NASA Public Access

Author manuscript

Conf Rec Asilomar Conf Signals Syst Comput. Author manuscript; available in PMC 2018 July 09.

Published in final edited form as:

Conf Rec Asilomar Conf Signals Syst Comput. 2018 April 16; 51: 858–862. doi:10.1109/ACSSC.

SPARSE BAYESIAN LEARNING USING VARIATIONAL BAYES INFERENCE BASED ON A GREEDY-BASED CRITERION

Mohammad Shekaramiz, Todd K. Moon, and Jacob H. Gunther

Elec. and Comp. Engr. Dept. and Information Dynamics Laboratory, Utah State University

Extended Abstract

Compressive sensing (CS) is an evolving area in signal acquisition and reconstruction with many applications [1–3]. In CS the goal is to efficiently measure and then reconstruct the signal under the assumption that such signal is sparse but the number and location of non-zeros are unknown. A linear CS problem is modeled as $\mathbf{y} = A\mathbf{x}_s + \mathbf{e}$, where $\mathbf{y} \in \mathbb{R}^M$ contains measurements, $\mathbf{x}_s \in \mathbb{R}^N$ is the sparse solution, and $\mathbf{e}$ is the noise with $M \ll N$ [4–6]. $A = \Phi\Psi$, where Φ is the sensing matrix and Ψ is a proper basis in which $\mathbf{x}_s$ is sparse. There are three approaches to solve for $\mathbf{x}_s$, i.e., greedy-based, convex-based, and sparse Bayesian learning (SBL) algorithms. Here, we consider the SBL approach. Specifically, we consider Gaussian-Bernoulli prior to promote sparsity in the solution and then use variational Bayes (VB) inference to estimate the variables and parameters of the model. In the Gaussian-Bernoulli model, the sparse solution is defined as $\mathbf{x}_s = (\mathbf{s} \circ \mathbf{x})$, where $\mathbf{s}$ is a binary support learning vector, $\mathbf{x}$ accounts for the values of the solution, and “ $\circ$ ” is the element-wise product [7, 8].

It turns out that using VB inference for CS problem has the over fitting issue mainly when the number of measurements are low. For example, for the CluSS-VB algorithm, Yu et al. [8] pointed out that the solution may tend to become non-sparse. In this work, we propose a VB-based SBL algorithm which uses a simple criterion to remove such effect and forces the solution to become sparse. We also discuss and compare the update rules obtained from the SBL using fully hierarchical Bayesian approach via Markov chain Monte Carlo (MCMC) [7], expectation-maximization (EM) algorithm, and the VB inference. As expected, there exist a very close relationship between all these algorithms and we provide some intuition on how to turn equations of one approach to another approach. Also, we will provide some simulation results to compare the performance of such algorithms for the CS problems.

G-OSBL (VB): An SBL algorithm using VB

Suppose there is a model with parameters Θ , hidden variables collected in $\mathbf{x}$, and a set of observations denoted by $\mathbf{y}$. Then, the approximation to the joint density $p(\mathbf{x}, \Theta | \mathbf{y})$ can be represented by $p(\mathbf{x}, \Theta | \mathbf{y}) \approx q_{\mathbf{x}}(\mathbf{x})q_{\Theta}(\Theta)$. Then, the lower bound on the model log marginal likelihood can be iteratively optimized by the following updates [9]

$$q_x^{[t+1]}(\mathbf{x}) \propto \exp \{E_{q_\theta^{[t]}}[\log p(\mathbf{x}, \mathbf{y} | \Theta)]\} \quad (1)$$

$$q_\Theta^{[t+1]}(\Theta) \propto p(\Theta) \exp \{E_{q_x^{[t]}}[\log p(\mathbf{x}, \mathbf{y} | \Theta)]\}. \quad (2)$$

