**Program Manual for State-Space Modeling Toolbox**

Types of observed signals which can be passed to the toolbox are:

1. Normal/Log-Normal
2. Gamma
3. Bernoulli
4. Normal/Log-Normal + Bernoulli
5. Gamma + Bernoulli

Observed signals might include both continuous or discrete scalar values.

The toolbox includes the following functions:

1. ay\_create\_state\_space
2. ay\_set\_learning\_param
3. ay\_em
4. ay\_filtering
5. ay\_sampling
6. ay\_set\_censor\_threshold

Internal functions of this toolbox re:

1. ay\_Qk
2. ay\_Tk
3. ay\_x\_2\_rp

**References:**

1. Yousefi, Ali, et al. "Cognitive state prediction using an EM algorithm applied to gamma distributed data." *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2015.
2. Eden, Uri T., and Emery N. Brown. "Continuous-time filters for state estimation from point process models of neural data." *Statistica Sinica* 18.4 (2008): 1293.

Written by Ali Yousefi - Oct 17, 2016

**ay\_create\_state\_space**

It creates the state space model for a specific task. The function defines how each continuous and discrete part of the model is defined, and how they are linked to the state variables. This is the first function requires to be called in this toolbox.

**Syntax**

Param = ay\_create\_state\_space(nx,nUk,nIn,nIb,xM,cLink,cLinkUpdate,dLink,dLinkUpdate)

**Description**

Param = ay\_create\_state\_space(nx, nUk,nIn,nIb,xM,cLink,cLinkUpdate,dLink,dLinkUpdate)

The function returns the parameters of the specified behavioral model defined by the function input arguments. The Param keeps the model structure and necessary parameters

**Input Arguments**

|  |  |
| --- | --- |
| nx | A scalar that defines dimension of state variable ()  **Example:** , this means is a vector with dimension 2 - |
| nUk | A scalar that defines the dimension of input to the state-space model  , is the input vector with “” elements.  **Example:** , for instance the has a dimension of 2. The “, ” are the components of the input to the state transition model.  For example, at time , , which means is equal to 0 at time 1.  Note that can be set to zero when there is no term in the model. |
| nIn | A scalar that defines the dimension of input passed to the continuous part of the model  ; is the input vector with “” columns and are the model free parameters.  Note that will be a matrix with size , where is the length of process.  For the Normal distribution, the function is a linear function of the and other model parameters:  Note that and are functions of and , and variable is the observation process noise.  For the Gamma distribution, you might check the function definition described in **Reference 1**.  **Example:**  for instance has a dimension of 5. The “, , , , and ” are indicator functions for the behavioral task which are the input to the continuous part of the model.  For example, at time , . |
| nIb | A scalar that defines the dimension of the input to the discrete – binary - part of the model  . is the input vector with “” elements, and the function is defined by a sigmoid function:  Note that and are functions of and .  **Example:** , for instance has dimension of 5. The “, , , , and ” are indicator functions for the behavioral task which are the input to the continuous part of the behavioral model.  For example, at time , . |
| xM | A transform matrix which determines how the state variables are used in both and observation processes. The default transform matrix is an identity matrix, but it can be a rectangular matrix with a larger number of rows than its columns.  Elements of the are 0 and 1, and sum of each row is equal to 1.  Note that rather than using in the continuous or discrete process, we use . Using the , we have more flexibility in defining the observation processes used in the model. When is an identity matrix, each state variable can be linked to one of the input elements, whereas using a rectangular , we will be able to define behavioral models for which each state variable can be linked to more than one input element. For example, we might have the following behavioral model:  here, the state variable is linked to two input elements. To build this model, we need to set ; thus, the state variable – here – is linked to both and . Note that and will be columns of input passed to the model.  **Default:** an identity matrix  **Example:** , the state variable utilized in the behavioral model is treated as a 3-component state variable, where the second and third components are the same. Note that the must be 2. |
| cLink | A matrix of size , where is the number of rows.  In this toolbox, we assume that there are state variables rather than , and defines which of column is linked to each state variable – check in ay\_em function. Elements of the matrix can have values between to .  **Example:** , this means there are 3 state variables, and the first state variable is linked to the first column of and so on – note that, should be at least with a length of 4. The suggests that dynamic part of the continuous process will be defined by: – here, () are model parameters, and .  We will explain in the cLinkUpdate definition, which of the parameters are fixed or need to be adjusted. |
| cLinkUpdate | A matrix of size , similar to cLink. Elements of the matrix are 0 and 1, and it determines whether the corresponding parameter - – will be updated or not. A value of 1 means will be updated, and value of 0 means is fixed. The default value for the fixed parameters is 1. Check the cLink argument description.  **Example:** , this means only the first variable will be updated using EM algorithm. Note that this refers to the parameter linked to the first state variable not the first column of . Check cLink example, this means the continuous part of the continuous process is defined by: . In the other words, it means and will be adjusted through the EM algorithm. |
| dLink | dLink is equivalent to cLink for discrete process. Check the cLink for further information. |
| dLinkUpdate | dLinkUpdate is equivalent to cLinkUpdate for discrete process. Check the cLinkUpdate for further information. |

