Optimized Bayesian Dynamic Advising – Examples

M. Kárný, J. Andrýsek, J. Böhm, T. V. Guy, J. Homolová, L. Jirsa,
J. Kracík, I. Nagy, P. Nedoma, L. Pavelková, E. Suzdaleva, L. Tesař

Read Use instructions **or go to** Contents **to select examples**

This CD complements the book *Optimized Bayesian Dynamic Advising* and provides a selection of interactive examples illustrating the book's content.

It begins with simple examples intended to help the reader understand the adopted Bayesian paradigm. Chapter 1 addresses tasks of a discrete type, while Chapter 2 addresses tasks of a continuous type. The advanced examples, Chapters 3, 4 and 5, give a deeper insight into the methodology and developed solutions. The advanced examples cover:

- *Learning techniques*, illustrated on normal mixtures, Chapter 8{243} of the book, as well as on the example related to controlled Markov chain mixtures, Chapter 10{377} of the book.
- *Advising designs*, illustrated on examples of academic, industrial and simultaneous advising for normal mixtures, Chapter 9{309} of the book, and for Markov chain mixtures, Chapter 11{411} of the book.

Particular tasks solved by an example are formulated as *decision-making (DM) problems*. Each example illustrates interconnected decision-making tasks of different complexity and abstraction. Interconnection concerns DM problems *internally solved* by the example and DM problems, where the reader is invited to play a role of *external decision maker*. The DM tasks solved by the example are provided with a detailed formal description using notions and notations introduced in the book.

The description of examples has the following uniform structure.

Aim:
General description of the DM problems, where the reader can act as an external decision maker.

Description:

An informal description of the DM problems, internally solved by an example.

Specification:
Formalization of the decision-making problem solved by an example. It includes description of the basic elements forming the presented decision problem under uncertainty.

System: the part of the world to which the decision problem relates.

Decision: quantities that can be chosen for reaching decision-making aims.

Experience: information available for selecting optimal decisions.

Ignorance: uncertainty influencing the solved decision-making problem.

Admissible strategies: the class of strategies within which optimal strategy is searched for.

Loss function: mapping of the behavior that quantifies the adopted decision-making strategy.

Some of these items may be missing, e.g., when admissible strategies are causal without other restrictions.

Recommended experiments:
Indications to the reader what effects and relationships are worth observing.

The examples are based on an extensive MATLAB toolbox Mixtools. See:

P. Nedoma, J. Böhm, T. V. Guy, L. Jirsa, M. Kárný, I. Nagy, L. Tesař, and J. Andrýsek, "Mixtools: User's Guide," Tech. Rep. 2060, ÚTIA AV ČR, Prague, 2002

P. Nedoma, M. Kárný, T. V. Guy, I. Nagy, and J. Böhm, *Mixtools.*, ÚTIA AV ČR, Prague, 2003.

USE

- Go to the **Contents** on the next page.
- Click on **Chapter** if you want to read the introduction to a whole group of examples.
- Click on **Section** if you want to read the introduction to a subgroup of examples.
- Click on **Subsection** if you want to go directly to an example.

You can always reach directly the Contents by clicking on the box or rely on navigating arrows.

References to the illustrated book have the forms Chapter 2{11}, Section 6.4{120}, Subsection 6.4.2{122}. The number in { } refers to the book page.

RUN OF PARTICULAR EXAMPLES

- Read the description of the example.

  You can get additional information by clicking on basic notions, like System, or on optional variables, like *ndat*.

  In a complex case, additional information can be obtained by clicking on the highlighted text like CCM model.

- Run the example by clicking on run example. It opens MATLAB and runs the corresponding script controlled by the following dialogues.

  1. Modification of optional parameters of the example:

     ```
     any changes ? ENTER = no, or type a command: >
     ```

     This is essentially keyboard mode of MATLAB. Thus, experts can see and modify other variables here. This dialogue is closed by the statement

     ```
     >>>  when ready, finish by ENTER
     ```

  2. Decision on continuation of the example:

     ```
     >> ENTER: continue  I: initial values  S: stop >
     ```

- Some examples provide ready results you can see by clicking on the box See results.
- Note that legends on figures can be moved by mouse to see what is behind them.

Naturally, the permanent improvements of the Mixtools toolbox are not reflected in this CD. Please, contact the authors at school@utia.cas.cz if you want to get the newest version. Authors will be thankful for any suggestion for improving the presented material.

Contents

www.utia.cas.cz/AS

Contents

Chapter 1

Basic Tasks with a Discrete Model: Controlled Coin with Memory (CCM)

This chapter contains examples related mostly to Chapter 2{11}. They illustrate the adopted Bayesian dynamic decision-making theory on rather simple examples that can be interpreted as a controlled tossing of a coin exhibiting a memory. The model of this coin is discrete in variables, which can have just two values labelled by numbers 1 and 2. The model has the form

$$f(y_t|\psi_t,\Theta)=\prod_{\psi\in\psi^*}\left[\Theta_{1|\psi}^{\delta_{y_t,1}}\Theta_{2|\psi}^{\delta_{y_t,2}}\right]^{\delta_{\psi_t,\psi}},$$

where

$y_t\in y^*=\{1,2\}$ and $u_t\in u^*=\{1,2\},\ t\in t^*$ are the output and input of the "coin" system;

$\psi_t\equiv[u_t,y_{t-1}]'\in\psi^*=\{[1,1]',[1,2]',[2,1]',[2,2]'\}$ is regression vector;

$\Theta=\Theta_{y|\psi}$ is a parameter of the CCM model;

$\delta_{\bullet,\bullet}$ is Kronecker symbol.

The model parameter Θ can be presented in the following form

$$\Theta_{y_t|\psi_t}\equiv f(y_t|u_t,d(t-1))\equiv f(y_t|u_t,y_{t-1})\equiv f(y_t|\psi_t)$$

$\psi_t=[u_t,\ y_{t-1}]$	$y_t=1$	$y_t=2$
[1, 1]	$\Theta_{1\|11}$	$\Theta_{2\|11}=1-\Theta_{1\|11}$
[1, 2]	$\Theta_{1\|12}$	$\Theta_{2\|12}=1-\Theta_{1\|12}$
[2, 1]	$\Theta_{1\|21}$	$\Theta_{2\|21}=1-\Theta_{1\|21}$
[2, 2]	$\Theta_{1\|22}$	$\Theta_{2\|22}=1-\Theta_{1\|22}$

The parameter Θ is fully determined by its first column and therefore only its values are required to be set. The first column is denoted `th1` in software context.

It is worth stressing that the simplicity of the system and its model allows us to illustrate the main subtasks of Bayesian decision-making, such as testing of hypotheses, parameter estimation, prediction and control design. For other types of models, the same tasks are solved but they have much higher descriptive and computational complexity.

Contents

1.1 Hypotheses Testing, Recursive Parameter Estimation and Prediction for CCM

This section relates to Section 2.5{33}. Dynamic predictors are discussed and exploited in Sections 7.1.2{197}, 9.1.2{318}.

Bayesian testing of hypotheses is a decision-making task, in which the decision reduces to acceptance of several alternatives from a finite set.

Generally, the decision is based on data observed on the system. First, the decision is assessed by

- the prior information on unknown quantities, corresponding to the particular variant of the model, typically model parameters;
- probabilities on the whole set of possible hypotheses.

Then, the decision rule is constructed that generates decisions as a function of observed data.

The Bayesian methodology solves "naturally" hypotheses testing for composed hypotheses by eliminating unknown, hypothesis-related, parameters via conditional marginalization. All considered hypotheses have to be explicitly formulated. Their adequate formulation decides on the efficiency of the result.

Throughout the text, the testing of hypotheses is often used, for instance, estimation of structure, Section 6.6{164}, model validation, Section 6.7{184}, etc.

This section also demonstrates advantageous properties of the exponential family, Section 3.2{47}, and represents a special case of controlled Markov chains treated in Chapter 10{377}.

Recursive parameter estimation consists of sequential repeating of one-step estimation using the currently observed data vector for updating the posterior *pdf*. Its use, possibly enriched by forgetting, Section 3.1{44}, forms the basis of adaptive systems, Section 4.1{57}.

The difficulty of the generalized Bayesian estimation, Proposition 2.14{37}, stems from the increasing complexity of the posterior pdf. The exponential family, see Section 3.2{47}, represents exceptional class of models for which there is a finite-dimensional sufficient statistics. Consequently, the conjugate prior *pdf* can be chosen whose functional form is preserved during recursive estimation. The controlled Markov chain, Chapter 10{377}, is a member of this family and the CCM model is its simplest example.

The Dirichlet pdf, Chapter 10{377}, is a conjugate pdf with an occurrence table as the finite-dimensional sufficient statistics. It contains positive initial values increased by counts of observed data vectors.

Construction of the predictive *pdf* representing the outer system model is the main motivation for estimation. This model is needed for the design of decision-making strategies and can be interpreted as a one-step-ahead predictor. Often, its quality cannot be judged from short horizon predictions, and compound multi-step-ahead predictions have to be evaluated. These predictions reveal clearly whether the inspected model describes well the dynamic properties of the modelled system.

Computationally, the multi-step-ahead predictions are more difficult than the one-step-ahead ones. Essentially, the joint pdf for all the variables up to the prediction horizon has to be evaluated. The final prediction is obtained by appropriate, computationally demanding, marginalization.

Contents

1.1.1 Testing of Hypotheses About Coin Tossing

Aim:

To inspect Bayesian hypothesis-testing machinery in its dependence on parameters defining the hypotheses and on possible experimental outcomes; see CT model, CT parameter, CCM hypotheses.

Description:

The considered CT model describes a simplified coin-tossing experiment. The tosses are supposed to be mutually independent and uncontrolled, i.e., the output $y_t \in \{1, 2\}$ is modelled on the pdf $f(y_t|y(t-1), \theta) = f(y_t = 1|y(t-1), \theta) = \theta$. Thus, it is specified by a single unknown parameter θ defining the probability of the event $y_t = 1$.

Three hypotheses about the unknown model parameter θ are formulated: $H_1 : \theta = \theta_1$, $H_2 : \theta = \theta_2$ and $H_3 : \theta = \theta_3$.

