Semester Thesis in Machine Learning

Peptide Assignment Validation: Telling what's wrong without actually knowing what's right

Patrick Pletscher*
ETH Zurich

4th April 2006

Supervised by:
Bernd Fischer, Prof. Joachim M. Buhmann

*Email: pat@student.ethz.ch

Peptide database searching can help quite significantly identifying peptides in a given cell and several algorithms and methods have been designed to make the search both, accurate and efficient. For a good summary of the different approaches see for example [SCY04]. However, not seldom this database search methods show to be useless, as the peptides the algorithm returns, have nothing to do with the peptides observed in the lab (in previous studies only around 10% of the peptide assignments showed to be correct ones). So the right way to do peptide database searches is to first match a peptide (or maybe several) in the database and afterwards to validate the result. In this semester thesis we concentrate on the second step – the validation – while of course the first step has still to be implemented as otherwise there wouldn't be anything to validate. We compare several methods to validate the peptide assignment, which we have all implemented in a small program, called PeptideFind.

Contents

1 Introduction

In biology and chemistry one is often interested in finding the proteins inside of a cell. The usual way to facilitate the identification of the proteins includes, among other steps, the collision of the digested proteins (then called peptides) with an inert gas, which is followed by a mass spectrometry. What we end up with, is a so called MS/MS spectra, like the one shown in Figure 1.1.

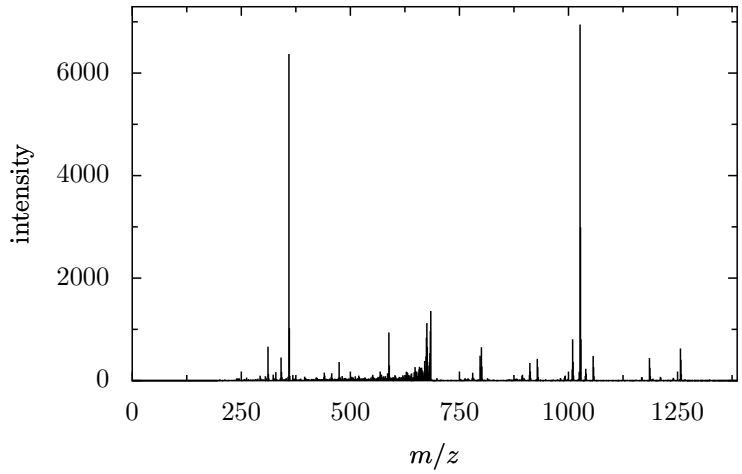

Figure 1.1: An experimental tandem mass spectrum of the peptide KAEQQLVNDPSR.

The goal is then to find the amino acid sequence of the peptide. There are different algorithms used in practice to identify the amino acids sequence, in recent time there were for example *de novo* algorithms proposed, see [FRR+05]. However, the vast majority of the algorithms in use today, use the information that is given to us by the genome of the creature of interest.

The peptide assignments, that these algorithms output are only correct for about 10% of the mass spectra. This makes a second step necessary, in which one classifies the assignment according to their correctness. In this semester thesis we discuss and compare several peptide validation algorithms. We especially focused ourselves on a hypergeometric propability model, proposed by Sadygov and Yates in [SY03], which leads to a simple, but powerful statistical test to decide on the correctness of an assignment.

2 Data Normalization

The normalization of the experimental spectrum is a very important step for the database search to succeed. However, there are several subtle details to pay attention to, which we describe in this chapter. These simple heuristics helped very much to improve the accuracy of our database search.

2.1 Discretization of Mass and Vector Representation

Throughout this project report, and as well in the code of PeptideFind, we always represent a mass spectrum as a simple vector. To do so, we first discretize the mass with steps of 1.0005 Dalton. The first component of the vector then denotes the intensity of the peak at 1.0005 Dalton, the second component the intensity of the peak at 2.001 Dalton and so on. Usually the elements of the vector are in $\mathbb{R}^+$. However, for the Hypergeometric Probability Model we binarize the intensities: either a peak is present or not. The motivation behind this should become clear in the relevant sections.

Introducing the vector notation has the advantage of using well-known mathematical formalism to denote operations on the spectra. Throughout this documentation we will denote vectors in bold fonts. Experimental spectra will be denoted by $\boldsymbol{e}$ and $\boldsymbol{f}$, while theoretical spectra, will be denoted by $\boldsymbol{p}$ and $\boldsymbol{q}$. The componentwise product of two vectors $\boldsymbol{p}$ and $\boldsymbol{e}$ will be denoted by $\boldsymbol{p} \star \boldsymbol{e}$.

2.2 Flatten the Experimental Spectrum

If one examines a mass spectrum in more detail, one observes, that there is a variation in the intensity over the peptide itself. The intensity of the peaks in the center of the spectrum is much higher than at the two ends. One should thus learn the distribution of the intensities over the spectrum and use this knowledge for the normalization of the experimental spectra. For an illustration of the intensity variations see Figure 2.1. It shows the learned distribution as a histogram of 200 bins. The small intensity in the middle of the histogram can be explained by doubly charged peptides.

After learning the distribution of the intensities over the protein for a training data set, we can multiply the experimental spectrum by the inverse of the learned distribution. As we need to invert the learned distribution, it makes sense to add a small regularization constant to each bin of the learned distribution, such that we don't have to bother with divisions by zero. In our tests we chose this constant to be 0.0005. The multiplication

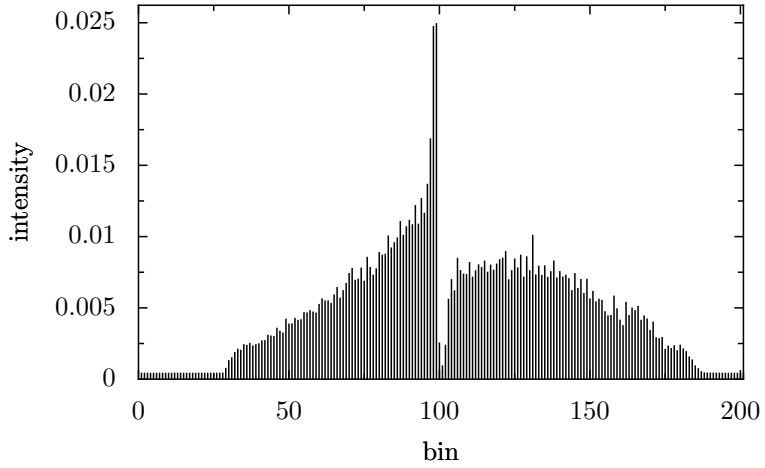

Figure 2.1: Distribution of the intensities as learned by PeptideFind for 200 bins.

with the inverse distribution should make all the peaks around equally intense. See Figure 2.2 for the inverse of the learned distribution.

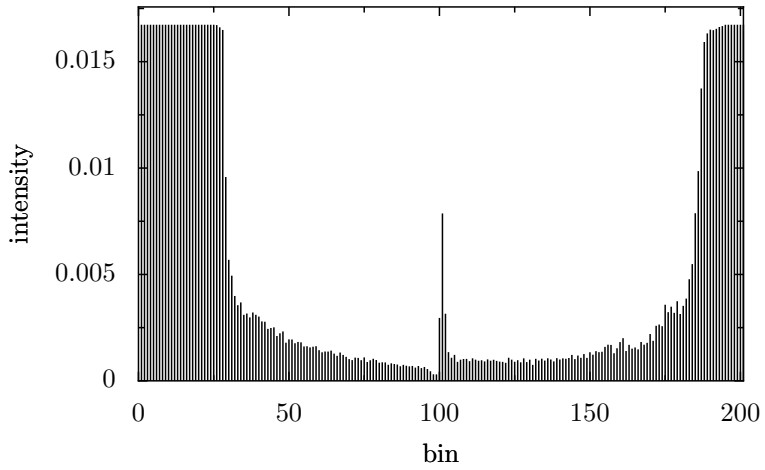

Figure 2.2: Inverse distribution of the intensities as learned by PeptideFind for 200 bins.

More formally given the learned distribution $\boldsymbol{d}$ and the experimental spectra $\boldsymbol{e}$. One can then flatten the experimental spectrum by multiplying it componentwise with the elementwise inverse of the learned distribution

$$\boldsymbol{e} = \boldsymbol{e} \star \boldsymbol{d}^{-1}.$$

Figure 2.3 shows an experimental spectrum preprocessed in the way just described.

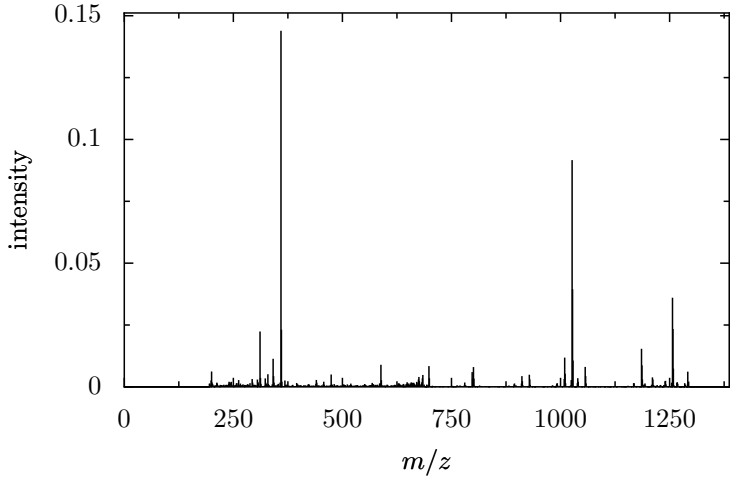

Figure 2.3: Flattened, normalized spectrum for the experimental spectrum shown in Figure 1.1.

2.3 Normalization for the Hypergeometric Probability Model

Especially the probability scoring showed to be highly dependent on the mass spectrometer. We can greatly decrease this dependency by only considering the biggest peaks of the experimental spectrum, and removing the other peaks. We've chosen the number of peaks proportional to the mass of the spectrum: around 4 peaks per 100 Dalton. This improved the probability of a correct assignment for `exp_041` from below 1% to around 99%.

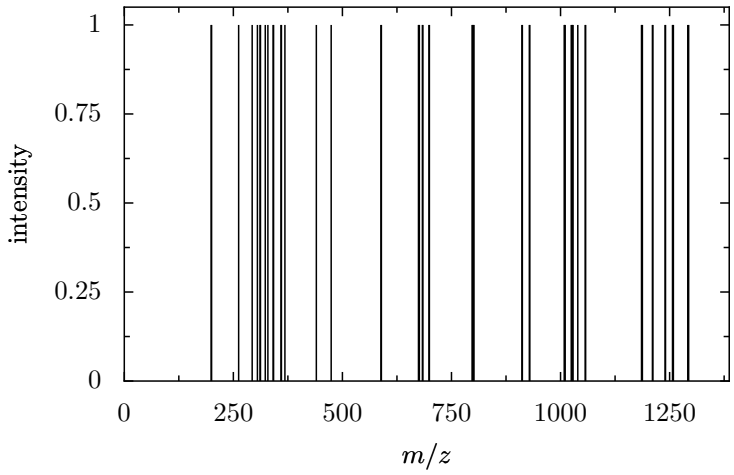

Figure 2.4: The same mass spectrum as in Figure 2.3, but this time we removed the small peaks and binarized the spectrum.

3 Searching in a Protein Database

Given an experimental tandem mass spectrum as for example shown in Figure 1.1 and a FASTA protein database, we want to find a peptide that results in a "similar" MS/MS spectrum. Although the concept is rather simple, there are two major issues, from the machine learning perspective:

- What does *similar* mean? We need to introduce some sort of metric/score, which assigns a high score to the correct peptide. As we work with mass spectra with measurement errors, we however can not hope to find the ultimate score. We can only aim for a small error.
- The *computational complexity* has to be carefully considered. Neither can we afford to use complex scores, nor can we afford to search through all the peptides. As a consequence we only consider simpler scores and only concentrate on the peptides that have almost the same mass as the experimental tandem mass spectrum.

Beside these two issues, there are biological reasons why a database search will not lead to a correct assignment: peptides can for example not fragment at all, because the binding energy of the molecules is too high. This issue leads to the problem, that the database search will assign a lot of peptides wrongly, which makes a second step necessary, the validation, where one uses heuristics to decide whether the peptide assignment is a correct one or not. In this chapter we concentrate on the database search and introduce two scores that are used in practice and are as well implemented in PEPTIDEFIND.

3.1 Generating a Theoretical Spectrum

For all peptides in the database that match the experimental peptide mass, we generate a theoretical spectrum. After the generation of the theoretical spectrum for each of the eligible sequences, we will compare each of these theoretical spectra with the normalized experimental spectrum by a scoring function. For two widely used scoring functions see section 3.2 and 3.3.
In this section we discuss how to generate a theoretical mass spectrum. As in the subsequent sections we will assume, that the experimental spectra are correctly normalized, we don't use any further heuristics to improve the quality of the theoretical spectra, beyond the ones described in this section.