**Output Arguments**

|  |  |
| --- | --- |
| Param | A structure array that defines the state-space and behavioral model structure and paramaters. Note that most of elements are filled with their default values and they need to be set either manually or by calling ay\_em function.  The Param structure fields are:   * **nx** – Check the input argument * **cLinkMap** – It is equal to cLink * **dLinkMap** – It is equal to dLink * **nIn** – Check the input argument, number of In elements. * **nIb** – Check the input argument, number of Ib elements. * **Ak** – The state-transition model is defined by:   The Ak defines the variable of the state-transition model.   * **Bk** – the state-transition model is defined by:   The Bk defines the variable of the state-transition model.   * **Wk** – the state-transition model is defined by:   The W defines the noise covariance term, and it is assumed to be stationary with a zero-mean.   * **X0** – The state-variable estimate at time 0. It is assumed that state variable at time 0 has a Normal distribution with mean X0 and variance W0. * **W0** – The covariance matrix of the state-variable estimate at time 0. * **Ck** – Model parameters for the section of the continuous observation process:   Note that at time is defined by: - is an element-by-element multiplication, and is a dot product.   * **Dk** – Model parameters for the of the continuous observation process:   Note that at time is defined by: . elements are either 0 or 1, and it defines which columns of the contribute in the stationary part of the observation process.  is equal to zero for the columns identified in cLink; thus, the stationary part of the observation process is define by a linear combination of input columns not being linked to state variables. Note the is an internal parameter of the toolbox, which is derived from cConstantUpdate.  Note that, if we want to have a term similar to: in the observation process, we should replicate twice in the definition.   * **cConstantUpdate** – It is a vector of length , and it determines which elements of will be set to zero. Elements of the cConstantUpdate are either 0 or 1, where elements with value 1 determines columns of used in the stationary part of the observation process. Note that is derived using the cConstantUpdate. The cConstantUpdate is 1 for the input columns not linked to the state variables and it is set 0 for the input columns lined to the state variables. * **Vk** – The variance for the continuous observation noise, . This will be dispersion term for the Gamma distribution. Check **Reference 1** for further information. * **Ek** – It is equivalent to Ck, and it is used for the discrete part of the state-space model. * **Fk** – It is equivalent to Ek, and it is used for the discrete of the state-space model. * **dConstantUpdate** – It is equivalent to cConstantUpdate, and it is used for the discrete part of the model. * **xM** – check the input argument |

**Example**

Param = ay\_create\_state\_space(3,1,4,4,eye(3,3),[1 3 4],[1 1 1],[1 3 4],[0 0 0]);

1. The state-variable dimension is set to be 3.
2. The input to the state-space transition model is a scalar, length 1.
3. Given items in 1 and 2, the state transition process is:
4. The input to both the continuous and discrete model of the behavior are vectors of the length 4.
5. The xM is an identity matrix, thus we have a state vector with 3 elements.
6. The cLink is [1 3 4], which means the first state variable is linked to the 1st element of , the second state variable is linked to 3rd element of , and the third state variable is linked to the 4th element of
7. The cLinkUpdate is [1 1 1], which means parameters will be updated. Check cLinkUpdate.
8. Given items 6 and 7, the observation process for a normal distribution is defined by:
9. The dLink is [1 3 4], which share a similar structure of the continuous model of the behavior.
10. The cLinkUpdate is [0 0 0], which means neither of parameters will be updated. Check dLinkUpdate.
11. Item 10 is very important to avoid miss-specified models of the observed signals.
12. Given items 9 and 10, the observation process for the discrete process is defined by:
13. Check items 8 and 11 to find the relationship between cLink, cLinkUpdate, dLink, and dLinkUpdate with the process model. Note that cLinkUpdate and dLinkUpdate must be carefully chosen, otherwise we might a miss-specified model.

