dcv : A set of MATLAB functions for blind deconvolution of discrete signals

Olivier Cappé
ENST Dpt. TSI / CNRS-URA 820,
46 rue Barrault, 75634 Paris cedex 13, France.
`cappe@tsi.enst.fr`

August 28, 2012

Keywords: Blind estimation, deconvolution, system identification, hidden Markov model, communications

Contents

1 About `dcv`

`dcv` is a set of functions for blind linear system identification and data detection in the case of a scalar discrete input signal (taking only a finite number of distinct values). The name itself originated from the phrase "Discrete ConVolution" which is not very informative! `dcv` is meant to handle both real and complex signals and has a number of maximum likelihood (EM algorithm, two stochastic versions of the EM algorithm) and Bayesian estimation schemes implemented. No elaborate data types are used and `dcv` functions may all be used on `MATLAB 4` as well as `MATLAB 5`. Among known limitations is the fact that `dcv` basically only handles Single Input Single Output (SISO) systems. The only way to (slighty) overcome this limitation consists in defining a single input - two outputs model by specifying real input symbols together with a complex filter.

The current version of the `dcv` 1.1. The only change from the first version (1.0, dated from january 1998) is the inclusion of functions that allow for a direct optimization of the log-likelihood using recursive update formulas (`dcv_llk`, `dcv_gllk` and `dcv_qn`).

2 Wish list

`(added November 25, 1999)` For MATLAB 5.3 users, the current version of *Optimization Toolbox* is 2.0 in which `fminu` is maintained only for backward compatibility. With *Optimization Toolbox 2.0*, `dcv_qn` should thus be modified so as to use `fminunc` which is more efficient and robust than the old `fminu` (but also has a slightly differing syntax)...

3 Model and hypotheses

`dcv` is based on the following Moving Average (MA) convolution model

$$X_t = \sum_{l=0}^{L-1} h_l^* \epsilon_{t-l} + N_t \,, \tag{1}$$

where $\{\epsilon_t\}$ is a sequence of iid. discrete (and possibly complex) random variables, and $\{N_t\}$ is a, possibly complex, white noise such that $E[N_t] = 0$, $E[|N_t|^2] = \sigma^2$ and $E[Re(N_t)Im(N_t)] = 0$. In the current version, it is assumed that all symbols (values of the input process $\{\epsilon_t\}$) are equiprobable: their probability of occurrence is $1/M$ where M is the number of different symbols. In (1), the $*$ superscript denotes conjugation (or the Hermitian adjoint for matrices). The reason why conjugation is needed at this point may become clearer when noting that (1) is equivalent to

$$X_t = \mathbf{h}^* \mathbf{V}_t + N_t \,, \tag{2}$$

where $\mathbf{h} \triangleq (h_0, h_1, \ldots, h_{L-1})^*$ (with the convention that for vectors the superscript $*$ indicates both transposition and complex conjugation) and $\mathbf{V}_t \triangleq (\epsilon_t, \epsilon_{t-1}, \ldots, \epsilon_{t-L+1})'$. Written as in (2), it is clear that X_t is a (deterministic) function of a discrete multivariate Markovian process $\{\mathbf{V}_t\}$ observed in additive noise. Stated differently, (1) corresponds to a finite state-space Hidden Markov Model (HMM).

Note that the dimension of the state space of $\{\mathbf{V}_t\}$ is M^L, which means that it grows exponentially with the length of the filter. However, it is easily seen that the transition matrix corresponding to $\{\mathbf{V}_t\}$ is sparse because each line (and each column) contains only N non-zero elements.

It is clear from the above discussion that the MA blind convolution model with a discrete input signal is just a particular case of finite state space HMM with Gaussian state conditional densities. There are however at least three good reasons that justify the development of specialized functions: (rather than using the more general framework of the `h2m` toolbox[1]):

- The sparsity of the transition matrix can be used to simplify the filtering procedures (such as the forward-backward recursion).
- The standard estimation procedures for HMMs have to be modified since there is a single set of parameters ($\mathbf{h}$ and σ) for all the state conditional distributions.
- `h2m` functions won't handle complex parameters or observations.