Bayesian estimation implies that the posterior probabilities of respective hypotheses are fully determined by the number of processed data *ndat* and by the relative frequency ρ of event $y = 1$ observed up to time $\mathring{t} \equiv$ *ndat*. The fact that the sufficient statistic $(\rho, \mathring{t})$ is only two-dimensional allows us to inspect the posterior probabilities of respective hypotheses for possible experimental results.

Specification:

System: uncontrolled, memoryless with two-valued scalar output.

Decision: the generalized Bayesian test of hypotheses $H \in H^*$ evaluating the $f(H|\rho, \mathring{t})$ on $H^* \equiv \{H_1, H_2, H_3\}$; acceptance of hypothesis $\hat{H} \in H^* \equiv \{H_1, H_2, H_3\}$ about the value of parameter θ.

Experience: the relative frequency ρ and the number of processed data records *ndat*.

Ignorance: the hypothesis $H \in H^*$.

Admissible strategies: causal $\rho, \mathring{t} \to \hat{H}$.

Loss function: zero for $\hat{H} = H$ and positive constant for $\hat{H} \neq H$.

Recommended experiments:

The generalized Bayesian test of hypotheses H_1 H_2, H_3 depends on parameter values $th(1) = \theta_1, th(2) = \theta_2, th(3) = \theta_3$ that specify them and on the statistics ρ and *ndat*. It is worth observing the

- probabilities of respective hypotheses if the "true" parameter $th1 = 0.1, 0.4, 0.5$ in conjunction with the influence of such setting,
- parameter values defining each respective hypothesis, $th(1) = 0.8, th(2) = th(3) = 0.1$ or $th(1) = 0.1, th(2) = 0.4, th(3) = 0.5$,
- influence of the increasing number of data records *ndat* $= 50, 200$.

coin/d21Hyp.m Contents

1.1.2 Recursive Parameter Estimation of the CCM Model

Aim:

To recognize the influence of prior knowledge and the number of data records on the time evolution of point parameter estimates of the CCM model.

Description:

The point estimates of unknown parameters depend on the prior pdf and information gained from the observed data. In the case of the CCM model, the sufficient statistic exists and can be written as a $(4, 2)$-occurrence table. Its rows correspond to the regression vector $\psi \equiv$ [current input, delayed output] and the columns to the current output values. Updating the statistics adds 1 to a single entry of the occurrence table. The entry is determined by the current data vector. For instance, for $\Psi_t =$ [current output, current input, delayed output] = [1, 2, 1] the entry (2,1) is updated. The entries of the prior statistics reflect prior expectation on the occurrence of respective combinations. For additional information, see CCM parameter, CCM prior.

Specification:

System: two-valued input and output; dynamic memory of the order 1.

Decision: point estimates of the model parameters.

Experience: past data; model structure.

Ignorance: future data; model parameters.

Admissible strategies: causal - the parameter estimate at time t depends on the observed data up to time $t - 1$.

Loss function: square of the difference between the parameter and its point estimate.

Recommended experiments:

The time course of the parameter points estimates depends both on the choice of the prior statistics and the number of data records used. The prior statistics specify our prior knowledge about the values of the parameters and also our confidence in these values. The data records carry objective information. The proportion of the entries in the prior occurrence table and the number of observations added determine the resulting influence of the subjective and objective information. To get experience in this respect try to change

- the values of the initial statistics $V0 = V0 \times 10$, $V0 \times 100$,
- the number of data records $ndat = 100$, 100 000,
- the character of the simulated data by changing the simulator parameters $th1 = 0.1$, 0.4, 0.5.

coin/d51Est.m Contents

1.1.3 Prediction with the CCM Model: Known Parameters, Feedforward Control

Aim:

To examine dependence of dynamic output prediction based on the discrete CCM model, the prediction length and the uncertainty of the investigated system.

Description:

The point output prediction is computed for the first-order CCM model, with known parameters and control predefined on the whole prediction interval. The prediction is computed for both possible initial conditions $y0 \equiv y_0 \in \{1, 2\}$.

For given values of the input, the transition probability $f(y_t|u_t, y_{t-1})$ reduces to the $(2, 2)$-transition matrix of the first-order Markov chain with input-dependent transition probabilities. The product of these transition matrices provides *nstep*-step prediction. The initial condition $y0$ selects the row in the resulting matrix determining probabilities $f(y_{nstep}|y_0 = y0)$ of $y_{nstep} \in \{1, 2\}$. The expected value evaluated with these probabilities determines the *nstep* point prediction presented.

The influence of the initial condition diminishes with an extended prediction horizon. For additional information, see CCM system, CCM model, CCM parameter, CCM predictive pdf, CCM prediction.

Note that without full knowledge of input up to the prediction moment *nstep* the prediction complexity is much higher.

Specification:

System: two-valued input and output; dynamic memory 1.

Decision: point predictions of the system output for both possible initial conditions.

Experience: all inputs; initial outputs; model structure and parameters.

Ignorance: future outputs.

Admissible strategies: the prediction at time *nstep* can use the sequence of all input values and initial values of the output.

Loss function: square of the difference between the output and its point prediction.

Recommended experiments:

The precision of predictions decreases with an increased prediction horizon and with increased system uncertainty. This fact can be seen in the example, where the predicted output for more than seven steps ahead is the same for both initial outputs $y0 = 1$ and $y0 = 2$. This effect is worth observing for various model parameters and prediction horizons. Try to change

- the uncertainty of the simulated model by changing the probabilities of the event $y = 1$ for possible combinations of regression vectors,
- the prediction horizon *nstep* = 10, 5, 1.

coin/d63Pre.m Contents

1.2 Control Design for the Discrete CCM Model

This section illustrates control design, Section 2.4{25}, which is widely exploited in Chapters 7{193}, 9{309} and 11{411}.

Taking into account that certainty equivalent strategy is used throughout, see Section 4.2.2{62}, the case with known model parameters is treated.

The dynamic design is considered as the firm basis for the receding horizon strategy, Section 4.2.1{58}. The static, one-step-ahead, control is known to be generally dangerous or, at least, very far from the optimum strategy designed for the whole control horizon.

The data-driven design, Agreement 2.8{27} is described by the functional equations given in Proposition 2.9{28}. A closed-form solution of this functional equation exists quite rarely. The considered case, the CCM model with known parameters, represents few exceptions: the involved Bellman function is the table indexed by the model state. This property is widely exploited for the general controlled Markov chains treated in Chapter 11{411}.

A similar "exception," the design of the ARX model with known parameters and quadratic loss function, is illustrated in Section 2.2 and is widely exploited in Chapter 9{309}.

Contents

1.2.1 Multistep Control for the CCM Model with Known Parameters

Aim:

To examine dependence of the quality of multi-step-ahead optimal control based on the CCM model, on the uncertainty of the controlled system, and on the control aim specified through the loss function.

Description:

The optimal control law, describing control strategy, minimizes expected additive loss with partial loss z given in a tabular form. The control horizon here covers the whole simulation horizon, i.e., with the number of data records *ndat*. The individual entries of the partial loss express our aims. They correspond to possible configurations of data vector $\Psi \equiv$[current output, current input, delayed output]. The loss table is a (4,2)-matrix with current output indexing columns and regression vectors ψ consisting of [current input, delayed output] pointing to rows. For instance, large z-entries (1,2), (2,1), (3,2) and (4,1) penalize output differences $y_t - y_{t-1}$.

For additional information, see CCM model, CCM parameter, CCM prediction, CCM loss, CCM control.

Properties of the simulated system are determined by its parameters and initial condition *y0*; see CCM system, CCM parameter.

Specification:

System: two-valued input and output; dynamic memory 1.

Decision: sequence of inputs until the control horizon.

Experience: past data (including the initial output value); model structure; model parameters.

Ignorance: future data.

Admissible strategies: causal - the input u_t computed at time t can depend only on data up to time $t-1$.

Loss function: tabular, penalizing individual data vectors $\Psi_t' = [y_t, u_t, y_{t-1}]'$.

Recommended experiments:

The control quality depends heavily both on the degree of influence of the input on the future outputs and the influence of the delayed output. The strength of this influence is specified by parameters of the simulated system and depends also on the initial condition *y0*. The aim-expressing loss function determines the control law, thus influences the achieved control quality. The example provides a possibility to inspect results obtained for various combinations of the system and loss function. The quality of the control can be judged from the figures presented. Try to

- change the simulated system by choosing different values of its parameters,
- change the initial output *y0* = 2; the influence of this change is more pronounced for a shorter horizon *ndat*,
- select loss function CCM loss z$\geq$ 0 (entrywise) expressing your control aim.

coin/d81Con.m Contents

Chapter 2

Basic Tasks for the Normal Auto-Regression Model with External Input (ARX)

This chapter contains examples related mostly to Chapter 2{11}. They illustrate the adopted Bayesian dynamic decision-making theory on rather simple examples of the normal, single-input single-output, auto-regression system with external input (ARX model). Its simplicity allows us to illustrate the main subtasks such as parameter estimation with forgetting, Section 3.1{44}, and prediction and control design, Section 2.9{28}.

This model represents the member of the dynamic exponential family, Section 3.2{47}, whose use simplifies substantially recursive estimation. Adopting certainty equivalent strategy, Section 4.2.2{62}, and using quadratic loss, dynamic control design, Proposition 2.9{28} can be solved.

The ARX model inspected here is a basic factor used extensively in Chapters 8{243}, 9{309} and 13{463}.

Contents

2.1 Recursive Parameter Estimation and Prediction for the Normal ARX Model

This subsection relates to Section 2.5{33}. It demonstrates advantageous properties of the exponential family, Section 3.2{47}, and represents a special case of the normal ARX factors treated in Chapter 8{243}. Dynamic predictors are discussed and exploited in Sections 7.1.2{197}, 9.1.2{318}.

Recursive parameter estimation consists of the sequential repetition of one-step estimation using currently observed data for correcting the posterior pdf. Its use, possibly enriched by forgetting, Section 3.1{44}, forms the basis of the adaptive systems, Section 4.1{57}.