For the inverse CS problem, one can define a set of priors as follows [7]. We model the elements of the binary vector “ $\mathbf{s}$ ” as Bernoulli random variables, whose probabilities are governed by the prior γ , that is,

$$s_n \sim \text{Bernoulli}(\gamma_n), \gamma_n \sim \text{Beta}(\alpha_0, \beta_0), n = 1, \dots, N. \quad (3)$$

In order to promote sparsity structure in $\mathbf{s}$, we set $\alpha_0 \ll \beta_0$. The solution-value vector components are assumed to be distributed as *i.i.d.* normal-gamma distribution

$$\mathbf{x} \sim \mathcal{N}(0, \tau^{-1} I_N), \tau \sim \text{Gamma}(a_0, b_0), \quad (4)$$

where a_0 and b_0 denote the shape and rate of the Gamma distribution, respectively. The entries of the noise component $\mathbf{e}$ are assumed outcomes of an *i.i.d.* Gaussian distribution with the precision ε .

$$\mathbf{e} \sim \mathcal{N}(0, \varepsilon^{-1} I_M), \varepsilon \sim \text{Gamma}(\theta_0, \theta_1). \quad (5)$$

Using the VB algorithm defined in (1) and (2), the update rule of the variables and parameters of the model can be simplified as follows.

- Update rule for the support learning vector $\mathbf{s}$

$$\left(s_n | -\right) \sim \text{Bernoulli}\left(Q_{s_n}\right), Q_{s_n} = \frac{1}{1 + c_n \kappa_n},$$

and therefore

$$\tilde{s}_n = Q_{s_n}, \forall n = 1, \dots, N, \quad (6)$$

where

$$c_n := e^{\psi(\beta_{1,n}) - \psi(\alpha_{1,n})},$$

$$\kappa_n := e^{\frac{1}{2}\tilde{\varepsilon}\left(\|\mathbf{a}_n\|_2^2(\tilde{\alpha}_n^2 + \sigma_{\tilde{\alpha}_n}^2) - 2\tilde{\alpha}_n\mathbf{a}_n^T <\tilde{\mathbf{y}}^{-n}>_x\right)},$$

$$\text{and } <\tilde{\mathbf{y}}_m^{-n}>_x = y_m - \sum_{l \neq n}^N a_{ml} \tilde{s}_l \tilde{x}_l.$$

- Update rule for the solution value matrix $\mathbf{x}$

$$(\mathbf{x} \mid -) \sim \mathcal{N}(\tilde{\mathbf{x}}, \tilde{\Sigma}_{\tilde{\mathbf{x}}}), \text{ where} \quad (7)$$

$$\tilde{\Sigma}_{\tilde{\mathbf{x}}} = (\tilde{\tau} I_N + \tilde{\varepsilon} \tilde{\Phi})^{-1} \text{ and } \tilde{\mathbf{x}} = \tilde{\varepsilon} \tilde{\Sigma}_{\tilde{\mathbf{x}}} \text{diag}(\tilde{\mathbf{s}}) A^T \mathbf{y},$$

$$\text{where } \tilde{\Phi} := [(A^T A) \circ (\tilde{\mathbf{s}} \tilde{\mathbf{s}}^T + \text{diag}(\tilde{\mathbf{s}} \circ (1 - \tilde{\mathbf{s}})))].$$

- Update rule for γ_n

$$(\gamma_n \mid -) \sim \text{Beta}(\alpha_{1,n}, \beta_{1,n}).$$

Therefore,

$$\tilde{\gamma}_n = \frac{\alpha_{1,n}}{\beta_{1,n}}, \quad \forall n = 1, \dots, N, \quad (8)$$

$$\text{where } \alpha_1 := \alpha_0 + \tilde{s}_n \text{ and } \beta_1 := \beta_0 + 1 - \tilde{s}_n.$$

