Description of Model Selector

Luca Iten

February 1, 2024

This document describes the implementations of class *ModelSelector*.

1 Model Description

The class *ModelSelector* implements, as the name suggests, a method to estimate which sections of a given sequence of observations have been generated by which model (i.e., it "selects" a model in each time step). In particular, it should be possible to select the same model on multiple occasions, making it fundamentally different from a simple piecewise constant level-fitting model.

Its working principle heavily relies on the estimation of an M dimensional vector quantity S_i, $i \in \{1, \ldots, N\}$, which will in the following be referred to as the *model selector vector*. This quantity has the following properties:

- All its elements lie between 0 and 1, i.e., $S_{i,m} \in [0, 1]$, $m \in \{1, \ldots, M\}$.
- The elements at each time index have to sum up to 1, i.e., $\sum_{m=1}^{M} S_{i,m} = 1$.
- All-$\{0, 1\}$ solutions should be preferred.

Considering these properties, the model selector vector can be described as a quantity assigning each model in each time step a factor between 0 and 1. Furthermore, its desired behaviour is that one of these factors will go to 1 while all others (due to the first and second property) have to go to 0. Ultimately, a factor $S_{i,m}$ close to 1 means that the observation at time index i is estimated to be generated by the m-th model, while a factor $S_{i,m'}$ close to 0 means it has most likely not been generated by model m'.

A factor graph depicting this desired behaviour is shown in Figure 1. Note how the evolution of the model selector vector S_i is modelled by a PWC model (as implemented by class *PWCModel*). To achieve the previously listed desired behaviours, "Positivity" and "One-Hot" NUV priors are applied to the outputs of this PWC model. Their exact working principles are described in the next Section. Finally, the individual elements of S_i are used to scale the differences between the corresponding model outputs and the actual observations. In this step, the values of $S_{i,m}$ can be viewed as indicators of how well the difference between the output of model m and the actual observation can be explained by the (assumed) model noise with covariance matrix $\Sigma_{i,m}$. This step is also further explained in the following Section.