**Note**

There are a group of behavioral model parameters which are defined by pre-defined assumptions. These parameters include: Ak, Bk, Wk, X0, W0, Ck, Dk, Ek, Fk, Vk, cConstantUpdate, dConstantUpdate

We will see later that some of these parameters will be updated, while some are fixed. These parameters can be changed by either ay\_create\_state\_space function or even manually. Specifically, the cConstantUpdate and dConstantUpdate are defined by a specific assumption that how the input to state-space model will be linked to constant parameters or state variables, which is part of the ay\_create\_state\_space function. We can change the structure of the behavioral model manually by changing elements of Param variables.

**ay\_set\_learning\_param**

It sets learning parameters used in EM algorithm. Note this function can be only called after ay\_create\_state\_space function. Using this function, we determine how the parameters of the models will be updated later using ay\_em function. Note that ay\_create\_state\_space defines a proper structure of the model, while the ay\_set\_learning\_param how parameters of the state-space model will be trained using the EM algorithm.

**Syntax**

Param = ay\_set\_learning\_param(Param,Iter,UpdateStateParam,UpdateStateNoise,UpdateStateX0,UpdateCModelParam,

UpdateCModelNoise, UpdateDModelParam,DiagonalA,UpdateMode,UpdateCModelShift)

**Description**

Param = ay\_set\_learning\_param(Param,Iter,UpdateStateParam,UpdateStateNoise,UpdateStateX0,UpdateCModelParam,

UpdateCModelNoise, UpdateDModelParam,DiagonalA,UpdateMode,UpdateCModelShift)

This function sets how the model parameters will be adjusted using the EM learning procedure run by ay\_em.

