Master thesis report

A baysian network for mass spectrometry

Sébastien Touzé

Table des matières

1 Background

The aim of my work is to create a machine learning program that predicts the relative abundance and the reproducibility of particular fragment ions that are produced when peptides are fragmented. This software module will be a part of a logical Bayesian network that models the full mass spectrometry process in proteomics.

1.1 Proteomics

The use of MS analysis for proteomics answers new questions and as a recent domain still undergoes rapid improvement. The idea is simple : using a mass spectrometer to separate the different proteins or peptides according to their mass and charge, and then concluding which proteins are present in the sample. Below I briefly describe the state of the art concerning mass spectrometry methods, more focused on what is useful for this study.

An example of a common experimental protocol can be seen in schema 1. It describes a tandem mass spectrometry (MS/MS) analysis.

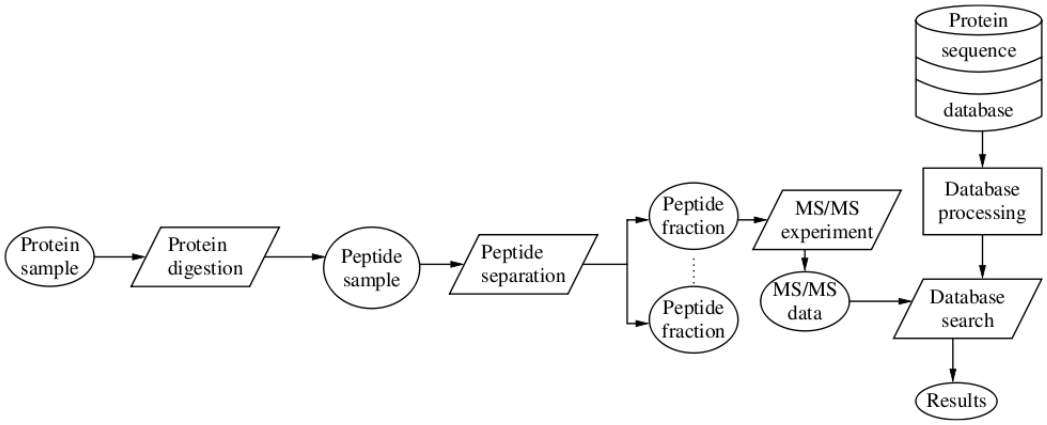

FIGURE 1: Experimental process schema, extracted from [1]

Most of the proteins have a mass that exceeds the sensitive range of a mass spectrometer, which makes it necessary to cleave the protein either chemically or biologically using *in vitro* protein digestion. The cleavage results in shorter and hence lighter-weight chains of amino acids, peptides. This type of approach, cutting a protein to smaller, easier to identify elements is called bottom-up, it is apposed to top-down approach in which a complete peptide is analyzed to get its composition.

To understand how the output of a mass spectrometer, a spectrum, is obtained a large number of fields are needed beyond biological knowledge on proteins proprieties. For example, some knowledge on the molecular structure of the peptide is necessary to predict where the charges will appear and why peptide may break. Then, in order to interpret the results, one must use some search algorithms on large databases of already known proteins to match the spectrum and the proteins present in the sample.

There are 5 main steps :

1. protein digestion is made to obtain sequences of lighter mass which can be detect by the mass spectrometer,
2. peptide (meaning amino acids sequences coming from digestion) separation,
3. mass spectrometry experiment and fragmentation of peptides,
4. fragment detection by the mass spectrometer
5. the resulting spectrum is used by search algorithms to match spectrum of already known proteins and then conclude on which proteins are present inside of the sample

The issue is that none of these steps are deterministic and the mechanisms underlying are not well known. But some factors that influence the results of each step are already identified. The goal of the InSPECtor team here is to use machine learning to build a Bayesian network showing the influence of all possible factors.

Another technological lock is that most of currently used search engines only uses already identified peptides' spectra databases. It means that those algorithms cannot identify any new peptide.

Data collection There are various data collections of identified spectra publicly available. Those database contain results of various previous MS analysis with the level of confidence in the identification (meaning the ling between the protein and the spectrum's peaks).

For this study I'll begin using the in-house database from the Mass Spectrometry Laboratory Information Management System (ms_lims)[2] data management system. Others database notably the the PRoteomics IDEntifications (PRIDE) database have been used in the different works of the InSPECtor work-group.

Protein digestion The first step of the analysis process is digestion, this study will focus on enzymatic cleavage as it is the most common approach. The trypsin protein is considered for the digestion. This choice is motivated primarily by the fact that this enzyme is commonly used because it exist in most of vertebrate species and have a good level of both selection (for cleavage site making fragments with a resulting mass suitable for mass spectrometer) and reliability (low number of miscleavage).

Protein digestion consists in cleaving a peptide into two smaller ones. The trypsin is known to cleave after whether Arginine (Arg) or Lysine (Lys) amino acids if not followed by Proline (Pro) according to one of the Keil rules currently used in proteomics to predict peptides cleavages by trypsin. A missed cleavage occurs when all the expected cleavages are not done in the experiment. If a miscleavage occurs the mass distribution of resulting peptides will change and so the final spectrum and add some mistakes on the peptides detection.

A paper describing a solution to predict the miscleavages of peptides using random forests have been published by the InSPECtor group[3]. In this publication, an ensemble of decision trees (which is called random forest) have been learned and then used to predict peptide cleavage on three different databases (iPRG, CPTAC and ms_lims). The comparison between state-of-the-art Keil rules and the predictions of the forest shows that the Cleavage Prediction with Decision Trees (CP-DT) algorithm gives better results than Keil rules on these 3 databases regarding the Area Under the ROC (AuROC) scoring and the shape of the curves (see **??**). A notable difference on the results of the CP-DT and the Keil rules as the first one gives a probability that a cleavage occurs, whereas Keil rules only classify as cleaved or not cleaved.

Peptide separation This step is made using chromatography which consist of a passing a mobile phase through a stationary phase. As the various elements of the mobile phase will interact with the stationary one, they will use different time to travel (named elution time) and then can be separated. Both column Gas Chromatography (GC) and column Liquid Chromatography (LC) are often used for organic molecules. In this work we consider that this problem is solved notably, the elution time prediction is correctly computed.

Peptide fragmentation Fragmentation is intentionally made when using tandem mass spectrometry (MS/MS). Indeed, for a peptide which carries multiple electronic charges it can break into pieces (with some of them still ionized) which will be detected by the mass spectrometer. This occurs inside the ionization source, so some are more specifically used for MS/MS as they generate more fragments. The most common fragmentation process is the collision-induced dissociation (CID) (see [1, subsection 8.2]), which uses in a collision cell containing an inert gas (like argon). An ionized precursor which enter in this cell will collide with the gas increasing its inner energy to fragmentation when a threshold energy level is reached. The principle of CID is illustrate is figure 2.

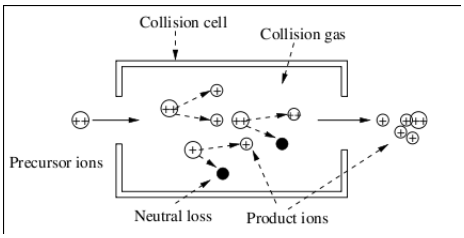

FIGURE 2: Schema of a CID fragmentation, the picture is extracted from [1]

In this study all the used data comes from a CID fragmentation. The results may be extend to other type of fragmentation but no test on this is made here.

