Optimal analysis of complex protein mass spectra

Dijkstra, Martijn; Jansen, Ritsert C.

Published in:
Proteomics

DOI:
10.1002/pmic.200701064

PROTEOMICS

Supporting Information for Proteomics

DOI 10.1002/pmic.200701064

Martijn Dijkstra and Ritsert C. Jansen

Optimal analysis of complex protein mass spectra

Supplementary information

A spectrum can be interpreted as a histogram with TOFs, say $t_1, t_2, \ldots, t_I \in \mathbb{R}$, ordered from small (left) to large (right) on the horizontal axis, where the histogram of the spectrum has $I \in \mathbb{N}$ bins. Generally, the spectra $(1, 2, \ldots, K)$ have horizontal axes with corresponding TOF-labels, because the detector frequency is generally not altered between different measurements. Let $n_{k,1}, n_{k,2}, \ldots, n_{k,I} \in \mathbb{R}_{\geq 0}$ denote the intensities of the detection signal in spectrum k after $t_1, t_2, \ldots, t_I$, respectively.

Self-calibration

For the analysis of spectra produced by other MS technologies such as ESI, in which molecules generally hold more charges, the Pearson correlation coefficient,

$$\rho(n_1, n_2) = \frac{\sum_i (n_{1,i} - \bar{n}_1)(n_{2,i} - \bar{n}_2)}{\sqrt{\sum_i (n_{1,i} - \bar{n}_1)^2 \sum_i (n_{2,i} - \bar{n}_2)^2}} \tag{8}$$

can be generalized to

$$\rho(n_1, n_2, n_3) = \frac{\sum_i (n_{1,i} - \bar{n}_1)(n_{2,i} - \bar{n}_2)(n_{3,i} - \bar{n}_3)}{\sqrt{\sum_i (n_{1,i} - \bar{n}_1)^2 \sum_i (n_{2,i} - \bar{n}_2)^2 \sum_i (n_{3,i} - \bar{n}_3)^2}} \tag{9}$$

and so on, where $n_{1,i}$, $n_{2,i}$ and $n_{3,i}$ are intensities in region 1, 2 and 3, and $\bar{n}_1$, $\bar{n}_2$ and $\bar{n}_3$ are the means of the intensities per region, respectively.

Models interconnecting peak parameters

Initialization

In the SELDI pre-processing pipeline, any peak detection method which can identify single charge peaks, can be used to initialize the parameters μ_j. We consider each data point as a peak, if it is higher than its 30 neighbor data points left, and also than its 30 neighbor data points right [21]. We initialize $r = 2 \cdot 10^{-7}$ and we label a peak at a given location μ_j as single charge peak, if the correlation between the data points within $\mu_j \pm r \cdot \mu_j^2$ and the data points within $\frac{\mu_j \pm r \cdot \mu_j^2}{2}$, *i.e.* at the double charge location, is > 0.80. The proportion parameters can be initialized randomly, as long as $p_* \in (0, 1]$ and $\sum p_{k,*} = 1$, per mixture k. However, a good guess is preferable to speed up the convergence of the algorithm and to avoid getting stuck into local best solutions. This is important, because the likelihood function is not convex, and *EM* is sensitive to initialization. A wrong initialization might lead to a local, not a global best solution. We use the height of a peak above the baseline together with its location to initialize the proportion parameters, whereby we the lower convex hull [22] as initial guess of the baseline.

Parameter estimation

We apply the iterative *EM*-algorithm [23] to calculate maximum likelihood values for the parameters in the model. Each iteration consists of an *E*-step and an *M*-step. The *E*-step calculates the component membership probabilities for the normal distributions by

$$p_{k,j,a,z|i} = \frac{p_{k,j,a,z} \cdot f_{j,a,z}(y_i)}{f_k(y_i)} \tag{10}$$

and for the baseline by

$$p_{k,\text{bl}|i} = \frac{p_{k,\text{bl}} \cdot f_{k,\text{bl}}(y_i)}{f_k(y_i)} \tag{11}$$

given the current parameter estimates. These component membership probabilities, derived from the likelihood equations described in (Dempster *et al.* 1997), estimate the probability that a random observation i with *m/z*-value y_i, in spectrum given k, belongs to a given molecular species j with a adducts and z charges.

The *M*-step uses these component membership probabilities to updated the parameter estimates in the model. Let

$$\varphi_{k,j,a,z,i} = \frac{n_{k,i} \cdot p_{k,j,a,z|i}}{z^2 \cdot \sigma^2_{j,a,z}} \tag{12}$$

The updated estimates for the molecular masses are

$$\hat{\mu}_j = \frac{\sum\limits_{k,a,z,i} \varphi_{k,j,a,z,i} \cdot (z \cdot y_i - a \cdot \mu_a)}{\sum\limits_{k,a,z,i} \varphi_{k,j,a,z,i}} \tag{13}$$

and the updated estimate for the mass of the adduct is

$$\hat{\mu}_a = \frac{\sum\limits_{k,j,a,z,i} \varphi_{k,j,a,z,i} \cdot (z \cdot a \cdot y_i - a \cdot \mu_j)}{\sum\limits_{k,a,z,i} \varphi_{k,j,a,z,i} \cdot a^2} \tag{14}$$

for $k = 1, \ldots, K$; $j = 1, \ldots, M$; $a = 0, \ldots, a_{\max}$; $z = 1, \ldots, z_{\max}$; $i = 1, \ldots, I$.

