THE EXPECTATION-MAXIMIZATION ALGORITHM

In this section we provide a brief summary of the Expectation-Maximization (EM) algorithm. For details and theoretical understanding of the EM algorithm we recommend readers to refer to the following references (Dempster *et al.* (1977); McLachlan and Krishnan (1997)).

The EM algorithm is an elegant and powerful method for finding the maximum likelihood of models with hidden variables. The key concept in the EM algorithm is that it iterates between the E-step and M-step until convergence. In the E-step, the algorithm estimates the posterior distribution of the hidden variables Q (also refer to as the responsibility) given the observed data and the current parameter settings; and in the M-step the algorithm calculates the ML parameter settings with Q fixed. At the end of each iteration the lower bound on the likelihood is optimized for the given parameter setting (M-step) and the likelihood is set to that bound (E-step), which guarantees an increase in the likelihood and convergence to a local maximum, or global maximum if the likelihood function is unimodal.

Formally, for N input data points at iteration t, given the current parameter settings θ^{t-1} calculated in the previous iteration, the E-step estimates the responsibility Q for the nth input data point as

$$Q_{nh_n} = P(h_n|v_n, \theta^{t-1}) = \frac{P(h_n, v_n|\theta^{t-1})}{\sum_{h_n} P(h_n, v_n|\theta^{t-1})}. \quad (1)$$

where h_n are the hidden (latent) variables, v_n are the observed variables and θ^{t-1} are the model parameters for iteration $t-1$.

In the M-step, while fixing Q_{nh_n}, the parameters are estimated by maximizing the expected log likelihood

$$\begin{aligned}\theta^t &= \arg\max_\theta E_{h_n}\left[\sum_n^N \ln P(h_n, v_n|\theta)\right], \quad (2)\\ &= \arg\max_\theta \sum_n^N \sum_{h_n} Q_{nh_n} \ln P(h_n, v_n|\theta),\end{aligned}$$

where $E_{h_n}\left[\sum_n^N \ln P(h_n, v_n|\theta)\right]$ is the expectation of the log likelihood respect to the hidden variables h_n for the nth data point. The EM algorithm can be altered to calculate MAP instead of ML parameter settings by switching the ML estimation in the M-step with MAP estimation.

As an example, the EM algorithm is applied to learn our proposed algorithm PTMClust (see below).

PTMCLUST - A GENERATIVE MODEL FOR FINDING PTM GROUPS

Below, we provide details of our generative model for finding PTM groups and derivation of our inference algorithm using the EM algorithm. This is additional to the description provided in the Method section of the main manuscript. To make it easy for the reader to follow we have repeated related information from the Method section in the main manuscript.

By accounting for combinatorial interactions between hidden variables that play a role in the protein modification process, our generative probability model aims to describe how each PTM observation is generated. For a given PTM type (PTM group), the observed modification mass is assumed to be a noisy version of the expected (mean) modification mass, and the modified amino acid is chosen from a distribution over amino acids that may be modified in that type. For example, modifications occur primarily on serine (S) and threonine (T) for phosphorylation. For a given peptide, the true modification position is assumed to be chosen uniformly among occurrences of that amino acid in the peptide. Finally, the observed modification position is assumed to be a noisy version of the true position. Below, we described the components of our model: the probability of choosing each PTM type, the probability of choosing each amino acid to be the modified amino acid given the PTM type, the probability of the true modification position given the modified amino acid, and the uncertainty in the observed modification mass and modification position. We then introduced an algorithm for learning the model parameters and inferring the hidden (latent) variables from the input data. Once a model is learned, we can refine the modification for each input peptide sequence by inferring its most likely PTM group, true modification mass and true modification position.