4 Model representation and data types

If you just want to use the basic simulation (`dcv_gen`) and estimation functions (`dcv_io`, `dcv_em`, `dcv_sem`, `dcv_saem`, `dcv_baye`, `dcv_vit`), you just need to specify

A vector of observations: (usually denoted `X`) Observed signal $X_1, \ldots X_T$.

A vector of symbols: (usually denoted `v`) M possible values of the input signal.

A vector of filter coefficients: (usually denoted `H`) Vector $\mathbf{h}$ in (2).

A positive variance: (usually denoted `sigma2`) Variance σ^2 of the noise N_t.

Note that all vectors have to be specified as *column vectors*.

If you need to use the lower level functions (`dcv_fb`, `dcv_psim`), you will have to create a representation of the transition matrix using `dcv_init`:

[1]http://www-sig.enst.fr/~cappe/h2m/html/

```
[AR,AC,V]=dcv_init(v,H);
```

This creates a $M^L \times M$ matrix `AR` which contains the indexes of the non null elements of the transition matrix for each row, `AC` is a $M \times M^L$ matrix which likewise gives the indexes of the non null elements in the transition matrix for each column and `V` is a $L \times M^L$ matrix which contains the M^L possible values of the state vector $\mathbf{V}_t$ stacked column-wise. If needed, the function `dcv_chk` can be used to check that `AR`, `AC`, `V` and `H` are compatible and to obtain the dimensions of the model. `dcv_chk` also sets a flag `REAL_COEFF` which indicates whether the model is real or complex (by looking at the symbols in the state vectors `V` and the filter coefficients in vector `H`).

5 Estimation functions

5.1 Input-output estimation

`dcv_io` estimates the parameters of the convolution model (filter and noise level) when the input symbols are observed. In this case the maximum likelihood estimate is obtained using a standard least squares procedure [9].

All other estimations procedures deal with the case of blind estimation and are iterative in nature.

5.2 Blind estimation with EM

`dcv_em` implements blind estimation (that is without observing the input symbols) using the Expectation Maximization (EM) algorithm. The E step is carried out in `dcv_fb` using the forward-backward recursions of Baum et al [13], [12]. The use of EM for this particular model is also detailed in many papers dealing with telecommunications applications such as [8] or [1].

Note that `dcv_em` (and all the iterative estimation procedures) returns the sequence of estimated parameters rather than just the parameters estimated during the last iteration. `H(:,1)` and `sigma2(1)` are the initial values (as given in the input arguments of `dcv_em`) and `H(:,Nit+1)` and `sigma2(Nit+1)` the values estimated at the last iteration.

5.3 Blind estimation using quasi-Newton optimization

`dcv_qn` implements blind estimation using a direct optimization on the log-likelihood using a quasi-Newton technique. It requires `fminu` for optimization and thus can only be used if `Optimization Toolbox` is installed.

As this is perhaps less well-known than EM, we briefly describe the procedure used to compute the gradient of the log-likelihood. A more detailed account of the method can be found in [2]. For a sequence of observations of length T, the log-likelihood can be written as

$$\log p(x_1, \ldots x_t) = \sum_{t=1}^{T} \log \left[\sum_{i=1}^{N} f_i(x_t) \phi_t(i) \right] \tag{3}$$

where $f_i(x_t) = p(\mathbf{v}_t | \mathbf{V}_t = \mathbf{v}_i)$ is the state conditional distribution (which is complex Gaussian with mean $\mathbf{h}^* \mathbf{v}_i$ and variance σ^2) and $\phi_t(i) \triangleq P(\mathbf{V}_t = \mathbf{v}_i | x_1, \ldots x_{t-1})$ denotes the state prediction filter. $\phi_t(i)$ can be updated recursively using (see [14])

$$\begin{aligned} \phi_1(j) &= P(\mathbf{v}_1 = j) \\ \phi_{t+1}(j) &= \frac{1}{c_t} \sum_{i=1}^{N} f_i(\mathbf{x}_t) \phi_t(i) A_{ij} \quad (t \geq 1) \end{aligned} \tag{4}$$

where $\mathbf{A}$ denote the transition matrix (which is sparse in the case of the model considered here). The normalization factors c_t are given by

$$c_t = \sum_{k=1}^{N} f_k(x_t) \phi_t(k) \tag{5}$$

The gradient of the log-likelihood is obtained by mere differentiation of (3) as

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} \log p(\mathbf{X}_1^T) = \sum_{t=1}^{T} \frac{1}{c_t} \sum_{i=1}^{N} \left[\phi_t(i) \boldsymbol{\nabla}_{\boldsymbol{\theta}} f_i(\mathbf{x}_t) + f_i(\mathbf{x}_t) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \phi_t(i) \right] \tag{6}$$