The difficulty of this task, which implements the generalized Bayesian estimation, Proposition 2.14{37}, stems from the increasing complexity of the posterior pdf. The exponential family, see Section 3.2{47}, represents an exceptional but important class of models for which there is a finite-dimensional sufficient statistics. Consequently, the conjugate prior pdf can be chosen whose functional form is preserved during recursive estimation. The normal ARX model, Chapter 8{243}, is a member of this family and the considered ARX model is its simplest example.

The Gauss-inverse-Wishart pdf, Chapter 8{243}, is the pdf conjugated to the normal ARX model. The extended information matrix and the number of degrees of freedom form the finite-dimensional sufficient statistics of this pdf.

Construction of the predictive pdf representing the outer system model is the main motivation for estimation. This model is needed for the design of decision-making strategies and can be interpreted as a one-step-ahead predictor. Often, its quality cannot be judged from short horizon predictions, and compound multi-step-ahead predictions have to be evaluated. These predictions reveal clearly whether the inspected model describes well the dynamic properties of the modelled system.

Computationally, the multi-step-ahead predictions are more difficult than the one-step-ahead ones. Essentially, the joint pdf for all the variables up to the prediction horizon has to be evaluated. The final prediction is obtained by appropriate, computationally demanding, marginalization.

Contents

2.1.1 Recursive Parameter Estimation with Forgetting

Aim:

To select an appropriate forgetting factor improving the parameter estimation when the data mean jumps.

Description:

The model parameters are estimated recursively, i.e., the generalized Bayesian parameter estimates are updated by the newest data vector. The exponential forgetting is used during the estimation that suppresses the influence of the older data records. The forgetting rate is determined by the forgetting factor frg, which can be interpreted as the probability that the estimated parameters are time invariant. Thus, the forgetting factor is in the interval (0,1] and the value 1 means that no forgetting is used.

In this example, the properties of forgetting are demonstrated in the situation when a strong impulse signal is added to the system output at simulation time around 100.

Specification:

System: single-output, autoregressive normal linear system stimulated by external single input, described by the ARX model.

Decision: generalized Bayesian estimate, point estimate of model parameters.

Experience: past data compressed into a sufficient statistics formed by the extended information matrix and the number of degrees of freedom and determining also past point parameter estimates; current data vector; model structure.

Ignorance: parameters and future data.

Loss function: quadratic Euclidean norm of the difference between the constructed point estimate and unknown parameters.

Recommended experiments:

The estimation properties depend on the chosen forgetting factor. They are influenced by the simulated system, by the signal-to-noise ratio determined by the range of the applied input and by the number of processed data. It is worth to observe influence of

- the forgetting factor, $frg = 1$ (no forgetting); 0.99 (slow forgetting); 0.98 (fast forgetting); 0.9 (very fast forgetting, a danger of numerical breakdown),
- the parameters of the simulated system; be careful to select stable auto-regression,
- the range of the stimulating input, $rg = 1, 0.001, 1000$.

normal/nEst3.m Contents

2.1.2 Prediction with the ARX Model: Known Parameters, Feedforward Control

Aim:

To examine the dependence of dynamic output prediction, based on the ARX model, on the prediction horizon *nstep* and on the uncertainty of the investigated system.

Description:

A single-output, single-input normal system with known parameters is considered. Its output is predicted *nstep*-steps ahead. The prediction conditions can be changed by changing the range *rg* of the input, which is uniformly distributed, zero mean white noise. Alternatively, the covariance of the system noise *cove* can be varied.

Specification:

System: normal system described by ARX model.

Decision: point *nstep*-step prediction of the output.

Experience: data observed up to the moment when the prediction is made and the input values up to the prediction horizon; system structure and parameters.

Ignorance: future output values.

Loss function: quadratic error between the output and its prediction.

Recommended experiments:

The prediction quality depends on the noise realization, and decreases with an increasing prediction horizon and with a decreasing signal-to-noise ratio. It is worth observing the influence of

- the prediction horizon, *nstep* = 1, 5, 20,
- the variance of the system noise, *cove* = 0.001, 0.1, 1,
- the range of the system input, *rg* = 0.1, 1, 5.

normal/nPred2.m Contents

2.1.3 Generalized Prediction with the ARX Model: Known Parameters, Feedforward Control

Aim:

To examine the dependence of dynamic generalized output prediction, based on the ARX model, on the prediction horizon *nstep* and the uncertainty of the investigated system.

Description:

The single-output, single-input normal system with known parameters is considered. Its output is predicted *nstep*-steps ahead. The prediction conditions can be changed by changing the range *rg* of the input, which is uniformly distributed, zero mean white noise. Alternatively, the covariance of the system noise *cove* can be varied.

Unlike in Example 2.1.2, the whole predictive pdf is computed and displayed. You can slow down the corresponding sequence of displayed images by selecting a bigger pause *pau*. Then it makes sense to shorten the experiment length by decreasing *ndat*.

Specification:

System: normal system described by the ARX model.

Decision: generalized *nstep*-step prediction of the output.

Experience: data observed up to the moment when the prediction is made and input values up to the prediction horizon; system structure and parameters.

Ignorance: future output.

Recommended experiments:

The prediction quality decreases with an increasing prediction horizon and with a decreasing signal-to-noise ratio. Both position and width of the predictive pdf are influenced. It is worth observing the influence of

- the prediction horizon, *nstep* = 1, 5, 20,
- the variance of the system noise, *cove* = 0.001, 0.1, 1,
- the range of the system input, *rg* = 0.1, 1, 5.

The images' appearance can be controlled by

- pauses between figures and simulation length (*pau*,*ndat*) = (0.5, 50), (2, 20).

normal/nPred3.m Contents

2.2 Control Design for the Normal ARX Model

The ideal solution of the decision-making under uncertainty is described by the combination of Bayesian estimation, Subsection 2.5{33} and dynamic programming, Proposition 2.8{26}. The functional equations describing them are mostly computationally infeasible and have to be solved approximately.

At the design stage, the complexity stems from

- the richness of the space of the ignorance part of the behavior that has to be inspected for the optimal selection of decisions,
- the complexity of the models describing the relationships of the experience and optional decisions to the ignorance part of the behavior.

The suboptimal design, Section 4.2{58}, tries to reduce the influence of one or both of these sources of complexity. The reduction of the design horizon is the most direct way to a simplified (suboptimal) design.

The receding horizon strategy, Section 4.2.1{58}, is used in the following examples. It offers a compromise between the ideal planning over the whole horizon of interest and short-sighted, locally optimizing strategies. Planning a few steps ahead provides just an approximation of the optimal design. Thus, it is reasonable to apply just the initial planned decisions and to redesign strategy whenever a new information about the system and its state is obtained.

The above-mentioned strategy is supplemented by another strategy, which simplifies the models by employing certainty equivalence strategy, Section 4.2.2{62}. It gets the approximate predictive pdf by replacing an unknown parameter in the parameterized model by its current point estimate.

Both these simplifications are widely used in the book.

Contents

2.2.1 Control with Known Parameters

Aim:

To gain experience with multi-step-ahead control based on the ARX model.

Description:

The closed control loop consists of the simulated ARX system and the suboptimal controller. It uses the receding horizon strategy, determined by *nhor*, quadratic loss function and known parameters of the system.

Specification:

System: single-input, single-output first-order system described by the normal ARX model with time invariant parameters.

Decision: sequence of input values.

Experience: past data; system structure; system parameters.

Ignorance: future input–output data up to the control horizon.

Admissible strategy: causal with unrestricted range of input values.

Loss function: additive with *nhor* terms; partial loss equals the squared difference of the output from set point plus squared input, weighted by *om*.

Recommended experiments:

Closed-loop behavior depends on the selected input weight, on the system parameters, noise level and information content of data that can be influenced by the set point changes. It is worth observing the influence of

- the weight, say *om* = 1 (output deviations and value of the input equally important), *om* = 0 (input values unimportant), *om* = 50 (low level input values are strongly preferred),
- the system parameters, a1 = -0.8 (damped oscillations), a1 = 0.99 (stable system with slow response), a1 = 1.2 (unstable system), b0 = 0.2, b0 = 1e-4 (almost zero static gain),
- the heights of set-point changes, st1 = 10 and st2 = 0; st1 = 0 and st2 = 10,
- the noise variance, *cove* = 1e-3, 1.

normal/nCont1.m Contents

2.2.2 Adaptive Control Based on Certainty Equivalence Strategy

Aim:

To gain an experience with multi-step-ahead control based on the ARX model whose unknown parameters are replaced by the recursively updated point estimates.

Description:

The closed control loop consists of the simulated ARX system and the suboptimal controller. It uses the receding horizon strategy, determined by *nhor* and quadratic loss function, and replaces unknown parameters of the system by their recursively updated estimates.

Specification:

System: single-input, single-output first-order system described by the normal ARX model with time invariant parameters.

Decision: sequences of input values and parameter estimates.

Experience: past data; system structure.

Ignorance: future data; parameters of the ARX model.

Admissible strategy: causal with unrestricted range of inputs.

Loss function: additive with *nhor* terms; partial loss equals the squared difference of the output from set point plus squared input, weighted by *om*.

Recommended experiments:

Closed-loop behavior depends on selected input weight, on system parameters, noise level and information content of data that can be influenced by the set point changes. It is worth observing the influence of

- the weight, say *om* = 1 (output deviations and value of the input equally important), *om* = 0 (input values unimportant), *om* = 50 (low-level input values are strongly preferred),
- the receding horizon *nhor* = 1, 20, 100,
- the heights of set-point changes, st1 = 10 and st2 = 0; st1 = 0 and st2 = 10,
- the system parameters, a1 = -0.8 (damped oscillations), a1 = 0.99 (stable system with slow response), a1 = 1.2 (unstable system), b0 = 0.2, b0 = 1e-4 (almost zero static gain),
- the noise variance, *cove* = 1e-3, 1.

normal/nCont2.m Contents

Chapter 3

Learning with Normal Mixtures

Learning means tuning a prior mixture model using observations of system data. This topic is discussed in Chapters 6{95} and 8{243}. Both online and offline modes are available.

