each dot represents a parameter combination and show the performance in recall and precision.

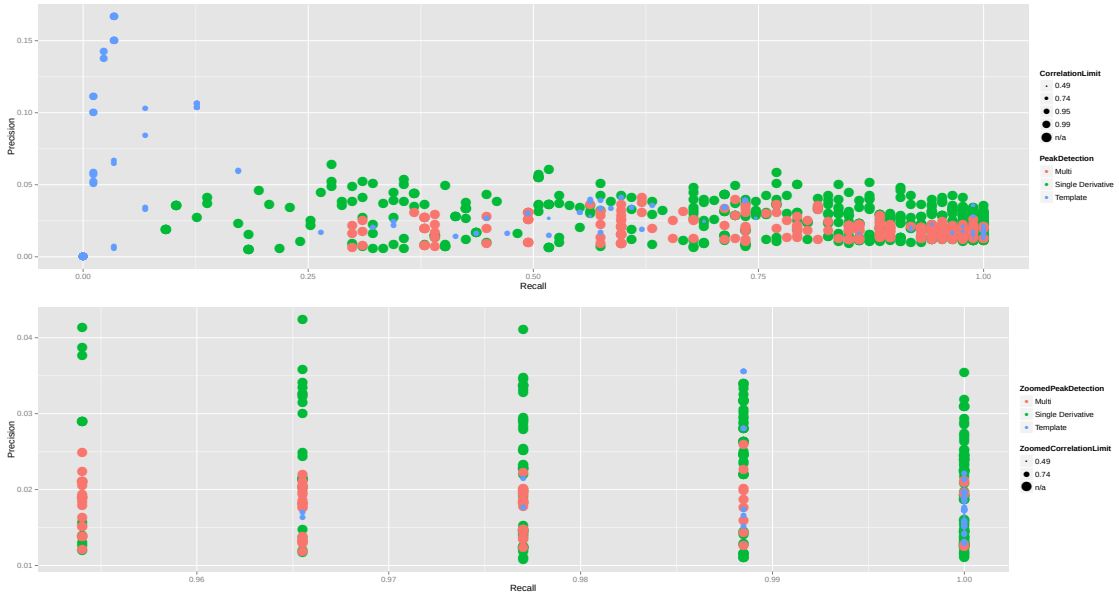

Figure 3.9: Performance of the different peak detection methods along with the threshold value of the template method on the high noise file. The first graph is a overview and the second is zoomed in on high recall. Each dot represents a parameter combination and show the performance in recall and precision.

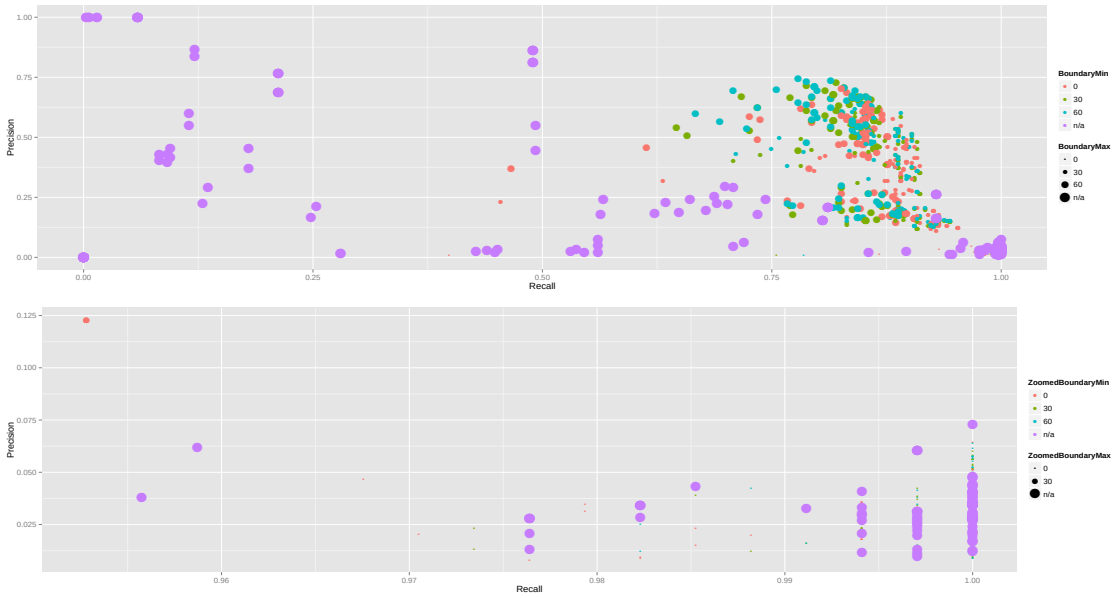

Figure 3.10: Performance of the different boundary values on the derivative methods on the low noise file. The first graph is a overview and the second is zoomed in on high recall. Each dot represents a parameter combination and show the performance in recall and precision.