Where $\boldsymbol{\nabla}_{\boldsymbol{\theta}} \phi_t(i)$ can be updated recursively using

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} \phi_{t+1}(j) = \frac{1}{c_t} \sum_{i=1}^{N} \left(A_{ij} - \phi_{t+1}(j) \right) \left[\phi_t(i) \boldsymbol{\nabla}_{\boldsymbol{\theta}} f_i(\mathbf{x}_t) + f_i(\mathbf{x}_t) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \phi_t(i) \right] \tag{7}$$

Since $f_i(x_t)$ is the Gaussian density, we have

$$\frac{\partial f_i(x_t)}{\partial \mathbf{h}} = \frac{(x_t \mathbf{v}_i - \mathbf{v}_i \mathbf{v}_i' \mathbf{h})}{\sigma^2} f_i(x_t) \tag{8}$$

and

$$\frac{\partial f_i(x_t)}{\partial \kappa} = \frac{1}{2} \left[\left(\frac{x_t - \mathbf{h}' \mathbf{v}_i}{\sigma} \right)^2 - 1 \right] f_i(x_t) \tag{9}$$

for a real model, and

$$\frac{\partial f_i(x_t)}{\partial \mathbf{h}} = 2 \frac{(x_t^* \mathbf{v}_i - \mathbf{v}_i \mathbf{v}_i^* \mathbf{h})}{\sigma^2} f_i(x_t) \tag{10}$$

and

$$\frac{\partial f_i(x_t)}{\partial \kappa} = \left[\left(\frac{|x_t - \mathbf{h}^* \mathbf{v}_i|}{\sigma} \right)^2 - 1 \right] f_i(x_t) \tag{11}$$

for a complex one. In both case $\kappa \triangleq \log(\sigma^2)$ is used as the noise parameter (rather than σ) so that all model parameters ($\mathbf{h}$ and κ) are unconstrained.

The computation of the log-likelihood and its gradient is done, respectively, in `dcv_llk` and `dcv_gllk`. `dcv_qn` (and its subroutines `dcv_qn_f` `dcv_qn_g`) is just a function that makes the interface between `dcv` parameter conventions and the `fminu` optimization routine. Because `fminu` does not provide the intermediate optimization steps, `dcv_qn` only returns the value of the estimated parameters `H` and `sigma2` for the last itteration, which is in contrast with the behavior of the other blind estimation routines.

In practise, it happens frequently that, at some point of the optimization, numerical problems (underflows) occur in `dcv_llk` because of large steps on the parameter κ required by `fminu`. To alleviate this problem, all functions as been modified to allow the use of an optional variance offset $V_{\min}$ such that $\sigma^2 = \exp(\kappa) + V_{\min}$ which guarantees that $\sigma^2 > V_{\min}$ and thus avoids numerical problems which appear when computing the Gaussian likelihood for very small variance values. The only change that is needed in the above equations consists in multiplying (9) or (11) by

$$(\sigma^2 - V_{\min})/\sigma^2$$

to account for the fact that differentiation is performed wrt. κ and not wrt. $\log(\sigma^2)$. By default, `dcv_qn` uses a variance offset (as defined by `VAR_OFFSET`) that corresponds approximately to a 25 dB Signal to Noise Ratio (SNR). Of course if the SNR is actually greater than that, this should be modified by specifying the optional argument `VAR_OFFSET` to `dcv_qn`. Note however that for lower values of the variance threshold you may experience some numerical problems if the algorithm is started from a value of the parameters which is far from any local maximum of the likelihood.