To illustrate the developed learning algorithms in the most transparent way, the system data for examples are simulated by the following artificial systems.

estim/zdata1.m

System description: continuous-valued data with two channels; static mixture of *ncom* normal components with the means placed on the unit circle. The component weights increase linearly anticlockwise from $< 0, 1 >$. The common covariance of components is specified by the diagonal value *diac*.

estim/zdata2.m

System description: continuous-valued data, *nchn* channels; static mixture of *ncom* normal components with means placed randomly. The common covariance of components is specified by the diagonal value *diac*.

estim/zdatadyn.m

System description: continuous-valued data, two channels, several normal components. Each component is composed of normal dynamic and static factors. Dynamic factors can be of the order $2, 3$ or 4 with the output noise covariance *covy*. The static factor models the input as normal noise with zero mean and covariance *covu*. For this example, a dynamic factor is always the first one in the component.

To simulate a multicomponent mixture, an optional algorithm of Markov jumps among components can be applied. The means of components are placed on the unit circle (see above). System data of length *ndat* are generated. The realization of a random number generator is specified by *seed*.

The mixture initialization is performed in *niteri* iterations. The initialized mixture is then estimated iteratively in *nitere* iterations. The prediction is computed from the pdf describing the first channel conditioned by the value of the zero-delayed regressor *psi*0. Resulting mixtures are displayed.

The **recommended experiments** are intended to demonstrate the influence on the learning results of particular system parameters and algorithm options. It is worth observing the influence of

- the system noise (*diac* = 0.5, 0.1, 0.01); the higher are values of *diac*, the higher is the probability that initialization will fail,
- the process complexity (*ncom* = 2, 4, 6, 10); more complex system needs more data for initialization and estimation,
- the number of iterations (*niteri* = 2, 6, 10)
- the number of iterations in estimation after initialization (*nitere* = 20, 50, 200),
- the values of a zero-delayed regressor on predictions (*psi*0 = -0.1, 0,10).

Contents

3.1 Construction of the Prior Estimate: Initialization

The initialization searches for a mixture model structure that maximizes the v-log-likelihood evaluated for the respective structures and data observed on the system considered.

This group of examples illustrates Sections 6.4{120}, 8.4{270}, 6.4.2{122}, 6.4.3{123}, 6.4.4{134}, 6.4.5{136}, 6.4.6{136}, 6.4.7{138}, 6.4.8{140}, 6.4.9{148}, 8.4.3{272}, 8.4.4{273}, 8.4.5{275}, 8.4.6{277}, 8.4.7{279}, 6.6{164}, 6.6.1{165}, 6.6.3{166}, 8.6{295},8.6.1{295}, 8.6.4{299}.

Initialization consists of the following, iteratively applied, operations:

1. Selection of the initial mixture that grasps the densely populated part of data space.
2. Approximate estimation of the mixture with fixed structure.
3. Selection of factors for splitting.
4. Splitting of factors.
5. Estimation of the factor structure.
6. Approximate estimation of the mixture with fixed structure.
7. Merging of similar components.
8. Cancelling of spurious components.
9. Repetition of the iterations starting from Step 2.

The majority of these steps are solved in a heuristic way, and thus the overall performance may vary with respect to the problems addressed. The examples should give an impression about the overall behavior of this conceptual algorithm and a feeling about its tuning knobs.

Contents

3.1.1 Computational Efficiency of Static Mixture Initialization

Aim:

To tune the mixture initialization algorithm to be adequate to the considered system. The techniques used are described in Section 6.4{120} and 8.4{270}.

Description:

Observed data are generated by a static simulated mixture. The data are used for initialization of the mixture model. The data can be optionally scaled (*scale* = 0|1). The performance of the algorithm is influenced by the number of components *ncomi* of an initial model, by the number of iterations *niter* and by the mixture estimation algorithm used inside the initialization.

The iterative estimation of *niteri* repetitions can be used in initialization. The estimation method is specified by *opt*. This is a character switch, as follows:

'q'	iterative quasi-Bayes mixture estimation (QB)
'b'	iterative batch quasi-Bayes mixture estimation (BQB)
'f'	iterative mixture estimation based on forgetting branching (BFRG)
'n'	without iterative estimation (default) (single pass of QB)

Simulator and estimated mixture are plotted and prediction-errors norm is displayed.

Specification:

System: realization of *ndat* data with the optional number of channels *nchn* generated by the normal static mixture with the *ncom* component determined by *seed*. The common covariance of components is *diac*.

Decision: model structure and values of its parameters.

Experience: *ndat* data records; knowledge that the system is a static one.

Ignorance: model structure and parameters; initial setting of the initialization algorithm.

Loss function: negative *v*-log-likelihood is internally used for decisions about initial model form.

Recommended experiments:

Performance and computational demands of the inspected heuristic initialization algorithms vary with the extent of data, noise level, system complexity, behavior realization and optional knobs of algorithms. It is worth inspecting the influence of

- extent of learning data, say *ndat* = 1000, 2000, 500,
- system-noise variance, *diac* = 0.1, 0.5,
- estimated system complexity, *ncom* = 3, 10, *nchn*= 2, 5,
- behavior realization, *seed* = any integer, zero if not fixed,
- number of iterations, *niter* = 10, 30, 100,
- number of initial components, *ncomi* = 1, 5,
- estimation method used, *opt* = $q|b|f|n$,
- number of iterations of the estimation algorithm, *niteri* = 10, 20,
- data scaling *scale* = 0 or 1.

Warning: Processing time can be quite long.

estim/benchinit.m | See results | Contents

3.1.2 Scanning of Static Mixture Initialization

Aim:

To tune the mixture initialization algorithm to be adequate for the system selected and to provide an insight into initialization – individual iterations are shown. The techniques used are described in Section 6.4{120} and 8.4{270}.

Description:

Measured data are generated by a static simulated mixture and are used for initialization of the mixture model.

The performance of the algorithm is influenced by the number of components *ncomi* of an initial model, by number of iterations *niter* and the mixture estimation algorithm used inside the initialization.

The iterative estimation of *niteri* repetitions can be used in initialization. The estimation method is specified by *opt*. This is a character switch, as follows:

'q'	iterative quasi-Bayes mixture estimation (QB)
'b'	iterative batch quasi-Bayes mixture estimation (BQB)
'f'	iterative mixture estimation based on forgetting branching (BFRG)
'n'	without iterative estimation (default) (single pass of QB)

The simulator and result of each iteration are plotted.

Specification:

System: realization of *ndat* data with the optional number of channels *nchn* generated by a normal static mixture with *ncom* components determined by *seed*. The common covariance of components have diagonals filled by *diac*.

Decision: model structure and values of its parameters.

Experience: *ndat* data records; knowledge that the system is a static one.

Ignorance: model structure and parameters; initial setting of the initialization algorithm.

Loss function: negative *v*-log-likelihood is internally used for decisions about initial model form.

Recommended experiments:

Performance and computational demands of the inspected heuristic initialization algorithms vary with the extent of data, noise level, system complexity, behavior realization and optional knobs of algorithms. It is worth observing the influence of

- extent of learning data, say *ndat* = 2000, 500,
- system-noise variance, *diac* = 0.1, 0.5,
- estimated system complexity, *ncom* = 3, 10, *nchn*= 2, 5,
- behavior realization, *seed* = any integer, zero if not fixed,
- number of iterations, *niter* = 1, 3, 5,
- number of initial components, *ncomi* = 1, 5,
- estimation method used, *opt* = $q|b|f|n$
- number of iterations of the estimation algorithm, *niteri* = 10, 20 .

Warning: Processing time can be quite long.

estim/inititer.m | See results | Contents

3.1.3 Static Mixture Initialization: "Banana Shape" Benchmark

Aim:

To expose a "hard" case, known as the "banana" shape, study static mixture initialization. The techniques used are described in Section 6.4{120} and 8.4{270}.

Description:

Measured data are generated by a static simulated mixture. Data are used for initialization of the mixture model. The performance of the algorithm is influenced by the number of components *ncomi* of the initial model and by the number of iterations *niter*.

After the initialization, the resulting mixture generates new data that can be visually compared with the original one. This is the simplest method of model verification.

The simulator, data clusters, estimated mixture and data clusters generated by the estimated mixture are plotted.

Specification:

System: realization of *ndat* data generated by a normal static mixture with *ncomi* components. The realization is determined by *seed*. The common covariance of components have diagonals filled by *diac*. The components are in a "banana" formation.

Decision: estimate of model structure and generalized Bayesian parameter estimates.

Experience: *ndat* data records; knowledge that the system is a static one.

Ignorance: mixture structure and parameters, initial setting of the initialization algorithm.

Loss function: negative *v*-log-likelihood is internally used for decisions about initial model structure.

Recommended experiments:

Performance and computational demands of the inspected heuristic initialization algorithms vary with the extent of data, noise level, system complexity, behavior realization and optional knobs of algorithms. It is worth inspecting the influence of

- extent of learning data, say *ndat* = 1000, 500,
- system-noise variance, *diac* = 0.1, 0.5,
- behavior realization, *seed* = any integer, zero if not fixed,
- number of system components, *ncom* = 10, 20,
- number of initial model components, *ncomi* = 30, 10, 5,
- number of iterations of the initialization algorithm, *niter* = 3, 5

Warning: Processing time can be quite long for higher values *ncom* and *ndat*.

estim/banan.m | See results | Contents

3.1.4 Dynamic Mixture Initialization

Aim:

To tune the mixture initialization algorithm to be adequate for the considered dynamic system. The techniques used are described in Section 6.4{120} and 8.4{270}.

Description:

Measured data are generated by a simulated dynamic mixture. The data are used for initialization of the mixture model. The performance of the algorithm is influenced by the order of the initial model *ord*, by the number of iterations *niter* and by the mixture estimation algorithm used inside the initialization. The initial model has only one component.

The iterative estimation of *niteri* repetitions can be used in initialization. The estimation method is specified by *opt*. This is a character switch, as follows:

'q'	iterative quasi-Bayes mixture estimation (QB)
'b'	iterative batch quasi-Bayes mixture estimation (BQB)
'f'	iterative mixture estimation based on forgetting branching (BFRG)
'n'	without iterative estimation (default) (single pass of QB)

The system output, the estimated mixture, histogram of prediction errors and plot of prediction versus data are plotted. The prediction-errors norm and number of components found are shown.