To generate a theoretical spectrum first one has to recall how a tandem mass spectrometer works: After splitting apart the peptide by colliding it with a inert gas, one gets a pre- and suffix part (in biology often called b- and y-ion, respectively), of which the

mass spectrometer then detects its mass. These peaks are the most intense and are thus called *main peaks*. As there occur as well different molecule losses, e.g. H_2O loss, there exist additional peaks near the main peak. This so-called *shifts* have however a smaller intensity, see Figure 3.1 for their relative intensities: the plots show the intensities as learned by PeptideFind.

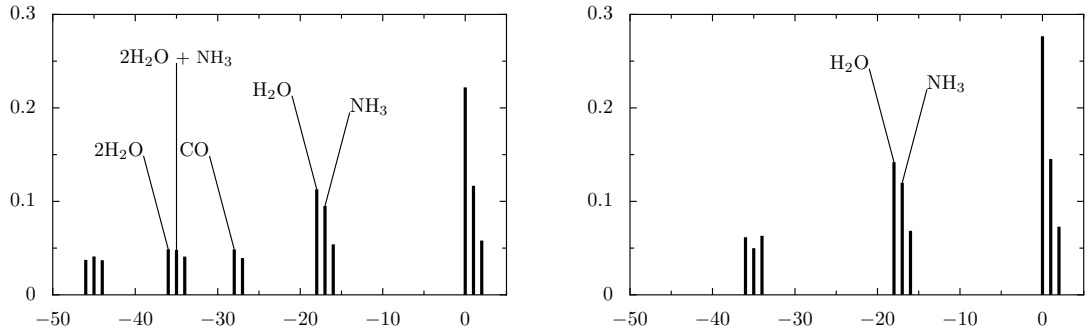

Figure 3.1: Learned intensities for different shifts. On the left for the prefix, on the right for the suffix.

To now generate a theoretical spectrum, one just goes through the whole sequence and generates for each possible split a main peak and the corresponding shifts.
It is important to note a small, but useful detail of our algorithm: we do *not* add the new peak of the prefix (suffix) unconditionally, but rather check first whether the new peak is bigger than the one already present at this position. Like this we set each peak to the maximum of the two peaks, from prefix and suffix, at this position. More formally

$$\boldsymbol{p}^{(i)} = \max(\boldsymbol{b}^{(i)}, \boldsymbol{y}^{(i)}),$$

This trick, with a maximum instead of a sum, can improve the quality of the cross-correlation scoring as much as 5%, as we otherwise would favour palindroms too much.

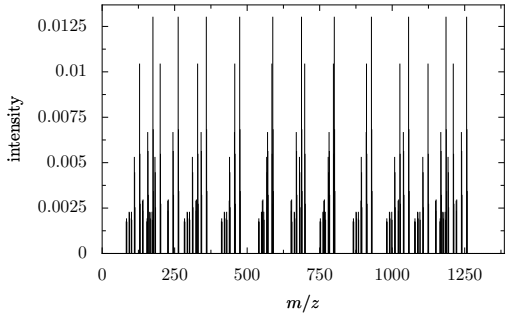

Figure 3.2: A theoretical spectrum generated by the algorithm described in this subsection.

3.2 Scoring based on Cross-correlation

Using our formalism introduced in section 2.1, we can now introduce a first scoring scheme. Given our normalized experimental spectrum $\boldsymbol{e}$, we compute

$$Xcorr = \boldsymbol{e}^t \boldsymbol{p},$$

the dot product between the normalized experimental spectrum and the theoretical spectrum of the peptide sequence under question. As both spectra are assumed to be normalized, we get a value in the interval $[0, 1]$, which measures the similarity of the two spectra.
See Figure 3.3 for an illustration of this approach. Because of the previous normalization we should weight all peaks around equally.

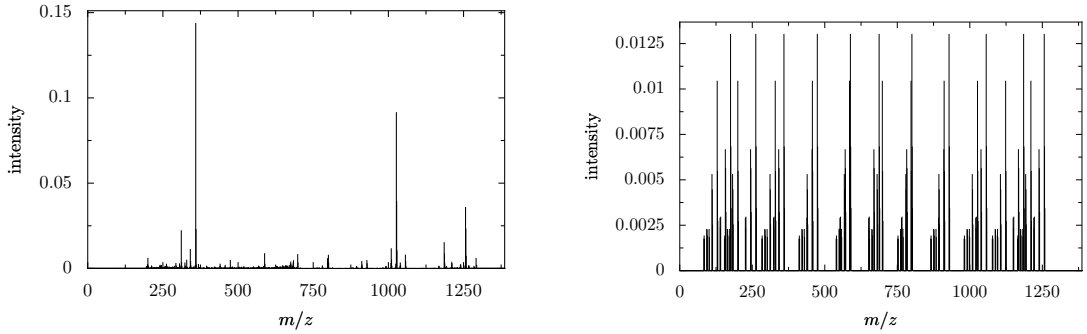

Figure 3.3: Flattened and normalized experimental spectrum (left) and corresponding theoretical spectrum (right) as generated by PeptideFind for the peptide KAEQQLVNDPSR.

3.2.1 Results

This approach assigns around 10% of the peptides wrongly on the two datasets used. See Table 3.1 for more details.

Data	`sbv`	`exp041`
correctly assigned	89.1%	87.8%
wrongly assigned	10.9%	12.2%

Table 3.1: Results with the cross-correlation scoring with inverse.

Figure 3.4 shows a histogram where one can see the position of the correct peptide assignment in the answer of PeptideFind.

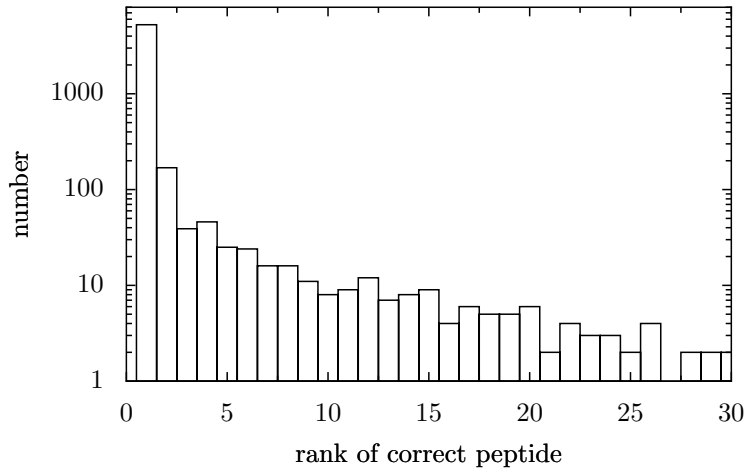

Figure 3.4: Position where the correct assignment occurs in the answer of a cross-correlation search.

3.2.2 Length dependence of Cross-correlation

The cross-correlation score has a tendency to score smaller peptide sequences with a higher score than longer peptide sequences, Figure 3.5 illustrates this behaviour.

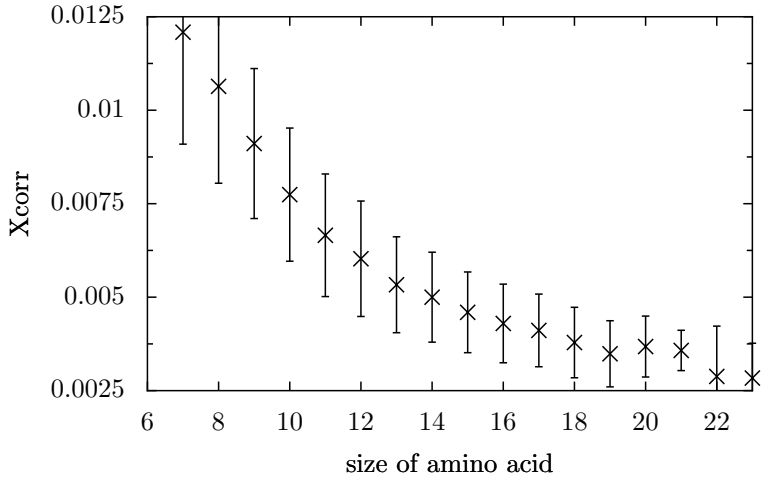

Figure 3.5: Length dependence: The cross-correlation score between the theoretical spectrum of the correct assignment and the experimental spectrum. This score is then averaged over all sequences with the same length. The error bars show the standard deviation.

3.3 Scoring based on a Hypergeometric Probability Model

One promising approach to database search validation is a hypergeometric probability-based method, proposed by Sadygov and Yates [SY03]. However this method is not only suited for peptide assignment validation, as we will see in section 4.1, but one can as well use it for scoring peptide assignments. In this section we describe the theory behind the hypergeometric probability model and present the results, we achieved when using this scoring method for the database search. In chapter 4, we will come back to this model and use it as well for the validation of the peptide assignments.

We want to know the probability that a database search for an experimental tandem mass is a *random match*. This approach depends on the size ℓ of the peptide assignment and on the number of matching main peaks between the assigned peptide spectrum and the experimental mass spectrum.

In this section we interpret a spectrum as a vector of zeros and ones: Either a peak is present or not. We completely forget about the intensity of the peaks. We assume as well that all the 0/1-vectors have the same length, which makes the formalism easier. We can achieve this easily be appending zeros at the end of the vectors. Assume we have m peptides within the database, matching the mass of the experimental tandem mass spectrum with a certain tolerance. We can thus imaging the database as 0/1-vectors $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_m$. Let us then compute

$$\bar{\boldsymbol{q}} = \operatorname{sign}\left(\sum_{i=1}^{m} \boldsymbol{q}_i\right),$$

where the signum is componentwise. With N we then denote the number of non-zero elements of $\bar{\boldsymbol{q}}$.

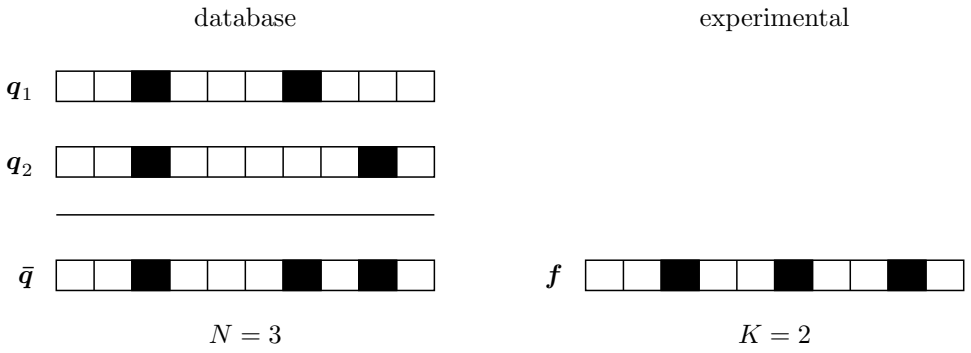

Figure 3.6: How N and K are computed. It is important to note that although $\boldsymbol{f}$ has three non-zero elements, K is still only equal to two, as one non-zero element of $\boldsymbol{f}$ (the fifth from the right) doesn't occur in the database.

Let us imaging as well our experimental tandem mass spectrum as a vector of zeros and ones, which we denote by $\boldsymbol{f}$. We then check if the non-zero elements of $\boldsymbol{f}$ occur at

least once in the database,

$$\boldsymbol{e} = \boldsymbol{f} \star \bar{\boldsymbol{q}}$$

K then denotes the number of non-zero elements of $\boldsymbol{e}$, or alternatively $K = \boldsymbol{f}^t \bar{\boldsymbol{q}}$. Figure 3.6 shows an illustration of the various steps described so far.
If ℓ_i denotes the number of amino acids of the peptide assignment $\boldsymbol{q}_i$, we have at most $n_i = 2(\ell_i - 1)$ non-zero elements in its theoretical spectrum (which we generate), as only the main peaks are used for the generation of the theoretical spectrum. We can then compute the actual number of matching peaks between $\boldsymbol{q}_i$ and $\boldsymbol{e}$ and denote this by k_i,

$$k = \boldsymbol{q}_i^t \boldsymbol{e}. \tag{3.1}$$

After the definitions we can now actually come to the model. We are interested in the probability that the observed configuration (N, K, n_i, k_i) is a random match. The answer is given by the *hypergeometric distribution*. The sample space has a total of N elements and K of them match peaks in the tandem mass spectrum ($K \leq N$). The probability of observing k matches among n_i tries is given by

$$\mathrm{P}_{N,K,n_i}(X = k_i) = \frac{\binom{K}{k_i}\binom{N-K}{n_i-k_i}}{\binom{N}{n_i}}, \tag{3.2}$$

An illustration of the hypergeometric distribution is given in Figure 3.7.