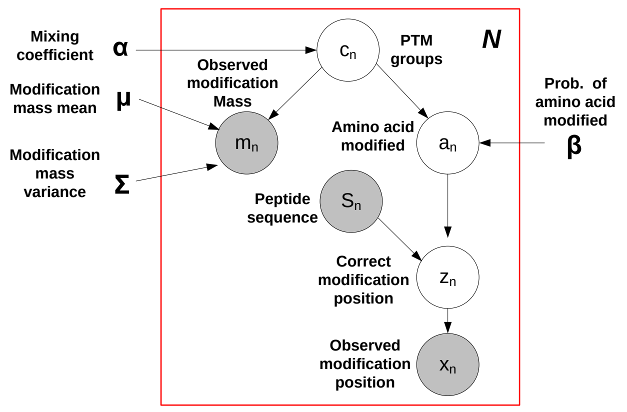
Mixing coefficient
α
Modification mass mean
μ
Modification mass variance
Σ
Observed modification Mass
m_n
c_n
PTM groups
N
Prob. of amino acid modified
β
Amino acid modified
a_n
Peptide sequence
S_n
Correct modification position
z_n
Observed modification position
x_n

Fig. 1. A Bayesian network describing our generative model, using plate notation. The shaded nodes represent observed variables, the unshaded nodes represent latent variables and the variables outside the plate are model parameters. The model describes how the observed modification mass and modification position are generated. Given the type of PTM (PTM group), we can generate the observed modification mass as a noisy version of the modification mass mean, and select an amino acid to have the modification as the modified amino acid. Given the peptide sequence, we can choose a position along it that matches the modified amino acid as the 'true' modification position. We can generate the observed modification position as a noisy version of the 'true' modification position. The plate notation indicates there are N copies of the model, one for each input data. This is the same figure presented in the main manuscript.

The structural relationship between the variables described below is indicated by the Bayesian network from Fig. 1. It describes the model for one input and is repeated for N inputs, as indicated by the plate notation (box in the figure).

In our model, each input peptide sequence S_n, indexed by $n \in \{1, ..., N\}$ where N is the number of peptides in the dataset, has a corresponding discrete peptide length L_n, observed modification position $x_n \in \{1, ..., L_n\}$ and observed modification mass m_n.

$S_n(j)$ is the amino acid in position j of the input sequence n. The total number of values $S_n(j)$ can take on is $A = 24$, which includes the 20 naturally occurring amino acids and four special characters indicating the beginning and end of proteins and peptides. The latent variable $c_n \in [1, .., K]$ denotes the unknown PTM group for peptide sequence n, where K is the number of PTM groups and will be adjusted depending on the desired false detection rate, as described later. The prior probability (mixing coefficient) for each PTM group is given as

$$P(c_n = k) = \alpha_k, \tag{3}$$

where it satisfies the constraints $\alpha_k \geq 0$ and $\sum_{k=1}^{K} \alpha_k = 1$, and is inferred from the data (see below for detail).

The probability that the PTM occurs on amino acid $i \in \{1, ..., A\}$, given that the PTM group is k, is

$$P(a_n = i | c_n = k) = \beta_{ki}, \tag{4}$$

where the latent variable a_n denotes the true (unobserved) modified amino acid and the β's satisfy the constraints $\beta_{ki} \geq 0$ and $\sum_{i=1}^{A} \beta_{ki} = 1$, and is inferred from the data (see below for detail).

Given the peptide sequence S_n and the modified amino acid a_n, each occurrence of that amino acid in the peptide sequence has equal probability of being the true (unobserved) modification position z_n. For completeness, our probabilistic model considers the likelihood of cases where an amino acid does not occur in S_n. To do so, we allowed for the event that the true PTM occurs outside of the given peptide sequence [1], indicated by $z_n = 0$, so that $z_n \in \{0, ..., L_n\}$. All other positions in the peptide sequence have zero probability of being the true modification position. This can be written as

$$P(z_n = j | a_n = i, S_n) = \begin{cases} \frac{1}{\delta_{ni}+1} & \text{if } S_n(j) = i, j \geq 1, \\ \frac{1}{\delta_{ni}+1} & \text{if } j = 0, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where δ_{ni} denotes the number of times amino acid i occurs in sequence n.