Specification:

System: realization of *ndat* data records generated by the normal dynamic mixture of two single-input single-output components. The realization is fixed by *seed*.

The components are of fourth order, the input is normal noise with the same input covariance *cove*. Both component weights are 0.5.

Decision: model structure and values of its parameters.

Experience: past data up to *ndat*; the richest model structure.

Ignorance: model structure and parameters; initial setting of the initialization algorithm.

Loss function: negative *v*-log-likelihood is internally used for decisions about model structure.

Recommended experiments:

Performance and computational demands of the inspected heuristic initialization algorithms vary with the extent of data, noise level, behavior realization and optional knobs of algorithms.

It is worth observing the influence of

- extent of learning data, say *ndat* = 300, 500, 1000,
- system-noise variance, *cove* = 0.1, 0.5,
- behavior realization, *seed* = any integer, zero if not fixed,
- number of iterations, *niter* = 1, 3,
- estimation method used, *opt* = $q|b|f|n$,
- number of iterations of the estimation algorithm, *niteri* = 10, 20,
- order of richest (initial) mixture, *ord* = 2, 3, 5.

estim/initdyn.m | See results | Contents

3.1.5 Model Checking by Cross-Validation of Learning Results

Aim:

To choose the key optional knobs of the algorithm performing Bayesian cross-validation, taking into account the degree of under-modelling and signal-to-noise ratio.

Description:

Initialization provides complete structure of mixture and the prior pdf, which is corrected by generalized Bayesian parameter estimation. The result should be checked before using it for subsequent tasks. In other words, we have to decide whether the available data, obtained here by simulation, allow us to accept the model as valid or not.

The test that uses variable cutting of the available data into the initial learning part — leaving the rest of for validation purposes — is applied here.

The behavior of the resulting Algorithm 8.18{305} can be actively influenced by selecting the number *n* of nodes in a uniform grid of inspected cutting moments. The influence is inspected for single-input single output systems of orders *ordi*=4 and *ordi*=2, respectively. The estimated model has order 2. Thus, it is correct in the latter case only.

Of course, the results depend also on properties of learning data, such as the number of data records *ndat* or the signal-to-noise ratio controlled by variance of the output, *covy* and *covu*.

The probability pH0 of the hypothesis that the model is acceptable is drawn as a function of inspected cutting moments. The hypothesis should be accepted if the maximum of this sequence is higher than a treshold value. Otherwise the model should be taken as invalid.

Specification:

System: realization of *ndat* two-dimensional data records, generated by a normal mixture, is determined by *seed*; the first factor with variance *covy* represents a single-input single-output ARX system of order *ordi*; the 2nd static factor models input as white noise with zero mean and variance *covu*.

Decision: accept or reject the hypothesis H0 that the model is valid.

Experience: all (!) *ndat* available data records.

Ignorance: model parameters, proper cutting moment, validity of H0.

Loss function: table with positive and equal loss assigned to false acceptance or false rejection of the hypothesis H0.

Recommended experiments:

Observe the impact of the chosen grid while modifying properties of learning data. It is worth inspecting the influence of:

- the number of nodes in the grid of cutting moments, say n = 10, 50, 100,
- the validity (invalidity) of the hypothesis H0, *ordi*= 2 (4),
- the testing data extent, say *ndat* = 500, 300, 200,
- the influence of signal-to-noise ratio, *covy* = 0.01, 0.05, 1, *covu* = 0.1, 0.5, 1.

estim/zverseg.m | See results | Contents

3.2 Approximate Parameter Estimation

The approximate parameter estimation fine-tuns of the regression coefficients of the factors and other parameters. The following methods are available:

- iterative quasi-Bayes mixture estimation, (QB),
- iterative batch quasi-Bayes mixture estimation, (BQB),
- iterative mixture estimation based on forgetting branching, (BFRG).

The examples of this section relate to the book Sections 6.5{154}, 8.5{289}, 6.5.1{155}, 8.5.1{289}, 8.5.1{289}, 6.5.3{162}, 8.5.3{293}, 8.5.1{289}, 6.4.7{138}, 8.4.6{277}.

Contents

3.2.1 Comparison of Mixture Estimation Algorithms: Static Case

Aim:

To choose a learning algorithm and its tuning knobs adequate to the considered system. Compared algorithms are quasi-Bayes, see Section 6.5.1{155} (QB), batch quasi-Bayes, see section 6.5.3{162} (BQB) and forgetting branching, see Algorithm 6.7{138} (BFRG), each of which approximately estimates the mixture of a fixed known structure.

Description:

Observed data are generated by a static simulated mixture. They are used for estimation of mixture models via the learning algorithms compared. Their performance is influenced by the number of iterations *niter* and the forgetting rate frg. The forgetting branching uses additionally an alternative small forgetting rate $frga$.

Specification:

System: realization of *ndat* two-dimensional data generated by normal static mixture with *ncom* components determined by *seed*. The components have common diagonal covariance with the diagonal filled by *diac*. Their centers are uniformly positioned on the unit circle. Component weights are linearly decreasing.

Decision: one-step-ahead output predictions, generalized Bayesian estimation of mixture parameters.

Experience: *ndat* two-dimensional data records (collected on two channels); correct system structure.

Ignorance: model parameters; future system outputs; quality of compared estimation algorithms.

Loss function: negative v-log-likelihood normalized by number of processed data; prediction-errors norm; contour plots of the posterior pdf on estimated parameters.

Recommended experiments:

Performance of the inspected heuristic algorithms varies with the extent of data, noise level, system complexity, behavior realization and optional knobs of algorithms. It is worth observing the influence of

- the extent of learning data, say *ndat* = 500, 300, 200,
- the system-noise variance, *diac* = 0.5, 1,
- the estimated system complexity, *ncom* = 10, 20,
- the behavior realization, *seed* = any positive integer
- the number of iterations, *niter* = 10, 30, 100 (computational time increases linearly),
- the involved forgetting factors on quality of branching-by-forgetting, $frg = 0.99, 0.999$ $frga = 0.5$, 0.9 .

estim/estcmpsta.m See results Contents

3.2.2 Comparison of Mixture Estimation Algorithms: Dynamic Case

Aim:

To choose a learning algorithm and its tuning knobs adequate to the considered system. Compared algorithms are quasi-Bayes, see Section 6.5.1{155} (QB), batch quasi-Bayes, see Section 6.5.3{162} (BQB) and forgetting branching, see Algorithm 6.7{138} (BFRG), each of which approximately estimates the mixture of a fixed known structure.

Description:

Observed data are generated by a dynamic simulated mixture. The simulator is shown in six time values to demonstrate the dynamic character of the model. Data are used for estimation of the mixture model via the learning algorithms compared. Their performance is influenced by the number of iterations *niter* and forgetting rate frg. The forgetting branching uses, in addition, an alternative small forgetting rate $frga$.

Specification:

System: realization of *ndat* two-dimensional data generated by normal, two-component dynamic mixture determined by *seed*. The components contain a fourth order dynamic factor and a static factor modelling noise on input. The input variance *covu* and component weight *alpha* can be specified.

Decision: one-step-ahead output predictions, generalized Bayesian estimation of mixture parameters.

Experience: *ndat* two-dimensional data records collected on two channels; correct system structure.

Ignorance: model parameters; future system outputs; quality of compared estimation algorithms.

Loss function: negative v-log-likelihood normalized by number of processed data; prediction-errors norm; contour plots of the posterior pdf on estimated parameters.

Recommended experiments:

Performance of the inspected heuristic algorithms vary with the extent of data, separation of clusters, signal-to-noise ratio, behavior realization and optional knobs of algorithms. It is worth observing the influence of

- the extent of learning data, say *ndat* = 500, 300, 200,
- the input variance, *covu* = 0.5, 1,
- the component weights, *alpha* = 0.1, 0.5, 0.9,
- the behavior realization, *seed* = any positive integer,
- the number of iterations, *niter* = 10, 30, 100 (computational time increases linearly),
- the involved forgetting factors on quality of branching-by-forgetting, frg = 0.99, 0.999 $frga$ = 0.5, 0.9 .

estim/estcmpdyn.m See results Contents

3.2.3 Computational Efficiency of Mixture Estimation Algorithms

Aim:

To choose a learning algorithm according to its computational efficiency. Compared algorithms are quasi-Bayes, see Section 6.5.1{155} (QB), batch quasi-Bayes, see Section 6.5.3{162} (BQB), and forgetting branching, see Algorithm 6.7{138} (BFRG), each of which approximately estimates the mixture of a fixed known structure.

Description:

Observed data are generated by a static normal simulated mixture. The data are used for estimation of mixture models via the learning algorithms considered. Their performance is influenced by the number of iterations *niter* and forgetting rate *frg*. The forgetting branching uses, in addition, an alternative small forgetting rate *frga*.

Specification:

System: realization of *ndat* data with the optional number of channels *nchn* generated by normal static mixture with *ncom* components determined by *seed*. The components have common diagonal covariance with the diagonal filled by *diac*.

Decision: one-step-ahead output predictions, generalized Bayesian estimation of mixture parameters

Experience: *ndat nchn*-dimensional data records; correct system structure.

Ignorance: model parameters; future system outputs; quality of compared estimation algorithms.

Loss function: negative *v*-log-likelihood normalized by number of processed data; prediction-errors norm.

Recommended experiments:

Performance and computational demands of the inspected heuristic algorithms vary with the extent of data, noise level, system complexity, behavior realization and optional knobs of algorithms. It is worth observing the influence of

- the extent of learning data, say *ndat* = 2000, 500,
- the system-noise variance, *diac* = 0.5, 1,
- the estimated system complexity, *ncom* = 3, 10, *nchn*=1, 2, 5,
- the behavior realization, *seed* = any positive integer,
- the number of iterations, *niter* = 10, 30, 100 (computational time increases linearly),
- involved forgetting factors on quality of branching-by-forgetting, *frg* = 0.99, 0.999, *frga* = 0.5, 0.9.

Warning: Processing time can be quite long.

estim/benchest.m | See results | Contents

3.2.4 Mixture Estimation Based on the Batch Quasi-Bayes Algorithm (BQB)

Aim:

To compare batch quasi-Bayes estimation, see Section 6.5.3{162}, with quasi-Bayes estimation, see Section 6.5.1{155}, when pointers to the system components form a Markov chain.