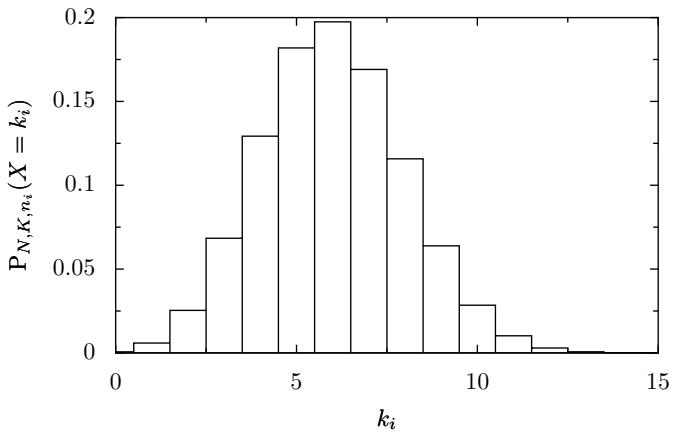

Figure 3.7: The hypergeometric distribution for $N = 1500$, $K = 500$ and $n_i = 18$.

The usage of this model as a scoring function is now straightforward: Given a theoretical spectrum $\boldsymbol{q}_i$ and the normalized experimental spectrum $\boldsymbol{e}$ (where we only keep the biggest peaks as described in section 2.3), we compute k_i, the number of matching peaks as in equation (3.1). Given k_i, we then compute the probability $P_{N,K,n_i}(X = k_i)$, as in equation (3.2), which gives us the probability that this peptide assignment is a random match. As we don't want to select a random match, but rather a correct assignment, we are interested in small probabilities of a random match. We thus select the peptide, that results in the smallest probability of a random match (this is contrary to the

cross-correlation scoring, where we select the peptide assignment with the biggest cross-correlation value). PeptideFind computes only $P_{N,K,n_i}(X = k_i)$ and not the p-value $P_{N,K,n_i}(X \leq k_i)$, which in theory should be a better score. We however will use the p-value in section 4.1, for the validation of an assignment. Using the p-value as scoring criterion would lead to an elegant way of validating and searching peptide assignments at the same time.

As already mentioned before, it is very important for this method, to only keep the biggest peaks and remove the noise.

3.3.1 Results

This approach assigns around 1% of the peptides wrongly on the two datasets used. See Table 3.2 for more details.

Data	`sbv`	`exp041`
correctly assigned	98.9%	98.9%
wrongly assigned	1.1%	1.1%

Table 3.2: Results with the probability scoring.

Figure 3.8 shows a histogram where one can see the position of the correct peptide assignment in the answer of PeptideFind.

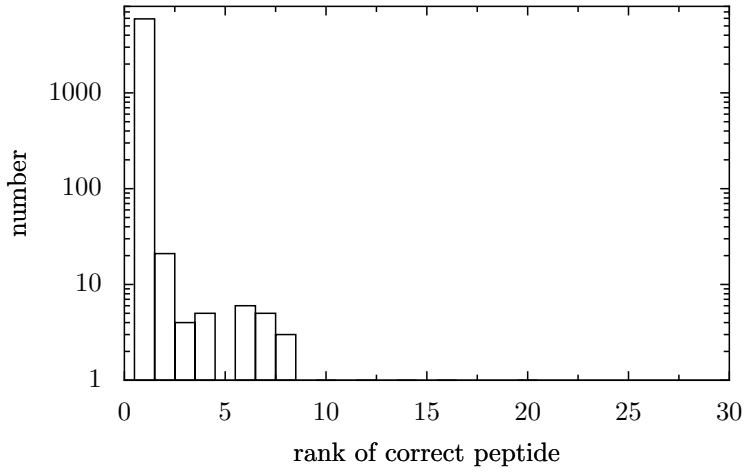

Figure 3.8: Position where the correct assignment occurs in the answer of a hypergeometric probability search.

3.3.2 Length independence of the Hypergeometric Probability Scoring

It seems that the hypergeometric probability scoring has a smaller tendency to favour smaller peptide sequences than the cross-correlation method, this is illustrated by Fig-

ure 3.9.

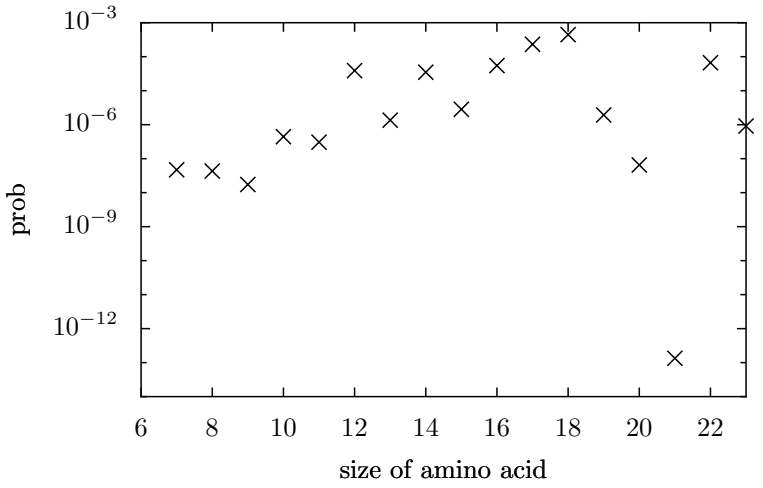

Figure 3.9: Length independence: The probability score between the theoretical spectrum of the correct assignment and the experimental spectrum. This score is then averaged over all sequences with the same length.

3.4 Conclusions

Both, the cross-correlation and hypergeometric probability scoring, lead to good results, considering the very few heuristics we used in our program. Especially the hypergeometric score shows to be very effective. One could possibly even improve on this, by already using the p-value for the search, not only for the validation. One would certainly need to run further tests to really compare the peptide assignments that our algorithms return, to other products, such as SEQUEST.

Although we feel that PEPTIDEFIND shows, that its possible to get good results without too many heuristics (often more sort of a hack than a heuristic), it seems clear that we would need further preprocessing steps to really outperform other software packages.

4 Validation Theory

In this chapter we describe three approaches to database search validation, that were proposed by other researchers in the field. We try to summarize the theoretical foundations of these methods in a consistent way and compare the underlying assumptions.

4.1 Validation based on the Hypergeometric Probability Model

The hypergeometric probability model, described in section 3.3, gives raise to several validation approaches, which we describe in this section.

4.1.1 χ^2-Test

A first method mentioned in [SY03] is a χ^2-test, where one does a statistical test whether the answer of the database search is meaningful. This is formulated as the null hypothesis H_0 which states that the peptide assignment is a random match. Formally this means that the number of matching peaks belongs to the hypergeometric distribution. H_1 is the alternative hypothesis, which states that the database search result is not a random match.

Assuming the null hypothesis is true, using the hypergeometric distribution, we can compute the expected number of peptides having i matches by

$$\eta_i := m\mathrm{P}_{N,K,n}(X = i),$$

where m is the number of peptides in the database, with a certain mass. We can then compare the expected number with the observed number of peptides having i matches by a χ^2-test

$$\bar{\chi}^2 = \sum_{i=0}^{2(\ell-1)} \frac{(\zeta_i - \eta_i)^2}{\eta_i},$$

where ζ_i is the observed number of peptides having i matches. We then numerically compute the so called p-value p_{val} of the χ^2 distribution for the test statistic $\bar{\chi}^2$,

$$p_{val} = P(\chi^2_{\mathrm{df}} > \bar{\chi}^2).$$

Where df are the degrees of freedom of the χ^2 distribution, for our case this is equal to the number of bins[1]. Given a significance level α, we can then either accept or reject

[1]For the result to be meaningful, one has to assert that $\eta_i > 5$, which has of course an influence on the number of degrees of freedom. For our case the degrees of freedom is just the number of terms of the sum.

the null hypothesis. If $p_{val} \leq \alpha$ we reject the null-hypothesis and say that the peptide assignment is *not* a random match, which means that the peptide assignment is correct. If $p_{val} > \alpha$, we accept H_0, which means that the peptide was wrongly assigned.

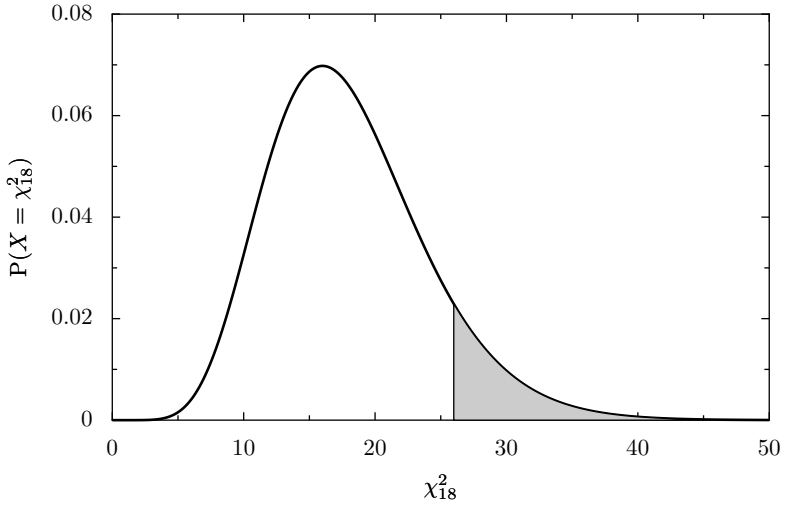

Figure 4.1: The chi-square density with 18 degrees of freedom. The p-value corresponds to the area of shaded region of the density, that to the right of the observed value of the test statistic. Here the p-value is selected such that it is equal to 0.1.

However in our tests, this method showed to be almost useless, as the χ^2-test tells us nearly always that the two distributions are different. We feel that this method gives us that bad results mainly because of one reason: The χ^2-test assumes that the two distributions under consideration are distorted by a normally distributed error. This might be an unjustified assumption for our case.

4.1.2 Cumulative Hypergeometric Distribution

As described in the introduction of the hypergeometric probability model, for a given peptide assignment $\boldsymbol{p}$, one can compute the number of matching non-zero elements with the experimental spectrum $\boldsymbol{e}$ and can then compute the probability of observing the configuration (N, K, n, k). But one can as well use the observed configuration (N, K, n, k), to compute a p-value p_{val}. As the hypergeometric distribution is a discrete distribution, one can do this quite easily by summing up over all $i < k$,

$$p_{val} = 1 - \sum_{i=0}^{k-1} P_{N,K,n}(X = i). \tag{4.1}$$

By comparing to Figure 3.7, one immediately sees that if p_{val} is near to zero or near to one, it is very unlikely that the observed number of matches is a random match. We are of course interested in the case where the p-value is close to 0, as we should

have more matching peaks for a correct peptide assignment, and not less, compared to random matches, as the peptide assignment is the result of a database search, where one selects a "similar" looking peptide. See Figure 4.2 for an illustration of a cumulative hypergeometric distribution function.

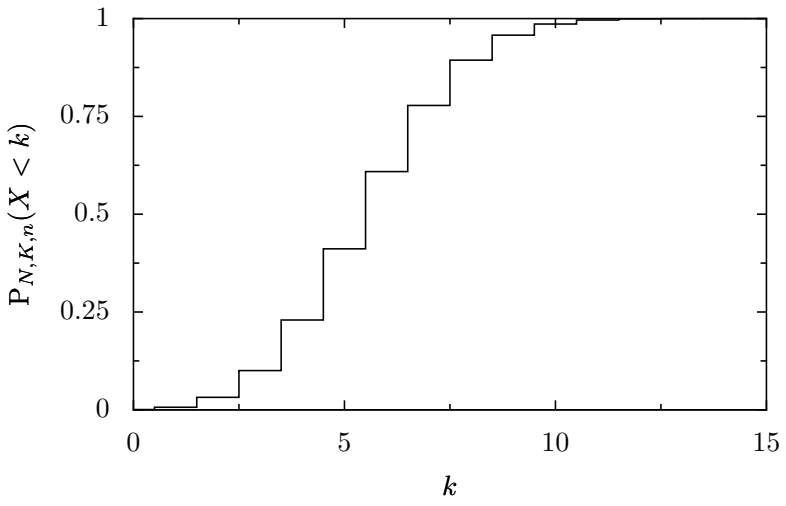

Figure 4.2: The cumulative distribution function for the density given in Figure 3.7.

4.1.3 Chebyshev Bound for the Confidence Interval

The *inequality of Chebyshev* provides us with an upper bound for the deviation from the mean, which we use in this subsection. Given a random variable X and $\lambda \in \mathbb{R}^2$ with $\lambda > 0$ the inequality of Chebyshev says that,

$$\mathrm{P}\left(|X - \mathbb{E}[X]| \geq \lambda\right) \leq \frac{\mathrm{Var}[X]}{\lambda^2}.$$

We can use the inequality of Chebyshev for an approximation of the p-value as in (4.1), without actually computing it, as both, the mean and variance of the hypergeometric distribution, can be computed analytically. The mean of the hypergeometric distribution as in (3.2) is

$$\mu = \mathbb{E}[X] = n\frac{K}{N},$$

and the variance is

$$\sigma^2 = \mathrm{Var}[X] = n\frac{K}{N}\left(1 - \frac{K}{N}\right)\frac{N-n}{N-1}.$$

Like this we can compute an upper bound for p_{val}, without ever using binomial coefficients, which are computationally inefficient. However in our experiments using this approximation didn't result in significant improvement of the running time and we thus rather used the exact p-value.

