Monte Carlo Variational EM

Proposed by Florence FORBES and Gersende FORT
Codes were developed by Gersende FORT
Version 1.0

October 4, 2006

Contents

1 Description and Version information

This is the MATLAB code that was used to produce the figures and tables in Section V of

> F. Forbes and G. Fort, *Combining Monte Carlo and mean-field like methods for inference in Hidden Markov Random Fields*, Accepted for publication in `IEEE Trans. on Image Processing`, 2006.

MATLAB has the capability of running functions written in C. The files which hold the source for these functions are called MEX-Files. Some functions of our codes are written in C.

The purpose of this software is to implement the **MCVEM** algorithm, described in the paper mentioned above, when applied to Image Segmentation. MCVEM consists in combining approximation techniques - based on variational EM - and simulation techniques - based on MCMC -.

This software is the first version that is made publicly available.

2 How to

2.1 Obtain the source code

Download it from

`http://www.tsi.enst.fr/~gfort/INRIA/MCVEM.html`

After unpacking the archive, you should obtain

- two MATLAB files *ExampleMCVEM.m* and *MCVEM.m.*
- directory `CodesMCVEM.` This directory contains the codes for running MCVEM. The main file is `InterfaceMCVEM.m`, where the different parameters and the different choices of implementation are fixed.
- directory `MEXfiles.` This directory contains the C-codes and the associated MEX-files.
- directory `Pictures.` This directory contains the synthetic and real images, used in the paper.
    1. For synthetic images called `pict`, three dat-files are available: `pict_true.dat`, `pict_obs.dat` and `pict_init.dat` that contain resp. the true image, the observed image (additive gaussian noise) and an initial segmentation for the iterative algorithm.
    2. For real images, only two dat-files are available: `pict_obs.dat` and `pict_init.dat`.

2.2 Check that it works

Start MATLAB and type EXAMPLEMCVEM. This will run the program MCVEM on four different images (two synthetic images and two real images).

You should see on the screen

- the list of the different displays when running.
- the files where these displays are defined.

After 5 sec, MCVEM starts.

2.3 Understand the program output

2.3.1 Display when running the MCVEM algorithm

MCVEM is an iterative procedure that produces a sequence of parameter estimate (a positive real number β, and a family of mean and variances) and a sequence of segmented images.

Table 1: We first display the initial value of the different parameters.

```
Initialisation
                     Mean 26.631337          Mean 460.228581
                     Std Dev 135.200515      Std Dev 183.897542
                     Beta 1.000000
```

Table 1:

```
Iteration 1.000000
                        Step 1 :    Determine the Q-distribution
                                    Nbr of iteration 46.000000
                                    Control :
                                         Diag of cvgce (1 if OK) 1.000000
```

Table 2:

Then per iteration :

Table 2: `Step 1` is an iterative procedure : we display the number of iterations till the stopping criterion is reached as well as a binary variable that indicates if the iterative procedure converged.

```
    Step 2a :   Update Mean and Variances
                Mean 38.931758              Mean 441.489396
                Std Dev 147.918913              Std Dev 217.574490
    Step 2b :   Update Beta
                Beta 0.998890
                Nbr Simul (and burn in : 0) : 103
                Control :
                     Cvgce of Dichotomy (1 if OK) : 1
                     -(GradLnW) init : 32282.000000
                     -(GradLnW) end : 31029.916931
                     -(MeanH) under Q : 31029.916931
```

Table 3:

Table 3: `Step 2` consists in updating the parameters. The update of the parameter β is done by running a MCMC algorithm. We thus indicate the number of initial samples that are discarded in the MCMC path (burn in) as well as the total length of the Markov Chain. This simulation step is coupled with an optimization step that consists in finding the root of a gradient by dichotomy. We give a binary indicator of convergence of the iterative optimization procedure, the value of the gradient when starting the optimization procedure and the final value of the gradient when stopping the optimization procedure.

Table 4: `Step 3` is the segmentation step. We display the number of pixels the class of which vary between two consecutive iterations, and the number of misclassified samples. The percentage is in parenthesis.

Four figure windows are opened :

Figure 1: At each iteration, we plot on `Figure 1` the observation, the initial segmentation, the current segmentation and the "true" image when available.