- Update rule for the noise precision

$$\tilde{\varepsilon} = \frac{\theta_0 + \frac{M}{2}}{\theta_1 + \frac{1}{2}\tilde{\Psi}}, \quad (9)$$

where

$$\tilde{\Psi} := (\mathbf{y}^T \mathbf{y} - 2(\tilde{\mathbf{x}} \circ \tilde{\mathbf{s}})^T A^T \mathbf{y} + \text{tr}((\tilde{\mathbf{x}} \tilde{\mathbf{x}} + \tilde{\Sigma}_{\tilde{\mathbf{x}}}) \tilde{\Phi})).$$

- Update rule for the solution precision

$$(\tau \mid -) \sim \text{Gamma}\left(a_0 + \frac{N}{2}, b_0 + \frac{1}{2}[\|\tilde{\mathbf{x}}\|_2^2 + \text{tr}(\tilde{\Sigma}_{\tilde{\mathbf{x}}})]\right),$$

where $\Sigma_{\tilde{x}} = \text{diag}(\sigma_{\tilde{x}_1}^2, \dots, \sigma_{\tilde{x}_N}^2)$. Therefore,

$$\tilde{\tau} = \frac{a_0 + \frac{N}{2}}{b_0 + \frac{1}{2}[\|\tilde{\mathbf{x}}\|_2^2 + \text{tr}(\Sigma_{\tilde{x}})]}. \quad (10)$$

The stopping criterion of the algorithm can be then made based on the log-marginalized likelihood defined as $L := \log \{p(\mathbf{y}|\mathbf{x}, \mathbf{s}, \boldsymbol{\varepsilon}, \boldsymbol{\tau})\}$. The marginalized likelihood is defined as $p(\mathbf{y}|\mathbf{x}, \mathbf{s}, \boldsymbol{\varepsilon}, \boldsymbol{\tau}) = \int p(\mathbf{y}|\mathbf{x}, \mathbf{s}, \boldsymbol{\varepsilon})p(\mathbf{x}|\boldsymbol{\tau})d\mathbf{x}$. After some simplification, the negative log-likelihood will be proportional to $-L \propto \log |\Sigma^{-1}| + \mathbf{y}^T \Sigma \mathbf{y}$, where $\Sigma = (\boldsymbol{\varepsilon}^{-1} I_M + \boldsymbol{\tau}^{-1} A \tilde{\mathbf{S}}^2 A^T)^{-1}$, and $\tilde{\mathbf{S}} := \text{diag}(\tilde{\mathbf{s}})$. Therefore, the stopping condition can be defined as [8]

$$\frac{|L^{[t+1]} - L^{[t]}|}{|L^{[t]}|} \leq \text{Threshold},$$

for some small value of “Threshold”.

Although for a large number of measurements the above algorithm has a good performance, it turns out that when the number of measurements decreases, the solution becomes very non-sparse in terms of support recovery, while the estimated solution vector $\tilde{\mathbf{x}}_s$ becomes almost zero and noninformative. We illustrate this over fitting issue via some examples in Fig. 1. In the simulations, we generated a trial, where the true solution vector is $\mathbf{x} \in \mathbf{R}^{100}$ with 25 non-zero elements. The supports of the true solution are randomly drawn so that the support set of the solution exhibits a clustered pattern. The non-zero values of the solution are drawn *i.i.d.* from a zero-mean Gaussian distribution with variance of 1. Each entry of the sensing matrix A is an outcome of a Gaussian distribution $\mathcal{N}(0, 1)$. After constructing A , we normalize A with respect to its columns. In Fig. 1, we show the performance for the cases where the number of measurements is set to 85, 70, and 65, respectively. The elements of measurement noise are drawn *i.i.d.* from $\mathcal{N}(0, \sigma^2)$ with SNR=25 dB, where $\text{SNR} := 20 \log_{10}(\sigma_x/\sigma)$.

Table Tab. 1 shows the obtained detection rate, false alarm rate, and the normalized mean-squared error (NMSE) between the true and the estimated solution.