The newly obtained $\hat{\mu}_j$'s and $\hat{\mu}_a$ are used to calculate the resolution parameter

$$\hat{r}^2 = \frac{\sum\limits_{k,j,a,z,i} n_{k,i} \cdot p_{k,j,a,z|i} \cdot \dfrac{(y_i - \hat{\mu}_{j,a,z})^2}{\hat{\mu}^4_{j,a,z}}}{\sum\limits_{k,j,a,z,i} n_{k,i} \cdot p_{k,j,a,z|i}} \tag{15}$$

The baseline is updated for each spectrum individually. In spectrum k, the fractions $p_{k,\mathrm{bl}|i}$ of the data $n_{k,i}$ are attributed to the baseline. The lowess curve is fit to

$$p_{k,\mathrm{bl}|i} \cdot n_{k,i}, \quad \text{for} \quad i = 1, 2, \ldots, I \tag{16}$$

Finally, the proportion parameters are updated by

$$\hat{p}_{k,j,a,z} = \frac{\sum_i n_{k,i} \cdot p_{k,j,a,z|i}}{\sum_i n_{k,i}} \tag{17}$$

and

$$\hat{p}_{k,\mathrm{bl}} = 1 - \sum_{j,a,z} \hat{p}_{k,j,a,z} \tag{18}$$

for $k = 1, \ldots, K$; $j = 1, \ldots, M$; $a = 0, \ldots, a_{\max}$; $z = 1, \ldots, z_{\max}$; $i = 1, \ldots, I$. The *E*-step and the *M*-step are alternated until convergence (generally 50–100 iterations).

Some small abundance peaks, which are not included in the model, may turn out to bias parameter estimates of nearby peaks. We tackle this issue by implementing robustness weights in the parameter estimates. We explained the details on robust estimation in [13]. Alternatively, one can correct such biases and include small abundance peaks in the model.

Visualization

The spectrum intensities can be plotted on the vertical axis versus the observed TOF values or *m/z* values on the horizontal axis. Note that converting TOF into *m/z* will change the area under the spectrum. Also note that equally sized TOF intervals correspond to differently sized *m/z* intervals. The fit of a mixture distribution, f_k, to spectrum k can be visually

inspected on the *m/z* scale by plotting f_k on top of the spectrum, after taking the following two steps.

First, we multiply the mixture distribution ($f_k(y_i)$), which has area 1, with the area under the spectrum on the time-scale,

$$A_k = \Delta t \cdot \sum_{k,i} n_{k,i} \tag{19}$$

where

$$\Delta t = t_{i+1} - t_i \tag{20}$$

is the regular distance between the bins on the axis, which corresponds to the detector frequency. The area under the mixture distribution is now equal to the area under the spectrum on the time-scale.

Second, we scale the fitted intensities. This is necessary because the regular distances between the bins on the TOF scale become variable on the *m/z*-scale, which affects the area under the peaks. We multiply the mixture distribution with the Jacobian of the transformation ($\frac{\delta}{\delta t} y(t_i|\alpha, t_0, \beta)$), described by the calibration equation (equation 1). The Jacobian of the transformation, is the derivative of the calibration function with respect to time,

$$\begin{aligned} \frac{\delta}{\delta t} y(t|\alpha, t_0, \beta) &= \frac{\delta}{\delta t}\Big(U\alpha(t - t_0)^2 + U\beta\Big) && (21) \\ &= 2 \cdot U\alpha(t - t_0) && (22) \end{aligned}$$

Plotting

$$f_k(y_i) \cdot A_k \cdot \frac{\delta}{\delta t} y(t_i|\alpha, t_0, \beta) \tag{23}$$

on top of spectrum k, shows the fit of the model to the spectrum on the *m/z*-scale.

Software

The running time of the *EM*-algorithm is more-or-less proportional to the number of spectra (here: 64), to the number of data points per spectrum (here: 15,000), and also to the number of peaks in the model (here: 29,952).

We use the *R* `library(snow)` [24] to parallelize the computations within each *EM*-step. The algorithm is a proof of principle and converged in less than five days with 100 *EM*-iterations on a computer with 64 processors (3 THz each). There are various ways to reduce the computation time of our prototype software, the most obvious one being a re-implementation of computational intensive parts in C. We plan to make our self-calibration software available as *R*-package. A prototype of our software is available upon request from the corresponding author.