We modeled the modification position error $(x_n - z_n)$ between the observed modification position x_n and the true modification position z_n with a discrete probability distribution, given as

$$P(x_n | z_n = j) = \begin{cases} \phi(x_n - j) & \text{if } j > 0, \\ \phi(L_n) & \text{if } j = 0, \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where the likelihood function ϕ accounts for the modification position error. This likelihood function is shared across all PTM groups and is inferred from our empirical observation of the yeast PTM dataset as follows (see Results section in the main manuscript for description of the dataset). We grouped the entries in the dataset by their peptide sequence and modification mass, allowing for mass difference of +/- 2 Da and removing groups with less than three entries. Then, we determined the average modification position for each group (rounded to the nearest position) and computed a histogram of the modification position error. Three entries was chosen because it is the minimum number of entries per group needed to calculate a reasonable mean and variance, provides a large enough dataset (1206) to estimate the likelihood function, and acts as a crude filter for false modified peptides. The frequency of peptides for each group size, shown in Supplementary Fig. 1, exhibits a heavy-tail curve where majority of modified peptides have low counts. More than 48% of the entries have a group of size exactly three. The resulting likelihood distribution is shown in Fig. 2.

[1] $z_n = 0$ is needed to avoid numerical issues since our algorithm considers each amino acid as a possible modification target.

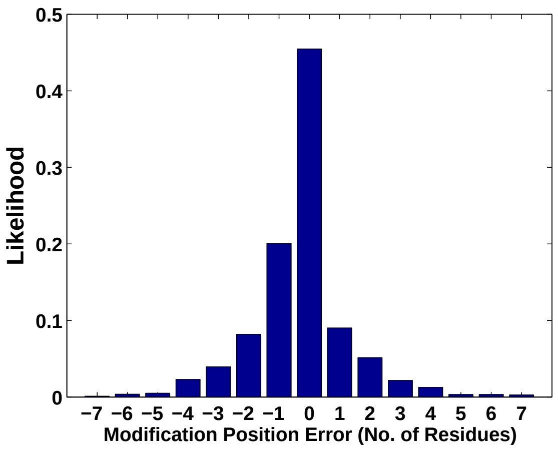

Fig. 2. Distribution of Modification Position Error used by PTMClust. This empirical distribution was derived using yeast PTM data (Krogan *et al.* (2006)) analyzes with SIMS (Liu *et al.* (2008)). A positive (negative) modification position error indicates the observed modification position is towards the C-terminus (N-terminus) of the expected modification position. This is the same figure presented in the main manuscript.

Lastly, we accounted for the variation (noise) in the estimated modification mass by assuming the observed modification mass for each PTM group is normally distributed around the true modification mass, given as

$$P(m_n | c_n = k) = \frac{1}{\sqrt{2\pi\Sigma_k}} \exp\left(\frac{-(m_n - \mu_k)^2}{2\Sigma_k}\right), \tag{7}$$

where μ_k and Σ_k are the modification mass mean and variance for the kth PTM group, and are inferred from the data (see below for detail).

Combining the structure of the Bayesian network and the conditional distributions described above, we can write the joint distribution as

$$P(c, a, z, x, m | S, \theta) = \prod_{n=1}^{N} \big(P(c_n|\theta) P(m_n|c_n, \theta) P(a_n|c_n, \theta) P(z_n|a_n, S_n, \theta) P(x_n|z_n, \theta)\big), \tag{8}$$

where θ represents the model parameters (α_k, β_{ki}, μ_k and Σ_k).

The input data are noisy and may contain false positives and modified peptides that do not fit into proper PTM groups. To account for these spurious data points, we included an additional PTM group (background component) that acts as a garbage collection process (background model). In the background component, we assumed there is no specific relationship between the modification mass and modified amino acid. Formally, the background component has a fixed modification mass mean μ^b and variance Σ^b set to be equal to the mean and variance of the data. Additionally, it has a fixed uniform probability over the modified amino acid $\beta_a^b = \frac{1}{A}$, $\forall$ $a = 1, ..., A$, and a mixing coefficient α^b, which will be used to adjust model complexity (see below).

0.1 Learning with the EM Algorithm

Below, we derived an EM algorithm to infer the unobserved variables and learn our model parameters, which in this case include the PTM group probabilities α_k, the modification mass means μ_k, modification mass variances Σ_k, and the probability that the PTM occurs on an amino acid β_{ki} for each PTM group.