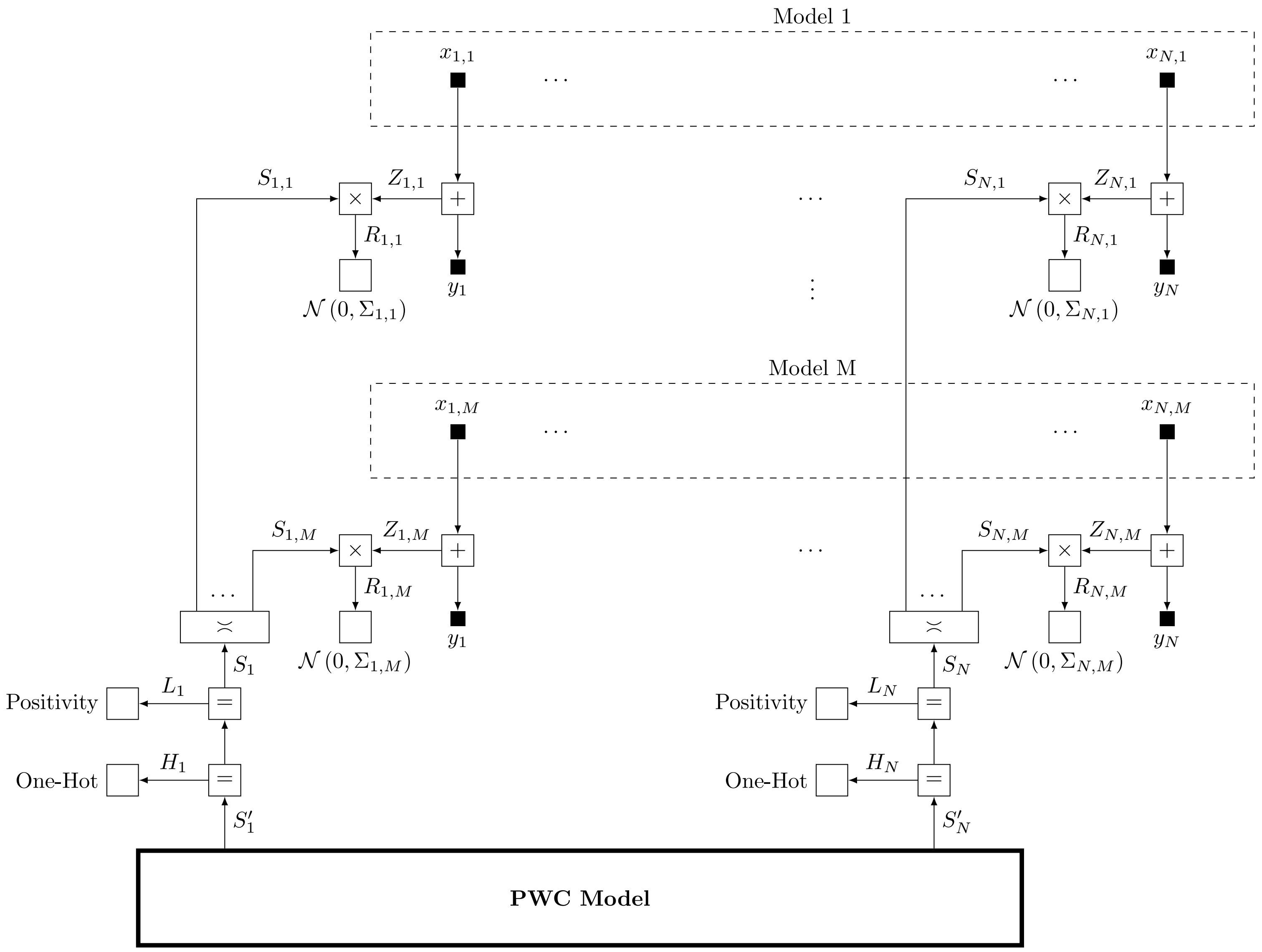

Figure 1: Factor graph of the implemented model selector applied to M models. Note that the generated outputs of the models $x_{i,m}$, $(i, m) \in \{1, \dots, N\} \times \{1, \dots, M\}$ are assumed to be observed (i.e., perfectly known).

2 Explanation of Implementations

This Section describes how the mean and covariances of S_i, $i \in \{1, \ldots, N\}$ are estimated by class *ModelSelector*. Note that the naming and directions of the messages are consistent in the factor graph shown in Figure 1, the following explanations, and the actual implementations. However, they can differ from the documentation of other models investigated in this project.

2.1 Initializing an Object of *ModelSelector*

For initialization, the number of observations N and their dimension D must be given. Furthermore, the assumed observation noise per model must be specified in 'sigmas', either by the covariance or precision matrices of the additive Gaussian noise. Their type must be specified in 'sigmas_type' $\in$ ['covariance', 'precision']. Similar to the initialization of an object of *PWCModel*, the initial values of S_i and U_i can optionally be specified to incorporate potential prior knowledge.

Next to saving all this initial information, the initializing function also generates an object of the class *PWCModel*. This "sub-model" will later be used to handle the estimation of the piecewise constant quantities S_i.

2.2 Positivity Prior

The 'Positivity' prior generating the backward messages through L_i, $i \in \{1, \ldots, N\}$ is needed to implement parts of the desired properties of the model selector (listed in Section 1). Note that L_i is an M dimensional vector quantity like S_i. Its implementation is based on the idea of half-space NUV constraints described in [1]. In particular, a scalar positivity constraint (i.e., ≥ 0) is applied to each element of L_i, resulting in the following (updated) backward messages

$$\overleftarrow{m}_{L_i} = \begin{bmatrix} |\hat{m}_{S_{i,1}}| \\ |\hat{m}_{S_{i,2}}| \\ \vdots \\ |\hat{m}_{S_{i,M}}| \end{bmatrix} \tag{1}$$