Description:

Observed data are generated by a static simulated mixture. Its component weights form a Markov chain. Markov jumps among components are controlled by a transition probability matrix whose diagonal *diam* can be chosen. Zero value induces independent transitions. Performance of compared algorithms is influenced by the number of iterations *niter* and forgetting rate frg.

The results are displayed as follows:

Mixture simulator	Data clusters	Active components in time
Mixture estimated by batch quasi-Bayes	Estimated by quasi-Bayes	v-log-likelihood left is batch quasi-Bayes

Specification:

System: realization of *ndat* two-dimensional data generated by normal static mixture with two components determined by *seed*. The common covariance of the components is specified by the diagonal element *diac*. The character of component weights is controlled by *diam*.

Decision: generalized Bayesian estimation.

Experience: *ndat* two-dimensional data records; correct system structure.

Ignorance: model parameters; future system outputs; quality of compared estimation algorithms.

Loss function: negative v-log-likelihood normalized by number of processed data.

Recommended experiments:

Performance of the inspected heuristic algorithms vary with the extent to which the assumption of independent jumps between components is violated. Performance also varies with the amount of data, noise level, behavior realization and optional tuning knobs of algorithms. It is worth observing the influence of

- the degree of dependence, *diam*= 0.99, 0.1, 0.3, 0.5, 0.9 in conjunction with changes of the following:
- the learning data extent, *ndat* = 500, 300, 200,
- the system-noise variance, *diac* = 0.5, 1,
- the behavior realization, *seed* = any positive integer,
- the number of iterations, *niter* = 10, 30, 100 (computational time increases linearly),
- the involved forgetting factors on quality of branching-by-forgetting, frg = 0.99, 0.999 .

estim/bquasi.m See results Contents

3.2.5 Mixture Estimation Based on the Branching by Forgetting Algorithm (BFRG)

Aim:

To choose the key optional knob of the approximate learning algorithm based on Branching by forgetting (BFRG) and compare it with the quasi-Bayes (QB) estimation. The branch and bound technique, see Algorithm 6.1{100}, is inspected. The bounding mechanism is based on forgetting, see Algorithm 6.7{138}.

Description:

Simulated data are used for estimation of mixture models by the inspected BFRG algorithm.
The estimation combines branching and bounding steps:

- Branching step: Two copies of the mixture estimates are updated with no forgetting and with a small value of forgetting rate $frga$, respectively. The updating runs until the difference between their respective v-log-likelihood exceeds a threshold value ro. Then, the bounding step is applied.
- Bounding step: The mixture with higher v-log-likelihood is used as the initial one for the subsequent branching step.

Specification:

System: realization of $ndat$ two-dimensional data generated by the normal static mixture with $ncom$ components determined by $seed$.

Decision: generalized Bayesian estimate prediction.

Experience: $ndat$ two-dimensional data records; correct system structure.

Ignorance: model parameters; future system outputs.

Loss function: negative v-log-likelihood normalized by number of processed data; processing time.

Recommended experiments:

Performance of the inspected heuristic algorithm varies with its tuning knobs in conjunction with the extent of data. It is worth to observe the influence of

- the forgetting rate, say $frga = 0.9, 0.5, 0.1$,
- t he threshold, $ro = 1, 2, 5$,
- the extent of learning data, say $ndat = 500, 300, 200$,
- the behavior realization, $seed$ = any integer, zero if not fixed.

estim/brafor.m | See results | Contents

3.2.6 Parameter and Structure Estimation of Normal Factors

Aim:

To demonstrate dependence of factor structure estimation on model settings and to inspect time evolution of point estimates of regression coefficients; see Sections 6.6{164}, 8.6{295}, 6.6.1{165}, 8.6.1{295}.

Description:

Observed data are generated by a dynamic simulated mixture.

A sufficiently large richest model order *ord* is selected. It specifies the space of possible regressor structures. In this space, the "best" regressor is searched for, being the one that has the highest posterior probability in the space of competing factor structures.

The results of structure estimation and estimate of regression coefficients are influenced by the initial model. It is built by specifying:

ord	model order
cove	estimate of the output covariance
diaCth	diagonal of covariance of the regression coefficients
dfm	number of degrees of freedom

The display of results is via the internal form of structures — as a list of channels and delays (channel 0 stands for offset). For instance, the structure

```
str = [1 1  2 2
       1 2  0 1]
```

means that the regression vector at time t is composed of the data value on channel 1 with delays 1 and 2 (i.e $d_{1;t-1}$, $d_{1;t-2}$) together with the data value on channel 2 with delays 0 and 1 ($d_{2;t}$, $d_{2;t-1}$).

Specification:

System: realization of *ndat* two-dimensional data generated by a normal dynamic mixture with one component. The realization is determined by *seed*. The first factor is an ARX model with regression coefficients $a1 = 1.81, a2 = -0.8187, b0 = 0.00468, b1 = 0.00438$ and the output variance *covy*. The second factor provides input as normal, zero-mean white noise with covariance *covu*.

Decision: point estimate of the model structure and recursively updated point estimates of regression coefficients.

Experience: past data up to *ndat* for structure estimation and the order *ord* of the richest regression vector; for parameter estimation, experience is growing with time.

Ignorance: model structure and parameters.

Loss function: negative *v*-log-likelihood.

Recommended Experiments:

Observe dependence of point estimates on different initial factor attributes. The presence of the second channel (input) in the estimated structure is an important indicator of the quality of the estimate. It is worth observing the influence of

- the extent of learning data, say *ndat* = 50, 100,
- the process-noise covariances, *covy*= 0.001, 0.01 and *covu* = 1, 0.1,
- the richest model order *ord* = 5, 4, 3,
- the other initial model attributes: noise covariance *cove* = 1, 0.1, *diaCth* = 100, 100000, degree of freedom *dfm* = 1, *ndat*.

estim/factor.m | See results | Contents

3.2.7 Prior Knowledge in Structure Estimation

Aim:

To show the influence of the prior knowledge model in factor structure estimation. This section refers to Sections 6.3{96} and 8.3{244}.

Description:

Observed data are generated by a dynamic simulated mixture. A sufficiently large richest model order *ord* is selected. It specifies the space of possible model structures.

A guess of the system gain serves as the processed prior knowledge. It is specified as a range of values *gain*. The range implies both the expected value of the gain as well as the weight of this piece of knowledge.

The initial model is built and its structure is estimated with and without prior knowledge. The display is done via the internal form of structures — as a list of channels and delays (channel 0 stands for offset), e.g, the structure

```
str = [1 1  2 2
       1 2  0 1]
```

means that the regression vector at a time t is composed of the data value on the channel 1 with delays 1 and 2 (i.e $d_{1;t-1}$, $d_{1;t-2}$) together with the data value on the channel 2 with delays 0 and 1 ($d_{2;t}$, $d_{2;t-1}$).

Specification:

System: realization of *ndat* two-dimensional data generated by a normal dynamic mixture with one component. The realization is determined by *seed*.

Decision: the estimate of the structure of the first factor.

Experience: past data up to *ndat*; prior knowledge about the system gain is either exploited or not.

Ignorance: model structure and parameters.

Loss function: negative *v*-log-likelihood.

Recommended experiments:

The structure estimation task depends on signal-to-noise ratio, space of possible regressors and prior knowledge. The experiments should show the contribution of prior knowledge to the structure estimation task. It is worth observing the influence of

- the extent of learning data, say *ndat* = 100, 500, 200,
- process-noise covariances, *covy*= 0.001, 0.01 and *covu* = 1, 0.1,
- richest model order *ord* = 5, 4, 3,
- gain range, gain = [0.09 1.01] or [0.9 1.1].

estim/zgain.m | See results | Contents

Chapter 4

Design with Normal Mixtures

This chapter contains case studies related to Chapters 5{67}, 7{193} and mainly 9{309}.

The design is based on the model of the o-system. The observed behavior, tailored to the rate of operator actions, is described by the mixture

$$f(d_t|\phi_{t-1},\Theta) = \sum_{c\in c^*} \alpha_c f(d_t|\phi_{c;t-1},\Theta_c,c).$$

Each component $f(d_t|\phi_{c;t-1},\Theta_c,c)$ is decomposed into the product of factors $f(d_{ic;t}|\psi_{ic;t},\Theta_{ic},c)$; see Agreement 5.4{77}:

$$f(d_t|\phi_{c;t-1},\Theta_c,c) = \prod_{i\in i^*} f(d_{ic;t}|\psi_{ic;t},\Theta_{ic},c),\ i^* \equiv \{1,\ldots,\mathring{d}\}.$$

The factors are parameterized by individual parameters Θ_{ic}. A collection of these parameters, together with probabilistic weights $\alpha_c \in \alpha^*$, form the multivariate parameter Θ of the mixture.A Structure of the mixture as well as this parameter are assumed to be known throughout this chapter. For the fixed advisory system, their reliable point estimates are supposed to be obtained during the offline learning phase.

The considered design of an optimal advisory system assumes that the aims of management and advising can be expressed by the user's ideal pdf describing the desired behavior of the data $d(\mathring{t}) \equiv d_p(\mathring{t})$.

The optional advising elements are optimized so that the resulting mixture is as close as possible to the user's ideal pdf. The result is then offered to the operator as the target to be followed. The design is performed in the fully probabilistic sense, Section 2.4.2{28}, i.e., proximity of the involved pdfs is judged via the KL divergence.

Basic types of advisory systems differ in the extent of optional ingredients of the modified model.

The advisory system designed without knowing which entries of $d_{o;t}$ belong to the action space of the operator is called the academic advisory system. The corresponding design is called academic design.

The advisory system designed with knowledge of a non-empty part of the action space of the operator, is called industrial advisory system. The design of recognizable actions is called industrial design. The joint academic and industrial design is called simultaneous design.

Contents

4.1 Academic Design

We have at our disposal the multiple-mode mixture model of the nonguided o-system $f(d(\mathring{t}))$. According to Agreement 5.5{82}, the advisory system maps $f(d(\mathring{t}))$ on an ideal pdf $\lfloor^I f(d(\mathring{t}))$, whose projections are presented to the operator.