Using the above notation, we derive the steps in EM for our model as follows. To avoid numerical underflow all the calculations are performed in the logarithmic domain. In the E-step, the log joint probability for iteration t is calculated as follows:

$$\begin{aligned}\ln P(c,a,z,x,m|S,\theta^{t-1}) &= \\ \ln P(c,a,z,x,m|S,\alpha^{t-1},\mu^{t-1},\Sigma^{t-1},\beta^{t-1})\end{aligned} \tag{9}$$

where $\theta^{t-1} = \{\alpha^{t-1}, \mu^{t-1}, \Sigma^{t-1}, \beta^{t-1}\}$ represents the set of parameters at iteration $t-1$. From (1), we have the responsibility Q as

$$Q^{(t)}(c,a,z) = P(c,a,z|x,m,S,\theta^{t-1}) = \frac{\sum\limits_{n=1}^{N} P(c_n,a_n,z_n,x_n,m_n|S,\theta^{t-1})}{\sum\limits_{n=1}^{N}\sum\limits_{c}\sum\limits_{a}\sum\limits_{z} P(c_n,a_n,z_n,x_n,m_n|S,\theta^{t-1})}. \tag{10}$$

Since the calculations are done in logarithmic domain, we will need to obtain the posterior distribution $P(c,a,z|m,x,S,\theta^{t-1})$ from the log-likelihood (this is known as the log-sum trick):

$$P(c,a,z|x,m,S,\theta^{t-1}) = \frac{\sum\limits_{n=1}^{N} \exp\left[\ln P(c_n,a_n,z_n,x_n,m_n|S,\theta^{t-1}) - \max\limits_{c,a,z}\Big(\ln P(c_n,a_n,z_n,x_n,m_n|S,\theta^{t-1})\Big)\right]}{\sum\limits_{n=1}^{N}\sum\limits_{c}\sum\limits_{a}\sum\limits_{z} \exp\left[\ln P(c_n,a_n,z_n,x_n,m_n|S,\theta^{t-1}) - \max\limits_{c,a,z}\Big(\ln P(c_n,a_n,z_n,x_n,m_n|S,\theta^{t-1})\Big)\right]}. \tag{11}$$

In the M-step, the parameters are estimated by maximizing the expected complete log-likelihood. We begin by writing down the expected complete log-likelihood as follows:

$$\begin{aligned}
&\left\langle \ln P(c,a,z,x,m|S,\theta^t)\right\rangle_{Q^t} = \\
&E_{caz}\left[\ln \prod_{n=1}^{N} P(c_n,a_n,z_n,x_n,m_n|S,\theta^t)\right] = \\
&E_{caz}\left[\sum_{n=1}^{N} \ln P(c_n,a_n,z_n,x_n,m_n|S,\theta^t)\right] = \\
&\sum_{n=1}^{N} E_{c_n a_n z_n}\left[\ln P(c_n,a_n,z_n,x_n,m_n|S,\theta^t)\right] = \\
&\sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{i=1}^{A}\sum_{j=0}^{L_n} Q^{(t)}(c_n=k,a_n=i,z_n=j) \\
&\qquad \ln P(c_n=k,a_n=i,z_n=j,x_n,m_n|S,\theta^t)
\end{aligned} \tag{12}$$

Accounting for the constraints $\sum\limits_{k=1}^{K}\alpha_k = 1$ and $\sum\limits_{i=1}^{A}\beta_{ki} = 1$ we add one Lagrangian term per constraint to the expected complete log-likelihood and expanding it, we get

$$\begin{aligned}
&\langle \ln P(c,a,z,x,m)\rangle_{Q^{(t)}} = \\
&\Bigg[\sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{i=1}^{A}\sum_{j=0}^{L_n} Q^{(t)}(c_n=k,a_n=i,z_n=j)\ \{-\ln(N)+ \\
&\ln(\alpha_k)+\ln(\beta_{ki})-\ln(\delta_i+1)[S_n(j)=i][j>0]- \\
&\frac{1}{2}\ln(2\pi\Sigma_k)-\frac{(m_n-\mu_k)^2}{2\Sigma_k}+\phi(x_n-j)[j>0]+ \\
&\phi(L_n)[j=0]\}\Bigg]-\lambda\left(\sum_{k=1}^{K}\alpha_k-1\right)-\tau\left(\sum_{i=1}^{A}\beta_{ki}-1\right).
\end{aligned} \tag{13}$$