As describe in [4], the fragmentation process is not deterministic and vary with the experimental setup, the experimental protocol applied, the type of instrument used or the MS/MS runs made before. Even if some chemical mechanisms are known to influence fragmentation, no complete model is able to completely predict the fragmentation behavior.

Mass spectrometry It's in this step that the spectrum will be generated. A mass analyses is a instrument which is able to detect ions according to their charge and mass. The output then is what is called a spectrum, which is the intensity detected for each possible value of the fraction $\frac{mass}{charge}$. As an ion charge is often denoted with the letter z the ratio notation $\frac{mass}{charge}$ is commonly simplified to m/z. The intensity can be seen as the count of how many ions are detected for this value of the m/z ratio. Each mass analyzer is able to detect ions in a specific range of m/z meaning that all ions at the edge or outside of this sensibility field will not be detected or with a less significant intensity.

Among other type of analyzer, we focus on this study on orbitrap. This type of mass analyzer is made of two coaxial electrodes around which the ions are trapped. The frequency of each ion

inside the orbitrap is proportional to z/m making possible to get the m/z value with a fast Fourier transform (FFT). This type of analyzer have a high precision which explain the choice of this instrument for this study.

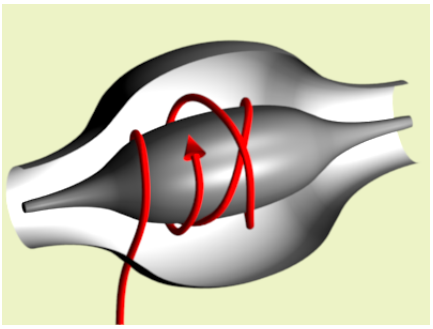

FIGURE 3: Schema of an orbitrap mass spectrometer, trapped ions' path is shown with the red curve, extracted from wikipedia [1]

1.2 Data mining

Data mining consists in various methods to extract patterns or a particular element in some data. Different model exists like decision trees, SVM, clustering, ...to take into account all possible type of information, and use it according to a specific goal. In this study we use decision trees to make our predictions. The final goal of InSPECtor group is to make a global logical Bayesian network containing all information and factor influencing an MS/MS run.

Learning Algorithms This section will focus and describe on two type of data mining methodologies : decision trees and Bayesian networks. Some elements on Random forests, a method derived from decision tree one will also be added as this method improve decision trees results.

Decision Tree A decision tree is a data structure made of a sequence of nodes. Changing from one node to another is made according to the answer to a query.

In the case of this study binary trees are used and learned. In a binary tree each node only has two child node as a maximum.

Random Forest [5]

Bayesian Network A Bayesian Network (BN) is an directed acyclic graph (DAG) $G = (V, E)$ with V the set of all G's vertices and E the edges. Each node represents a random variable and each edge connected to this node shows the conditional dependencies of the random variable [6].

1. `http://en.wikipedia.org/wiki/Orbitrap`, seen on 2013-04-22
2. `http://en.wikipedia.org/wiki/Bayesian_network`, seen on 2013-04-22

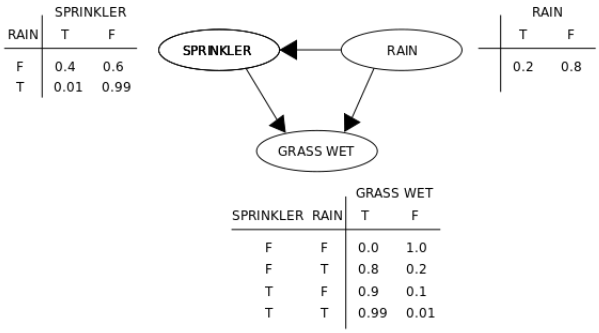

FIGURE 4: Example of a simple Bayesian network. The relation between the observation `the grass is wet` and both `rain` and `sprinkler` is clear here. Extracted from wikipedia [2]

The use of BN allow to have a very quick overview of all dependencies a random variable. This is also a strong way to calculate very quickly probabilities knowing the graph properties.

Logical Bayesian Network

This specific type of Bayesian network is a combination of Bayesian network with logic (see [7] and [8] for more details). Logical Bayesian Network (LBN) is an modeling language which main advance compare to other models also derived from both BN and logic (like Probabilistic Relational Model or Bayesian Logic Programs) is that it allows to make the distinction between deterministic and probabilistic information.

This study do not transform the learn trees and forests to LBN even if this is the final goal of the InSPECtor team.

1.3 Data mining for MS fragmentation

Review of some state of the art articles Fragmentation allows to obtain more information on the structural formula of a peptide — a process called structural elucidation.

Two classes of methods are used in order to match a peptide with a spectrum :

- prediction models, which can use data-driven algorithms or kinetic models of the fragmentation to predict the final spectrum of a peptide and then be used to recognize the original peptides from an experimental spectrum,
- direct search against a peptide library containing identified peptides and corresponding fragments (with their spectra).

Database search The idea of database search for peptide identification is to use a database containing identified peptides and compares the mass in the spectrum with those. Then the different peptides which match will be ranked to obtain the peptide-spectrum matching. The main limitation of this method is that it is not able to identify an amino acid sequence which is not already present in the database.

To solve this point, *de novo* sequencing have been used to complete the database. De novo sequencing consist in creating all the possible peptide sequences from a 6-frame translation of the genome assuming that no other peptides are possible. Then a search can be made against all these

sequences to discover new peptides. This solution brings two other issues which are the size of the database as it may contain a very large number of peptides and only a few of them are relevant to the spectrum under consideration. This will increase computation time. Another factor which decreases the efficiency of this method are posttranslational modifications which occur in the amino acid sequence and modify its mass, amount other properties.

To answer those problems different approaches are described in the literature.

Another way to predict a peak intensity of a given peptide fragment is to compare unknown amino acid sequences from de novo sequencing to sequences with a known spectrum. This method is described in [9]. This is possible due to the high correlation between the spectra of two fragmentation experiments [10] and the correlation between a spectrum and the amino acid sequence as shown in this article.

Furthermore, to improve the results of current database search engines it is possible to use a different ranker or a re-ranker which will return more accurate order of the possible peptides matching an experimental spectrum.

The Rankboost method is used in [11] to learn a classifier to rank matches between peptides from a database with experimental spectra. This technique can be applied to both database search and a full 6-frame translation of a genome. Compared to classic techniques using tags and scoring functions this method appears to be more relevant and able to identify more peptide sequences. Moreover, some comparisons are made between their ranker and another in-house classifier, the method used for comparison seems unclear and may be biased. However, the results presented are not compared with most current methods of identification such as MASCOT, so there is no possibility to compare further this idea.

The same algorithm is used to predict the rank of the peaks intensities in a spectrum. This work, describe in [12] uses a similar algorithm as describe above (Rankboost) to rank the relative intensities of peaks. The results compared with kinetic and (in-house ?) heuristic database search methods are better. According to the authors, this algorithm can still be improved using a different learning method such as a decision tree.

prediction models **Data-driven algorithms**

The use of data-driven models to predict the result of a peptide fragmentation is highly connected with the reproducibility of MS/MS experiments. Indeed if for the same peptide the resulting spectra of fragmentation vary strongly, the possibility to predict it becomes very limited. This point has been proved in [10]. This paper shows that for a given instrument the results of a peptide fragmentation are highly reproducible. When changing the instrument and settings used for the MS/MS process, the results are still quite reproducible, so training a learner on results from different instrument may be used to predict the final value of any MS/MS instrument (with any settings). The reproducibility of data tested stays between 76% and 92% for experiments using the same devices and between 40% and 71% when using different instruments.
However, the figures set a maximum to the expectation of data-driven predictions.

