Master the Usage of T-MSBL in 3 Minutes

Zhilin Zhang (z4zhang@ucsd.edu)
University of California, San Diego
Nov. 11, 2011

0. What is T-MSBL?

The algorithm was developed for the multiple measurement vector (MMV) model, i.e.

$$Y = Phi\ X + V,$$

where *Phi* is the known dictionary matrix. T-MSBL exploits the correlation exists in each nonzero row of X (namely, temporal correlation). For details, see Ref.[1].

1. When Shall I use T-MSBL?

- ✓ **Based on our knowdge, it is the only algorithm performs well when the matrix *Phi* is highly coherent (see:http://dsp.ucsd.edu/~zhilin/papers/comparison.pdf).**

- ✓ **MMV model with temporally correlated source vectors**
  By extensive experiments, we found T-MSBL has the best recovery performance among existing algorithms for this model, especially when the inverse problem is more difficult, such as highly under-determinacy ratio, more nonzero rows, and/or low SNR (6-15dB)

- ✓ **Single measurement vector (SMV) model**
  T-MSBL can be used for SMV models. Of course, in this case there is no temporal information can be exploited. However, due to an effective learning rule to automatically choose the optimal regularization value, it has better performance than most compressed sensing algorithms for this model, especially when SNR is around 6-15dB.

- ✓ **Compressed sensing for cluster-structured signals (in the SMV model)**
  Although T-MSBL is not for compressed sensing of cluster-structured signals, it shows better performance than many algorithms for cluster-structured signals.

- ✓ **Time-varying sparsity model, or dynamic compressed sensing model**
  The support of each source vector X(:,i) is slowly time-varying, we can use the concatenate of MMV models to approximate this scenario. See Ref [2] and the demo file: `demo_time_varying.m`. By extensive experiments, it shows better performance than most existing algorithms for this model.

- ✓ **MMV model with identical source vectors**
  This model sometimes appears in the theoretical work or is combined with Kalman filtering model (random walk model). See the demo file `demo_identicalVector.m` for the usage.

2. The most convenient way to use T-MSBL

- You can simply use the command to call T-MSBL for most cases (MMV model with temporal correlated source vectors, when SNR varies from 6 dB to 22 dB)

```
X_est = TMSBL(Phi, Y);
```

- In most cases you can roughly know or estimate the noise level (of course you need not to know the exact value). For example, when solving practical problems, you may say: "the noise is large/mild/small". In this case, you can choose these commands for a better performance:

  - When noise is large (e.g. SNR <=6 dB)

    ```
    X_est = TMSBL(Phi, Y, 'noise', 'large')
    ```

  - When noise is mild (e.g. 6 dB <= SNR <=22 dB)

    ```
    X_est = TMSBL(Phi, Y, 'noise', 'mild'), or X_est = TMSBL(Phi, Y);
    ```

  - When noise is small (e.g. SNR >22 dB)

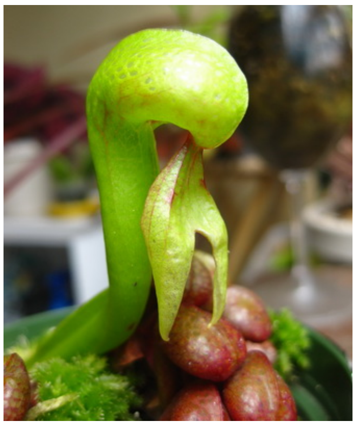
    ```
    X_est = TMSBL(Phi, Y, 'noise', 'small')
    ```

  - When no noise

    ```
    X_est = TMSBL(Phi, Y, 'noise', 'no')
    ```

Note: The above five commands uses a set of pre-defined parameter values, which work well for most standard compressed sensing experiment settings. When your problem is not a standard compressed sensing problem (e.g. signals are not sparse, the matrix Phi is highly coherent), you need to set some specific parameters' values (see the next page).

If you encounter any problems when using it, please feel free to contact me (z4zhang@ucsd.edu).



The following is a detailed instruction on how to tune parameters (for those who are familiar with SBL and want better performance in their specific problems).