$$\overleftarrow{V}_{L_i} = \begin{bmatrix} \frac{|\hat{m}_{S_{i,1}}|}{\beta_\ell} & 0 & \ldots & 0 \\ 0 & \frac{|\hat{m}_{S_{i,2}}|}{\beta_\ell} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \frac{|\hat{m}_{S_{i,M}}|}{\beta_\ell} \end{bmatrix} \tag{2}$$

or

$$\overleftarrow{\xi}_{L_i} = \begin{bmatrix} \beta_\ell \\ \beta_\ell \\ \vdots \\ \beta_\ell \end{bmatrix} \tag{3}$$

$$\overleftarrow{W}_{L_i} = \begin{bmatrix} \frac{\beta_\ell}{|\hat{m}_{S_{i,1}}|} & 0 & \dots & 0 \\ 0 & \frac{\beta_\ell}{|\hat{m}_{S_{i,2}}|} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{\beta_\ell}{|\hat{m}_{S_{i,M}}|} \end{bmatrix}. \tag{4}$$

Here, $\hat{m}_{S_{i,m}}$ denotes the m-th element of the posterior mean estimate of S_i from the previous iteration. The parameter $\beta_\ell \geq 0$ can be used to "tune" the effect of this prior, where higher values correspond to a stronger prior. Generally, high values of β_ℓ make the estimation more stable, but decrease the speed of convergence. Note that for the implementation of class *ModelSelector*, exclusively the dual representations given in Equations (3) and (4) are used.

2.3 One-Hot Prior

The desired behaviour of this 'One-Hot' prior is two-folded. Firstly, it should "select" one out of M elements of the applied vector H_i, $i \in \{1, \dots, N\}$ and set it to 1, while all others should go to 0 (i.e., emphasizing all-$\{0, 1\}$ solution). Secondly, all the elements have to sum up to 1, i.e., $\sum_{m=1}^{M} H_{i,m} = 1$. Together with the previously described 'Positivity' prior, the desired properties of the model selector vector should therefore be enforced.

A factor graph achieving these two behaviours is shown in Figure 2. Thereby, the row vector C is defined as

$$C \triangleq \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix} \in \mathbb{R}^{(1 \times M)}. \tag{5}$$

Forcing the inner product of H_i with this C to be equal to 1 restricts the solution space of H_i to a $M - 1$ dimensional hyperplane. Together with the positivity constraint introduced in Subsection 2.2, this space is further reduced to

$$\left\{ x \in \mathbb{R}^M \, : \, \sum_{m=1}^{M} x_m = 1, \, 0 \leq x_m \leq 1 \right\}. \tag{6}$$

Due to the reduction in dimension of the solution space, the quantities this constraint is applied to (i.e., the backward messages through H_i') don't have a well defined dual representation (in terms of dual-mean and precision matrix). This needs to be considered in the following derivations of the implemented algorithm. For numerical stability, it is generally a good idea to introduce a small amount of uncertainty to this condition, causing the solution space not

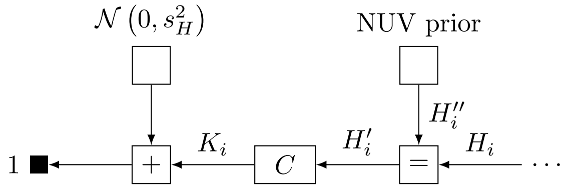

Figure 2: Factor Graph of One-Hot prior.

to decrease in dimension (s_H^2 in Figure 2). However, solutions lying close to (6) are still heavily preferred.

The second desirable condition of the "One-Hot" constraint (emphasis to all-$\{0, 1\}$ solution) is less rigorous and can be achieved with different NUV priors. In the implementation of class *ModelSelector*, four different versions are included. They can be selected via the variable 'priorType' ∈ ['sparse', 'repulsive_laplace', 'repulsive_logCost', 'discrete']. Their respective implementations and the resulting generated messages are further explained in the following paragraphs.