5.4 Blind estimation with SEM

`dcv_sem` uses the Stochastic EM (SEM) procedure [5]. This algorithm belongs to the class of stochastic extensions of the EM algorithm since it uses simulation and imputation of the missing data (the unobserved sequence of input symbols) in place of the E step. The simulation of the unobserved state vectors is carried out by `dcv_psim` using the principle explained in [3] (forward filtering and backward conditional simulation). The simulation is based on the relation

$$p(\mathbf{V}_t|\mathbf{V}_{t+1},\ldots,\mathbf{V}_T,X_1,\ldots,X_T) \propto p(X_1,\ldots,X_t,\mathbf{V}_t)p(\mathbf{V}_{t+1}|\mathbf{V}_t)$$

The first term on the right-hand side is computed by forward recursion (it is exactly the forward variable in the Baum algorithm). The above relation is then used for backward simulation (using it for $t = T, T-1, \ldots, 1$).

Rather than just a single number of iteration, `dcv_sem` expects the length of the burn-in period (during which the estimated values are discarded) and the number of subsequent iterations that will be used to obtain averaged estimates. Note that contrary to what happens for EM, the likelihood of the sequence of estimates has to be computed separately afterwards. This extra computation is performed only if the likelihood is requested (as an output argument) and can be omitted in order to save some time (the same is true for `dcv_saem`).

5.5 Blind estimation with SAEM

`dcv_saem` implements the Stochastic Approximation EM (SAEM) algorithm which is slightly different in spirit from the SEM, although it is also based on the simulation of the unobserved input data. Almost sure convergence of the SAEM iterates towards a local maxima of the likelihood under suitable regularity conditions has been proved in [11].

`dcv_saem` uses a three stages strategy: first a burn-in period (no decrease of the stochastic approximation step size which is set to 1), then a pure stochastic approximation procedure with proper step size decay so as to ensure convergence and finally an averaged stochastic approximation procedure [10]. After the burn-in period, `dcv_saem` uses stochastic approximation step sizes that decrease proportionally to $n^{-0.6}$ which is enough to guarantee convergence but has no claim to optimality.

5.6 Bayesian estimation with the Gibbs sampler

`dcv_baye` also deals with the blind convolution model but is different in nature since its goal is "fully Bayesian analysis" of the model using a Markov Chain Monte Carlo (MCMC) approach [7]. Note that in the present case, what prevent the use of a more direct analysis is the fact that the posterior is a mixture of $M^{(T+L-1)}$ component distributions (as many as the number of different sequences of symbols). Thus, when T is large the direct computation of the a posteriori distribution is not feasible.

`dcv_baye` uses a standard "non informative" prior for normal regression models (with $p(\mathbf{h}, \sigma^2) \propto \sigma^2$) as described in [6, Sec. 8.3]. For a related (though more elaborate) Bayesian model analysis, see [4]. `dcv_baye` uses a systematic scan Gibbs sampler based on data augmentation [15]. Each cycle of the Gibbs sampler features the following simulation steps:

- $\mathbf{V}_1, \ldots \mathbf{V}_T | X_1, \ldots, X_T, \mathbf{h}, \sigma$
- $\sigma | X_1, \ldots, X_T, \mathbf{V}_1, \ldots \mathbf{V}_T, \mathbf{h}$
- $\mathbf{h} | X_1, \ldots, X_T, \mathbf{V}_1, \ldots \mathbf{V}_T, \sigma$

In the first step the unknown state vectors are drawn jointly (in block) using `dcv_psim` [see [4] for a different implementation]. As you may experience yourself (using `xample_c` for instance), in many cases, the above sampling scheme does not provide a full exploration of the posterior density but rather visits only one of its mode. This is due to the particular mixing properties of the hidden Markov chain $\{\mathbf{V}_t\}$ and to the marked multimodal character of the posterior distribution of $\mathbf{h}$.

5.7 Estimation of the symbols

`dcv_vit` implements the so-called "Viterbi" or dynamic programming algorithm to estimate the most likely sequence of symbols together with the initial state. `dcv_vit` also returns the value of the symbol-optimized joint log-likelihood

$$\max_{\mathbf{V}_1,\ldots,\mathbf{V}_T} p(\mathbf{V}_1,\ldots,\mathbf{V}_T,X_1,\ldots,X_T;\mathbf{h}.\sigma^2)$$

6 List of functions

dcv_baye Bayesian analysis of the blind convolution model with the Gibbs sampler using a noninformative prior.

dcv_chk Checks the parameters of a convolution model and returns its dimensions.