Figure 2: At each iteration, we plot on `Figure 2` [top] Follow the iterative procedure that solves the fixed point equation verified by the product distribution $Q(z) = \prod_k q_k(z)$: at each pixel k, $q_k(z)$ is a distribution on $1, \cdots, G$. We observe the evolution of $q_k(1)$ along this iterative

Step 3 :	Segmentation
	Variation in the segmentation : 1583 / 18886 (8.382)
	Nbr errors in segmentation : 3528 / 18886 (18.681)

Table 4:

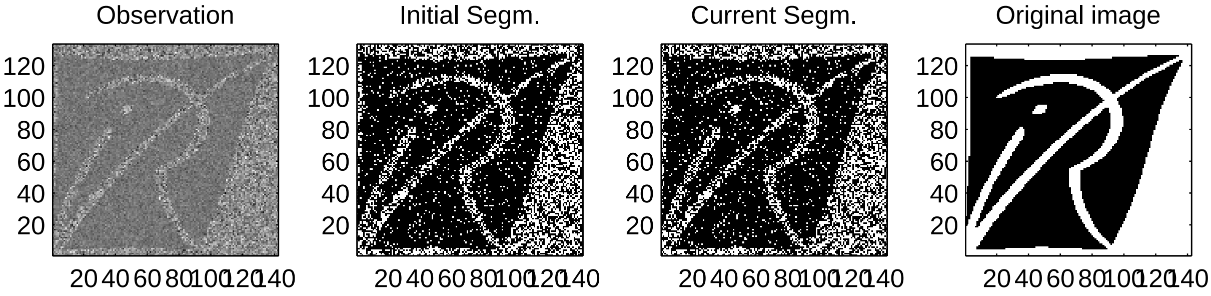

Figure 1:

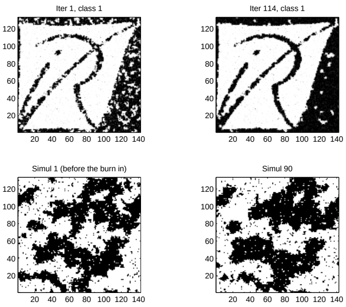

Figure 2:

procedure. [bottom] Follow the path of the Markov chain, produced at each iteration of MCVEM.

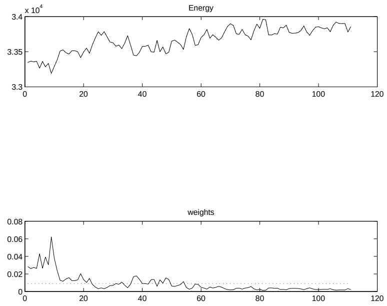

Figure 3:

Figure 3: At each iteration, we plot on `Figure 3` different controls of the β-parameter update. [top] energy of each Markov chain sample [bottom] weights in the computation of $\nabla(\log \tilde{W})$ (in red dots, the uniform weights).

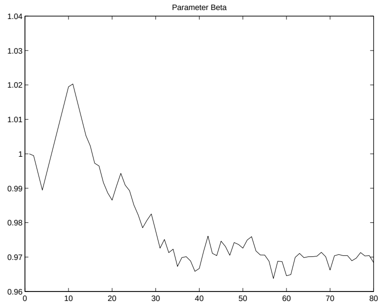

Figure 4:

Figure 4: We plot on `Figure 4` the evolution of the parameter β as a function of the iterations of MCVEM.

2.3.2 Output file

A `.mat` file is created. It contains different quantities computed when running the algorithm (see Section 3 for the description of the algorithm and the possible different values for the following variables).

G**:** the number of classes for the segmentation.

NbrNeigh**:** the number of neighbors for each pixel (4 or 8).

HatZ**:** the segmented picture.

Mu**:** a matrix that contains the estimated means at each iteration of the algorithm.

Sigma2**:** a matrix that contains the estimated variances at each iteration of the algorithm.

Beta**:** a vector that contains the estimates of β-parameter at each iteration of the algorithm.

HQdist**:** at each iteration of the algorithm, the expectation of $\ln \pi(X; \beta)$ is computed for the current value of the parameter β. This vector contains the value of this expectation at each iteration of the algorithm.

DeltaSegm**:** the number of pixels that differ between two consecutive segmentations.

MethSimul**:** the Monte-Carlo sampler.

InitChainMethod**:** the way in which the Markov Chains are initialized at each iteration of the algorithm.

BurnIn**:** the number of samples that are discarded when computing the Monte-Carlo sum.

NbrSimulMin**:** the minimal length of the Monte-Carlo sum.

DiagCvgQ**:** At each iteration of the algorithm, a mean-field equation is solved. This is done by an iterative procedure. This binary-valued vector contains the diagnostic of convergence of this procedure, at each iteration (1 if convergence).

NbrIterQ**:** a vector that contains, for each iteration of the algorithm, the number of iterations for solving the mean-field equation.

DiagCvgNR**:** At each iteration of the algorithm, an optimization procedure is run. This is done iteratively. This binary-valued vector contains the diagnostic of convergence of this procedure, at each iteration (1 if convergence).

Initialization**:** Initial segmentation when available. Otherwise, the initial segmentation is obtained by a `k-means` procedure applied to the observed picture.