2.3.1 Sparsity per Dimension

The idea of the prior used for 'priorType' = 'sparse' is to apply a sparsifying NUV prior to each element of H_i. This causes the solution to strongly prefer most of its elements to be 0 without penalizing outliers too heavy (i.e., those elements corresponding to selected models).

A sparsity encouraging prior is the log-cost NUV prior ([2]). When applying this to each element of H_i'' individually, the resulting generated forward messages become

$$\overrightarrow{mi}_{H_i''} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{7}$$

$$\overrightarrow{V}_{H_i''} = \begin{bmatrix} \frac{\hat{m}_{S_{i,1}}^2 + \hat{V}_{S_{i,1}}}{\beta_h} & 0 & \cdots & 0 \\ 0 & \frac{\hat{m}_{S_{i,2}}^2 + \hat{V}_{S_{i,2}}}{\beta_h} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\hat{m}_{S_{i,M}}^2 + \hat{V}_{S_{i,M}}}{\beta_h} \end{bmatrix} \tag{8}$$

or

$$\overrightarrow{\xi}_{H_i''} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{9}$$

$$\overrightarrow{W}_{H_i''} = \begin{bmatrix} \frac{\beta_h}{\hat{m}^2_{S_{i,1}} + \hat{V}_{S_{i,1}}} & 0 & \cdots & 0 \\ 0 & \frac{\beta_h}{\hat{m}^2_{S_{i,2}} + \hat{V}_{S_{i,2}}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\beta_h}{\hat{m}^2_{S_{i,M}} + \hat{V}_{S_{i,M}}} \end{bmatrix} . \tag{10}$$

2.3.2 Repulsive Prior at Origin

In this approach, instead of attracting most elements of H_i to 0, all solutions close to the origin are de-emphasized. This causes the cost-function to be minimal at all corners of the solution space described in Equation (6).

For this repulsive prior, either a log-cost or Laplace NUV prior can be used, both with negative tuning factor $\beta_h < 0$. Lets first have a look at the case with the log-cost NUV prior. The resulting forward messages through H_i'' are

$$\overrightarrow{m}_{H_i''} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{11}$$

$$\overrightarrow{V}_{H_i''} = {\sigma^2}_{\text{RP}} \cdot \mathbf{I}_M \tag{12}$$

or

$$\overrightarrow{\xi}_{H_i''} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{13}$$

$$\overrightarrow{W}_{H_i''} = \frac{1}{{\sigma^2}_{\text{RP}}} \cdot \mathbf{I}_M \, , \tag{14}$$

where

$${\sigma^2}_{\text{RP}} = \frac{\text{Tr}\left\{\hat{V}_{S_i}\right\} + ||\hat{m}_{S_i}||^2}{\beta_h} \tag{15}$$

and $\mathbf{I}_M$ denotes the M dimensional identity matrix. Furthermore, $\text{Tr}\{\cdot\}$ denotes the trace and $||\cdot||$ the $L-2$ norm. The EM update rule used to derive these expressions is given in [3]. This prior is used when 'priorType' = 'repulsive_logCost' is selected.

The Laplace NUV prior is used for the configuration 'priorType' = 'repulsive_laplace'. With this choice, Equation (15) needs to be adapted to

$$\tilde{\sigma}^2_{\mathrm{RP}} = \frac{||\hat{m}_{S_i}||}{\beta_h} \tag{16}$$

Note that the values of Equations (15) and (16) are negative (because $\beta_h < 0$). Even though they correspond to variances in a statistical model, these negative values do not pose a problem as long as the posterior variances are all non-negative. Therefore, β_h should be chosen appropriately (i.e., "large enough").