**Input Arguments**

|  |  |
| --- | --- |
| Param | A structure array consisting both the structure and parameters of the state-space model parameters along with EM learning procedure.  Check Param in ay\_create\_state\_space for a more complete description. |
| Iter | Number of EM training iterations.  **Example:** . The EM training iteration number is 100. |
| UpdateStateParam | It can be either 0 or 1.  Value 1 means that the state-transition model parameters, elements of A and B matrices, will be updated. Value 0 means that these parameters are not updated in the EM training process.  State transition process model is defined by:  **Example:** . The state-transition process A, and B matrixes are being trained in the EM learning process.  Note that we either update and or none of them. It is also possible to train only or ; this requires change in the ay\_em code. |
| UpdateStateNoise | It can be either 0 or 1.  Value of 1 means that the state-transition model covariance matrix will be updated. It is assumed the W, covariance matrix, is diagonal. Value 0 means that the W won’t be updated in the training process.  State transition process model is defined by:  **Example:** . The state-transition process parameters are trained in the EM learning process. |
| UpdateStateX0 | It can be either 0 or 1.  Value of 1 means that the state variable parameters at time 0 – mean and covariance matrix - will be updated. Value 0 means that the initial parameters of the state variable at time 0 won’t be updated in the EM training process.  State transition process model is defined by:  **Example:** . It means and – mean and covariance matrix - of the initial value of the state-variable is estimated in ay\_em function. |
| UpdateCModelParam | It can be either 0 or 1.  Value of 1 means parameters of the continuous observation process will be updated in the EM training process. The parameters are a sub-set of parameters in the set to be trained and all the paramaters in vectors. Read the cLinkUpdate in ay\_create\_state\_space function to find which subset – or index - of the parameters will be updated.  Value of 0 means parameters of the continuous observation process won’t be updated in the EM training process.  **Example:** . It means paramaters of the continuous pbservation process – elements of and vectors – will be updated in the EM training process. |
| UpdateCModelNoise | It can be 0 or 1.  Value of 1 means the observation process noise-term will be updated through the EM training process, ay\_em. For the continuous observation process with a normal or log-normal distribution, the noise term – – represents additive Normal white noise. For the observation process with a Gamma distributed signal, the term corresponds to the dispersion term. For further information about the canonical form of the Gamma distribution and dispersion term definition check the **Refernece 1**.  For the Normal observation process, we have:  here, is a white noise with a Normal distribution with a variance of .  **Example:** . It means the noise-term of the observation process is getting updated through the EM training process. |
| UpdateDModelParam | It can be 0 or 1.  It is equivalent to the UpdateCModelParam for continuous observation process. Check the UpdateDModelParam for further explanations.  Note that for the discrete process, there is no noise-term to be updated. |
| DigonalA | It can be 0 or 1. Default is 1.  Value 1 means the matrix in the state-transition process model will have a diagonal matrix. The diagonal assumption is the favorable choice for EM learning, note that state variables will be dependent through observation process even is a diagonal. Note the posterior estimates of the state variables are not necessarily independent.  Value 0 means the matrix is a full matrix. For the EM algorithm, this might lead to a less accurate estimate or even instable estimation of both parameters and state-variables.  Note that whether is going to be updated ot not, will be determined by UpdateCModelParam. In many occasion, we look forward a random-walk type process, where we set to be an identity matrix and we exclude it from the training process.  **Example:**  – default value. It means the matrix is a diagonal matrix. |
| UpdateMode | It can be 1 or 2. Default is 2.  It determines which update method is getting used in the Filter mean and covariance update. Note that we use a Gaussian approximation method, we generally might have different update rules for the mean and co-variance of the state variables. We are using two different update rules:   1. First update Mean, and then update Covariance - **UpdateMode 1** 2. First update Covariance, and then update Mean – **UpdateMode 2**   Picking the right update rule is dependent on the observation process and is characteristics, specifically variability and noise. In practice, the good starting point is **UpdateMode 2.** To identify the better update rule, we can check both estimation and ML result using both update rules and find the one which gives a higher ML or a more stable estimation result.  For a further detail about which of these approximate might give a better result, you can check **Reference 2**. **Reference 2** describes the Gaussian approximation for the point process model, which can be extended for other distributions including Gamma, Bernouli, and or mixture of these distributions.  **Example:**  – default value. It means the Filter estimate of the state-variable is based on the second method used in the posterior Gaussian approximation. |
| UpdateCModelShift | It can be 0 or 1.  It determines whether the estimation of a positive time shift will be part of the EM training process or not. For the Gamma distribution, we define a positive time shift and we can tune this parameter using the EM algrotihm. A value of 1 means that the time shift parameter will be updated in the EM training process.  A value of 0 means that this time shift parameter is pre-set and it is not getting updated in the EM training process, ay\_em function.  Note that this is the specific parameter of the Gamma distribution, and it is not used in the observation process with Normal or Log-Noraml distributions.  For a further information about the Gamma distribution, please check the **Reference 1**.  **Example:**  . The time shift parameter will be updated if the continuous observation process is set to be Gamma. |

**Output Arguments**

|  |  |
| --- | --- |
| Param | An updated structure array consisting of both the state-space model structure/parameters and EM learning parameters.  The new fields of the Param structure are:   * Iter – check the input argument * UpdateStateParam – check the input argument * UpdateStateNoise – check the input argument * UpdateStateX0 – check the input argument * UpdateCModelParam – check the input argument * UpdateCModelNoise – check the input argument * UpdateDModelParam – check the input argument * DigonalA – check the input argument * UpdateMode - check the input argument * UpdateCModelShift - check the input argument   Note that Param will also include the state-space model parameters defined by ay\_create\_state\_space as well. |

**Example**

% create state space model

Param = ay\_create\_state\_space(3,1,4,4,eye(3,3),[1 3 4],[1 1 1],[1 3 4],[0 0 0]);

% set the learning parameters

Iter = 300;

Param = ay\_set\_learning\_param(Param,Iter,0,1,1,1,1,1,1,2,1);

**Note**

In ay\_create\_state\_space, we determine which part of the proposed dynamical model will have free parameters. Using the ay\_set\_learning\_param function, we determine if these free parameters will be updated or not.

For the state-transition model, we can:

1. Set whether A to be diagonal or full matrix
2. Set whether elements of A and B matrices will be updated or not – Both *A* and *B* will determine the state-transition dynamics, and for this reason, we either update both *A* and *B* or none of them. We might change the ay\_em function, if separate update of *A* and *B* is desired.
3. Set whether the noise covariance matrix – which is assumed to be diagonal – will be updated or not.
4. Set whether X0 variable distribution will be updated or not.