To find the new estimate of each parameter, we set the derivative of the expected complete log-likelihood (13) with respect to each of the parameters to zero and solve for the corresponding parameter. Setting the derivative of the expected complete log-likelihood with respect to the means μ_k of the PTM groups to zero, we obtain

$$\begin{aligned}
0 &= \frac{d}{d\mu_k}\left\langle \log P(c,a,z,x,m|\theta^t)\right\rangle_{Q^{(t)}}, \\
0 &= -\sum_{n=1}^{N}\sum_{i=1}^{A}\sum_{j=0}^{L_n} Q^{(t)}(c_n=k,a_n=i,z_n=j)\frac{1}{\Sigma_k}[m_n-\mu_k], \\
0 &= -\sum_{n=1}^{N} Q^{(t)}(c_n=k)\frac{1}{\Sigma_k}[m_n-\mu_k].
\end{aligned}$$

Multiplying by Σ_k and rearranging to solve for μ_k we obtain

$$\mu_k = \frac{\sum\limits_{n=1}^{N} Q^{(t)}(c_n=k)m_n}{\sum\limits_{n=1}^{N} Q^{(t)}(c_n=k)}. \tag{14}$$

In the above expressions, $Q^{(t)}(c_n = k)$ is computed from $\sum_i\sum_j Q^{(t)}(c_n=k,a_n=i,z_n=j)$.

Similarly, if we set the derivative of the expected complete log likelihood with respect to the variances Σ_k to zero and, follow a similar line of reasoning to solve for Σ_k, we obtain

$$\Sigma_k = \frac{\sum_{n=1}^{N} Q^{(t)}(c_n = k)\Big(m_n - \mu_k\Big)^2}{\sum_{n=1}^{N} Q^{(t)}(c_n = k)}. \tag{15}$$

Next, we calculate the new estimate for the mixing coefficients α_k. Here the constraint that the mixing coefficients must sum to one $\sum_{k=1}^{K} \alpha_k = 1$ is enforced by the addition of a Lagrangian term in (13). When we maximize the expected complete log likelihood with respect to the mixing coefficients α_k, we obtain

$$\begin{aligned} 0 &= \sum_{n=1}^{N}\sum_{i=1}^{A}\sum_{j=0}^{L_n} Q^{(t)}(c_n = k, a_n = i, z_n = j)\frac{1}{\alpha_k} - \lambda, \\ 0 &= \sum_{n=1}^{N} Q^{(t)}(c_n = k)\frac{1}{\alpha_k} - \lambda. \end{aligned}$$

If we multiple both sides by α_k and sum over k, making use of the constraint $\sum_{k=1}^{K} \alpha_k = 1$, we get $\lambda = N$. Using this to eliminate λ and rearranging, we obtain

$$\alpha_k = \frac{1}{N}\sum_{n=1}^{N} Q^{(t)}(c_n = k). \tag{16}$$

Finally, we maximize the expected complete log likelihood with respect to β_{ki}, which also consists of a Lagrangian term to account for the constraint $\sum_{i=1}^{A} \beta_{ki} = 1$, we get

$$\begin{aligned} 0 &= \sum_{n=1}^{N}\sum_{j=0}^{L_n} Q^{(t)}(c_n = k, a_n = i, z_n = j)\frac{1}{\beta_{ki}} - \tau, \\ 0 &= \sum_{n=1}^{N} Q^{(t)}(c_n = k, a_n = i)\frac{1}{\beta_{ki}} - \tau. \end{aligned}$$

Using this to eliminate τ and rearranging we get

$$\beta_{ki} = \frac{\sum_{n=1}^{N} Q^{(t)}(c_n = k, a_n = i)}{\sum_{n=1}^{N} Q^{(t)}(c_n = k)}. \tag{17}$$

At the end of each pair of E- and M-steps, we calculate the log-likelihood and check for convergence. If the convergence criteria (the difference between current and previous expected log-likelihood is smaller than 10^{-5}) are not satisfied, we cycle through to the next iteration and repeat both the E- and M-steps.