2.3.3 Discrete Phase Prior to Target Solutions

Finally, the approach used when setting 'priorType' = 'discrete' relies on the discrete-phase prior described in [1]. To match the desired behaviour, the target vectors are chosen equal to the Cartesian bases, i.e.,

$$t_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad t_2 = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \quad \ldots, \quad t_M = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}. \tag{17}$$

Accordingly, the forward messages through H''_i become

$$\overrightarrow{m}_{H''_i} = M \cdot \sigma^2_{\mathrm{DP}} \cdot \sum_{m=1}^{M} \frac{t_m}{\mathrm{Tr}\left\{\hat{V}_{S_i}\right\} + ||\hat{m}_{S_i} - t_m||^2} \tag{18}$$

$$\overrightarrow{V}_{H''_i} = \sigma^2_{\mathrm{DP}} \cdot \mathbf{I}_M \,, \tag{19}$$

or

$$\overrightarrow{\xi}_{H''_i} = M \cdot \sum_{m=1}^{M} \frac{t_m}{\mathrm{Tr}\left\{\hat{V}_{S_i}\right\} + ||\hat{m}_{S_i} - t_m||^2} \tag{20}$$

$$\overrightarrow{W}_{H''_i} = \frac{1}{\sigma^2_{\mathrm{DP}}} \cdot \mathbf{I}_M \,, \tag{21}$$

where

$$\sigma^2_{\mathrm{DP}} = \frac{1}{M} \cdot \left(\sum_{m=1}^{M} \frac{1}{\mathrm{Tr}\left\{\hat{V}_{S_i}\right\} + ||\hat{m}_{S_i} - t_m||^2} \right)^{-1} . \tag{22}$$

Note that this approach does not come with a "natural" way to tune the NUV update (similar to β_h in the previous two paragraphs). However, by reducing / increasing all other tuning parameters in the model, this NUV prior can implicitly be tuned too.

2.3.4 Resulting Message Through H_i

In the previous paragraphs different possible forward messages through H_i'' have been described. Choosing any of these representations, the resulting backward messages through H_i (i.e., the messages generated by the "One-Hot" prior) can now be calculated as

$$\overleftarrow{m}_{H_i} = \overrightarrow{m}_{H_i''} + \overrightarrow{V}_{H_i''} C^\mathsf{T} G_i \left(\overleftarrow{m}_{K_i} - C\overrightarrow{m}_{H_i''}\right) \tag{23}$$

$$= \overrightarrow{m}_{H_i''} + \overrightarrow{V}_{H_i''} C^\mathsf{T} G_i \left(1 - C\overrightarrow{m}_{H_i''}\right) \tag{24}$$

$$\overleftarrow{V}_{H_i} = \overrightarrow{V}_{H_i''} - \overrightarrow{V}_{H_i''} C^\mathsf{T} G_i C \overrightarrow{V}_{H_i''} \,, \tag{25}$$

where

$$G_i \triangleq \left(\overleftarrow{V}_{K_i} + C\overrightarrow{V}_{H_i''} C^\mathsf{T}\right)^{-1} \tag{26}$$

$$= \left(s_H^2 + \sum_{m=1}^{M} \overrightarrow{V}_{H_{i,m}''}\right)^{-1} . \tag{27}$$

The corresponding dual representations are

$$\overleftarrow{\xi}_{H_i} = C^\mathsf{T} \overleftarrow{\xi}_{K_i} + \overrightarrow{\xi}_{H_i''} \tag{28}$$

$$= \frac{1}{s_H^2} C^\mathsf{T} + \overrightarrow{\xi}_{H_i''} \tag{29}$$

$$\overleftarrow{W}_{H_i} = C^\mathsf{T} \overleftarrow{W}_{K_i} C^\mathsf{T} + \overrightarrow{W}_{H_i''} \tag{30}$$

$$= \frac{1}{s_H^2} C^\mathsf{T} C + \overrightarrow{W}_{H_i''} \,. \tag{31}$$

For both representations, the parameter $s_H^2 \geq 0$ determines how much the final solution of H_i is allowed to deviate from the solution space described in (6). Its value needs to be specified during initialization in 'sh_squared' (default is 10^{-6}). Note that in the implementation of *ModelSelector*, messages described by their dual representations have been used exclusively.