For the continuous behavioral model, we can:

1. Set whether model parameters to be updated or not – subset of the *C* and all D elements.
2. Set whether observation noise will be updated or not.
3. For the Gamma distribution, we can determine whether the shift term will be updated or not.

For the discrete behavioral model, we can set whether model parameters to be updated or not.

It is possible to change the code to give more flexibility to the training process. For instance, we might consider to have *A* matrix fixed and just train *B*. This requires a small change in the EM training algorithm.

**ay\_set\_censor\_threshold**

It sets censor threshold for the continuous observation process. The censored threshold will be added to the Param structure. We assume on the censored trials, the continuous observation process exceeds the censoring threshold and both continuous and disrcetre – possible extra – obseravtions are unobserved. Check obs\_valid argument in the **ay\_em** for further definition of the censored or missed trials.

**Syntax**

Param = ay\_set\_censor\_threshold(Param,censor\_thr)

**Description**

Param = ay\_set\_censor\_threshold(Param,censor\_thr)

This function gets the Param structure plus the censored value, censor\_thr, and it updates the Param structure. The censor\_time in the Param keeps the value of censoring threshold.

**Input Arguments**

|  |  |
| --- | --- |
| Param | A structure array consisting both the state-space and observation process model parameters along EM learning parameters.  Check Param in ay\_create\_state\_space and ay\_set\_learning\_param for a more complete description. |
| censor\_thr | It define the continuous observation process censor time.  The censor\_time variable of the Param will be equal to censor\_thr.  **Example:** . It means the censored tials have a value larger than censor\_thr. |

**Output Arguments**

|  |  |
| --- | --- |
| Param | An updated structure array of Param consisting of model structure, model parameters and EM learning parameters. The new field of the Param structure is: censor\_time - check the input argument |

**Example**

censor\_time = 1;

Param= ay\_set\_censor\_threshold(Param,censor\_time);

This means that the censored trials will have values larger than censor\_time which is 1 here.

**ay\_em**

This function runs EM algorithm to adjust model parameters. This is the **core** function of the toolbox; it gets model structure, input and observation data and it returns updated model parameters plus its state estimates. It also returns the ML estimate which show progression of the ML through multiple iterations.

**Syntax**

[rXSmt,rSSmt,Param,rXPos,rSPos,ML,EYn,EYb,rYn,rYb]=ay\_em(DISTR,Uk,In,Ib,Yn,Yb,Param,obs\_valid)

**Description**

[rXSmt,rSSmt,Param,rXPos,rSPos,ML,EYn,EYb,rYn,rYb]= ay\_em(DISTR,Uk,In,Ib,Yn,Yb,Param,obs\_valid)

The function gets input arrays (Uk,In,Ib), observation signals (Yn,Yb, obs\_valid), model parameters (Param), and type of observation processes (DISTR), and returns an estimate of state variables (smoother - rXSmt,rSSmt -, plus filter - rXPos,rSPos) along with the likelihood value on each iteration (ML), expected observed signals (EYn,EYb), and updated observed signals along imputated data for censored trials (rYn,rYb).