First, let's see the full command to call TMSBL:

```
[X, gamma_ind, gamma_est, count, B_est] = TMSBL(Phi, Y, 'Prune_Gamma', 1e-5, 'Learn_Lambda',1, 'Enhance_Lambda',1, 'Matrix_Reg', 2*eye(L), 'lambda', 0.015, 'MAX_ITERS', 500, 'Fix_B', eye(L), 'EPSILON', 1e-8, 'PRINT', 1);
```

You can see the m-file for descriptions about these parameters. When tuning these parameters, you only need to pay attention to four input parameters, indicated by red color. Fortunately, tuning them is not a tough job. See the following golden rules.

3. Four golden rules when you tune parameters

- ### 'Prune_Gamma': Larger noise level means larger threshold to prune out small γi

  Theoretically, when the i-th row of X is zero, γi should also be zero. However, due to noise, γi won't be zero. Larger noise means larger γi. So if your threshold is too small, there would be many nonzero γi that should be pruned out but are not. So you need to adjust the threshold according to the noise level. But you need not to know exactly the noise level. Empirically,
  - At strongly noisy cases: e.g. set 'prune_gamma' = 1e-2;
  - At most noisy cases: e.g. set 'prune_gamma' = 1e-3.
  - At noiseless cases: set it to any small number (e.g. 1e-4) .
  - When solving a noisy large scale problem (e.g. Phi's size is thousands by thousands, say. 2000 x 5000), set 'prune_gamma' = 1e-2.
  - When signals are not sparse, set a smaller 'prune_gamma' value in any cases, e.g. 'prune_gamma'=1e-3.

- ### 'Learn_Lambda': In most noisy cases, use the λ learning rule (i.e. set 'Learn_Lambda'=1). In noiseless case, you do not need to use the λ rule (i.e. set 'Learn_Lambda'=0), and also set 'Lambda' to a very small value, e.g. 1e-15.

- ### 'Enhance_Lambda': In mildly/strongly noisy case (e.g. SNR <= 22 dB) use the modified λ learning rule (set 'Enhance_Lambda' = 1). When noise is small, set 'Enhance_Lambda' =0. When the matrix *Phi* is highly coherent, set 'Enhance_Lambda'=1.

- ### 'Matrix_Reg': In mildly/strongly noisy cases (<22dB), regulate the estimated covariance matrix B (set 'Matrix_Reg' = c*eye(L)).

  Note that in the development of T-MSBL, we used an approximation. The approximation is exact only when there is no noise or there is no temporal correlation. This approximation requires regulating the estimated covariance matrix B in each iteration in noisy cases. The higher noisy level requires stronger regularization, i.e. larger c in c*eye(L). For example,

- In mildly noisy case (7-22dB): c = 2
- In strongly noisy case (<7 dB): c = 4

4. Some notes

- **The default value of 'MAX_ITERS' is 2000. In some cases (e.g. large-scale data set, strongly noisy cases), 2000 iterations may not be enough or too many.**

- **The argument 'MAX_ITERS', 'EPSILON', and 'PRUNE_GAMMA' all determine the speed. For fast speed, you must tune these parameters for a good tradeoff between performance and speed.**

5. Case Study for small or medium scale problems

Note: You can set the input argument 'print' = 1, so you can check the parameters you used.

Case Study 1: Noiseless Cases

- The default value of 'prune_gamma' is 1e-3. In noiseless cases, you can use smaller values, e.g. 1e-4.
- No need to use the λ learning rule ('learn_lambda' = 0). Freeze 'lambda' to a very small value, eg. 1e-14 ('lambda' = 1e-14)
- No need to regulate B, so set 'matrix_reg'= zeros(L) (note: **the default value is 2*eye(L) , L is the number of measurement vectors**).