0.2 Recursive Merge Method for Model Selection

In our model, the only free parameter is the number of PTM groups (mixture components) K. We devised a recursive merge method, similar to the split and merge model selection methods [2], that will effectively evaluate and identify the optimal setting for K, to achieve a desired false detection rate.

Instead of adjusting K directly, we used the mixing coefficient of the background component α^b, which represents the prior probability that a data point belongs to the background model, as a 'control knob' to adjust model complexity. For each specific setting of α^b (i.e. a control knob setting) our method infer and learn the hidden variables, parameter settings and K, as describe above. Using maximum likelihood estimation, as α^b increases (we used step size of 0.01), more and more of the loosely clustered peptide sequences are redistributed to other components, including the background component, and the number of non-background components decreases. This is accomplished by pruning away 'empty' components, where we define a component to be empty when it has less than or equal to one peptide sequence assigned to it. In effect, the non-background components are slowly merging with each other and the background component as α^b increases until the non-background components are empty and pruned away, which decreases the model complexity. In our algorithm, we started with a large value for K and a small value for α^b (0.01), and slowly merge the non-background components and the background component, pruning away any empty clusters, by increasing α^b each time. In essence, we learn M models where M is the number of different α^b settings. We chose a single model (i.e. a specific setting for α^b) by analyzing the results from our model selection method using the measures rate of detection (RD) and rate of false detection (RFD). We define RD as the number of real peptides that are not assigned to the background model, divided by the total number of real peptides and RFD as the number of decoy peptides that are not assigned to the background model, divided by the total number of decoy peptides. The choice of which model to use depends on the desire RD and RFD as there is no one optimal setting and is dataset dependent.

0.3 Summary of the Algorithm

Here, we present a summary of the major steps in our PTM clustering algorithm. We used $\rho = 0.01$.

1. Initialize the model parameters and K
2. Train model with EM and set α^b to the learned α for the background model
3. Evaluate the false detection rate
4. Increment α^b by ρ, retrain the model with EM and prune away 'empty' clusters
5. Repeat Steps 3 and 4 until α^b = 1

REFERENCES

Dempster, A., Laird, M., and Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, **39**(1), 1–38.

Krogan, N., Cagney, G., Yu, H., Zhong, G., Guo, X., Ignatchenko, A., Li, J., Pu, S., Datta, N., Tikuisis, A., Punna, T., Peregrin-Alvarez, J., Shales, M., Zhang,

[2] Our approach only makes use of merge steps.

X., Davey, M., Robinson, M., Paccanaro, A., Bray, J., Sheung, A., Beattie, B., Richards, D., Canadien, V., Lalev, A., Mena, F., Wong, P., Starostine, A., Canete, M., Vlasblom, J., Wu, S., Orsi, C., Collins, S., Chandran, S., Haw, R., Rilstone, J., Gandi, K., Thompson, N., Musso, G., Onge, P. S., Ghanny, S., Lam, M., Butland, G., Altaf-Ul, A., Kanaya, S., Shilatifard, A., O'Shea, E., Weissman, J., Ingles, C., Hughes, T., Parkinson, J., Gerstein, M., Wodak, S., Emili, A., and Greenblatt, J. (2006). Global landscape of protein complexes in the yeast saccharomyces cerevisiae. *Nature*, **440**(7084), 637–643.

Liu, J., Erassov, A., Halina, P., Canete, M., Nguyen, D., Chung, C., Cagney, G., Ignatchenko, A., Fong, V., and Emili, A. (2008). Sequential interval motif search: unrestricted database surveys of global ms/ms data sets for detection of putative post-translational modifications. *Analytical Chemistry*, **18**(20), 7849–7854.

McLachlan, G. and Krishnan, T. (1997). *The EM Algorithm and its Extensions*. Wiley.