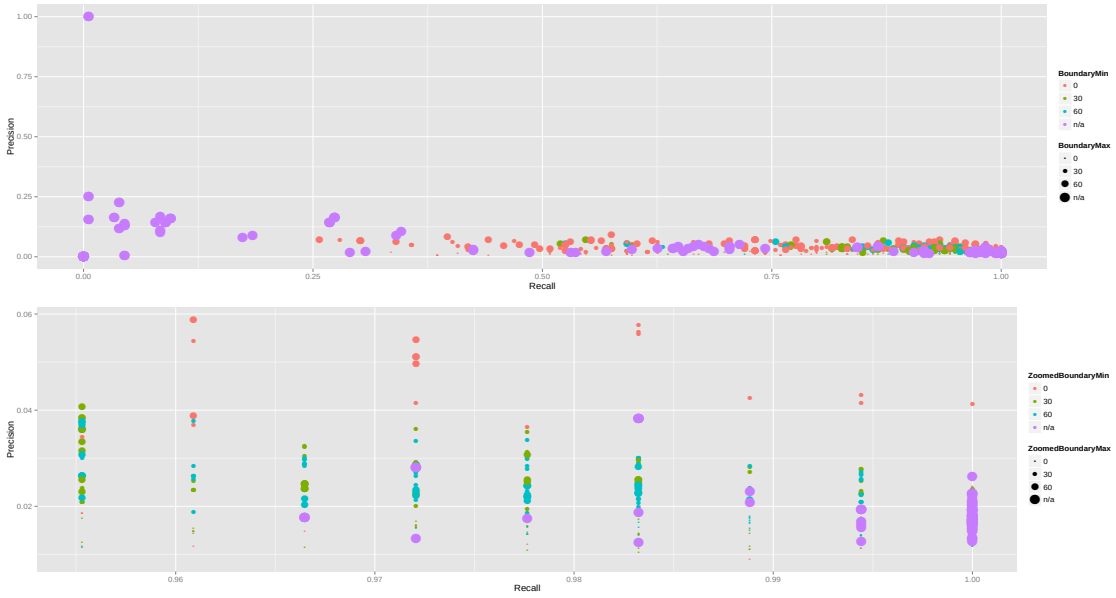

Figure 3.11: Performance of the different boundary values on the derivative methods on the medium noise file. The first graph is a overview and the second is zoomed in on high recall. Each dot represents a parameter combination and show the performance in recall and precision.