Another technique [13] uses an ordinal regression model to predict fragments. This method is quite fast and outputs a lower number of fragments than other solutions like the naive fragmentation model used by SEQUEST without losing the right fragments. The direct consequence of this is the increase of peak prediction accuracy. A second effect, which is interesting when trying to identify new peptide, is that this improvement of precision allows to limit the number of mass spectrometry (MS) experiments in the first iteration of this search.

Kinetic models

The MS-Simulator described in [14] uses four assumptions on the fragmentation process to model the fragmentation of peptide into y-ions (fragmentation into others ions is not predicted in this article).

Selecting a fragmentation method Another interesting approach based on decision trees allows to choose which is the best method of fragmentation for a given peptide [15]. This method uses the full mass spectrum MS1 to predict whether collision-activated dissociation (CAD) or electron transfer dissociation (ETD) should be used to fragment the peptides. This increases the probability of MS/MS success. Although this is not directly linked with our topic as it does not predict fragments, this aspect is another idea which can be taken into account when wanting to improve MS/MS identification results.

2 Predicting fragmentation with decision trees

This part will focus on the program we wrote to predict the fragmentation behavior. The program is written in C++ and uses the MiPS library which provides functions to build decision trees and random forests.

As everyone of the InSPECtor team can collaborate on the code, we use a git repository hosted in BitBucket. The idea here is to push steadily some working documented code. The documentation is made using Doxygen so the main part of documentation is embedded in the code.

2.1 Database structure

In order to make any prediction we need to have a suitable database containing both enough examples and appropriate features to test on. Furthermore, as the result of each MS/MS experiment can vary according to the instrument used and the setting applied, all the tests of the program must be made using the same settings to allow comparison. For this purpose we will use the database from ms_lims [2]. This in-house database gathers MS/MS runs made by the Gent University. A simplified version of the database structure is shown in figure 5. We can see in this schema the different features used by the tree learner like the amino-acid sequence (including both the post-translational modifications and without them), charge and mass for the precursor ; the type of ion (meaning the location of the fragmentation, see 1.1 p. 5 for more details), the ion charge, the mass (in reality the m/z value) the intensity of the detected peak for this ion. For each identified peptide, some spectrum data are also displayed. The raw spectrum files are also available but are not used in this study. All the spectra used in this work come from MS/MS runs using an orbitrap for mass analysis.

We need to have a short overview of the used data as it will influence the way learners build their model and can cause biased results.

As expected in MS/MS experiments, ions distribution in this database (see figure 6) shows a high number of y and b ions and very few other ions. The reasons for this are detailed in section 1.1, page 5.

The mass distribution displayed in figure 7 shows the limit of sensibility of the mass spectrometer. Indeed under a m/z ratio of $60Da/e$ and above $2,000Da/e$ no fragments are detected. A mass of $60Da$ correspond approximatly to the mass of an iminium ion which consist in a single amino acid. Thus, this is not a real limitation contrary to the maximum m/z ratio which sets a limit to the size of peptide sequence detected.

2.2 Decision Tree

Before going to random forest we choose to start with decision tree, as a tree is slightly easier to set up and faster to run as explained in part 1.2. This must give first results to validate the principle of using decision trees to predict spectrum from fragmentation.

Peptide sequence representation and features choice All the predictions and learning will be made on some examples which are in this study peptides sequence. We modeled this with a specific class which contains all the needed information. The attributes of that class are the information considered as disciminative feature of a real peptide sequence. This includes :

– the amino-acid sequence,

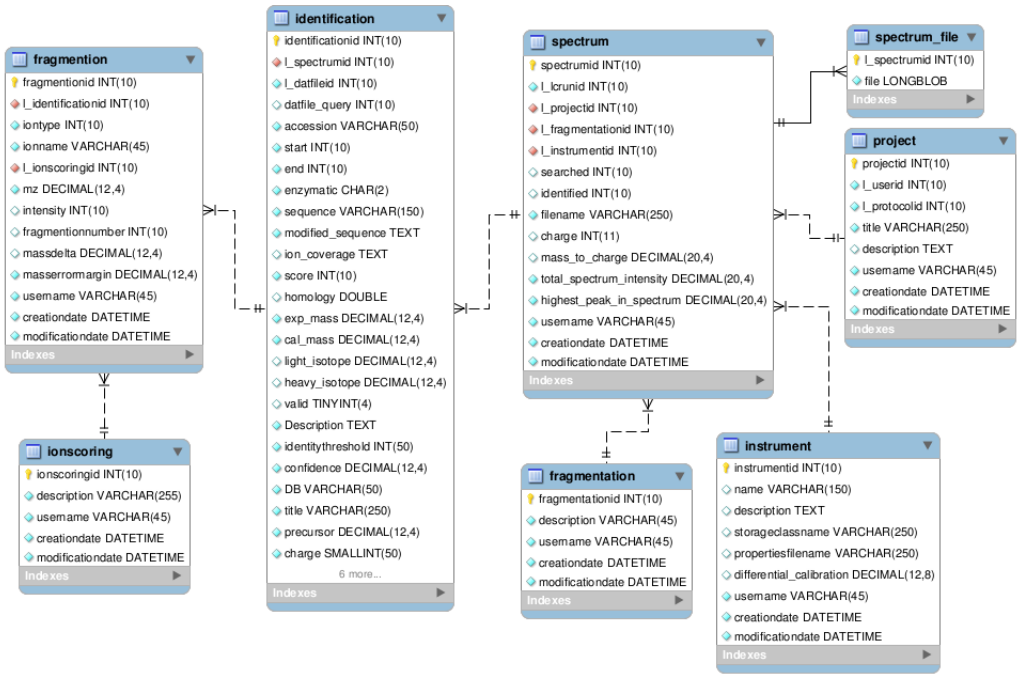

FIGURE 5: Simplified ms_lims database structure, only the relevant tables for this work are displayed.

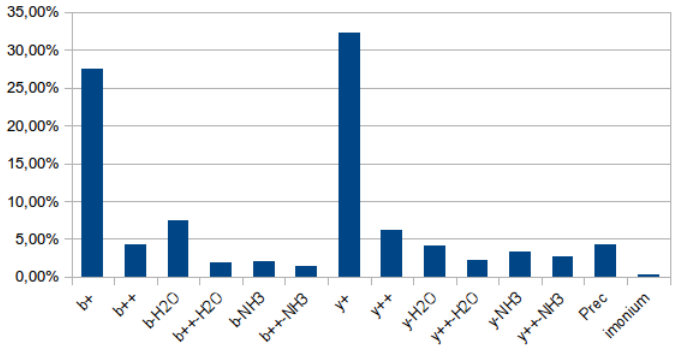

FIGURE 6: Ion distribution in the ms_lims database. The b and y ions represent almost half of the database (45% and 51% each). The notation Prec stand for precursor ion and corresponds to a complete peptide sequence which don't break and only lose C- or N-terminus (usually a small peptide sequence).