The joint pdf $f(d(\mathring{t}))$ describes the probability distribution of achievable modes within the data space of the advisory system. Thus, the reachable ideal pdf should be created from these modes. The selection of modes leading to a higher management quality should be advised. The academic design selects the recommended mode through the recommended pointer $c_t \in c^*$ by defining the ideal pdf

$$\lfloor^I f(d(\mathring{t}), c(\mathring{t})) \equiv \prod_{t \in t^*} \lfloor^I f(d_t, c_t | d(t-1)) \equiv \prod_{t \in t^*} f(d_t | d(t-1), c_t) \lfloor^I f(c_t | d(t-1)).$$

In other words, the basic idea of academic design is to find new, optimized component weights so that data generated by the optimized guided system has its pdf as close as possible to the user's ideal pdf in the KL sense.

The first example uses a simple two-dimensional static system. It makes visualization of the involved pdfs simple.

Contents

4.1.1 Academic Design with Static Mixtures

Aim:

To inspect typical results of academic design for various choices of true user's ideal pdf defining the desired data ranges and preferences on particular quantities.

Observe the role of the user's ideal pdf on recommended pointers to components. This optional design parameter Ufc is out of the *direct* user's choice, but it can be influenced indirectly by specifying additional requirements on the advising strategy (e.g., exclusion of dangerous components).

Description:

Academic design is performed according to Algorithm 9.8{343}. The advices supplied to the operator are recommended pointers to the components. The design provides such probabilities of the components that the resulting mixture, has minimal KL divergence from the user's ideal pdf.

The static mixture model is used for the system description. An optional design parameter, optimization horizon, can be set to 1, as the resulting advising strategy for the static mixtures does not depend on the data. Mean value Uth and covariance $Ucov$ of the true user's ideal pdf can be changed to observe different results of the design; see detailed description.

Specification:

System: two-dimensional data records, simulated by static normal mixture of three components.

Decision: recommended pointers to the components.

Experience: past data, complete model of the system, i.e., structure and parameter.

Ignorance: future data.

Admissible strategies: causal mapping on recommended pointers.

Loss function: KL divergence from the user's ideal, which is obtained by extension of the true user's ideal, described by one static component.

Recommended experiments:

The advices vary for different true user's ideal pdfs. To observe this, choose, for example, the mean of the true user's ideal pdf to be equal

- to a mean of some system component, for example, $Uth1 = 3$, $Uth2 = 1$;
- to an approximate center of the system's components, for example, $Uth1 = 2.5$, $Uth2 = 1.5$;
- to the value outside the system components means, for example, $Uth1 = 0$, $Uth2 = 4$.

The influence of the changes of covariances for the particular quantities can be tested as well.

design/acdes1.m | See results | Contents

4.1.2 Academic Design with Dynamic Mixtures

Aim:

To inspect typical results of academic design for various choices of the true user's ideal pdf defining the desired data ranges and preferences on particular quantities.

Observe the role of the user's ideal pdf on recommended pointers to components. This optional design parameter Ufc is out of the *direct* user's choice, but it can be influenced indirectly by specifying additional requirements on the advising strategy (e.g., exclusion of undesirable components).

Description:

Academic design is performed according to Algorithm 9.8{343}. The advices supplied to the user are recommended pointers to the components. The design provides such probabilities of the components that the resulting mixture has minimal KL divergence from the user's ideal pdf.

The dynamic mixture model is used for the system description. Unlike in Example 4.1.1, the optimization is substantially influenced by the optimization horizon. Mean value Uth and covariance $Ucov$ of the true user's ideal pdf can be changed to observe different results of the design; see detailed description.

Specification:

System: two-dimensional data record, simulated by dynamic normal mixture of three components.

Decision: recommended pointers to the components.

Experience: past data, complete model of the system, i.e., structure and parameter.

Ignorance: future data.

Admissible strategies: causal mapping on recommended pointers.

Loss function: KL divergence from the user's ideal, which is obtained by extension of the true user's ideal, described by one normal component.

Recommended experiments:

The advices vary for each different user's ideal and different values of the optimization horizon. To verify this, observe the influence of

- changes of the value of covariance $Ucov$ of the true user's ideal, in particular, for the second factor: $Ucov1 = 1$, $Ucove2 = 1000$;
- changes of the optimization horizon hor, for example, hor= (1; 10; 50; [1 50]; [5 10]; [5 50]);
- changes of the user's ideal pdf on recommended pointers to components by setting, for example, Ufc= [1/2 1/2 0]. This experiment imitates exclusion of undesirable advices, here the third component.

The influence of the changes of mean values Uth for the particular quantities can be observed as well.

design/acdes2.m See results Contents

4.2 Industrial and Simultaneous Designs

The industrial design optimizes recommended recognizable actions $u_{o;t}$. Ideally, these actions are directly fed into the o-system and their consequences are predicted by the model of the unguided o-system. They are similar to ordinary inputs of the o-system with the operator serving as an imperfect actuator. The constructed randomized strategy is described by the pdfs $\{ \lfloor^I f(u_{o;t}|d(t-1))\}_{t\in t^*}$. These pdfs replace $\{f(u_{o;t}|d(t-1))\}_{t\in t^*}$ forming a part of the estimated unguided model $f(d_t|d(t-1))$. Thus, the ideal pdf generated by this design has the form

$$\lfloor^I f(d_t|d(t-1)) = f(\Delta_t|u_{o;t}, d(t-1))\, \lfloor^I f(u_{o;t}|d(t-1)) = \tag{4.1}$$

$$= \lfloor^I f(u_{o;t}|d(t-1)) \frac{\sum_{c_t\in c^*} \alpha_{c_t} f(\Delta_t|u_{o;t}, d(t-1), c_t) f(u_{o;t}|d(t-1), c_t)}{\sum_{c_t\in c^*} \alpha_{c_t} f(u_{o;t}|d(t-1), c_t)}.$$

The optimal strategy described by the pdf $\lfloor^I f(u_{o;t}|d(t-1))$ is obtained through the fully probabilistic design, Proposition 2.11{30}. The needed user's ideal pdf

$$\lfloor^U f(d(\mathring{t})) = \prod_{t\in t^*} \lfloor^U f(\Delta_t|u_{o;t}, d(t-1))\, \lfloor^U f(u_{o;t}|d(t-1)),$$

which includes the target for the recognizable actions, is constructed exactly as described in Section 5.1.5{73}.

The industrial design has to be used whenever component weights have objective meaning and cannot be influenced by the operator.

The joint academic and industrial design is called simultaneous design. The simultaneous design optimizes both recommended pointers to components and recommended recognizable actions. It should lead to a better advising strategy than a sequential use of academic and industrial designs.

Contents

4.2.1 Industrial Design with Dynamic Mixtures

Aim:

To inspect typical results of industrial design for various choices of the true user's ideal pdf defining the desired data ranges and preferences on particular quantities and observe influence of the optimization horizon on the advises. For this type of design, the user's ideal coincides with the true user's ideal.

Description:

Industrial design is performed according to Algorithm 9.11{352}. The advices provided to the operator are the recommended recognizable actions while the component weights are kept unchanged. The design provides such a strategy that the resulting mixture has minimal KL divergence from the user's ideal pdf. The dynamic mixture model is used for the system description. The example supposes recognizable actions coincide with system inputs. As a fully cooperating user is assumed, the designed advisory strategy for this example is equivalent to an optimal feedback controller.

The results of the design are substantially influenced by the selected optimization horizon *hor*; see detailed description.

Specification:

System: two-dimensional data record consists of system output and input and is simulated by dynamic normal mixture of three components.

Decision: recommended recognizable actions.

Experience: past data, complete model of the system, i.e., structure and parameter.

Ignorance: future data.

Admissible strategies: causal recommended recognizable actions.

Loss function: KL divergence from the user's ideal, which is obtained by extension of the true user's ideal, described by one normal component.

Recommended experiments:

The advices vary with changes of the true user's ideal and with the horizon of optimization. To verify this, observe the influence of:

- different choices of means of the true user's ideal pdf, for example, [$Uth1 = 5$, $Uth2 = 0$]; [$Uth1 = 5$, $Uth2 = 5$];
- different choices of covariances of the true user's ideal pdf, for example, $Ucov1 = 1$, $Ucov2 = 1000$;
- different choices of optimization horizon $hor = \{1, 10, 100\}$.

design/inddes2.m See results Contents

4.2.2 Simultaneous Design with Dynamic Mixtures

Aim:

To inspect typical results of simultaneous design for various choices of the true user's ideal pdf defining the desired data ranges and preferences on particular quantities and observe influence of the optimization horizon on the design results.

Observe the role of the user's ideal pdf on recommended pointers to components. This optional design parameter is out of the *direct* user's choice, but it can be influenced indirectly by specifying additional requirements on the advising strategy (e.g., exclusion of unstable advises).

Description:

Simultaneous design combines features of academic and industrial designs, see Examples 4.1.2 and 4.2.1, and is performed according to Algorithm 9.13{362}. The simultaneous design searches for the such advising strategy, selecting both recommended pointers to components and recommended recognizable actions, that the resulting optimized mixture has minimal KL divergence from the user's ideal pdf. The recognizable actions coincide with system inputs in this example. The dynamic mixture model is used for the system description; see detailed description. The results of the design are substantially influenced by the chosen horizon *hor*.

Specification:

System: two-dimensional data record consists of system output and input and it is simulated by dynamic normal mixture of three components.

Decision: recommended recognizable actions and recommended pointers to components.

Experience: past data, complete model of the system, i.e., structure and parameter.

Ignorance: future data.

Admissible strategies: causal mapping on recommended pointers and recommended recognizable actions.

Loss function: KL divergence from the user's ideal, which is obtained by extension of the true user's ideal, described by one normal component.

Recommended experiments:

The advices vary with changes of the user's ideal and with horizon of optimization. To verify that, observe the influence of:

- different choices of means of the true user's ideal pdf, for example, [$Uth1$ = -5, $Uth2$= -2]; [$Uth1$ = 0, $Uth2$= 0],
- different choices of covariances of the true user's ideal pdf, for example, $Ucov1$ = 1, $Ucov2$= $\{1, 10, 100, 10000\}$,
- different choices of optimization horizon hor= $\{100, [10,\ 10]\}$.