4.2 Linear Discriminant Analysis

In [KNKA02] Keller et al. use a Linear Discriminant Analysis to find a discrimination function between correct database answers and incorrect ones. Among other scores they use the cross-correlation ($Xcorr$) score and the difference between the first and second highest Xcorr score for all peptides queried from the database ($\Delta Xcorr$).

Most of the notation is taken from [HTF01]. For our problem we have two classes – correctly (class 1) and incorrectly (class 0) assigned peptides, we thus describe the Linear Discriminant Analysis for the special case of only two classes. We collect all our scores (e.g. cross-correlation or difference between the first and second highest cross-correlation score) in a vector $\boldsymbol{x}$. Using Bayes theorem we can compute the posterior probabilities

$$p(G = 1|X = \boldsymbol{x}) = \frac{p(X = \boldsymbol{x}|G = 1)\pi_1}{p(X = \boldsymbol{x}|G = 1)\pi_1 + p(X = \boldsymbol{x}|G = 0)\pi_0},$$

and

$$p(G = 0|X = \boldsymbol{x}) = \frac{p(X = \boldsymbol{x}|G = 0)\pi_0}{p(X = \boldsymbol{x}|G = 1)\pi_1 + p(X = \boldsymbol{x}|G = 0)\pi_0},$$

respectively. Here $p(X = \boldsymbol{x}|G = 1)$ and $p(X = \boldsymbol{x}|G = 0)$ denote the probabilities of observing score $\boldsymbol{x}$ given a correctly or incorrectly assigned peptide, respectively. Furthermore π_1 and π_0 denote the prior probabilities of a correct and incorrect peptide assignment, respectively.
If we can compute the two posterior probabilities, we can assign the label of the maximizing class to our peptide assignment during the validation phase. For the LDA one assumes that $p(X = \boldsymbol{x}|G = 1)$ and $p(X = \boldsymbol{x}|G = 0)$ are distributed according to a multivariate Gaussian

$$\phi_k(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_k)\right).$$

We further assume that the classes have a common covariance matrix $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}\ \forall k$.

Instead of maximizing the posterior distribution, one can compute the *linear discriminant functions*

$$\delta_1(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 - \frac{1}{2}\boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 + \log \pi_1,$$

and

$$\delta_0(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_0 - \frac{1}{2}\boldsymbol{\mu}_0^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_0 + \log \pi_0.$$

If we then compute $\delta_1 - \delta_0$ we get a simple decision rule: if the result is positive we say that the peptide assignment is correct, otherwise, we say that the assignment is wrong.

In practice we don't know the parameters of the Gaussian distributions, and will need to estimate them using our training data:

- $\hat{\pi}_k = N_k/N$, where N_k is the number of class-k observations;

- $\hat{\boldsymbol{\mu}}_k = \sum_{g_i=k} \boldsymbol{x}_i / N_k$;
- $\hat{\boldsymbol{\Sigma}} = \sum_{k=0}^{1} \sum_{g_i=k} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)^T / (N-2)$.

We are in the two class case and thus $k \in \{0, 1\}$.

The Linear Discriminant Analysis essentially linearly re-weights the input score $\boldsymbol{x}$ by weights $\boldsymbol{w}$, and we get only one score,

$$x^\star = w^{(0)} + \sum_{i=1}^{n} w^{(i)} x^{(i)},$$

where n is the size of $\boldsymbol{x}$. This interpretation helps us to identify important and less important input scores.

Figure 4.3 shows data generated by two Gaussians (with mean $\boldsymbol{\mu}_0 = (0.2, 0.2)$ and $\boldsymbol{\mu}_1 = (0.5, 0.5)$, respectively) and the same covariance matrix for both Gaussians. Furthermore the results of the LDA implementation of PeptideFind are plotted: estimated means, the covariance matrix and the decision boundary. However it is important to note that in real nature we will never have data that is really sampled from a Gaussian, as it is here the case.

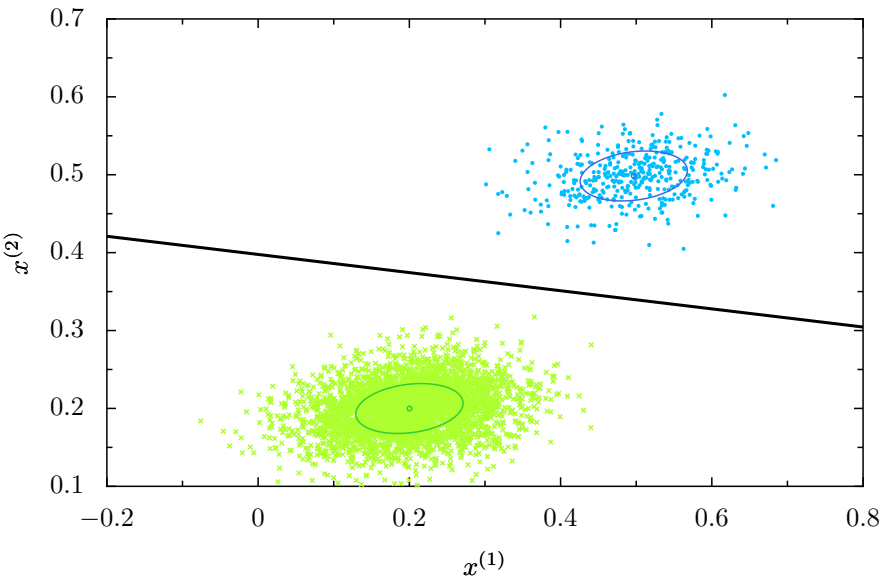

Figure 4.3: Results of the LDA implementation of PeptideFind on test data.

4.3 Gaussian Mixture Model (EM Algorithm)

In the previous section we assumed that the class labels (in our case correct or incorrect peptide assignment) are given, so we are in a supervised machine learning problem.

However, the labels are usually very costly, as each peptide assignment would need to be validated manually by a human, and you would still not be assured that the label is correct. If other approaches are taken, like the ones described in section 5.1.1, it is again not guaranteed that the label is correct. Given this fact, it makes sense to consider unsupervised algorithms to tackle the problem of peptide assignment validation. In this section we use a Expectation-Maximization (EM) algorithm to identify a possible bi-modality in the cross-correlation score distribution. In the following we assume we are given a data set of N peptide assignment scores $\mathcal{D} = \{\boldsymbol{x}_i\}$ for $i = 1, \ldots, N$, of which we want to model the density, and in particular identify the bi-modality. The usage of an EM algorithm for peptide assignments was first introduced in [KNKA02], however there the EM algorithm, as far as we understand, is only used to find a good threshold for the LDA. The notation of this section is again greatly inspired by the corresponding section in [HTF01].

We would like to model a bi-modality, in our case correct and incorrect assignments, as a *mixture of two Gaussians*:

$$\begin{aligned} X_1 &\sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \\ X_2 &\sim \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \\ X &= (1 - \Delta) \cdot X_1 + \Delta \cdot X_2, \end{aligned}$$

where $\Delta \in \{0, 1\}$ with $P(\Delta = 1) = \pi$. This *generative* representation is explicit: generate a $\Delta \in \{0, 1\}$ with probability π, and then depending on the outcome, deliver either X_1 or X_2. Let $\phi_{\boldsymbol{\theta}}(\boldsymbol{x})$ denote the normal density with parameters $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Then the density of X is

$$g_X(\boldsymbol{x}) = (1 - \pi)\phi_{\boldsymbol{\theta}_1}(\boldsymbol{x}) + \pi\phi_{\boldsymbol{\theta}_2}(\boldsymbol{x}).$$

Now suppose we wish to fit this model to our given data (in our case the distribution of the peptide assignment scores) by maximum likelihood. The parameters are

$$\boldsymbol{\theta} = (\pi, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = (\pi, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2).$$

The log-likelihood based on the N training cases is

$$\ell(\boldsymbol{\theta}, \mathcal{D}) = \sum_{i=1}^{N} \log[(1 - \pi)\phi_{\boldsymbol{\theta}_1}(\boldsymbol{x}_i) + \pi\phi_{\boldsymbol{\theta}_2}(\boldsymbol{x}_i)].$$

Direct maximization of $\ell(\boldsymbol{\theta}, \mathcal{D})$ is quite difficult numerically, because of the sum of terms inside the logarithm. There is, however, a simpler approach. We consider unobserved latent variables Δ_i taking values 0 or 1: if $\Delta_i = 1$ then X_i comes from model 2, otherwise it comes from model 1. Suppose we knew the values of the Δ_i's. Then the

log-likelihood would be

$$\ell_0(\boldsymbol{\theta}; \mathcal{D}, \boldsymbol{\Delta}) = \sum_{i=1}^{N} [(1-\Delta_i) \log \phi_{\boldsymbol{\theta}_1}(\boldsymbol{x}_i) + \Delta_i \log \phi_{\boldsymbol{\theta}_2}(\boldsymbol{x}_i)] + \sum_{i=1}^{N} [(1-\Delta_i) \log \pi + \Delta_i \log(1-\pi)] \tag{4.2}$$

and the maximum likelihood estimates of $\boldsymbol{\mu}_1$ and $\boldsymbol{\Sigma}_1$ would be the sample mean and variance for those data with $\Delta_i = 0$, and similarly those for $\boldsymbol{\mu}_2$ and $\boldsymbol{\Sigma}_2$ would be the sample mean and variance of the data with $\Delta_i = 1$.

Algorithm 4.1 EM algorithm for two-component Gaussian mixture.

1. Take initial guesses for the parameters $\hat{\boldsymbol{\mu}}_1, \hat{\boldsymbol{\Sigma}}_1, \hat{\boldsymbol{\mu}}_2, \hat{\boldsymbol{\Sigma}}_2, \hat{\pi}$.

2. *Expectation Step*: compute the responsibilities

$$\hat{\gamma}_i = \frac{\hat{\pi}\phi_{\hat{\boldsymbol{\theta}}_2}(\boldsymbol{x}_i)}{(1-\hat{\pi})\phi_{\hat{\boldsymbol{\theta}}_1}(\boldsymbol{x}_i) + \hat{\pi}\phi_{\hat{\boldsymbol{\theta}}_2}(\boldsymbol{x}_i)}, \ i = 1, 2, \ldots, N.$$

3. *Maximization Step*: compute the weighted means and variances:

$$\hat{\boldsymbol{\mu}}_1 = \frac{\sum_{i=1}^{N}(1-\hat{\gamma}_i)\boldsymbol{x}_i}{\sum_{i=1}^{N}(1-\hat{\gamma}_i)}, \qquad \hat{\boldsymbol{\Sigma}}_1 = \frac{\sum_{i=1}^{N}(1-\hat{\gamma}_i)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_1)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_1)^t}{\sum_{i=1}^{N}(1-\hat{\gamma}_i)},$$

$$\hat{\boldsymbol{\mu}}_2 = \frac{\sum_{i=1}^{N}\hat{\gamma}_i\boldsymbol{x}_i}{\sum_{i=1}^{N}\hat{\gamma}_i}, \qquad \hat{\boldsymbol{\Sigma}}_2 = \frac{\sum_{i=1}^{N}\hat{\gamma}_i(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_2)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_2)^t}{\sum_{i=1}^{N}\hat{\gamma}_i},$$

   and the mixing probability $\hat{\pi} = \sum_{i=1}^{N}\hat{\gamma}_i/N$.

4. Iterate steps 2 and 3 until convergence.

Since the values of the Δ_i's are actually unknown, we proceed in an iterative fashion, substituting for each Δ_i in (4.2) its expected value

$$\gamma_i(\boldsymbol{\theta}) = \mathbb{E}(\Delta_i | \boldsymbol{\theta}, \mathcal{D}) = P(\Delta_i = 1 | \boldsymbol{\theta}, \mathcal{D}),$$

also called the *responsibility* of model 2 for observation i. We use a procedure called the EM algorithm, given in Algorithm 4.1 for the special case of Gaussian mixtures. In the *expectation* step, we do a soft assignment of each observation to each model: the current estimates of the parameters are used to assign responsibilities according to the relative density of the training points under each model. In the *maximization* step, these

responsibilities are used in weighted maximum-likelihood fits to update the estimate of the parameters.

A good way to construct initial guesses for $\hat{\boldsymbol{\mu}}_1$ and $\hat{\boldsymbol{\mu}}_2$ is simply to choose two of the $\boldsymbol{x}_i$ at random. Both $\hat{\boldsymbol{\Sigma}}_1$ and $\hat{\boldsymbol{\Sigma}}_2$ can be set equal to the overall sample variance $\sum_{i=1}^{N}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^2/N$. The mixing proportion $\hat{\pi}$ can be started at the value 0.5.

Note that the actual maximizer of the likelihood occurs when we put a spike of infinite height at any one data point, that is $\hat{\boldsymbol{\mu}}_1 = \boldsymbol{x}_i$ for some i and $\boldsymbol{\Sigma}_1 = 0$. This gives infinite likelihood, but is not a useful solution. Hence we are actually looking for a good local maximum of the likelihood, one for which $\hat{\boldsymbol{\Sigma}}_1, \hat{\boldsymbol{\Sigma}}_2 > 0$. In PeptideFind, we run the EM algorithm with a number of different initial guesses for the parameters, and choose the run that gave us the highest maximizing log-likelihood. Figure 4.4 shows the Gaussian Mixture learned by the algorithm implemented in PeptideFind.