2.4 Specify your own input

2.4.1 When running the function `MCVEM.m`

MCVEM(Observation,Original,Initialization,NbrNeigh,NbrClass,NameFile)

See `ExampleMCVEM.m` for an example.

Observation : The observed image (matrix).

Original : The “true” image when available (matrix). `[ ]` otherwise.

Initialization : The initial segmentation when available (matrix). `[ ]` otherwise.

NbrNeigh : Either `’4-Neigh’` or `’8-Neigh’`. Specify the number of neighbors for each pixel.

NbrClass : The number of class for segmentation.

NameFile : The name of the output file.

2.4.2 Through the file `CodesMCVEM/InterfaceMCVEM.m`

There are different options when running the algorithm. These options are described in Section 3.

3 Technical Appendix

3.1 Model

The distribution of the hidden field is a K-color Potts model

$$p_Z(z;\beta) = W(\beta)^{-1}\ \exp(\beta \sum_{i\sim j} z_i^t z_j) = W(\beta)^{-1}\ \exp(0.5\beta \sum_{i=1}^{N} \sum_{j\in\mathcal{N}_i} z_i^t z_j) = \exp(-\ln W(\beta) - H(z;\beta))$$

where $W(\beta)$ is the normalizing factor and

$$H(z;\beta) = -\beta \sum_{i\sim j} z_i^t z_j.$$

The conditional distribution of the observations $\{y_i\}_{1\le i\le N}$ given the missing data $\{z_i\}_{1\le i\le N}$ is a factorized distribution :

$$\prod_{i=1}^{N} \mathcal{N}(\mu_{z_i}, \sigma_{z_i}^2)(y_i), \qquad\qquad y_i \in \mathbb{R}, z_i \in V.$$

3.2 Algorithm

Step 1 Initialization of the algorithm

Step 1-a Initialize the β-parameter.

Step 1-b Initialize the θ-parameter (means and variances).

Step 2 While non-converged

Step 2-a Update the q-component.

Step 2-b Update the θ-component (means and variances).

Step 2-c Update the β-component.

Step 2-c-i Sample a Markov Chain.

Step 2-c-ii Compute the maximum of the approximated gradient.

3.3 Step 2-a

3.3.1 Description : $(q^t, \Psi^t) \to q^{t+1}$

Find $\{q_k(e)\}_{1\le k\le N, e\in V}$ such that $\sum_{e\in V} q_k(e) = 1$ and

$$q_k(e) = \frac{\alpha_k}{\sigma_e} \exp\left(-0.5\left\{\frac{(y_k-\mu_e)^2}{\sigma_e^2} - 2\beta e^t \sum_{j\in\mathcal{N}_{z_k}} \sum_{z\in V} z q_j(z)\right\}\right).$$

α_k is the normalizing constant. This is of the form

$$q_k(e) = \mathcal{F}\left(\{q_k(e)\}_{1\le k\le N, e\in V}\right),$$

where $\mathcal{F}$ depends upon the current value of the parameters Ψ^t. This fixed point equation is solved iteratively.

Initialization : $q_k^{(0)} = \{q_k^{(0)}(e)\}_{1\le k\le N, e\in V}$.

While *stop* = *False*

$$q_k^{(n+1)} = \mathcal{F}\left(\{q_k^{(n)}(e)\}_{1\leq k\leq N, e\in V}\right).$$

End while.

Set $q^{t+1} = q^{(n+1)}$.

3.3.2 Parameters

InitQdistMethod**:** describes the initialization procedure.

- either `' AllTheSame'` i.e. at each loop of MCVEM, the initial point $q_k^{(0)} = \{q_k^{(0)}(e)\}_{1\leq k\leq N, e\in V}$ is the same and is defined as the initial segmentation if provided by the user, or by `k-means` otherwise.
- or `'LastIn'` i.e. at loop $(t+1)$ of MCVEM, the initial point $q_k^{(0)} = \{q_k^{(0)}(e)\}_{1\leq k\leq N, e\in V}$ is the last point $q_k^{(n)} = \{q_k^{(t)}(e)\}_{1\leq k\leq N, e\in V}$ obtained at loop t. For the first loop, $q_k^{(0)} = \{q_k^{(0)}(e)\}_{1\leq k\leq N, e\in V}$ is defined as the initial segmentation if provided by the user, or by `k-means` otherwise.

TolQdist, NbrIterMaxQdist**:** defines the stopping criterion. The iterative procedure is stopped when the maximal number of iterations `NbrIterMaxQdist` is reached, or when the total variation of the updated vector $\{q_k(e)\}_{1\leq k\leq N, e\in V}$ is lower that a threshold `TolQdist`. If the iteration stops because `NbrIterMaxQdist` is reaches, the procedure is non-converging and `DiagCvgQ` is set to $'0'$. Otherwise, it is set to $'1'$.