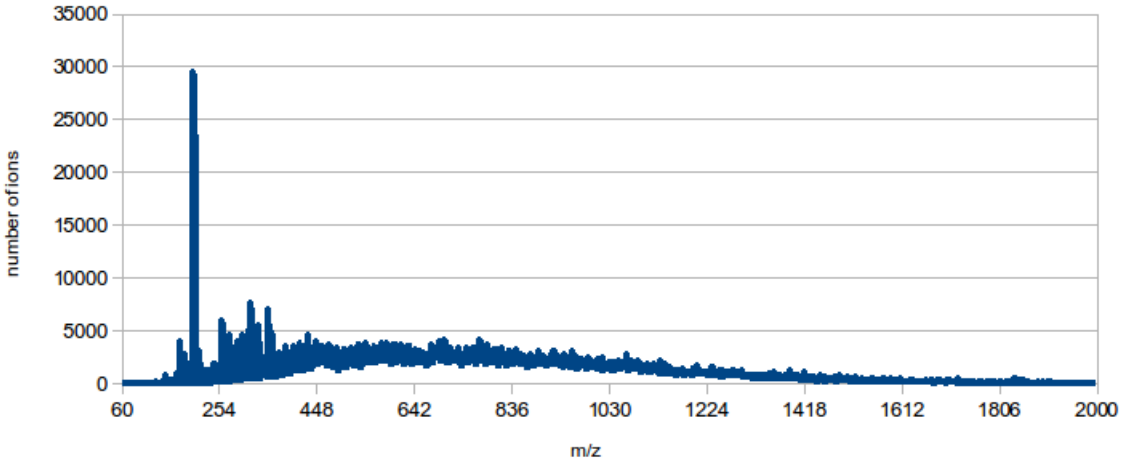

FIGURE 7: m/z distribution of the b and y ions in ms_lims database. m/z values are rounded down to the next integer. This chart shows an important number of ions with a mass of $203Da/e$ and $206Da/e$. Those two peaks are likely to be caused by the high number of some similar peptides in the database. This is due to the specific type of experiments recorded in the ms_lims database and show a limit of it.

- the modifications list on this sequence,
- the type of ion which includes both the type of fragment (a, b, c, x, y, z, ...) and the ion charge,
- the charge of the fragment precursor,
- the experimental mass of the ion precursor,
- the fragmentation point,
- the normalized intensity detected for this ion (the normalization process is detailed in 2.2),
- we also add an information for the ion presence learner to know if this example is a positive example or not.

Then, using the MiPS library we need to start with implementing tests on chosen features of the ion fragments. The type of feature tested will influence the efficiency of the decision tree. Another point to take into account is that for a given test, it may improve the efficiency of the prediction without meaning that the corresponding feature has an influence on the final spectrum (meaning that this feature may not cause the spectrum, but can be correlated with another feature which directly influence the aspect of the spectrum). So, later on when considering the results one must take care of this point. Then, as it includes all the algorithms to build the tree and we only need to give an example of the data structure used, which is a class modeling a peptide sequence, one test to make the decision, and a data set. A classified list of all created tests is shown figure 8.

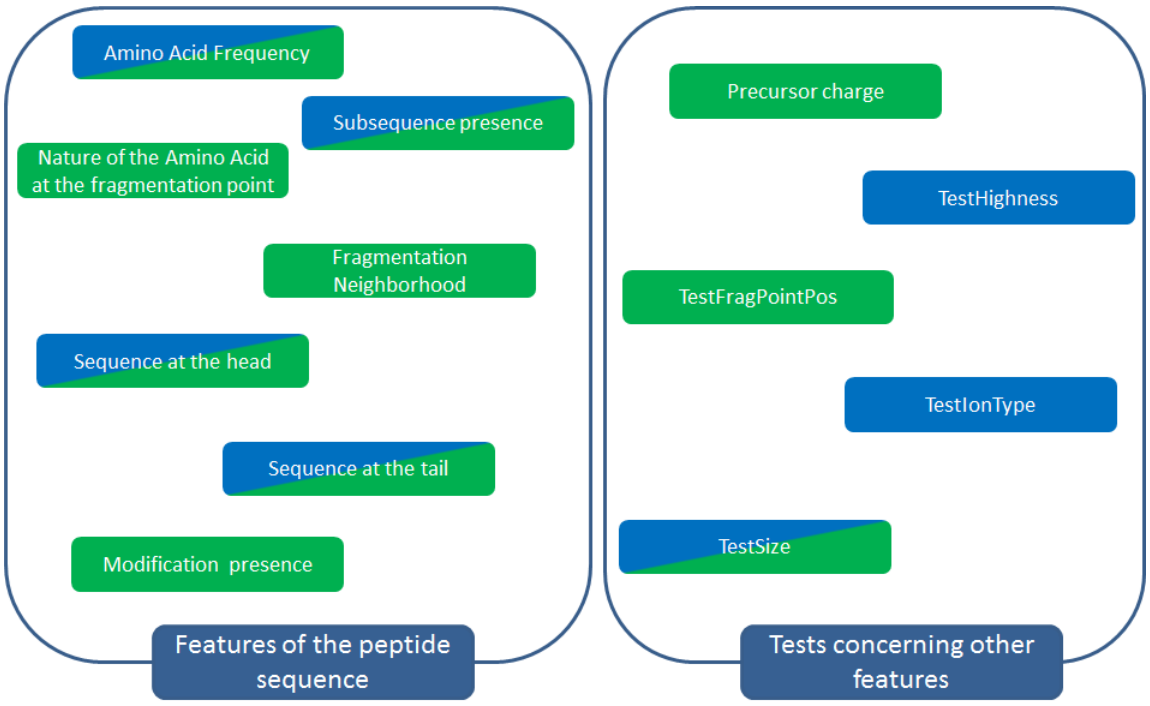

FIGURE 8: The tests available to the learner. A green background denotes that the test applies to the ion precursor. Blue tests apply to the fragmented ion (some apply to both cases).

Then, the idea is to select only the most useful tests in order to decrease as much as possible the computation time of the learner. Indeed, the complexity of the tree learning algorithm depends on the number of tests to perform and the number of examples to learn on as detailed in 1.2. For each tree, different run have been made with different tests to see how each one influences on the prediction. Some tests were not used at all and were removed straight away, some others where less used, but this information does not allow us to remove the test from the learner as according to the position of this test in the tree, it can highly influence the prediction results. The tests distribution statistics are displayed in figure 11. Concerning other less used tests, in order to make the distinction between those which have little influence in the tree performance and those which have a high separation power, we simply compare the results of the tree validation with and without one of those tests. If there is no important influence on the results we can infer that this feature do not improve the algorithm performance.

To avoid over-fitting in our tree, a pruner which sets a minimum leaf size (meaning a minimum number of training examples in each leaf) is used.

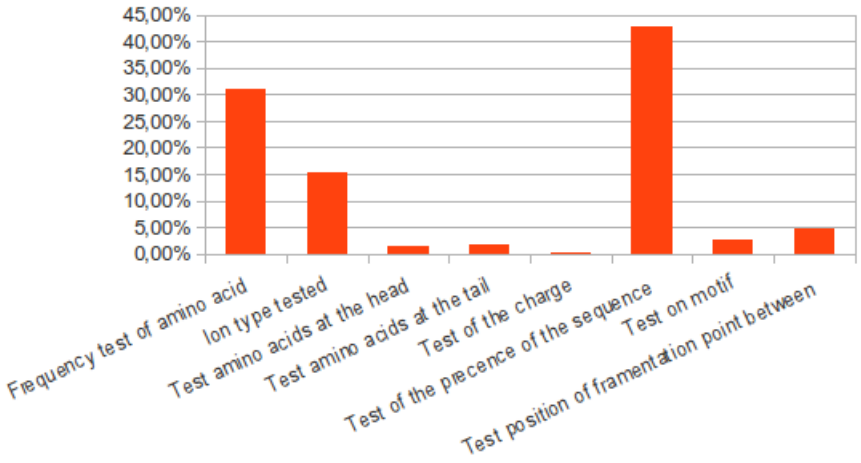

FIGURE 9: Features tests distribution and use in a tree predicting fragmentation. This schema is a mean of the use of each test in a 25-tree random forest).