dcv_em Blind identification using the EM algorithm.

dcv_fb Computes likelihood and a posteriori state probabilities using the forward-backward recursions.

dcv_gen Generates synthetic data for a convolution model.

dcv_gllk Computes the gradient of the log-likelihood using forward prediction.

dcv_init Returns transition matrix and state vectors for a convolution model.

dcv_io Input-output estimation of the convolution parameters.

dcv_llk Computes the log-likelihood using the forward predictor.

dcv_par Re-estimates the convolution parameters (filter and noise level).

dcv_psim Simulates sequence(s) of states conditionnally to the observed data.

dcv_qn Blind identification using quasi Newton optimization.

dcv_saem Blind identification using the SAEM algorithm.

dcv_sem Blind identification using the SEM algorithm.

dcv_vit Computes the most likely sequence of symbols and initial state (using "Viterbi" dynamic programming).

dcv2hmm Converts DCV parameters to HMM representation for the H2M toolbox.

randgamm Generates one deviate from the Gamma(alpha, beta) distribution.

randindx Generates random indexes with a specified probability distribution.

xample_c Illustration of the use of DCV functions for complex signals.

xample_r Illustration of the use of DCV functions for real signals.

`randgamm` is not a very efficient gamma random generator, you may want to use `gamrnd` instead if you have the `Statistics` toolbox installed. Note that none of the settings of the example files `xample_c` and `xample_r` should be considered as "optimal" as these files are just intended to illustrate the various function calls. In particular, the number of iterations may have to be set to higher values for practical applications (but you probably don't want to spend too much time in front of your screen for a simple demo). You will also notice from these little examples that blind identification is indeed a difficult task and that convergence of the estimation procedures strongly depends on their initialization.

7 Downloading `dcv`

`dcv` is available as a unix gz-compressed tape archive at address `http://perso.telecom-parsitech.f`

References

[1] C. Antón-Haro, J. A. R. Fonollosa, and J. R. Fonollosa. Blind channel estimation and data detection using hidden Markov models. *IEEE Trans. Signal Process.*, 45(1), January 1997.

[2] O. Cappé, V. Buchoux, and E. Moulines. Quasi-Newton method for maximum likelihood estimation of hidden Markov models. In *IEEE Int. Conf. Acoust., Speech, Signal Processing (ICASSP)*, pages IV–2265–IV–2268, Seattle, US, May 1998.

[3] C. K. Carter and R. Kohn. On Gibbs sampling for state space models. *Biometrika*, 81(3):541–553, 1994.

[4] R. Chen and T. Li. Blind restoration of linearly degraded discrete signals by gibbs sampling. *IEEE Trans. Signal Process.*, 43(10):2410–2413, 1995.

[5] J. Diebolt and E. H. S. Ip. Stochastic EM: method and application. In W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, editors, *Markov Chain Monte Carlo in Practice*, pages 259–273. Chapman & Hall, 1996.

[6] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian data analysis.* Chapman & Hall, 1995.

[7] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice.* Interdisciplinary Statistics Series. Chapman & Hall, 1996.

[8] G. K. Kaleh and R. Vallet. Joint parameter estimation and symbol detection for linear or non-linear unknown channels. *IEEE Trans. Communications*, 42(7), 1994.

[9] S. M. Kay. *Fundamentals of statistical signal processing: Estimation theory.* Signal processing series. Prentice-Hall, 1993.

[10] H. J. Kushner and G. G. Yin. *Stochastic approximation algorithms and applications.* Springer-Verlag, New York, 1997.

[11] M. Lavielle, B. Delyon, and E. Moulines. Convergence of a stochastic approximation version of the EM algorithm. *To appear in the Annals of Statistics*, 1998.

[12] I. L. MacDonald and W. Zucchini. *Hidden Markov models and other models for discrete-valued time series.* Chapman & Hall, 1997.

[13] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–285, February 1989.

[14] J.B. Moore R.J. Elliot, L. Aggoun. *Hidden Markov models: Estimation and control.* Springer-Verlag, New York, 1994.

[15] A. F. M. Smith and G. O. Roberts. Bayesian computation via the gibbs sampler and related Markov chain Monte Carlo methods. *J. Royal Statist. Soc. B*, 55(1):3–23, 1993.