As can be seen from Fig. 1 and Tab. 1, the SBL using VB fails to estimate the solution once the sampling ratio decreases. The reason is essentially due to the overfitting problem since there are many parameters to be learned via low number of measurements and the update rule of the support learning vector $\mathbf{s}$ is obtained from a soft thresholding rather than the outcome a Bernoulli distribution for the posterior distribution found in the ordinary sparse Bayesian learning (SBL) approach. Due to the overfitting issue, the estimated precision on the components of the solution increased in our simulations, and as a result the energy of the estimated solution became almost negligible. Notice that, there is a chance to remove this issue by smartly selecting the hyper-parameters on the Gamma prior distribution which has a role on governing the distribution on the supports of the solution. But, when having no

information on the support set, even the setting of the hyper-parameters becomes challenging. The main issue of the failures can be found in the update rule of the support learning vector $\mathbf{s}$ defined in (6). It becomes important to balance between the terms c_n and κ_n , where c_n imposes the effect of hyperprior on $\mathbf{s}$ accompanied by the current estimate of s_n , and κ_n imposes the contribution of the current estimates of noise precision, solution, and other supports in fitting the model to the measurements. Therefore, if we impose a strong effect on the sparsity via c_n , then the solution tends to neglect the effect of κ_n and vice versa. To tackle this issue, we combine the intuition behind the SBL approach with the greedy algorithm of orthogonal basis pursuit (OMP). At each iteration, OMP makes decisions on the supports based on the usefulness defined as the normalized version of the inner product between the residual and the columns of the sensing matrix. The problem with the OMP is that it is sensitive to noise. Also, when the number of measurements become close to the dimension of the signal, the OMP solution becomes nonsparse unless some prior knowledge on the sparsity level of the signal is fed to the algorithm.

The main approach for solving the CS problem is as follows. We need to learn where the non-zero locations are and then what value the active locations possess. If we knew the active locations, then it would be easy to project the original problem to a lower dimension and estimate the sparse signal (e.g., with least-squares), and then based on the nonzero locations turn the obtained estimate of the signal to the original dimension. Suppose that there are K non-zero locations and the length of signal is N , where $K \ll N$. If the locations were known, then K measurements would suffice to solve the problem. Since we do not know the non-zero locations, we expect to have more than K measurements i.e., with M greater than K . Therefore, it is not possible to do sparse recovery with M measurements less than K i.e., K number of measurements is the lowest possible value that the algorithm needs to be able to perform exact sparse signal recovery in the ideal case. Since the sparsity level is unknown in almost all real compressive sensing problems, we avoid this issue by rejection of having more than M supports in the iterative estimates of $\mathbf{s}$, meaning that no matter how many more than M non-zero locations exist in the true solution, we put effort on finding only M because we do not have more resources to find more. Even for the case where the true number of supports is less than M , we would still let the algorithm decide on the M most probable supports and then let the algorithm figure out the contribution of each remained estimated support. In this case, the algorithm is given more chance to avoid the over fitting problem, since it will work on the lower dimension (M), where fewer parameters are left to be learned.

The proposed approach can be explained as follows. We first estimate the contribution of each location in fitting the model to the measurements using SBL-VB defined in (6). Then, we only keep those locations with the M highest contribution in the support set and discard the other estimated supports. In this way, we can remove the issue of non-sparse $\mathbf{s}$ based on our available resources. It is worth mentioning that in our proposed algorithm we first estimate the supports, then we perform the aforementioned post-processing on the supports, and then update the other variables and parameters of the model. In this way, we remove the misleading effect of changing the estimates of the supports after updating the other parameters and variables. The pseudocode of the proposed algorithm is as follows.