**Input Arguments**

|  |  |
| --- | --- |
| DISTR | A 2 element vector which defines the type of the observed signals and their corresponding distributions.  **DISTR=[1 0]** means there is only continuous observation signal and the proper distribution for the signal is Normal. Note that the input argument Yn carries the continuous signal.  **DISTR=[2 0]** means there is only continuous observation signal and the proper distribution of the signal is Gamma. Note that the input argument Yn carries the continuous signal.  **DISTR=[0 1]** means there is only discrete observation – a Bernoulli observation process.  Note that the input argument Yb carries the continuous signal.  **DISTR=[1 1]** means there are both continuous and discrete observation signals and the proper distribution for the signal is a mixture of Normal and Bernoulli distributions. Note that the input arguments Yn and Yb carry the mixed signals.  **DISTR=[2 1]** means there are both continuous and discrete observation signals and the proper distribution for the signal is a mixture of Gamma and Bernoulli distributions. Note that the input arguments Yn and Yb carry the mixed signals.  **Example:** , it means the observed signal includes only continuous observation. Note that we might define the state-space model with both continuous and discrete observations; the EM algorithm might process either continuous or discrete parts of the process given the elements. |
| Uk | A matrix of size which defines the input to the state-transition model – is the length of observation, and is number of elements per each time index. The state-transition process is defined by:  **Example:** , it mean that input is a fixed value of 1 at each time index. Note that it is assumed that is equal to 100, which is the length of the observation. |
| In | A matrix of size which defines the input to the continuous observation process. For further information about how columns of is utilized in the continuous observation process, check the ay\_create\_state\_space function.  **Example:** ; it means a random input with 5 elements at each time index is passed to the continuous observation process. Note that it is assumed that is equal to 100, which is the length of the observation.  Practically, the elements could be indicator functions of the task factors or even continuous values, including history terms.  Note that , , and share the same number of rows. |
| Ib | A matrix of size which defines the input to the discrete observation process. It is equivalent to of the continuous part.  **Example:** ; it means a random input with 5 elements at each time index is passed to the discrete observation process. Note that it is assumed that is equal to 100, which is the length of the observation.  Practically, the elements could be indicator functions of the task factors or even continuous values, including history terms.  Note that , , and share the same number of rows. |
| Yn | A vector of size which carries the continuous observation signal - is the length of observation process.  Values of Yn can real values or NaN. Note that for a NaN observation signal, the corresponding obs\_valid must be either 0 or 2. Check obs\_valid for further information.  Note that for an observation process with a Normal distribution, , the Yn elements might be real values. For the observation process with a Gamma distribution, , the Yn elements might be positive real-valued.  **Example:** ; it means a positive random input with 100 rows.  Note that , , , and share the same number of rows. |
| Yb | A vector of size which carries the discrete observation signal - is the length of observation process.  Values of Yb are 0, 1, or NaN. Note that for a NaN observation signal, the corresponding obs\_valid must be either 0 or 2. Check obs\_valid for further information.  Value of 0 means an incorrect response, and a value of 1 means a correct response.  **Example:** , which means a vector with random values of 0 or 1 at each row.  Note that , , , , and share the same number of rows. |
| Param | A structure array consisting both the state-space model structure plus EM learning parameters.  Check Param in ay\_create\_state\_space and ay\_set\_learning\_param for a more complete description. |
| obs\_valid | A vector of length Kwith elements of 0, 1, and 2.  An element with value 1 means the observed signal is valid.  An element with value 0 means the observation is missing.  An element with value 2 means the observation is censored. The censoring threshold is defined by the Param.censor\_time.  Note that a NaN value in either Yn or Yb must be accompanied by a value 0 or 2 in obs\_valid, otherwise the EM algorithm fails to have a correct estimate of the model parameters and state variables.  Note that obs\_valid determines how the EM algorithm treats each row of the observed signal in running either E- or M-step of the EM process. Thus, setting correct values of the obs\_valid has a significant importance in running the model correctly.  **Example:** , it means all elements of the observed signals are filled with valid values.  Note that , , , , , and share the same number of rows. |

**Output Arguments**

|  |  |
| --- | --- |
| rXSmt | The mean value of the state variable smoother result.  A structure array of length K, where K is the number of observation samples. Each element of the is a vector of length . |
| rSSmt | The covariance matrix of the state variable smoothing result.  A structure array of length K, where the K is number of observation samples.  Each element of the is a matrix of size . |
| Param | Updated Param structure array which returns EM learning result.  Note that Param keeps all the model parameters. |
| rXPos | The mean value of the state variable filter result.  A structure array of length K, where K is the number of observation samples.  Each element of the is a vector of length . |
| rSPos | The covariance matrix of the state variable filter result.  A structure array of length K, where the K is the number of observation samples.  Each element of the is a matrix of size . |
| ML | Maximum likelihood value of the EM algorithm at each iteration.  The ML is a vector of the length Iter. Iter is the number of EM iterations, and it is passed through Param to ay\_em.  Practically, the ML might be an increasing function of the iteration. In the other words, it is expected to grow as the number of iteration increases.  ML growth rate or reaching a plateau might be an indication of enough number of EM iteration. The ML value can be used as the stop criterion of the algorithm. This can be added to ay\_em if it is needed.  Because we start with a random start value for model parameters, there is a chance to see a non-increasing ML on first iterations of the algorithm. It is suggested to set parameters of the model using a static version of the model using results of a GLM analysis. Using a reasonable starting point, we might reduce number of EM iterations. Furthermore, we might get a more well-behaved ML curve as well. |
| EYn | A vector of length which returns the expected value of the continuous process at each time index.  In the Normal observation process, this is the deterministic part of the observation process estimated by the model – or simply mean estimate of the process at each time index. This is defined by:  In the Gamma observation process, this is equal to the mean of the observation process at each each. Check **Reference 1** for further information. This is defined by:  Note that parameters in and are updated paramaters. |
| EYb | A vector of length which returns the expected probability of the correct choice at each time index.  The expected probability at each time index is defined by:  Note that parameters in and are updated paramaters. |
| rYn | Updated Yn Vector. It has the same length of Yn.  Elements of the vector corresponding to a censored data are replaced by an imputation technique. Check ay\_sampling for further information. |
| rYb | Updated Yb Vector. It has the same length of Yb.  Elements of the vector corresponding to a censored data are replaced by an imputation technique. Check ay\_sampling for further information |