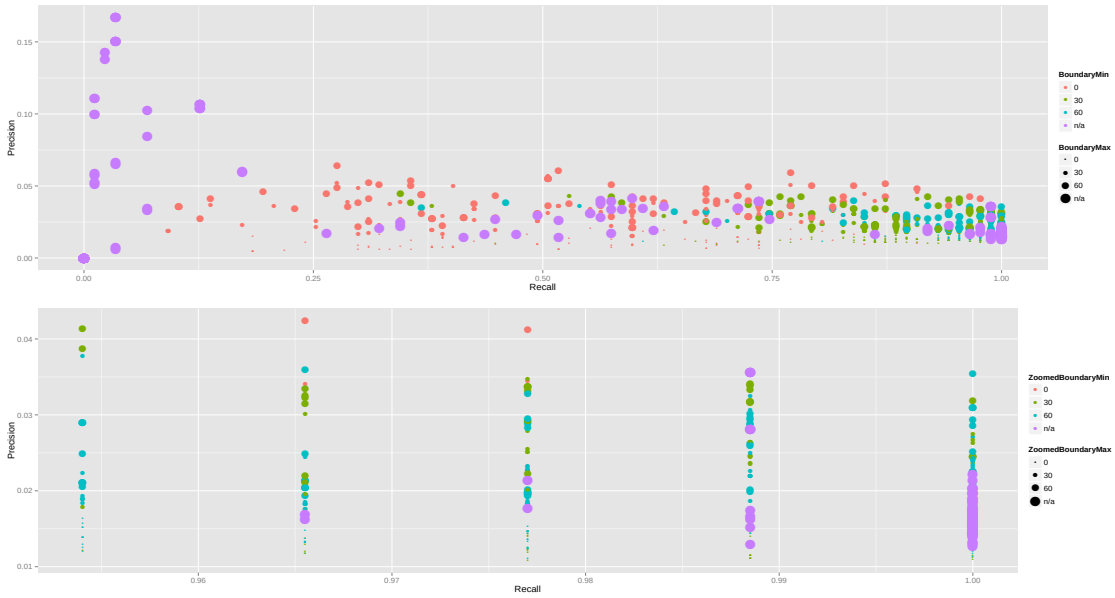

Figure 3.12: Performance of the different boundary values on the derivative methods on the high noise file. The first graph is a overview and the second is zoomed in on high recall. Each dot represents a parameter combination and show the performance in recall and precision.

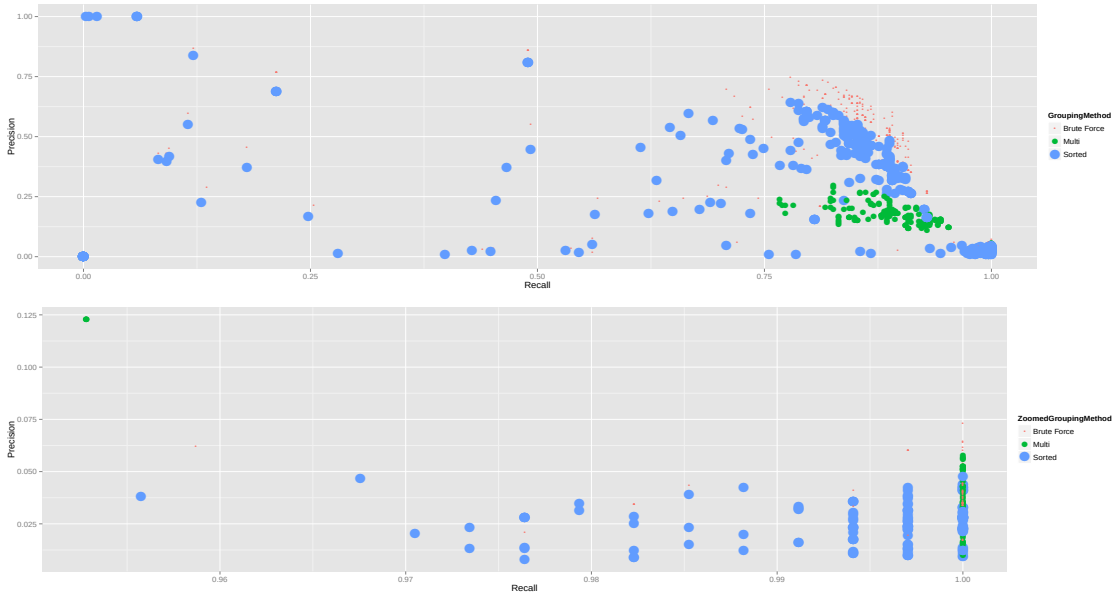

Figure 3.13: Performance of the different grouping methods on the low noise file. The first graph is a overview and the second is zoomed in on high recall. Each dot represents a parameter combination and show the performance in recall and precision.