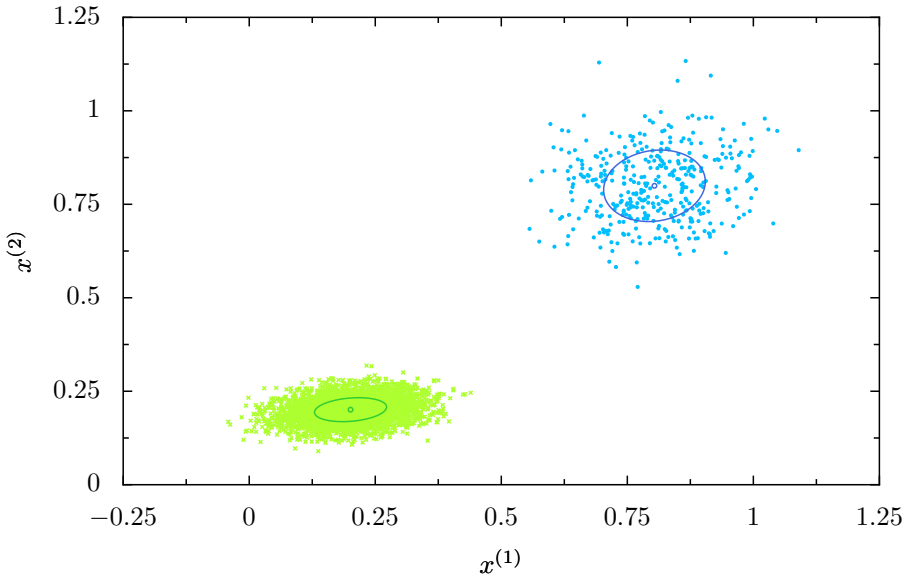

Figure 4.4: Results of the GMM implementation of PeptideFind on artificial normally distributed test data.

To now actually validate a peptide assignment, there exist at least two ways of doing so:

- Use the learned Gaussian distributions for the classification. This is straightforward: as the EM algorithm will give us all the parameters of the two Gaussians we can simply compute

$$\gamma = \frac{\hat{\pi}\phi_{\hat{\boldsymbol{\theta}}_2}(\boldsymbol{x})}{(1-\hat{\pi})\phi_{\hat{\boldsymbol{\theta}}_2}(\boldsymbol{x}) + \hat{\pi}\phi_{\hat{\boldsymbol{\theta}}_2}(\boldsymbol{x})},$$

  for a peptide assignment with scores $\boldsymbol{x}$. $\gamma \in [0, 1]$ indicates the probability of being

a correct assignment: values close to 0 are wrong assignments, values close to 1 should be correct assignments.

- Use again a linear discriminant. Compute the between-scatter $\hat{\mathbf{\Sigma}} = (1-\hat{\pi})\hat{\mathbf{\Sigma}}_1 + \hat{\pi}\hat{\mathbf{\Sigma}}_2$ and then use the same approach as in section 4.2 to discriminate between correct and incorrect labels.

Although the first approach would be the way to go in a picture-perfect world (i.e. everything is normally distributed), the latter is the clear winner for non-Gaussian distributed data. To see this, compare to Figure 4.5.

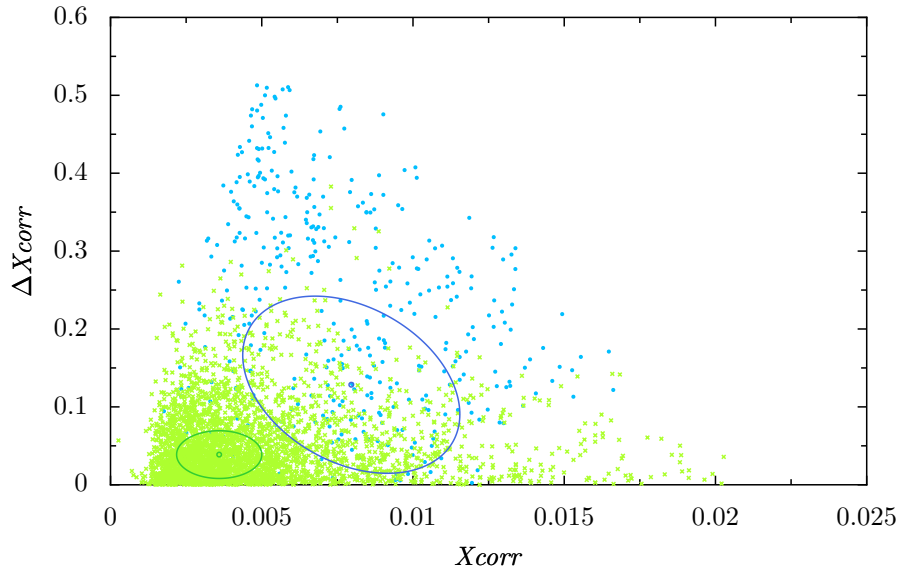

Figure 4.5: The GMM implementation of PeptideFind for real peptide assignment scores (instead of normally distributed artificial data).

Points with both, large $Xcorr$ and $\Delta Xcorr$, for which we can be pretty sure that they belong to the "correct assignments" class, will have smaller values of γ, than the ones close to the mean of the "correct assignments" class. We however want a validation method that does not only discriminate between correct and incorrect assignments, but does as well assign large values to clearly correct assignments. The second approach has this two properties.

5 Results and Discussion

In this chapter we describe the tests we used to determine the strengths and weaknesses of the different validation approaches described in chapter 4. After the description of the tests and their results, we try to interpret them and give some hints which algorithm should be used in practice.

We tested the following validation methods: hypergeometric p-value (for both search scorings), the LDA and the Gaussian Mixture Model. We didn't include the χ^2 square validation, as it seems to not work at all, and the Chebyshev approximation, as the exact computation of the p-value is already sufficiently fast. For the Gaussian Mixture Model we first cluster the data, to get the two means and the covariance matrices, afterwards we estimate a between-scatter and proceed like in the LDA validation, it is thus just as means of using the LDA on unlabeled data.

5.1 Data Sets and Evaluation

As in the previous tests, we used `contArath.fasta` as our FASTA database (it contains 28760 proteins). As test data we used `dta_summary.txt`, a collection of around 60000 MS/MS spectra, and as a second data set `summaryA.txt`, a collection of around 14000 spectra. Other than the summaries, used in chapter 3, these data sets are however unlabeled. So we do not know the correct assignment of the spectra. It is important to test the validation algorithms on unlabeled data, as these are the circumstances you have to deal with in real life.

5.1.1 How do we know what's right?

To compare the algorithms, we will however need to know which assignments are correct and which not. We basically used two heuristics to solve this problem:

- Search in both, the normal FASTA database and as well in the inverted one (where we just reverse each protein of the original database). If we then get a peptide assignment from the inverse database we know that the assignment is wrong. This method is proposed in [KSC+05].
  If the hypergeometric p-value (or the LDA output) is indeed a good measurement of the quality of an assignment, we should get very few assignments from the inverted database for scores with a small p-value (large LDA output, respectively).
- One carefully prepares a second FASTA database containing a small number of proteins of which we know that they occur in the MS/MS spectra collection

(`omics_control_proteinmix.fasta` and `summaryA.txt`, respectively). If a peptide assignment originates from the second database, we assume that it is a correct one, otherwise it is a wrong assignment. This is proposed in [KPN+02].
If the hypergeometric p-value (or the LDA output) is indeed a good measurement of the quality of an assignment, it should give very few assignments from the big peptide database for scores with a small p-value (large LDA output, respectively).

5.1.2 Finding a good threshold

In practice one would need to define a threshold, according which one either rejects or accepts the assignments. For the evaluation, we however don't need this, because we are only interested in the ROC curve (or precision-recall) and not in an actual decision. In this subsection we nevertheless include some hints, how one could select an appropriate threshold.

The validation with the hypergeometric p-value has the major advantage, that the calibration for a new mass spectrometer is quite easy: We define a significance level α and let the database search run for some spectra of the particular mass spectrometer with the database containing both, the normal peptides and the inverted ones. We then select the threshold p_{val} for which we accept a proportion of α assignments out of the inverted database. The only thing we need to remember is the p-value p_{val}: we then accept assignments with a p-value below p_{val} and reject the other assignments. In theory one would assume that p_{val} should be equal to α, as significance is nothing other than another name for the p-value. However, this is only true, if the assignments from the reverted database are truly random matches, which isn't the case in practice, as the assignments were intentionally selected looking as similar as possible to the experimental MS/MS spectrum during the database search. We thus will get much smaller p-values as a threshold, in our tests the threshold usually was around 10^{-8} for a significance of 3%.

The validation with the LDA is slightly more involved, as the LDA has to be trained on some data. In our tests we just trained it on `summaryB.txt` from the omics protein mix. The LDA might however look quite different for other mass spectrometer, which is a major problem of this approach. Finding a threshold is afterwards similar to the hypergeometric model. Only that we accept all values larger than the threshold, whereas we accepted all the values smaller than the threshold in the hypergeometric model.

5.1.3 Dealing with the Randomness

The approach with the inverse database has one smaller technical problem: A random validation, which just flips a coin to decide whether it accepts or rejects the assignment has a probability of success of around 1/2 (assuming that both, assignments from the normal and inverse database have the same probability). The significance level α has thus to be replaced by $\alpha/2$ for the evaluations with the inverse database.

5.2 Receiver Operating Characteristic (ROC)

We will use ROC curves to show the results of the validation methods. In this section we repeat the information a ROC plot shows, and we as well shortly discuss how to compute it.

A ROC plot as in Figure 5.1, shows the sensitivity plotted against the specificity. They are defined as follows:

Sensitivity: probability of predicting correct assignment given true state is correct assignment.

$$sens = \frac{tp}{tp + fn}$$

Specificity: probability of predicting wrong assignment given true state is wrong assignment.

$$spec = \frac{tn}{fp + tn}$$

Here true positive (for short tp) denotes the number of assignments predicted as correct assignments by the validation algorithm, being indeed correct assignments. The other definitions are similar.

To actually compute a ROC curve, we use the score returned by the validation algorithms (which in our case is continuous). We can then order the data according to this score, go through the data and compute for each possible cut-off the sensitivity and specificity, this will lead to a plot as in Figure 5.1.

The area under curve (AUC) is often used as a measurement of the quality of a classifier. The best possible classifier will give an AUC of 1, while a random classifier will lead to an AUC of 0.5. However the AUC is a bad measurement for peptide assignment validation, as the data returned by the mass spectrometer will just not allow a identification of correct assignments for more than around 10% of the spectra. Because of this, we are mostly interested in the area of specificity in the interval of $[0.95, 1]$, as an algorithm, that returns more than 5% wrongly assigned peptides is not useful at all in practice. We will thus always include a second ROC curve that shows only this specific area.

It is important to note, that a ROC curve (or equivalently the sensitivity and specificity) give us no information, about the proportion of incorrectly selected peptide assignments out of the selected peptide assignments, or as a formula $fp/(tp + fp)$. This is however the information we are interested in, as we usually ask: how many wrong assignments can we expect in the validated peptide assignments? To answer such questions, one uses the precision and recall, as described in section 5.3. We still include the ROC curves in this project report, as it is for example used in [KSC+05] to evaluate different validation and search methods.

5.3 Precision-Recall

Precision and recall are a way to measure how well the retrieved information matches the intended information and is especially important in the context of information retrieval. They are defined as follows:

Precision: The proportion of correct peptide assignments to all the assignments retrieved:

$$precision = \frac{tp}{tp + fp}$$

Recall: The proportion of correct peptide assignments retrieved, out of all relevant assignments available

$$recall = \frac{tp}{tp + fn}$$

In contrast to specificity and sensitivity, we can now really measure the proportion of incorrect assignments out of the assignments viewed as correct by the validation step.

5.4 Results

5.4.1 Validation of dta_summary

In this subsection we applied the different combinations of validation and search algorithms to the experimental MS/MS spectra in `dta_summary.txt`. Figure 5.1 shows a graph containing several ROC curves: (1) hypergeometric p-value validation of peptide assignments returned by the hypergeometric probability search (prob/prob), (2) hypergeometric p-value validation of peptide assignments returned by the cross-correlation search (xcorr/prob), (3) LDA validation of peptide assignments returned by the cross-correlation search (xcorr/lda), where the LDA was trained on `summaryB.txt` (which is data from quite a different mass spectrometer) and (4) Gaussian Mixture Model validation of assignments returned by the cross-correlation search (xcorr/gmm).