A more advance way to proceed is to make inner cross-validation using small sub-sets for each fold of cross-validation, and compare results of predictions using different sub-sets of all available tests. This have not been implemented by lake of time. The current choice for tests still seems reliable as it is correlated with biological and chemical considerations and because of the few tests describe above we made.

pre-processing the data According to the experimental conditions, the absolute intensity of two spectra from the same peptide can vary a lot. This is true even for identical machine settings or

identical samples. The direct consequence is the need of normalizing each spectrum before learning from it and then that an outputted spectrum will have relative intensities. Furthermore, this modification does not cause a loss of information as only the normalized intensities of the peaks are used when analyzing the result of a MS/MS run and comparing quantitatively different spectra.
More details on the normalization can be found in [1, section 15.4]. The following will only focus on the post-experimental methods i.e. only considering normalization made on the outputted values of the spectrum and not adding components to the samples.

Simple normalization Considering the normalization methods used in the literature, we chose to normalize the intensity of each peak by the total intensity of the spectrum. Doing this, each normalized intensity can be interpreted as a probability of presence among the whole spectrum. This stays close to our main goal to predict some presence probability and an intensity with a given confidence.

We must be aware that doing that we give much more influence on the high peaks of a spectrum whereas a homogeneous will keep close values when normalizing. This is not important when predicting the spectrum, but will become when analyzing a spectrum coming from a tree or forest prediction.
For example, considering a two spectra with four peaks : 22,23,29,26 for the first one and 19,19,40,22 for the second one. The total intensity is 100 (we suppose for this example that the noise is null). When normalizing using our method the high peak of the second spectrum becomes prevailing even if the other intensities are comparable between the two spectra.

Another limit to this method is that low quality spectra with a high noise will have lower peaks. Indeed, each peak will be divided by the total intensity detected, including the noise. As the learner works only on identified fragments this can be seen as the difficulty to discriminate a peak from the noise around.

At last, when considering the peak list, if too intense peaks remain unidentified by MASCOT

The last two limits underline that using good quality spectra to learn (low noise and most intense peaks identified) is key to get relevant results. A simple efficient way to evaluate spectrum quality is to compare the sum of identified peaks with the spectrum total intensity.

Improve data scattering Looking at a spectrum, we can see that there are few high peaks and many small ones. As it is easier to work on data with a more homogeneous distribution, we will apply a function to do this. When looking at the intensities distribution in the database, we can see that it has a logarithmic or square root shape, so we choose to compare the effect of those two functions. As square root normalization shows slightly better correlation between actual data and predictions considering intensity prediction, we choose to use this function for normalization.
The effect of square root and log normalization on a spectrum are displayed in figure 11.

Other normalization processes, used in microarray bioinformatics are described in [16]. Two approaches are most common : complete data methods and baseline methods. The first consists in plotting the average of log intensity versus the log intensity difference and fit a smooth line to this scatter plot. The latter consists in using a reference (the baseline) to calculate the normalizing factor. For example, linear methods will use a constant factor $\beta = \frac{\tilde{x}_{base}}{\tilde{x}_i}$ (with $\tilde{x}$ a trimmed mean value) apply to each intensity to get the normalized values. Those methods are not used in his work because it is not very common in proteomics contrary to microarray field.

Adding negative examples

Results Using the program described above to learn trees on an extract of our database we get some result which vary whith the type of feature learned. The evaluations methods also vary for the different learners as ion presence learner simply output a binary classification whereas intensity or variation learner output a regression. It must be noticed that the goal of this program, when each learner results will be good enough, is to use the outputted prediction of a learner as an input for the next one. Thus, given a peptide sequence a complete spectrum with variation around the intensity of each peaks is outputted.

Ion presence learner As describe above, the ion presence learner create a binary classifier after a supervised learning on positive. More precisely, it gives the probability for each ion to be observed in the spectrum of a real MS/MS run.

Evaluation method As it is a binary classifier the performance evaluation will be done using the area under the Receiver Operating Characteristic (ROC) curve (AUROC or simply AUC for area under the curve). The use of AUROC for evaluation the performance of a decision tree is very frequent as detailed for example in [17]. The use of AUROC is also used in biology to evaluate tests diagnostics performances as [18] describes, so it's not surprising for biologist to find this type of evaluation process.

The ROC curve plots the percentage of true positive ($\frac{TP}{TP+FN}$), called sensitivity, the against the percentage of false positive ($\frac{FP}{FP+TN}$), which is $1-specificity$. Formally written, the ROC curve represents this function f :

$$f : \begin{cases} [0;1] & \rightarrow & [0;1] \\ \frac{TP}{TP+FN} & \rightarrow & \frac{FP}{TN+FP} \end{cases} \tag{1}$$

Then the area under this curve gives a evaluation of the performances of the classifier. Two extreme type of classifier can be distinguished, the perfect one which do not make any mistake on predictions. That mean the true positive rate is always 1 and the false positive rate always 0. The resulting area is then 1 which is the maximum possible value. On an other hand a random classifier would have 50% of wrong prediction, would be represented by a strait line in the plot from $(0;0)$ to $(1;1)$. The corresponding area is 0.5 and is a minimum possible value. Indeed, if a classifier get an AUC under 0.5 that mean that the inverse classifier (using the inverse prediction of this one) have a AUC above 0.5 so is better than the random classifier.

To compute the points of our ROC curve we used a regression which allow us to work with the probabilities calculated by the tree without any other treatment on the data. The complete method will be to determinate a discrimination threshold between the two classes *observed ion* and *non observed ion*. This would be done plotting the ROC for different threshold values and selecting the one which maximize the area under the curve.

Results

Tests have been made to learn a tree on both 5,000, 50,016 and 1,000,000 positive example fragment ions from the database. Equal number of negatives ions have been added when pre-processing the data as detailed in 2.2. The learner work therefore with 10,000, 100,032 and 2,000,000 examples in total using a 5-fold cross-validation. It takes from a bit more than one minute for the first set to half an hour for the second on a single core of a quad-core intel i3 running at 2.27GHz and more than a day on a 8-core intel i7 running at 3.40GHz (still using only one of those cores).

Peak presence learner The comparison of average AUROC results are showed in figure 12. This results clearly shows that performances of the learner increases with the number of examples to

learn on. This seems to suggest than using the full 3,231,103 ion fragments present in the database the score will be event higher. The score will indeed increase but first because of computational time consuming the gain may not be really relevant. Secondly, as explained above in part 2.1, p.10, the type of experiments in our database is not neutral. The direct consequence is that learning only on this database the results may be too specific to be used in any real situation. The use of an increasing number of examples from this database may increase this bias.