**Example**

Here, we assume input elements are properly loaded and assigned.

%% First step, build state space model - create behavioral model

Param =

ay\_create\_state\_space(3,0,3,3,eye(3,3),[1 2 3],[0 0 0],[1 2 3],[0 0 0]);

%% Second step, define learning parameters

Iter = 300;

Param = ay\_set\_learning\_param(Param,Iter,0,1,1,1,1,1,1,2,1);

%% Third step, define censore time if it is necessary

censor\_time = 1;

Param= ay\_set\_censor\_threshold(Param,censor\_time);

%% Fourth Step, EM Algorithm

[XSmt,SSmt,Param,XPos,SPos,ML,EYN,EYb,rYn,rYb]=ay\_em([1 0],[],In,[],Yn',[],Param,valid);

**ay\_filtering**

This function runs the filter algorithm given current observed signals. This function is appropriate for real-time filtering, when the observed signals are updated trial-by-trial. The function gets current observation signals plus an estimate of the state variable from the previous trial to estimate the current estimate of the state variables. The Param structure normally comes from the ay\_em ran over training dataset.

**Syntax**

[XPos,SPos,YP,YB]=ay\_filtering(DISTR,Uk,In,Ib,Yn,Yb,Param,obs\_valid,XPos0,SPos0)

**Description**

[XPos,SPos,YP,YB]=ay\_filtering(DISTR,Uk,In,Ib,Yn,Yb,Param,obs\_valid,XPos0,SPos0)

It runs an on-line filter algorithm on each new trial – or time-index – of the data.

**Input Arguments**

|  |  |
| --- | --- |
| DISTR | Check ay\_em for further information |
| Uk | A vector of size which defines the current values of the input to the state-transition model.  Note that the ay\_filtering will be called per trial.  For further information, check ay\_em function  **Example:** , it means the current input to the state-transition process is a scalar value of 1. |
| In | A vector of size which defines current value of the input to the continuous process of the model.  Note that the ay\_filtering will be called per trial.  For further information, check ay\_em function  **Example:** , it means a random input with 5 elements at each trial. |
| Ib | A vector of size which defines current value of the input to the continuous process of the model.  Note that the ay\_filtering will be called per trial.  For further information, check ay\_em function  **Example:** , it means a random input with 5 elements at each trial. |
| Yn | A scalar value either real value or NaN.  For further information, check ay\_em function definition.  **Example:** , it means a random input for current observation of the continuous signal.  Note that in practice, this is observed behavior or equivalent signal |
| Yb | A scalar value either 0, 1, or NaN.  For further information, check ay\_em function definition.  **Example:** , it means a correct response for current observation of the discrete signal.  Note that in practice, this is observed behavior or equivalent signal. |
| Param | Updated Param structure. In practice, this is the output of the ay\_em ran over the training dataset.  A structure array consisting both the state-space and behavioral model parameters plus EM learning setup.  Check Param in ay\_create\_state\_space and ay\_set\_learning\_param for a more complete description. |
| XPos0 | The mean value of the state variable filter on the previous trial.  The is a vector of length . |
| SPos0 | The covariance matrix of the state variable filter on the previous trial.  The is a matrix of size . |
| obs\_valid | A scalar variable with values of either 0, 1, or 2.  A value of 1 means a valid observed signal/s.  A value of 2 means a censored observed signal/s.  A value of 0 means a missing observed signal/s.  For further information, check ay\_em function definition. |