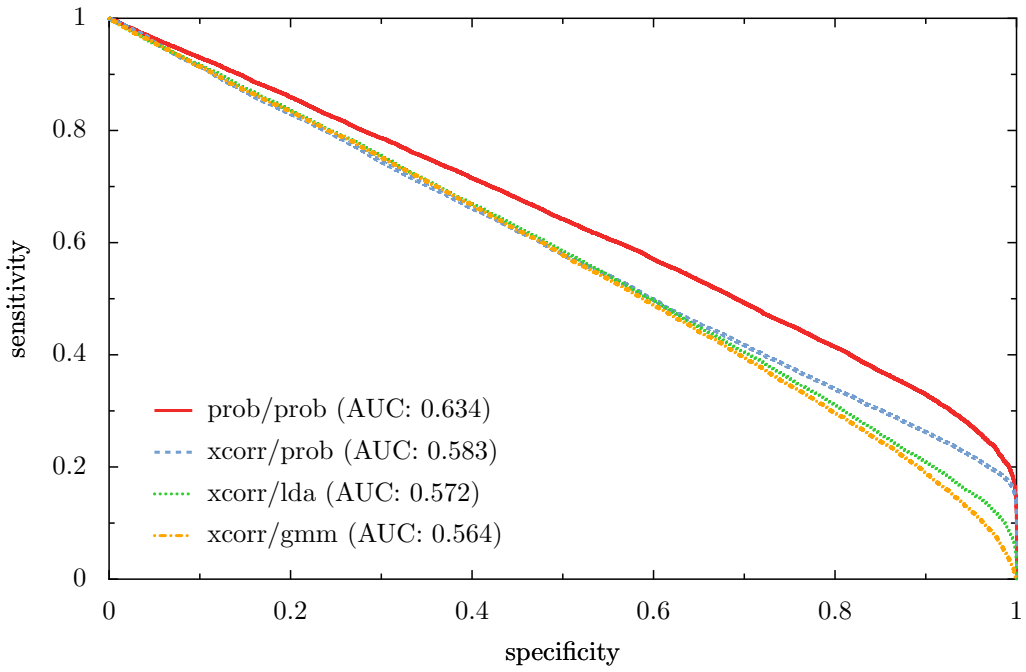

Figure 5.1: ROC curves for different combinations of validation and search algorithms for `dta_summary.txt`.

See Figure 5.2 for the relevant interval of specificity. One can clearly see that the combination of hypergeometric search and p-value validation gives better results than the other combinations. As the LDA is trained on data from another spectrometer, this however shouldn't come as a surprise. See section 5.5 for a more detailed discussion of the problems with the LDA validation.

As previously discussed, we think that precision-recall is the measurement of choice in the context of peptide assignment validation. We thus as well include precision-recall curves, see Figure 5.3 and Figure 5.4. For this dataset however the differences between ROC and precision-recall results are not drastically: The validation with the p-value

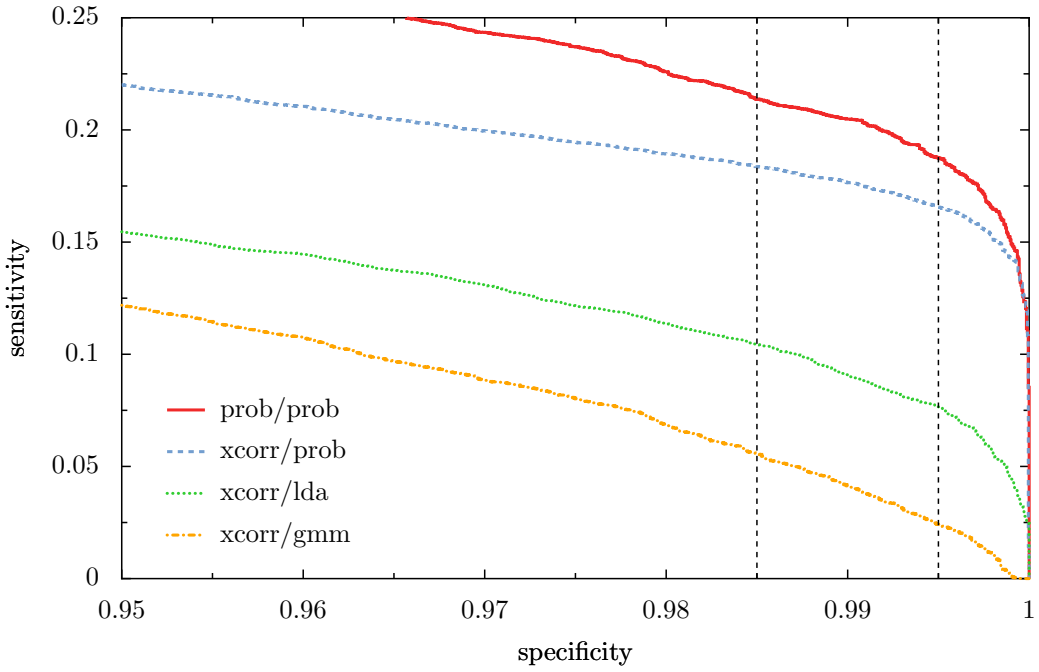

Figure 5.2: More detailed ROC curves for different combinations of validation and search algorithms for `dta_summary.txt`.

does look like a clear winner for both measurements. This is in sharp contrast to the next subsection, where the ROC results do not match the ones of the precision-recall.

	precision 97%		precision 99%	
	accepted	threshold	accepted	threshold
prob/prob	10.6%	$5.58 \cdot 10^{-9}$	8.9%	$4.25 \cdot 10^{-10}$
xcorr/prob	8.9%	$8.47 \cdot 10^{-8}$	7.9%	$4.3 \cdot 10^{-9}$
xcorr/lda	2.3%	3.17	1.5%	5.09
xcorr/gmm	0.0%	-	0.0%	-

Table 5.1: Two precision levels and their corresponding thresholds together with the number of peptide assignments accepted by the validation algorithm for `dta_summary.txt`.

Table 5.1 shows some more information about the most interesting significance levels (1% and 3%, respectively). A combination of a hypergeometric search and a validation with the p-value leads to around 9% accepted peptide assignments (with a significance of 99%). The really bad results of the Gaussian Mixture Model are discussed in section 5.5.

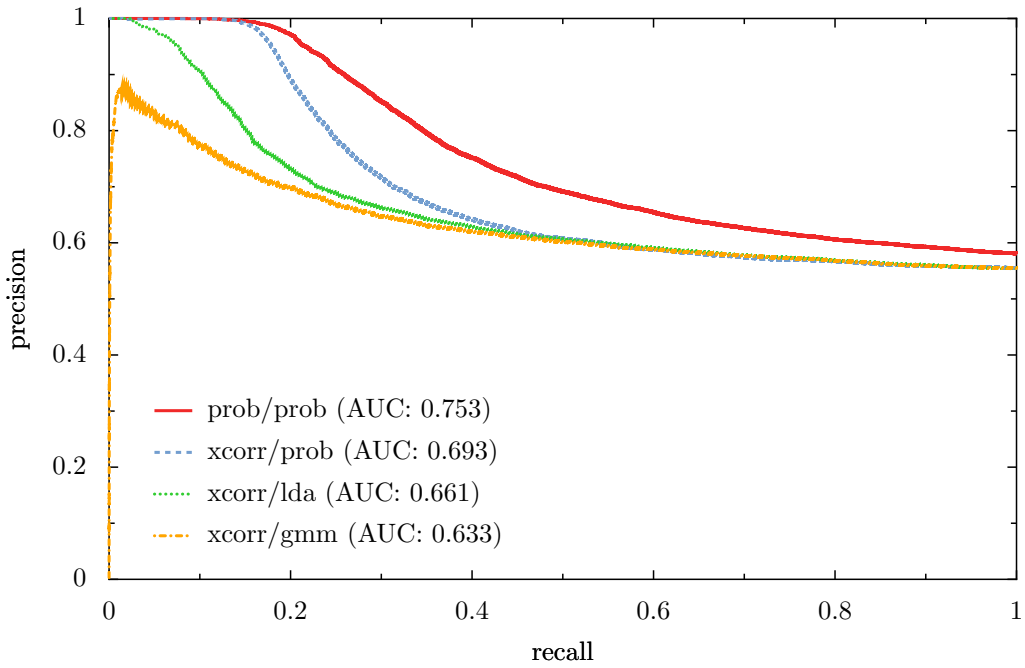

Figure 5.3: Precision-recall curves for different combinations of validation and search algorithms for `dta_summary.txt`.

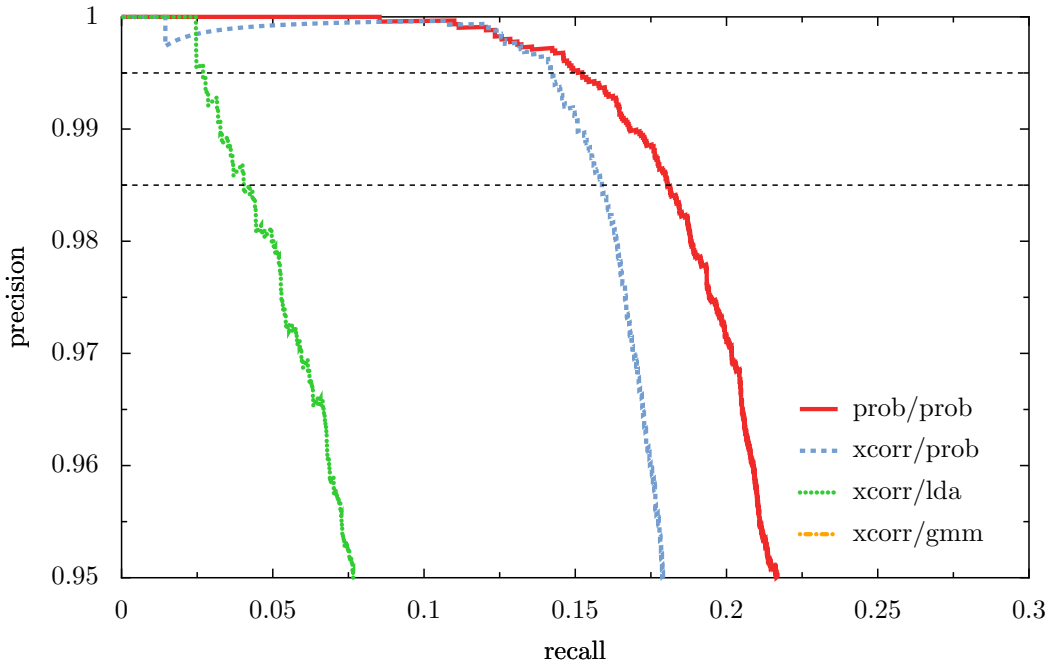

Figure 5.4: Detailed precision-recall curves for different combinations of validation and search algorithms for `dta_summary.txt`.

5.4.2 Validation of summaryA – Protein Mix

In this subsection we applied the different combinations of validation and search algorithms to the experimental MS/MS spectra in `summaryA.txt`. As we had better labels available as in subsection 5.4.1, namely the information whether the assignment originates from the omics protein mix or not, we searched against the contamined Arabidopsis database and as a second (small) database the omics protein mix and did not include the inverse database. Figure 5.5 shows a graph containing several ROC curves: (1) hypergeometric p-value validation of peptide assignments returned by the hypergeometric probability search (prob/prob), (2) hypergeometric p-value validation of peptide assignments returned by the cross-correlation search (xcorr/prob), (3) LDA validation of peptide assignments returned by the cross-correlation search (xcorr/lda), where the LDA was trained on `summaryB.txt` and (4) Gaussian Mixture Model validation of assignments returned by the cross-correlation search (xcorr/gmm).

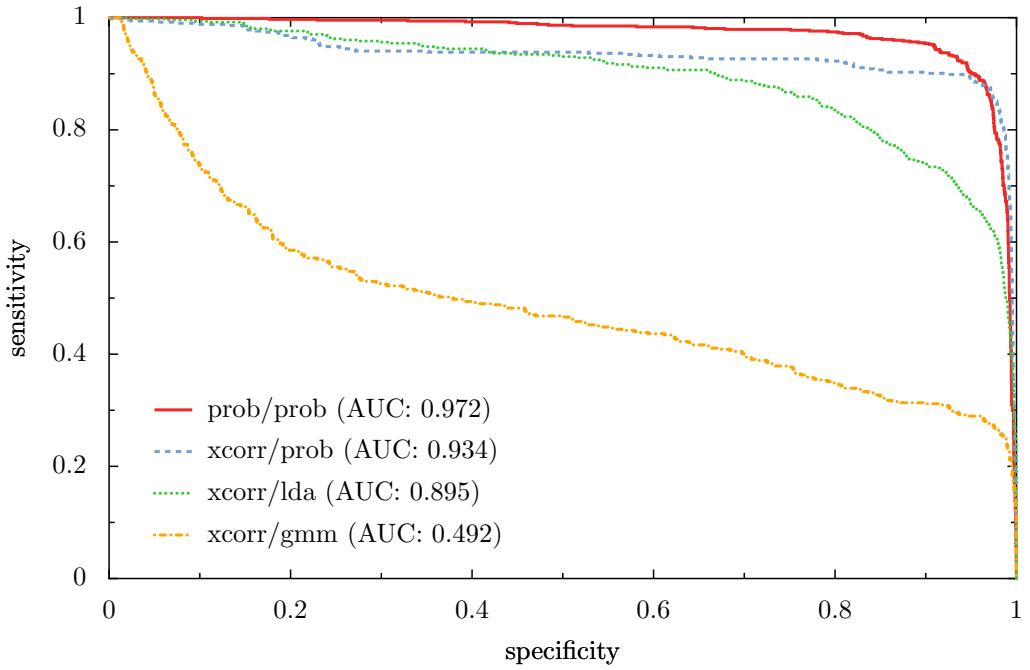

Figure 5.5: ROC curves for different combinations of validation and search algorithms for `summaryA.txt`.