Peak intensity learner Intensity and also variation predictions are quite different of above one because the goal is here to learn a regression tree to get continuous values from 0 to 1. The direct consequence is that the performance evaluation must be done using other criteria. The tests done here have been made only on the positive examples. The final goal of this study is indeed to use the conclusions of the previous learner but this have not been done here to get results easier to interpret.

Evaluation method In reality we will use two different criteria to get a more precise idea of the . First, the root-mean-square error, using the formula 2 where the a_i are the actual intensity, p_i the predicted one and N the total number of ions tested. This measure will show how close the predicted intensities are from the real spectrum.

$$\epsilon = \sqrt{\frac{\sum_{iintests}(a_i - p_i)^2}{N}} \tag{2}$$

Secondly, a correlation factor is used to check how close the shape of our prediction is to the real one. Indeed, if for a given spectrum the prediction have for example a 0.1 intensity offset, the final error will be 0.1 but the predicted spectrum will be very correlated with the real one (in this case the correlation factor is 1). THe pearson correlation factor can be expressed as :

$$r_p = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

where σ_x is the standard deviation and σ_{xy} the covariance. Developing the σ_{xy} expression the correlation factor can be rewrote as :

$$r_p = \frac{\sum_{i=1}^{N}(x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}}$$

Results

The figure 13 shows the results for this learner when learning on 500 and 5000 examples.

Intensity variation leaner For this learner the same evaluation methods than in the section 2.2 above are used.
We get the following results in figure 14. Those are not very good mainly due to the fact that the tests used are less optimised for this task. Indeed, in this learner no particular selection on feature tested have been made.

2.3 Random Forest

Evolution of the algorithm The use of MiPS library will here be very useful as it gives an abstraction layer between the learners, our examples and example features. That mean that no change will be made to both the class `fragmentedSequence` which is the model for each example and to the features tested.

Then we change the learner from a decision tree learner to a random fraction forest learner. That means the forest learner used will randomly use a portion of available feature tests each time adding a node to a tree. We chose a fraction of 10% of the total number of tests. Knowing that a learner like the ion presence learner have 2,870 tests on features, each tree of this forest will then use 287 tests. The number of trees in each forest will be changed, from 50 to 200, in order to compare the results and discuss the improvement of prediction.
Evaluation methods remain the same as each forest.

Results We can clearly see that forest learners improve the results of predictions as shown by the differents figures. The gain doesn't increase with the number of trees in the forets but reach a maxumun value arrond 50 or 100-trees forest. Thus, it seems not useful to increase further the number of trees per forest.

2.4 Limitations and predictions correction

As seen above the learner works on a limited proportion of all negatives examples. In our experiments we only work with as much negative examples as positive ones, but the possibilities for negatives examples are much higher than the number of positive one. The possible ions for a given protein is approximately the number of amino acids in the sequence times the number of ion types (for ms_lims database this number is 17, this number includes the different ion charge saw in database).
The real number of possible fragments can be approximate considering all possible fragment ions which are, according to [19], 15 different ions, with a charge from 1 to 3, which make 45 possibly observed fragment ions. Then if we consider the average size of peptide sequence 13 amino acids, for ms_lims database, we get 585 different possible ions. Among them, only 22 are positives examples (this is an average value from ms_lims database) meaning 3.7% of all possible fragments.

This lag with the real data may influence the prediction of the learner. Indeed, as we used 50% true and false example the learner consider that by default for any sequence the probability of being observed is 50%. Whereas in reality this probability may fall under 3.7%. Consequently, as we won't try to learn on all possible ions because the learning data will be too big in this situation, we need to correct the predictions of the learner when testing.

Let's denote by f the observation of a particular ion of a peptide that enters the mass spectrometer. Then follows from the definition of conditional probability :

$$P(f) = P_{prior}(type(f)) \times P(fispositive|type(f))$$

For all peptides in the training database, there are T ions that could possibly have formed (where ions from different peptides are separately counted). In the training set are P peaks. All other $(T - P)$ possible ions are not observed. For computational reasons, detailed above, the model

is however built with only P negative examples (or $2P$, or $3P$). So, the value predicted by the tree or forest :

$$V' = \frac{n_+}{n_+ + n_- \times \frac{P}{T-P}} \tag{3}$$

where n_+ and n_- are the number of positive and negative ions for this sequence respectively. The desired probability is :

$$V = \frac{n_+}{n_+ + n_-} \tag{4}$$

Knowing that, we can now calculate a correction factor that will be applied to each of the predictions of the tree (or forest) learner on a test set containing all possible fragments of each tested sequence.

Lets use 3 to express n_-/n_+

$$V' \times \left(1 + \frac{n_-}{n_+} \times \frac{P}{T-P}\right) = 1 \tag{5}$$

$$\frac{1-V'}{\frac{P}{T-P} \times V'} = \frac{n_-}{n_+} \tag{6}$$

Then we can rewrite 4 using this expression :

$$V = \frac{1}{1 + \frac{1-V'}{\frac{P}{T-P} \times V'}} \tag{7}$$

$$V = \frac{\frac{P}{T-P} \times V'}{\frac{P}{T-P} \times V' + 1 - V'} \tag{8}$$

The expression 8 will be used for each sequence considering the probability V' outputted by the learner to get the real probability V. From the resulting probabilities, the figure 17 is made representing the ROC curve with and without the correction factor. The AuROC is lower with this correction factor according to the expectation as explained above.

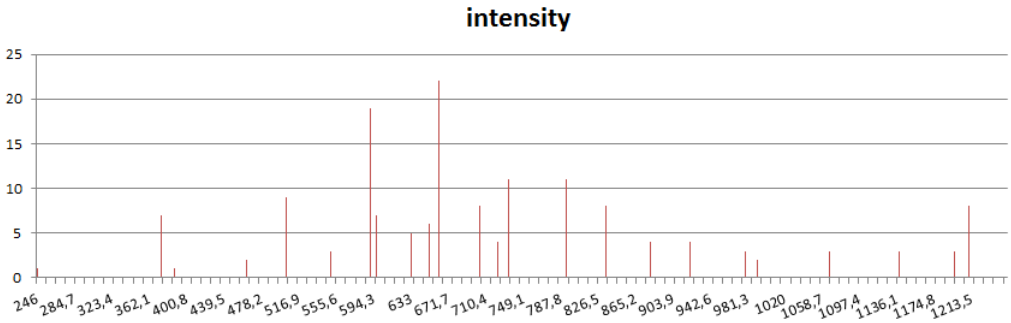

FIGURE 10: A spectrum considered by the learner. It only contains identified ions with their intensity at the measured m/z. This spectrum is extracted from the ms_lims library (spectrum number 7519, identified precursor : Ace-M¡Mox*¿ESGSTAASEEAR-COOH)