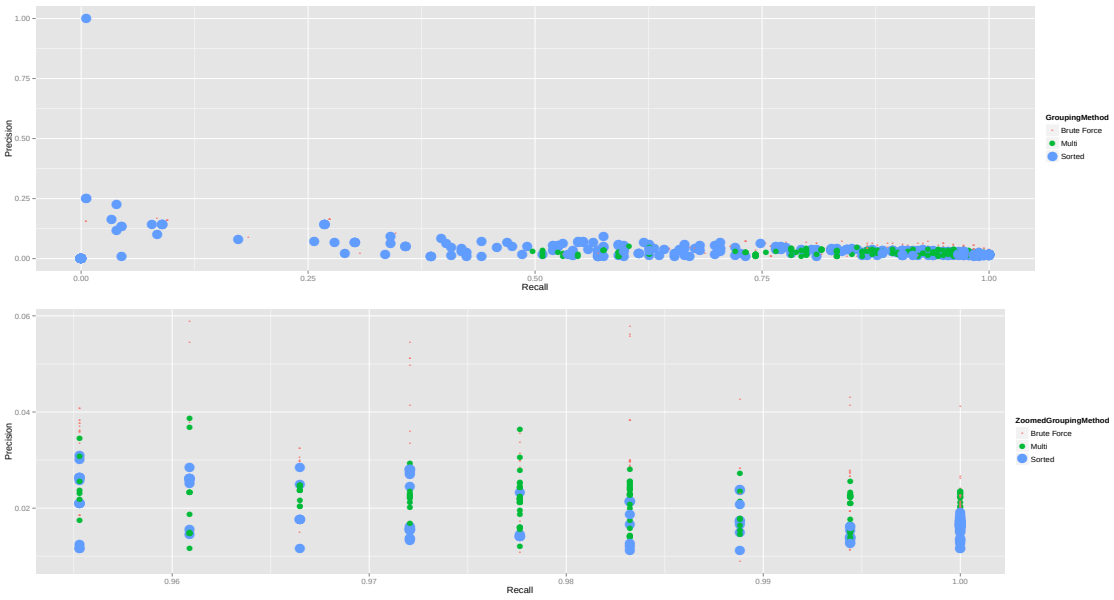

Figure 3.14: Performance of the different grouping methods on the medium noise file. The first graph is a overview and the second is zoomed in on high recall. Each dot represents a parameter combination and show the performance in recall and precision.

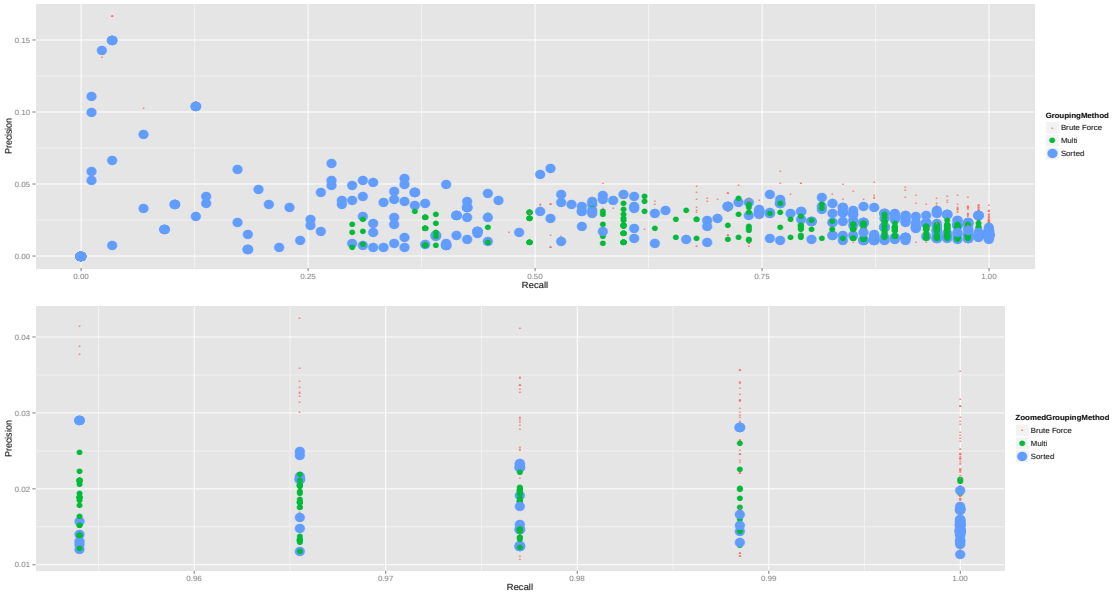

Figure 3.15: Performance of the different grouping methods on the high noise file. The first graph is a overview and the second is zoomed in on high recall. Each dot represents a parameter combination and show the performance in recall and precision.

Table 3.2: Results of the unseen test data. The File column marks which of the three test files it was tested on. The Recall column describes how many of the manual annotated peaks that were detected in percentage (1.0 corresponds to all). The Precision column describes how many of the found peaks that were correct peaks in percentage (1.0 corresponds to all). The Run time column describes how long the running time for the program was.