See Figure 5.6 for the relevant interval of specificity. While the results of the LDA are promising, the ones of the Gaussian Mixture Model are certainly disappointing, as if the EM algorithm computes a good clustering (in the sense that it's similar to the labeled data), the results should be around the same as with the LDA validation. However, looking at the results in some more depth, we note that the scores of the search algorithms are by far not normally distributed and thus the Gaussian Mixture Model does not lead to good results. For a more detailed discussion, see again section 5.5.

Looking at the ROC curves, the p-value validation does still look like a clear winner.

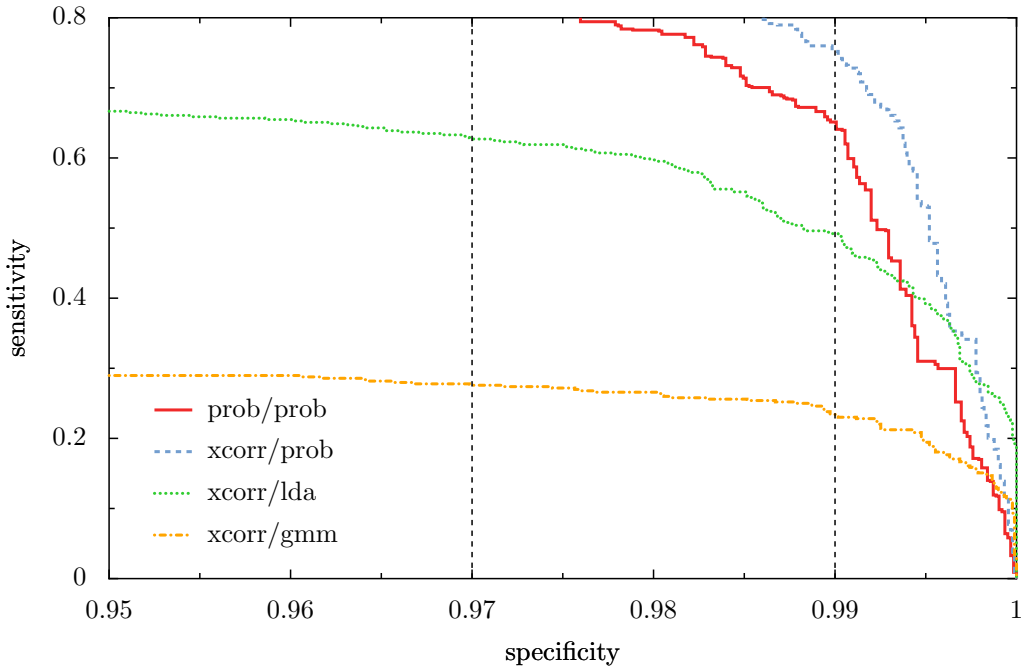

Figure 5.6: More detailed ROC curves for different combinations of validation and search algorithms for `summaryA.txt`.

The improvement of the LDA can simply be explained by the learning step: we learn from scores of the same mass spectrometer, and thus the discrimination should be much better than in the previous subsection, where we trained the LDA on data from another mass spectrometer.

For the precision-recall, the results look quite different, compare Table 5.2 and Figure 5.8: the LDA clearly outperforms the hypergeometric p-value validation. While the latter only accepts 0.01% with a significance of 1%, the LDA validation accepts 1.1% of the assignments.

	precision 97%		precision 99%	
	accepted	threshold	accepted	threshold
prob/prob	0.01%	0.0	0.01%	0.0
xcorr/prob	0.01%	0.0	0.01%	0.0
xcorr/lda	1.2%	9.03	1.1%	9.65
xcorr/gmm	0.5%	4.04	0.1%	4.54

Table 5.2: Two precision levels and their corresponding thresholds together with the number of peptide assignments accepted by the validation algorithm for `summaryA.txt`.

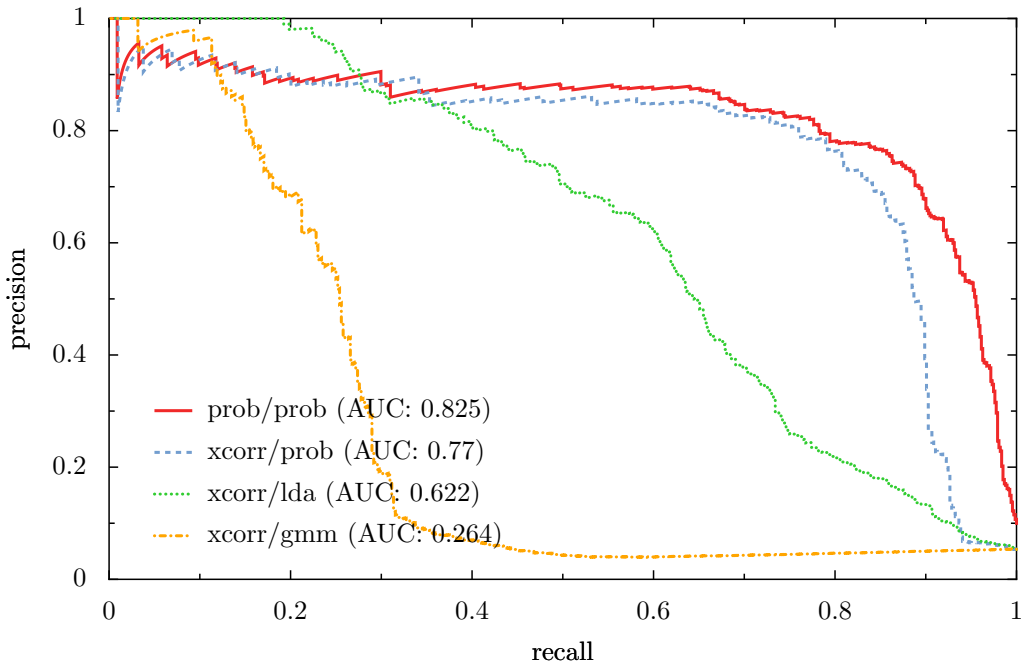

Figure 5.7: Precision-recall curves for different combinations of validation and search algorithms for `summaryA.txt`.

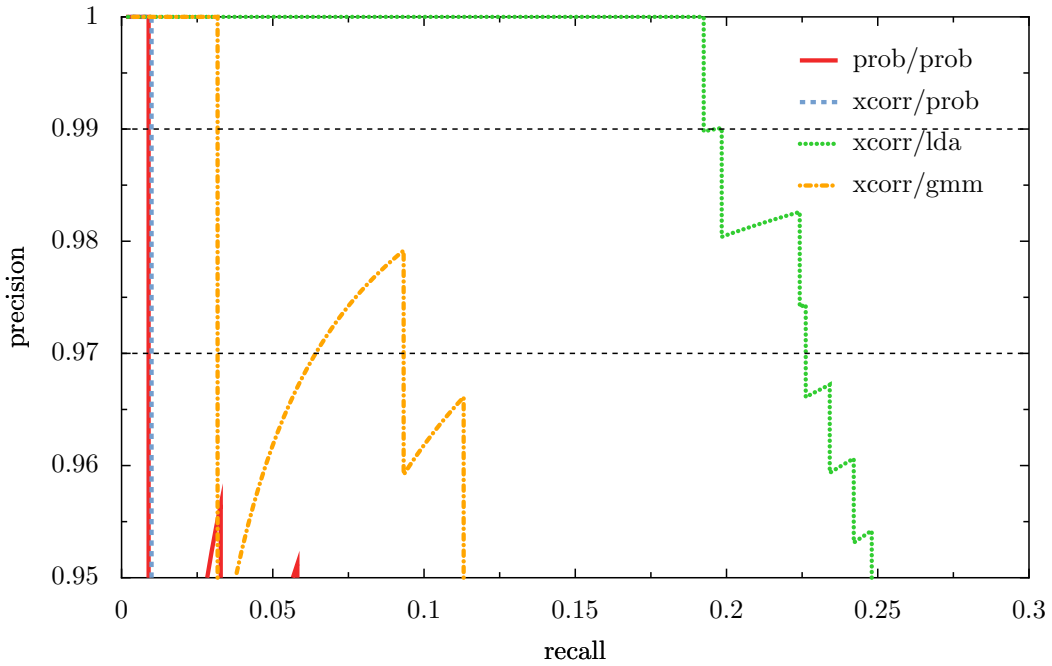

Figure 5.8: Detailed precision-recall curves for different combinations of validation and search algorithms for `summaryA.txt`.

5.4.3 Validation of summaryA – Inverse Database

In this subsection we applied the different combinations of validation and search algorithms to the experimental MS/MS spectra in `summaryA.txt`. Comparing to the previous subsection we however do not use the protein mix approach, to get the correctness label, but use the inverse database method. So the setting is the same as in subsection 5.4.1, we only use experimental spectra from another mass spectrometer. Figure 5.9 shows a graph containing several ROC curves: (1) hypergeometric p-value validation of peptide assignments returned by the hypergeometric probability search (prob/prob), (2) hypergeometric p-value validation of peptide assignments returned by the cross-correlation search (xcorr/prob), (3) LDA validation of peptide assignments returned by the cross-correlation search (xcorr/lda), where the LDA was trained on `summaryB.txt` and (4) Gaussian Mixture Model validation of assignments returned by the cross-correlation search (xcorr/gmm).

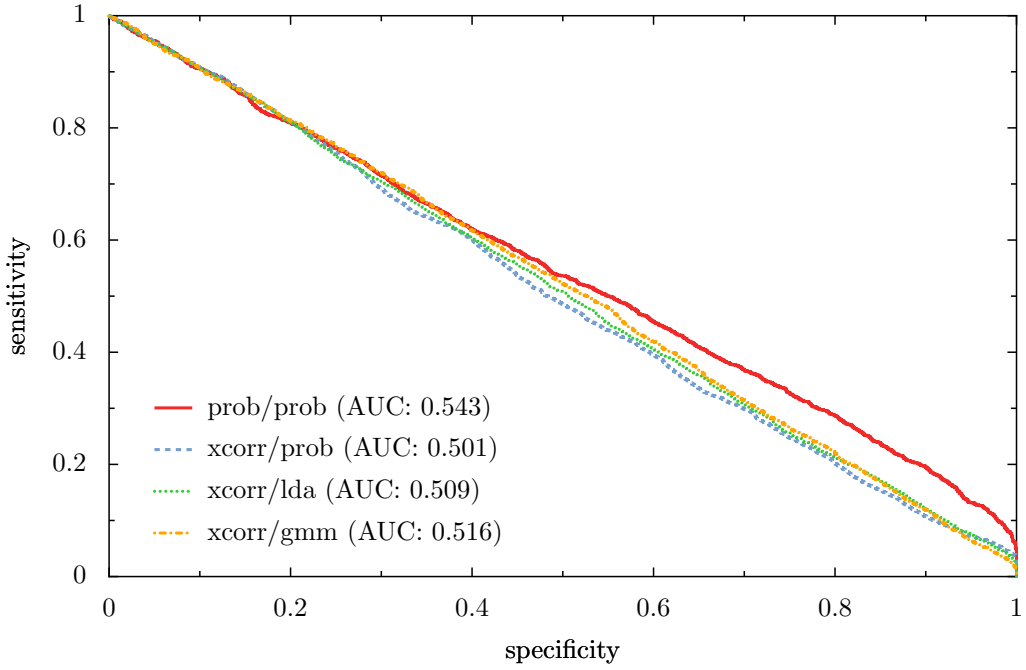

Figure 5.9: ROC curves for different combinations of validation and search algorithms for `summaryA.txt`.

The results we got look quite similar to the ones in subsection 5.4.1. Again the hypergeometric p-value validation leads to better results for both measurements, ROC and precision-recall. As both tests were done using the inverse database approach, one has to ask, whether the peptides selected from the inverse database show the same characteristics as the incorrect assignments of the normal database. If this is not the case, the inverse database approach is not meaningful. We try to answer this question in more detail in section 5.5.