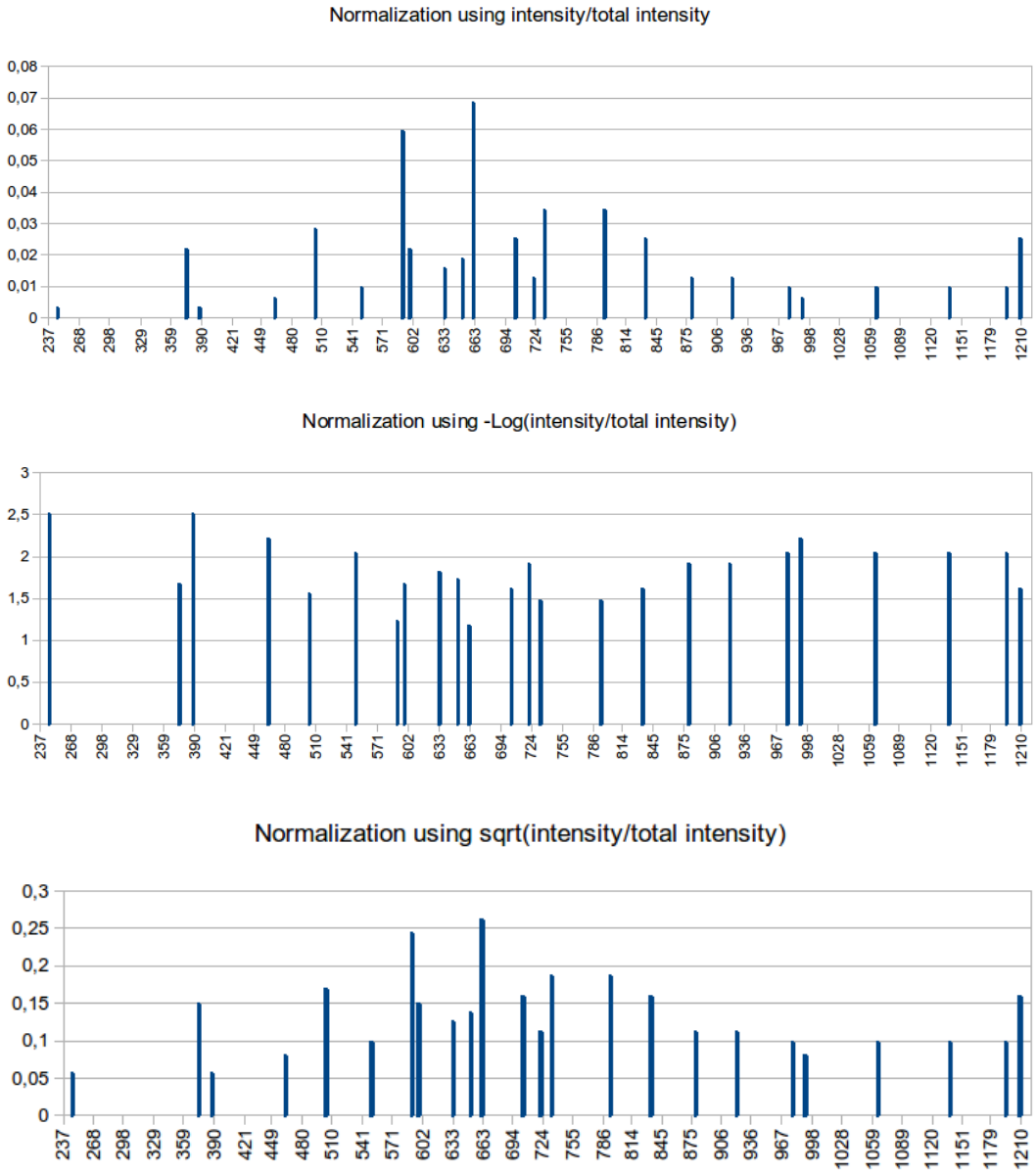

FIGURE 11: Comparison of the effect on the spectrum of normalization. (spectrum number 7519, identified precursor : Ace-M¡Mox*¿ESGSTAASEEAR-COOH)

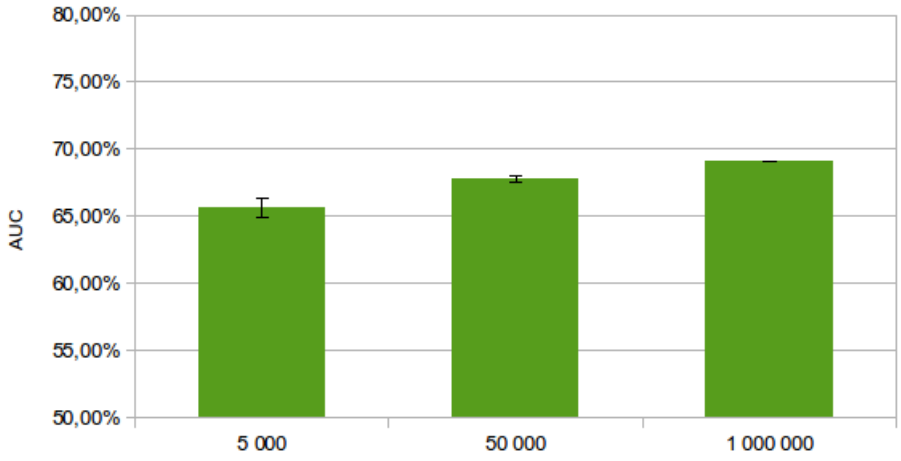

FIGURE 12: Comparison of AUROC scoring after a learning on (from left to right) 10k , 50k and 2M examples. The improvement of the learner performance with the number of examples is clearly underlined here.

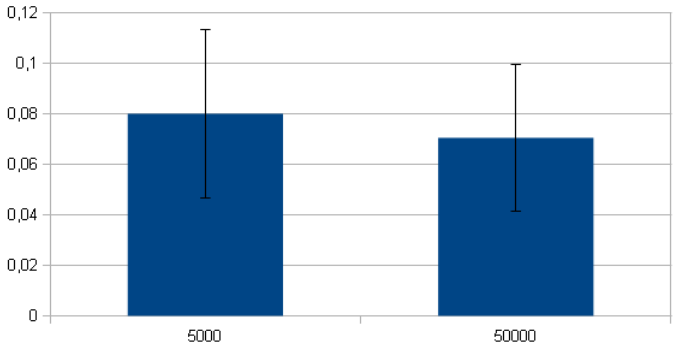

FIGURE 13: Comparison of mean RMS error for intensity learnings with 5 000 and 50 000 examples. The standart deviation is used as an estimation of the error made using only a 5 fold cross validation

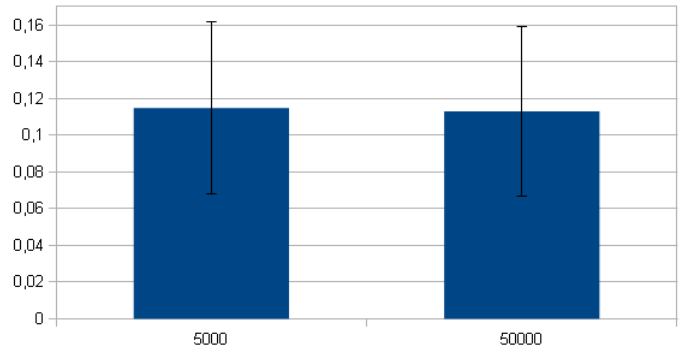

FIGURE 14: Comparison of mean RMS error for variation learnings with 5 000 and 50 000 examples. The standart deviation is used as an estimation of the error made using only a 5 fold cross validation