3.4 Step 2-b

3.4.1 Description $q^{t+1} \rightarrow \theta^{t+1}$

θ contains the means $\{\mu_k\}_{1\leq k\leq G}$ and variances $\{\sigma_k^2\}_{1\leq k\leq G}$ for the Gaussian distributions. At loop $(t+1)$ of MCVEM, given q^{t+1}, the parameter is updated by the explicit formula

$$\mu_k^{t+1} = \frac{\sum_{i\in S} y_i q_i^{t+1}(e_k)}{\sum_{i\in S} q_i^{t+1}(e_k)}, \qquad 1\leq k\leq G,$$

and

$$(\sigma_k^2)^{t+1} = \frac{\sum_{i\in S}(y_i - \mu_k^{t+1})^2 q_i^{t+1}(e_k)}{\sum_{i\in S} q_i^{t+1}(e_k)}, \qquad 1\leq k\leq G.$$

3.4.2 Initialization θ^0

The parameters are updated as the empirical means and variances of the classes in the initial segmentation. This initial segmentation is either provided by the user or obtained by `k-means`. The classes are sorted so that the means are in the ascending order : $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_G$.

3.5 Step 2-c

3.5.1 Description $(q^{t+1}, \beta^t) \rightarrow \beta^{t+1}$

Sample a Markov Chain $\{Z^{j,t}\}_{1\leq j\leq J_t}$ with invariant distribution π^t.

Calculate

$$\tilde{W}^{J_t,\pi^t}(\beta) = \frac{1}{J_t} \sum_{j=1}^{J_t} \exp\left(-H(Z^{j,t};\beta) - \ln \pi^t(Z^{j,t})\right).$$

Set

$$\beta^{t+1} = \mathrm{argmax}_{\beta \in \{b, |b-\beta^t| \leq \gamma^t\}} - \left\{ \sum_{z \in \mathcal{Z}} H(z;\beta) q^{t+1}(z) + \ln \tilde{W}^{J_t,\pi^t}(\beta) \right\}.$$

3.5.2 Sample a Markov Chain with target distribution π^t

InitChainMethod **:** defines the initial value of the chain by one of the following policy

- `'LastIn'` set $Z^{0,t+1} = Z^{J_{t-1},t}$.
- `'AllTheSame'` set $Z^{0,t+1} = Z^{0,1}$ where $Z^{0,1}$ is obtained at random.
- `'AtRandom'` choose $Z^{0,t+1}$ at random.

BurnIn **:** number of samples that are discarded when computing $\tilde{W}^{J_t,\pi^t}(\beta)$. Set it to $'0'$ if the variable `InitChainMethod` is set to `'LastIn'`.

NbrSimulMin **:** is the minimal length of the Markov chain.

NbrSimul **:** is the sequence $\{J_t\}_t$ of the number of simulations at each MCVEM iteration. Choose it as a polynomially increasing sequence.

MethSimul **:** Specifies the MCMC sampler and the invariant target distribution π^t

- `'PriorGibbsSyst'` Run a Gibbs sampler with deterministic scan (columns and rows are visited according to the lexicographical order). The invariant ditribution is the Potts distribution for the current value of the parameter : $\pi^t(\cdot) = p_Z(\cdot;\beta^t)$.
- `'PriorGibbsNonSyst'` Run a Gibbs sampler with deterministic scan (columns and rows are visited in staggered rows). The invariant ditribution is the Potts distribution for the current value of the parameter : $\pi^t(\cdot) = p_Z(\cdot;\beta^t)$.
- `'PriorGibbsRandom'` Run a Gibbs sampler with random scan : 50% of the picture is updated and the pixels are chosen at random. The invariant ditribution is the Potts distribution for the current value of the parameter : $\pi^t(\cdot) = p_Z(\cdot;\beta^t)$. This option is not available when the variable `NbrNeigh` is set to `'8-Neigh'`.
- `'PriorGibbsRandScan'` Run a Gibbs sampler with invariant ditribution is the Potts distribution for the current value of the parameter : $\pi^t(\cdot) = p_Z(\cdot;\beta^t)$. All the pixels are updated, but they are visited in a random order. This option is not available when the variable `NbrNeigh` is set to `'8-Neigh'`.

3.5.3 Optimization procedure

Gamma **:** Sequence $\{\gamma^t\}_t$ for the definition of the optimization domain. Can be a constant (if too large, the sequence $\{\beta^t\}_t$ may be periodic; in that case, re-start the algorithm with a smaller value of `Gamma`.