File	Recall	Precision	Run time
1	1.0	0.01519	7597.0
2	1.0	0.01637	8210.0
3	1.0	0.01498	12983

Chapter 4
Improvements and future work

The core of our project consists of peak detection in a DIA analysis. Our algorithm could find multiple parameter combination providing us with a 100 percent recall on the test set. This was the goal of this project and we reached it and therefore consider this project a success. However, this does not guarantee a 100 percent recall on all data sets as the manually annotated sets were limited.

The biggest reason we included recall as an evaluation parameter is that we thought that the execution time of the whole DIA-analysis could become too long if too many peaks were included. In our implementation we did not connect the peak detection to the subsequent steps of the analysis and our theory was therefore not tested. A future work on the subject could be inserting our implementation into a current implementation of the DIA-analysis.

Several aspects of our approaches could be improved. As our test data was limited we think that there are data structures our solution could not handle. Using more approaches than the ones we tried could provide a solution to this possible problem. The improvements that are the easiest to achieve would be a higher precision as well as a faster running time for the program. In the four different sections, we consider the following changes the ones we would look into first:

1. The signal processing could possibly be improved in the following ways: Add other filters that are not currently used or by changing the thresholds limits. By adding another filter we may be able to process data that the current program can not handle. Changing the thresholds may yield higher precision, but may also destroy correct peaks and should therefore be handled carefully.

2. The peak detection can probably be improved by optimizing boundary limits in the derivative approach, or possibly by adding a baseline to ignore low amplitude data noise. The template approach in the peak detection could possibly be improved by using a more advanced template, or by varying templates depending on the data structure.

3. When it comes to the grouping methods, one could include more grouping algorithms with different criteria in order to create more groups, possibly consisting of one peak alone. This would lead to even more peaks being kept that could yield the correct result. This might however impact the precision negatively.

4. The thing that definitely could be improved in the supervised learning is the feature choices. By changing or adding features, one could probably create a better regression which would lead to a higher precision. If our implementation turns out to have a recall less than 100 percent in the future, one might want to implement predictors instead, which might have a good success rate at predicting parameter choices.

Chapter 5
Conclusions

In this project, we tried to improve the first part of the DIA analysis pipeline using different peak detection algorithms, different signal processing methods, different grouping methods, and adding a machine learning aspect. Our implementation was able to find 100 percent of the correct peaks of the manually annotated results and we consider this a success. We also show that the varying solution of the subproblems we implemented are necessary as different approaches suit different structures of data. Our results will hopefully improve the currently used pipeline of DIA analysis and can also hopefully lay the foundation for further improvements in the peak detection step of this analysis.

We think that our results will help pinpoint what areas that should be the focus of further improvements. Our results clearly show that trying to improve one part of the peak detection might not be helpful without changing the other parts, which is a helpful conclusion for anyone doing similar work.

Bibliography

Balambigai Subramanian, A. R. and Rangasamy, K. (2014). Performance comparison of wavelet and multiwavelet denoising methods for an electrocardiogram signal. *Journal of Applied Mathematics*, 2014:1–8.

Brittain, J.-S., Halliday, D. M., Conway, B. A., and Nielsen, J. B. (2007). Single-trial multiwavelet coherence in application to neurophysiological time series. *Biomedical Engineering, IEEE Transactions on*, 54(5):854–862.

Caruana, R. and Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM.

Daubechies, I. (1992). *Ten Lectures on Wavelets*. SIAM: Society for Industrial and Applied Mathematics, Philadelphia, Pa.

Getreuer, P. (2006). Filter coefficients to popular wavelets. *MATLAB Central, May*.

Haar, A. (1910). Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69(3):331–371.

Hu, A., Noble, W. S., and Wolf-Yadlin, A. (2016). Technical advances in proteomics: new developments in data-independent acquisition. *F1000Research*, 5.

Keinert, F. (2004). *Wavelets and Multiwavelets*. Chapman & Hall/CRC Press, Boca Raton, FL.

Krasteva, V. and Jekova, I. (2007). Qrs template matching for recognition of ventricular ectopic beats. *Annals of Biomedical Engineering*, 35(12):2065–2076.

Loh, W.-Y. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23.