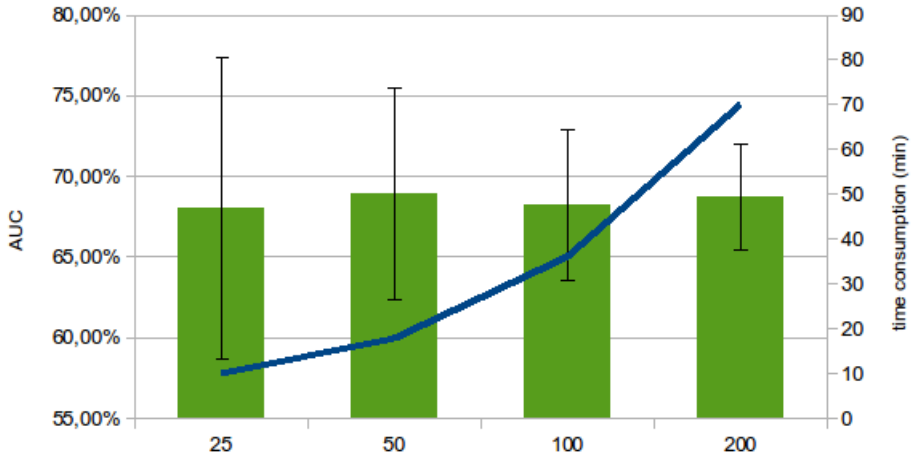

FIGURE 15: Comparison of average AUC for forest containing 25, 50, 100 and 200 trees. The values go from % for a 25-trees forest to 89.3% for 50 trees and 89.6% for a 100-tree forest. The test was made using a 5 folds cross validation. Errors bars shows the maximum and minimum values of the 5 folds. The blue line represents the time consumption for learning one forest, it is multiplied by 4 to 7 hours when considering a 100 trees forest.

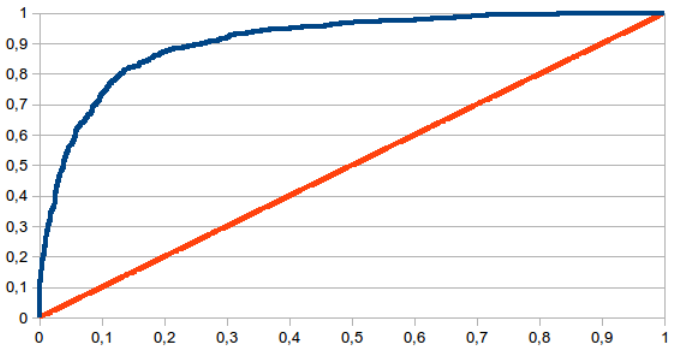

FIGURE 16: ROC curve (blue line) for a forest of 30 trees using a total of 10,000 examples (1/5 for testing the rest for learning). The area under the curve is 90.77%. The red line from (0 ;0) to (1 ;1) represents random predictions.

FIGURE 17

3 Project conclusion and future work

This work shows the possibility to use decision trees in proteomics to predict the fragmentation of peptides and the resulting spectrum. To evaluate and make easier predictions, the learning step has been divided in 3 sub-parts. In this paper they are only used independently to validate their efficiency. We manage to predict with a quite good accuracy the fragmentation of a peptide sequence. Predictions for intensity and variation prediction show less interresting results and still need some improvement to be used in biology. On the way learning is made in the program multi-processing can be added to reduce the time cunsumption of the program. More tests on different MS/MS database have to validate the results we get on ms_lims. Cross-database validation, which means learning on one database and testinng on another, must also be made to evaluate the over-fitting of the learning. Then from the conclusions of a large learning, a logical baysian network can be made and together with other conclusions from the InSPECtor group allow to gives a better understanding of MS/MS process.

4 Bibliographie

Références

1. Ingvar Eidhammer, Kristian Flikka, Lennart Martens, and Svein-Ole Mikalsen. *Computational methods for mass spectrometry proteomics*. 2007.
2. Kenny Helsens, Niklaas Colaert, Harald Barsnes, Thilo Muth, Kristian Flikka, An Staes, Evy Timmerman, Steffi Wortelkamp, Albert Sickmann, Joe Vandekerckhove, Kris Gevaert, and Lennart Martens. ms_lims, a simple yet powerful open source laboratory information management system for ms-driven proteomics. *Proteomics*, 10'6 :1261–1264, 2010.
3. Thomas Fannes, Elien Vandermarliere, Leander Schietgat, Sven Degroeve, Lennart Martens, and Jan Ramon. Predicting tryptic cleavage from proteomics data using decision tree ensembles. *Journal of Proteom research*, 2013.
4. Harald Barsnes, Ingvar Eidhammer, and Lennart Martens. A global analysis of peptide fragmentation variability. *Proteomics*, 6'11 :1181–1188, 2011.
5. Leo Breiman. Random forests. *Machine Learning*, 45 :5–32, 2001.
6. Patric Naïm, Pierre-Henri Wuillemin, Philippe Leray, Olivier Pourret, and Anna Becker. *Réseaux bayésiens*. 2007.
7. Daan Fierens, Hendrik Blockeel, Maurice Bruynooghe, and Jan Ramon. Logical bayesian networks. *3rd International Workshop on Multi-Relational Data Mining location :Seattle, USA, Août 22, 2004*, 2004.
8. Daan Fierens, Hendrik Blockeel, Maurice Bruynooghe, and Jan Ramon. Logical bayesian networks and their relation to other probabilistic logical model. *15th International Conference, ILP 2005, Bonn, Germany, Août 10-13, 2005*, 2005.
9. Chao Ji, Randy J. Arnold, Kevin J. Sokoloski, Richard W. Hardy, Haixu Tang, and Predrag Radivojac. Extending the coverage of spectral libraries : A neighbor-based approach to predicting intensities of peptide fragmentation spectra. *Proteomics*, 13-5 :756–765, 2013.
10. Sujun Li, Randy J Arnold, Haixu Tang, and Predrag Radivojac. On the accuracy and limits of peptide fragmentation spectrum prediction. *Analytical Chemistry*, 83-3 :790–796, 2011.
11. Ari M. Frank. A ranking-based scoring function for peptide-spectrum matches. *Journal of proteome research*, 8-5 :2241–2252, 2009.

12. Ari M. Frank. Predicting intensity ranks of peptide fragment ions. *Journal of proteome research*, 8-5 :2226–2240, 2009.

13. Dong Wang, Surendra Dasari, Matthew C. Chambers, Jerry D. Holman, Kan Chen, Daniel C. Liebler, Daniel J. Orton, Samuel O. Purvine, Matthew E. Monroe, Chang Y. Chung, Kristie L. Rose, and David L. Tabb. Basophile : Accurate fragment charge state prediction improves peptide identification rates. *Genomics, Proteomics & Bioinformatics*, 11-2 :86–95, 2013.

14. Shiwei Sun, Fuquan Yang, Qing Yang, Hong Zhang, Yaojun Wang, Dongbo Bu, and Bin Ma. Ms-simulator : Predicting y-ion intensities for peptides with two charges based on the intensity ratio of neighboring ions. *Journal of Proteome Research*, 11-9 :4509–4516, 2012.

15. Danielle L. Swaney, Graeme C. McAlister, and Joshua J. Coon. Extending decision tree–driven tandem mass spectrometry for shotgun proteomics. *Nature Methods*, 11 :959–964, 2008.

16. B. M. Bolstad, R. A. Irizarry, M. Astrand, and T. P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19-2 :185–193, 2003.

17. Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30-7 :1145–1159, 1997.

18. B. M. Bolstad, R. A. Irizarry, M. Astrand, and T. P. Speed. Roc (receiver operating characteristics) curve : principles and application in biology. *Bioinformatics*, 19-2 :185–193, 2003.

19. Matrix Science Inc. Mascot database search : Peptide fragmentation. `http://www.matrixscience.com/help/fragmentation_help.html`, juillet 2012.