**Output Arguments**

|  |  |
| --- | --- |
| XPos | The mean value of the state variable filter on the current trial.  Note that, it is being a function of the previous trial state estimate plus the new observed signals.  The is a vector of length . |
| SPos | The covariance matrix of the state variable filter on the current trial.  Note that, it is being a function of the previous trial state estimate plus the new observed signals.  The is a matrix of size . |
| YP | A scalar value which returns the expected value of the continuous process at current trial.  It is a function of the current estimate of the state variable and the trial input variables – and . |
| YB | A scalar value which returns the expected probability of correct decision at current trial.  It is a function of the current estimate of the state variable and the trial input variables – and . |

**Example**

Here, we assume the Param is the output of the ay\_em algorithm ran over the training dataset. The ay\_filtering is called twice, once for the first trial and then on the second trial of the task. Note that how we set our initial estimates of the mean and covariance of the state variables.

% Sample, this function is required for real-time state estimate

XPos0=[0;0];

SPos0=[10 0;0 10];

% first observation (note In,Ib are vectors of length 1xnIn, 1xnIb and similarly Uk)

% XPos0 and SPos0 are the estimate of state mean and variance from previous one

% we assume data is valid

[XPos0,SPos0]=ay\_filtering([1 0],[],In(1,:),[],Yn(1),[],Param,1,XPos0,SPos0);

% second observation

[XPos0,SPos0]=ay\_filtering([1 0],[],In(2,:),[],Yn(2),[],Param,1,XPos0,SPos0);

**ay\_sampling**

This function is called to generate a sample of continuous and discrete signals on the censored trials. This function is used in the imputation technique used in ay\_em on the censored trials. Current version of the ay\_em fetches a sample of data given the most update of the observation distribution on the censored trials, and run the EM algorithm using the imputated sample.

**Syntax**

[Yn,Yb]=ay\_sampling(DISTR,Cut\_Time,Uk,In,Ib,Param,XPos0,SPos0)

**Description**

[Yn,Yb]=ay\_sampling(DISTR,Cut\_Time,Uk,In,Ib,Param,XPos0,SPos0)

It generates a sample (Yn,Yb) for continuous and discrete observation signals, given current inputs of the model (Uk,In,Ib) and previous estimate of the state-variables (XPos0,SPos0). Cut\_Time variable defines the censoring threshold; note that censoring criteria is only conditioned on the value of the continuous signal. Param keeps the updated set of the model parameters.

**Input Arguments**

|  |  |
| --- | --- |
| DISTR | It is similar to DISTR defined in ay\_filtering and ay\_em . Check ay\_em or ay\_filtering for further information |
| Cut\_Time | It is a scalar value which define the censoring threshold.  A copy of this value is carried by Param as well. |
| Uk | It is similar to Uk defined in ay\_filtering. Check ay\_filtering for further information. |
| In | It is similar to In defined in ay\_filtering. Check ay\_filtering for further information |
| Ib | It is similar to Ib defined in ay\_filtering. Check ay\_filtering for further information  . |
| Param | It is similar to Param defined in ay\_filtering. Check ay\_filtering for further information |
| XPos0 | It is similar to XPos0 defined in ay\_filtering. Check ay\_filtering for further information |
| SPos0 | It is similar to SPos0 defined in ay\_filtering. Check ay\_filtering for further information |

**Output Arguments**

|  |  |
| --- | --- |
| Yn | A scalar variable returning a sample of the continuous signal over the censored threshold. |
| Yb | A scalar discrete variable returning a sample of the discrete signal over the censored threshold.  Yb can be either 0 or 1. |

**Example**

Here, the ay\_sampling is called inside the ay\_em function to create a sample data for the censored trials.

if obs\_valid(k)==2

[tYP,tYB]=ay\_sampling

(DISTR,censor\_time,Uk(k,:),In(k,:),Ib(k,:),Param,XPre{k},SPre{k});

if DISTR(1)>0 Yn(k)=tYP; end;

if DISTR(2)==1 Yb(k)=tYB; end;

end