[Example] `X_est = TMSBL ( Phi, Y, 'prune_gamma',1e-4, 'lambda',1e-14, 'learn_lambda', 0, 'matrix_reg', zeros(L) );`

Case Study 2: Small Noise Level (>22 dB)

- Use the default value for 'prune_gamma' (1e-3)
- Use the λ learning rule ('learn_lambda' = 1), but not use the enhance strategy ('enhance_lambda' = 0).
- No need to regulate B since the noise is very small ('matrix_reg'= zeros(L))

[Example] `X_est = TMSBL ( Phi, Y, 'prune_gamma' ,1e-3, 'learn_lambda', 1, 'enhance_lambda', 0, 'matrix_reg', zeros(L) );`

Case Study 3: Mild Noise Levels (6-22 dB)

- The default values in TMSBL aim to this scenario
- Use the default value for 'prune_gamma' (1e-3)
- Use the λ learning rule ('learn_lambda' = 1), AND use the enhance strategy ('enhance_lambda' = 1).
- Regulate the estimate of B: 'matrix_reg'= c * eye(L), where c can be chosen from 1 to 3

[Example] `X_est = TMSBL ( Phi, Y, 'learn_lambda', 1, 'enhance_lambda', 1, 'matrix_reg', 2.5 *eye(L) );`

Case Study 3: Large Noise Levels (e.g. SNR < 6 dB)

- Use the default value for 'prune_gamma' (1e-3) or a larger one (e.g. 1e-2)

- I do not suggest using the λ learning rules. But if you use them, the algorithm may still obtain better performance than many algorithms; thus set 'learn_lambda' = 1, 'enhance_lambda' = 1
- If you do not use the λ learning rule, then estimate a good value for λ and freeze it. In many cases, a good value is about **2-4** times of the true noise variance (note: for another algorithm, T-SBL, a good value is around the true noise variance).
- Use a stronger regularization to B, e.g. : 'matrix_reg'= c * eye(L), where c > 3

[Example] (Assume the estimated noise variance is 0.02)

```
X_est = TMSBL ( Phi, Y,  'prune_gamma' , 1e-2,  'lambda', 0.06,  'learn_lambda', 0,  'matrix_reg', 4*eye(L) );
```

Case Study 4: Source Vectors are Identical (i.e. Temporal Correlation is exactly 1)

- Basically, T-MSBL is suitable for the cases when the temporal correlation is large **but not exactly 1**. When the correlation is 1, the learning of B could have problems. However, you can freeze B to an identical matrix ('Fix_B' = eye(L)). Although the resulting algorithm is basically the same as MSBL, the performance is still better than MSBL due to the superiority of the λ learning rule in T-MSBL.
- When there is no temporal correlation **AND** the noisy is very large, you can also set 'Fix_B' = eye(L) and other suitable input argument values to run T-MSBL.
- See the demo file demo_identicalVector.m

[Example] (Assume a mildly noisy case)

```
X_est = TMSBL ( Phi, Y,  'noise', 'mild',  'fix_B', eye(L) );            % use the default values in the code
X_est = TMSBL ( Phi, Y,  'prune_gamma', 1e-4,  'learn_lambda', 1,  'enhance_lambda', 1,  'fix_B', eye(L) );
```

Case Study 5: Time-Varying Sparsity Model (Dynamic Compressed Sensing)

- Set a suitable sliding time-window (i.e. choosing L, which determines your input argument Y), in which the data can be approximately modeled as an MMV model.
- Then according to previous examples, choose suitable input arguments.
- See Ref [2] and the demo file: demo_time_varying.m.

References

[1] Zhilin Zhang, Bhaskar D. Rao, Sparse Signal Recovery with Temporally Correlated Source Vectors Using Sparse Bayesian Learning, IEEE Journal of Selected Topics in Signal Processing, vol. 5, no. 5, pp. 912-926, 2011

[2] Zhilin Zhang, Bhaskar D. Rao, Exploiting Correlation in Sparse Signal Recovery Problems: Multiple Measurement Vectors, Block Sparsity, and Time-Varying Sparsity, [online] http://arxiv.org/abs/1105.0725

[3] Zhilin Zhang, Bhaskar D. Rao, Clarify Some Issues on the Sparse Bayesian Learning for Sparse Signal Recovery, Technical Report, 2011

Feel free to contact me if you have any questions (send email to z4zhang@ucsd.edu)