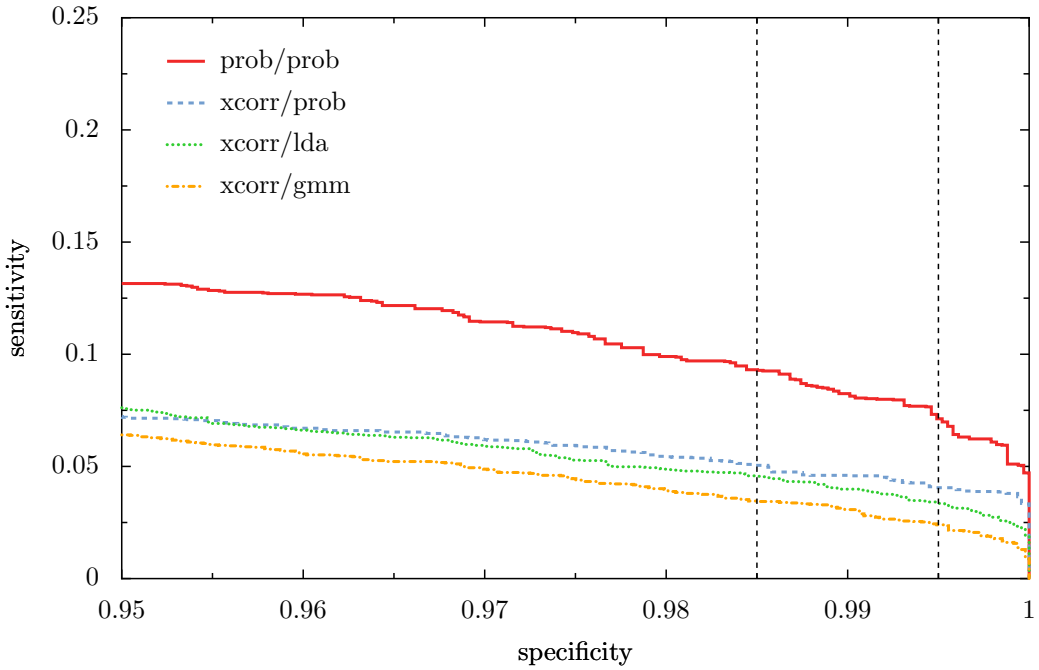

Figure 5.10: More detailed ROC curves for different combinations of validation and search algorithms for `summaryA.txt`.

	precision 97%		precision 99%	
	accepted	threshold	accepted	threshold
prob/prob	2.7%	$4.22 \cdot 10^{-12}$	2.4%	$1.33 \cdot 10^{-12}$
xcorr/prob	1.8%	$8.27 \cdot 10^{-10}$	1.7%	$1.36 \cdot 10^{-10}$
xcorr/lda	1.1%	4.49	0.9%	6.09
xcorr/gmm	0.4%	4.96	0.4%	4.96

Table 5.3: Two precision levels and their corresponding thresholds together with the number of peptide assignments accepted by the validation algorithm for `summaryA.txt`.

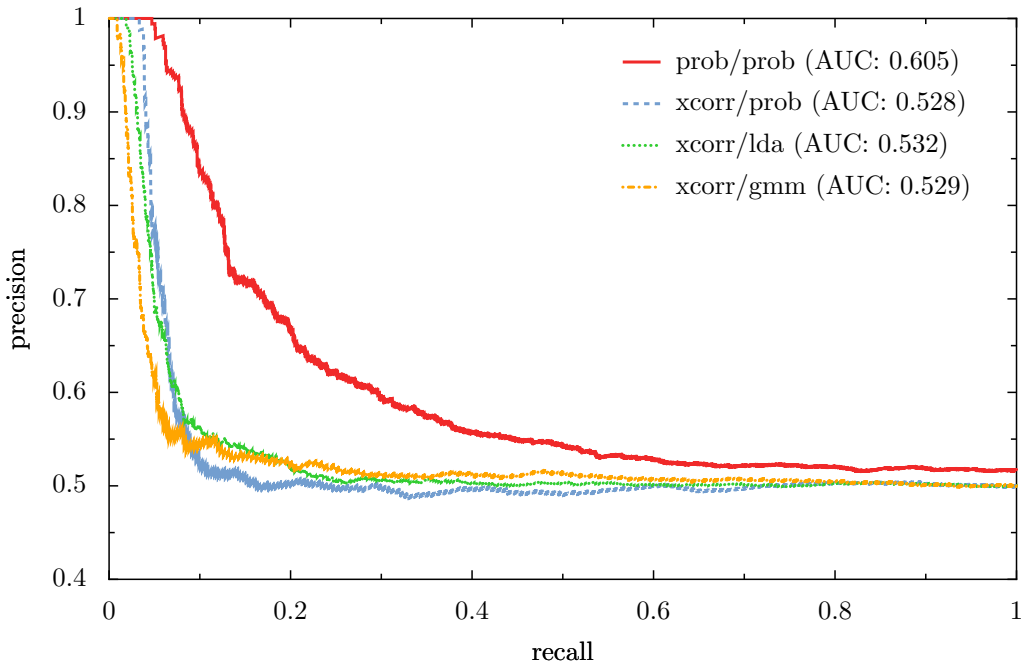

Figure 5.11: Precision-recall curves for different combinations of validation and search algorithms for `summaryA.txt`.

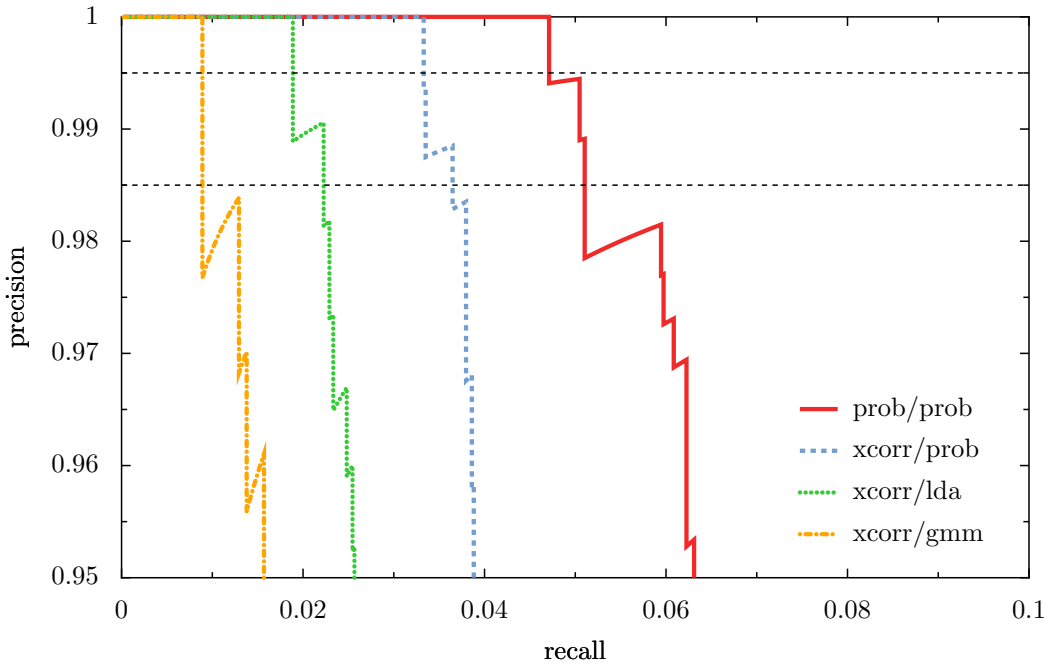

Figure 5.12: Detailed precision-recall curves for different combinations of validation and search algorithms for `summaryA.txt`.

5.5 Discussion

The tests did not lead to a clear winner. On the dataset, which should describe best the real conditions (omics proteinmix), the LDA is the best validation strategy. However, there are two issues with it:

1. The proportion of accepted peptide assignments is very small and
2. it is unclear how one should train the LDA, if we do not know the correct labels. Using the labeled data from another mass spectrometer does not seem very ingenious.

On the other hand the p-value validation has the appealing feature of being very simple. We do not need any training data to learn a discrimination function. This is a clear advantage, compared to the LDA. We only need to find a good threshold, which seems quite a bit easier than producing labeled scores for a mass spectrometer. There exist however as well a big problem with the p-value validation: It did not lead to good results on the omics proteinmix. As this test data models best the real conditions, this is a clear indication that the p-value alone might be a too weak score for the validation of peptide assignments. One would certainly need to run further tests with data of other mass spectrometers and would need to create other protein mixes to give a final decision, whether the p-value is a good score or not. There seems to be quite a difference between distinguishing assignments from the normal and inverted database to distinguishing correct from incorrect ones.

The Gaussian Mixture Model could solve the issue of knowing the labels for the LDA, as it should identify the bi-modality in the data in an unsupervised fashion. The GMM however did not lead to good results on the test data, as the mean and covariance matrices did only vaguely match the ones of the LDA and thus the LDA step of the GMM validation did project into an non-optimal direction. This is illustrated in Figure 5.13.

We experimented as well with Support Vector Machines for the discrimination, this did not lead to fundamentally different results than the LDA. It might however be a good alternative to the LDA, as we could then introduce much higher costs for missclassifying incorrect assignments, than for missclassifying correct assignments. The soft-margin SVM cost function then becomes

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\boldsymbol{w}^t\boldsymbol{w} + C_+ \sum_{y_i=1} \xi_i + C_- \sum_{y_i=0} \xi_i,$$

which is for example already implemented in LibSVM [CL01]. We however would still need to have labels for the training data. One could perhaps use the p-value as a measurement how much we believe that an assignment is a correct one.

One can quite easily introduce a better validation algorithm by combining for example the p-value, the cross-correlation score and the corresponding relative differences from

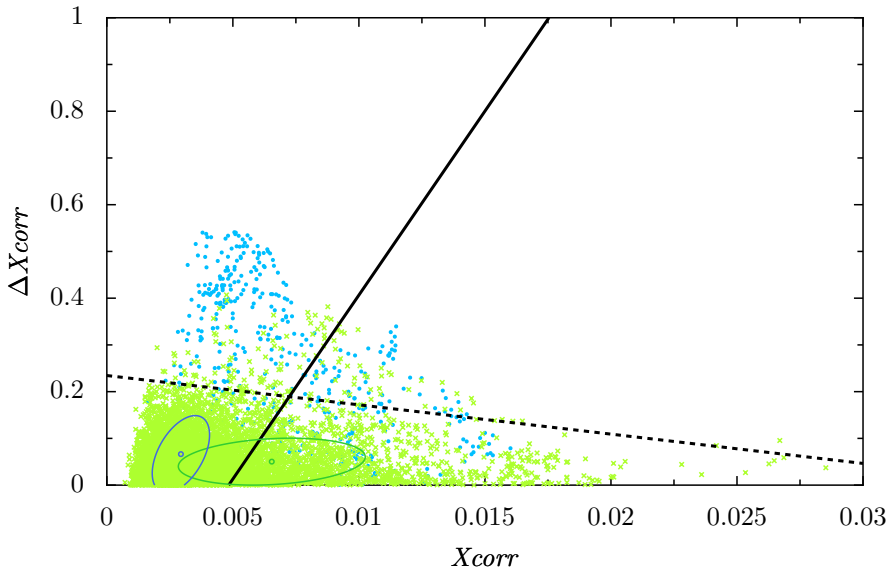

Figure 5.13: Comparison of the normal LDA (dashed line) and the one obtained when first clustering the data with the GMM EM algorithm and then computing the LDA (solid line).

the second best match to get a 4 dimensional vector. One could then compute a LDA for this data, which should hopefully discriminate better than the LDA used in this project report.

Bibliography

[CL01] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[FRR+05] Bernd Fischer, Volker Roth, Franz Roos, Jonas Grossmann, Sacha Baginsky, Peter Widmayer, Wilhelm Gruissem, and Joachim M. Buhmann. NovoHMM: A hidden markov model for de novo peptide sequencing. *Analytical Chemistry*, 77(22):7265–7273, 2005.

[HTF01] Trevor Hastie, Robert Tibshrani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[KNKA02] Andrew Keller, Alexey Nesvizhskii, Eugene Kolker, and Ruedi Aebersold. Empirical statistical model to estimate the accuracy of peptide identification made by ms/ms and database search. *Analytical Chemistry*, 74(20), 2002.

[KPN+02] Andrew Keller, Samuel Purvine, Alexey Nesvizhskii, Serg Stolyar, David Goodlett, and Eugene Kolker. Experimental protein mixture for validating mass spectral analysis. *OMICS*, 6:207–212, 2002.

[KSC+05] Eugene Kapp, Frédéric Schütz, Lisa Connolly, John Chakel, Jose Meza, Christine Miller, David Feny, Jimmy Eng, Joshua Adkins, Gilbert Omenn, and Richard Simpson. An evaluation, comparison, and accurate benchmarking of several publicly available ms/ms search algorithms: Sensitivity and specificity analysis. *Proteomics*, 5:3475–3490, 2005.

[SCY04] Rovshan Sadygov, Daniel Cociorva, and John Yates. Large-scale database searching using tandem mass spectra: Looking up the answer in the back of the book. *Nature Methods*, 1(3):195–202, 2004.

[SY03] Rovshan Sadygov and John Yates. A hypergeometric probability model for protein identification and validation using tandem mass spectral data and protein sequence databases. *Analytical Chemistry*, 75(15), 2003.