Besides, experiment imitating exclusion of undesirable advises can be performed by changing the user's ideal pdf on recommended pointers to components Ufc.

design/simdes2.m See results Contents

Chapter 5

Learning and Design with Markov Mixtures

This chapter contains examples related to learning and design with models describing discrete-valued data — mixtures of generalized Markov chains. The corresponding theory is discussed in Chapters 10{377}, and 11{411}. It is specialization of general results presented in Chapters 5{67}, 6{95}, and 7{193}.

The case studies in this chapter deal with discrete systems consisting of two quantities — input u_t and output y_t or of output y_t only. Both input and output are two-valued, $u_t \in \{1, 2\}, y_t \in \{1, 2\}$. The models describing these systems are mixtures of controlled Markov chains.

For Markov chains it holds that

$$f(y_t|u_t, d(t-1), \Theta) = f(y_t|u_t, \phi_{t-1}, \Theta) = \Theta_{y_t|u_t,\phi_{t-1}},$$

where

u_t is input at time t,

y_t is output at time t,

$d(t) \equiv (y_t, u_t, y_{t-1}, u_{t-1}, \ldots, y_1, u_1)$ is a sequence of data records until time t,

ϕ_{t-1} is the observable state, i.e., a subvector of $d(t-1)$ of a given structure, for instance, $\phi_{t-1} = (y_{t-1}, y_{t-2})$,

$\Theta_{y|u,\phi}$ are parameters of the model.

Note that u_t and y_t are denoted as `u(t)` and `y(t)` in MATLAB examples.

Contents

5.1 Learning

Algorithms for learning parameters of Markov mixtures, see Sections 10.5{400}, are special cases of general algorithms for learning with probabilistic mixtures, see Section 6.5{154}. Of course, a mixture of Markov chains is again a Markov chain with a regression vector, which contains regression vectors of all components. Thus, a mixture of Markov chains can be estimated as a single Markov chain without any approximation. Nevertheless, using a mixture of Markov chains instead of a single "common" one often requires a lower number of parameters. This justifies employment of general approximate learning techniques relying on the mixture-type parameterization. It is easy to demonstrate that this parameterization is not unique. Thus, two Markov mixtures may significantly differ in their parameters even if they describe (almost) the same processes. For this reason, the learning results are compared by expressing mixtures as single Markov chains.

Contents

5.1.1 Estimation of Markov Chain Mixture Parameters

Aim:

To inspect approximate parameter estimation of a Markov chain mixture with a known structure and see consequences of nonuniqueness of the adopted parameterization. The approximate estimation exploits the quasi-Bayes algorithm; see Sections 6.5.1{155}, and 10.5.1{400}.

Description:

The simulated system is a mixture of two Markov chains, each representing coin-tossing with memory. The output in the first component depends on the previous output with delay 1. The second component depends on the output with delay 2.

First, *ndat* data records are simulated from the given model. The parameters of the model of the same structure as the simulated system are recursively estimated from these data using the quasi-Bayes (QB) algorithm; for additional information see Section 10.5.1{400}.

For comparison, the simulated system and estimated model are transformed to Markov chains with a structure of regression vector common for both components.

For comparison, the simulated and estimated mixtures are transformed to Markov chains which are equivalent to the considered mixtures.

Specification:

System: two-valued outputs, dynamic memory of length 2, two components with different regression vectors. The parameters of the system are vectors determining transition probabilities *th1*, *th2*, and a component weight *alpha*.

Decision: generalized Bayesian parameter estimates and point estimates *Eth1*, *Eth2* of mixture parameters and point estimate *Ealpha* of component weight.

Experience: past observed data: the experience is extending up the full set *ndat* data; the estimated mixture is of the Markov chain type with structure identical with the simulated one.

Ignorance: mixture parameters, future data.

Loss function squared Euclidean norm of the difference between the unknown parameter and its point estimate.

Recommended experiments:

The estimation results depend on the extent of processed data and simulated system. With increasing number of data, the individual point estimates are converging to constant values that may substantially differ from the simulated ones due to the nonuniqueness of the adopted parameterization. The overall ability to express relations between past and future data is a decisive indicator of the model quality. It is worth observing the influence of

- the extent of learning data, say *ndat* = 500, 300, 200,
- the properties of the simulated system by changing mixture parameters; try *th1* = [0.25 0.75], *th2* = [0.8 0.5], *alpha* = 0.333.

Notice that the parameters of a Markov chain, which is equivalent to the original mixture, are close to the parameters of a Markov chain, which is equivalent to the estimated mixture — even in the case that parameters of original and estimated mixture significantly differ.

markov/MarkEst.m Contents

5.2 Design

Decision strategy is selected using fully probabilistic design; a model of the system, user's objectives, and a decision strategy are described by pdfs. The optimal decision strategy is designed so that the joint pdf given by the model and the decision strategy is close — measured by KL divergence — to the user's ideal pdf expressing the user's objectives.

For general theory related to design, see Chapter 7{193}, especially, Section 7.2{210}. For design with Markov mixtures, see Chapter 11{411}, Section 11.2{419}.

Note that unlike the normal mixture case, see Chapter 4, the design for Markov chains with known parameters needs no approximation.

Contents

5.2.1 Academic Design with Markov Chain Mixtures

Aim:

To inspect typical results of academic design for various choices of the user's ideal pdf in the case of mixtures of Markov chains.

Description:

The considered system is a mixture of two Markov chains, each representing coin-tossing with memory — CCM model — but without input. Both components have the same regression vectors — the previous observed output.

The user's ideal pdf is specified as a single Markov chain component that has the same structure as the components of the system. The true user's ideal pdf is the desired transition probability, determined by the parameters *Uth*. The true user's ideal is extended to the user's ideal pdf by specifying the pdf *Ualpha* on recommended pointers to components.

The optimal advising strategy assigns probabilities to the recommended pointers on respective components. These probabilities serve as optimal component weights, for complementary information see Sections 7.2{210}, and 11.2{419}. The optimal advising strategy and its projection on o-data are displayed.

Ideally, the receding design horizon *nhor* should coincide with the simulation length *ndat*. It is, however, computationally intensive. Thus, it makes sense to choose $nhor < ndat$. Naturally, it is paid for by a loss of quality.

In order to see the influence of the optimization, *ndat* data are simulated with the original component weights (unguided system), and with optimized weights (guided system); see Section 5.3{70}.

Specification:

System: two-valued output, dynamic memory of length 1, two components with the delayed output as regression vector.

Decision: recommended pointers to respective components.

Experience: model structure and parameters, past observed data.

Ignorance: future output up to the control horizon.

Admissible strategies: randomized causal, assigning ideal component weights.

Loss function: KL divergence to the user's ideal pdf.

Recommended experiments:

Satisfaction with optimization results depends on the ability to express the aims using the true user's ideal pdf and on the properties of individual components of the system. For certain combinations of system parameters *th1*, *th2*, and user's ideal pdf *Uth*, the desired "behavior" can be hardly approached. It is worth observing this dependence by changing the parameters *th*1, *th*2, *Uth*, *nhor*, and *ndat*

- parameters of the system, *th1* = [0.7 0.6], *th2* = [0.8 0.9], *Uth* = [0.1 0.1],
- simulation length, *ndat* = 200,
- user's ideal pdf, *Uth* = [0.9 0.9],
- receding design horizon, *nhor* = *ndat* = 50 or *nhor* = 1.

markov/MarkConA.m Contents

5.2.2 Simultaneous Design with Markov Chain Mixtures

Aim:

To inspect typical results of simultaneous design for various choices of the user's ideal pdf in the case of mixtures of Markov chains.

Description:

The considered system is a mixture of two controlled Markov chains each representing controlled coin-tossing with memory — the CCM model. Both components have the same regression vector, consisting of the current input and the delayed output.

The user's ideal pdf is specified as a single Markov chain component that has the same structure as the components of the system. The true user's ideal pdf is the desired transition probability for the output and the input determined by the parameters *Uth1*, *Uth2*. The true user's ideal is extended to the user's ideal pdf by specifying the pdf *Ufc* on recommended pointers to components.

The optimal advising strategy assigns joint probabilities to the recommended pointers to components and values of input; for complementary information, see Sections 7.2.4{225}. The optimal advising strategy and its projection on o-data are displayed as results.

In order to see the influence of the optimization, *ndat* data are simulated with the original component weights and distribution of input (unguided system), and with optimized weights and distribution of input (guided system); see Section 5.3{70}.

Ideally, the receding control horizon *nhor* should coincide with the simulation length *ndat*. It is, however, computationally intensive. Thus, it makes sense to choose $nhor < ndat$. Naturally, it is paid for by a loss of quality.

Specification:

System: two-valued output, dynamic memory of length 1, two components with the current input and the previous output as a regression vector.

Decision: input values and recommended pointers to respective components.

Experience: model structure and parameters, past observed data.

Ignorance: future output values up to control horizon.

Admissible strategies: randomized causal, assigning joint probabilities of pointers to components and input values.

Loss function: KL divergence to the user's ideal pdf.

Recommended experiments:

It is worth observing how the optimization results depend on the true user's ideal pdf, especially for true user's ideal pdfs, expressing user's aims, e.g., when a certain value of the output is highly preferred. Satisfaction with optimization results depends on the properties of individual components of the system. For certain combinations of system parameters *th1*, *th2*, *th3*, *th4* and user's ideal pdf *Uth1*, *Uth2* the desired "behavior" of the system can be hardly approached. This dependence can be observed by changing the parameters $th1$, $th2$, $Uth1$, $Uth2$, $nhor$, and $ndat$

- user's ideal pdf, *Uth1* = [0.1 0.1 0.1 0.1],
- system parameters, *th1* = [0.9 0.7 0.6 0.8], *th2* = [0.6 0.9 0.8 0.6], user's ideal pdf, *Uth1* = [0.1 0.1 0.1 0.1],
- receding control horizon, *nhor* = *ndat* or *nhor*=1.

markov/MarkConS.m Contents