2.4 Multiplication Node

Generally, the product of two Normally distributed random variables is no longer Gaussian. In the multiplication nodes shown in Figure 1 however, the values of $Z_{i,m}$ are assumed to be perfectly known. Therefore, the resulting backward messages through $S_{i,m}$ are still Gaussian and can be computed as

$$\overleftarrow{\xi}_{S_{i,m}} = \hat{m}_{Z_{i,m}}^\mathsf{T} \overleftarrow{\xi}_{R_{i,m}} \tag{32}$$

$$= 0 \tag{33}$$

$$\overleftarrow{W}_{S_{i,m}} = \hat{m}_{Z_{i,m}}^\mathsf{T} \overleftarrow{W}_{R_{i,m}} \hat{m}_{Z_{i,m}} \tag{34}$$

$$= \hat{m}_{Z_{i,m}}^\mathsf{T} \Sigma_{i,m}^{-1} \hat{m}_{Z_{i,m}} \,, \tag{35}$$

where

$$\hat{m}_{Z_{i,m}} = y_i - x_{i,m} \, . \tag{36}$$

With these results in mind, the backward messages through S_i can be constructed as

$$\overleftarrow{m}_{S_i} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{37}$$

$$\overleftarrow{V}_{S_i} = \begin{bmatrix} \left(\overleftarrow{W}_{S_{i,1}}\right)^{-1} & 0 & \dots & 0 \\ 0 & \left(\overleftarrow{W}_{S_{i,2}}\right)^{-1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \left(\overleftarrow{W}_{S_{i,M}}\right)^{-1} \end{bmatrix} \tag{38}$$

or

$$\overleftarrow{\xi}_{S_i} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{39}$$

$$\overleftarrow{W}_{S_i} = \begin{bmatrix} \overleftarrow{W}_{S_{i,1}} & 0 & \dots & 0 \\ 0 & \overleftarrow{W}_{S_{i,2}} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \overleftarrow{W}_{S_{i,M}} \end{bmatrix} . \tag{40}$$

For the implementations of class *ModelSelector*, the dual representations given by Equations (39) and (40) have been used exclusively. Note that if the assumed observation noises have been specified by their precision matrices (which is possible as explained in Subsection 2.1), calculating $\overleftarrow{W}_{S_i}$ does not require any matrix inversions!

2.5 Estimation of S_i

Similar to the method used to estimate the quantities in a PWC model, these implementations again rely on the ideas of IRLS.

Given the backward messages through L_i, H_i, and S_i in terms of their dual representations, the backward messages through S_i' can easily be calculated as their sum, i.e.,

$$\overleftarrow{\xi}_{S_i'} = \overleftarrow{\xi}_{L_i} + \overleftarrow{\xi}_{H_i} + \overleftarrow{\xi}_{S_i} \tag{41}$$

$$\overleftarrow{W}_{S_i'} = \overleftarrow{W}_{L_i} + \overleftarrow{W}_{H_i} + \overleftarrow{W}_{S_i} \, . \tag{42}$$

Recall that the messages through L_i and H_i are generated by NUV priors and, therefore, depend on the current estimates of $\hat{m}_{S_i}$ and $\hat{V}_{S_i}$. Given $\overleftarrow{\xi}_{S'_i}$ and $\overleftarrow{W}_{S'_i}$, these messages are now passed to the PWC model (i.e., an object of class *PWCModel*). This object then handles forward- / backward- message passing via BIFM, calculating the new estimates of S_i. Following the idea of IRLS, these new estimates of S_i are then used to "update" the messages generated by the 'Positivity' and 'One-Hot' NUV priors. This whole procedure is repeated till either the specified maximum number of iterations is reached (specified in 'n_it_irls) or the relative change of $\hat{m}_S$ falls below a certain threshold (specified in 'met_convTh').

References

[1] R. Keusch, "Composite nuv priors and applications," Ph.D. dissertation.

[2] H.-A. Loeliger, "Lecture notes for model-based estimation and signal analysis," 2023.

[3] ——, "On nup priors and gaussian message passing," 2023.