Reiter, L., Rinner, O., Picotti, P., Hüttenhain, R., Beck, M., Brusniak, M.-Y., Hengartner, M. O., and Aebersold, R. (2011). mprophet: automated data processing and statistical validation for large-scale srm experiments. *Nature methods*, 8(5):430–435.

Röst, H. L., Rosenberger, G., Navarro, P., Gillet, L., Miladinović, S. M., Schubert, O. T., Wolski, W., Collins, B. C., Malmström, J., Malmström, L., et al. (2014). Openswath enables automated, targeted analysis of data-independent acquisition ms data. *Nature biotechnology*, 32(3):219–223.

Savitzky, A. and Golay, M. J. E. (1964). Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639.

Shin, J., Choi, B., Lim, Y., Jeong, D., and Park, K. (2008). Automatic ballistocardiogram (bcg) beat detection using a template matching approach. In *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, pages 1144–1146. IEEE.

Strela, V., Heller, P. N., Strang, G., Topiwala, P., and Heil, C. (1999). The application of multiwavelet filterbanks to image processing. *IEEE Transactions on image processing*, 8(4):548–563.

Teleman, J., Karlsson, C., Waldemarson, S., Hansson, K., James, P., Malmström, J., and Levander, F. (2012). Automated selected reaction monitoring software for accurate label-free protein quantification. *Journal of proteome research*, 11(7):3766–3773.

Teleman, J., Röst, H. L., Rosenberger, G., Schmitt, U., Malmström, L., Malmström, J., and Levander, F. (2015). Diana—algorithmic improvements for analysis of data-independent acquisition ms data. *Bioinformatics*, 31(4):555–562.

Tsou, C.-C., Avtonomov, D., Larsen, B., Tucholska, M., Choi, H., Gingras, A.-C., and Nesvizhskii, A. I. (2015). Dia-umpire: comprehensive computational framework for data-independent acquisition proteomics. *Nature methods*, 12(3):258–264.

Vidakovic, B. and Mueller, P. (1991). Wavelets for kids: A tutorial introduction.

Weisberg, S. (2005). *Applied linear regression*, volume 528. John Wiley & Sons.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTATIONSDAG 2016-06-03

EXAMENSARBETE Peak Detection in Data Independent Analysis
STUDENT Viktor Stymne
HANDLEDARE Pierre Nugues (LTH), Johan Teleman (LTH) och Fredrik Levander (LTH)
EXAMINATOR Jacek Malec (LTH)

Topp-detektion i masspektrometridata av proteiner

POPULÄRVETENSKAPLIG SAMMANFATTNING **Viktor Stymne**

För att upptäcka människors sjukdomar kan det hjälpa att veta vilka proteiner som finns i deras celler. Det går att göra manuellt från ett cellprov, men det är effektivare att göra det med hjälp av automatiserade datorprogram.

Det finns olika metoder för att identifiera vilka proteiner som finns i cellprov, men inga av dessa är perfekta. Det här arbetet fokuserar på metoden "data independant acquisition analysis". För att sammanfatta processen så består den av två huvuddelar: dataframtagning och analys av datan. För dataframtagningen så körs ett cellprov igenom en masspektrometer. Maskinen tar fram data från cellprovet som sedan bearbetas och analyseras. Exempel på data visas i bilden.

Metoden är relativt ny och är ännu inte optimal, och kan därför vara värd att undersöka djupare. I detta arbetet förbättrade vi den första delen av analysen. Första steget består av att hitta särskilda toppar i datan vi fick fram från masspektrometern. Processen går att göra manuellt vilket också har gjorts av andra. Vi byggde istället en automatisk programvara för att försöka hitta dessa toppar. Vi använde sedan de manuella resultaten för att utvärdera vår automatiska programvara. Vårt program lyckades hitta alla manuella toppar och möjligen mer därtill.

Det är svårt att säga vad utfallet har för effekt på hela analysen, men vi vill tro att det kan förbättra de senare stegen och förhoppningsvis även det totala resultatet så att fler proteiner kan identifieras. Vårt resultat kan öka förståelsen för vad som orsakar sjukdomar samt hjälpa till att diagnostisera patienter tidigare så att det är lättare att behandla dem. Programmet använder sig av flera olika delar. Totalt finns det 1120 olika kombinationer av parametrar som programmet kan välja mellan. Valet mellan dessa parametrar görs helt automatiskt beroende på vilken typ av data den försöker hitta topparna i.