G-OSBL(VB) Algorithm:

 $[\hat{\mathbf{x}}, \hat{\mathbf{s}}] = \mathbf{G-OSBL}(Y, A, \text{Iter})$

iter= 0

$$\mathbf{While} \frac{|L^{[\text{Iter}]} - L^{[\text{Iter} - 1]}|}{|L^{[\text{Iter} - 1]}|} \geq 10^{-6}$$

% Support-learning vector component

Compute $\tilde{s}_n$ from (6), $\forall n = 1, \dots, N$ % Post processing on $\hat{\mathbf{s}}$ % Keep only the M supports with the highest contribution[-, Indx]=sort($\hat{\mathbf{s}}$, "Ascend") $\hat{\mathbf{s}}(1 : N - M) = 0$

% Solution-value matrix component

Compute $\Sigma_{\hat{\mathbf{x}}}$ and $\hat{\mathbf{x}}$ from (7)

% Parameters of mixing coefficients

Compute $\mathbf{a}_1$ and $\mathbf{\beta}_1$ from (8)

% Precision on the solution components

Compute τ from (10)

% Precision on the measurement noise

Compute e from (9)**End While**

In order to show the effect of the post filtering over the estimated $\mathbf{s}$, in Fig. 2 we illustrate the performance of GOSBL algorithm with the same $\mathbf{x}$, $\mathbf{s}$, $\mathbf{y}$, A , and noise that was used in Fig. 1 for the case of $M/N = 0.65$.

According to Fig. 2, the algorithm was able to avoid the overfitting problem and provided a good performance in sparse signal recovery. In Tab. 2 we show the obtained detection rate and false alarm rate, and the NMSE using G-OSBL(VB). Comparing Tab. 2 with Tab. 1, we see the improvement in the performance when using G-OSBL(VB).

O-SBL, O-SBL(EM), and O-SBL(VB) Duality

In this section we provide the update rules for the parameters of the O-SBL algorithm proposed in [7], the EM-based version of the O-SBL, and the O-SBL based on variational Bayes inference. In the original O-SBL [7], we used MCMC inference implemented via Gibbs sampler in order to approximate the posterior of the model variables and parameters. Tables Tab. 3–5 describe the obtained rules.

According to Tab. 3–5, there is a very close relationship between the update rules in all three aforementioned approaches for the posterior estimates. More specifically, the update rules of EM approach are essentially the same as the variational Bayes estimates, on average. In each update rule, both algorithms incorporate the variations of the estimates of the other related

parameters and variables, as well. Similarly, variations are considered in the Gibbs method due to the fact that the estimates of the other parameters and variables in each update rule are drawn from their corresponding approximated posteriors. For investigating the performance of all of these three approaches in CS, we will also apply these approaches to a set of simulated data.

Conclusion and Work In Progress

By combining variational Bayes inference and a greedy-based criterion, we proposed G-OSBL algorithm to perform sparse signal recovery in the compressive sensing problem. Initial simulations showed encouraging performance in detecting the supports and diminishing the over-fitting issue of the model parameters when having low number of measurements. In this case, we will provide a comprehensive study in comparing the overall performance of the algorithm compare to the other existing algorithm in the CS area. Furthermore, we showed the duality between a hierarchical SBL, the EM-based version, and variational Bayes inference approach algorithms. We will also include a performance comparison of the approaches on a simulated data.

References

1. Tibshirani R, Saunders M, Rosset S, Zhu J, Knight K. Sparsity and smoothness via the fused lasso. *J of the Royal Statistical Society Series B*. 2005; 67(1):91–108.
2. Hernandez-Lobato D, Hernandez-Lobato JM, Dupont P. Generalized spike-and-slab priors for Bayesian group feature selection using expectation propagation. *J Mach Learn Res*. 2013; 14(1): 1891–1945.
3. Fang J, Shen Y, Li H, Wang P. Pattern-coupled sparse Bayesian learning for recovery of block-sparse signals. *IEEE Trans Sig Proc*. 2015; 63(2):360–372.
4. Candes EJ, Romberg J, Tao T. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans Info Th*. 2006; 52(2):489–509.
5. Donoho DL. Compressed sensing. *IEEE Trans Info Th*. 2006; 52(4):1289–1306.
6. Candes EJ, Wakin MB. An introduction to compressive sampling. *IEEE Sig Proc Mag*. 2008; 25(2): 21–30.
7. Shekaramiz, M., Moon, TK., Gunther, JH. Hierarchical Bayesian approach for jointly-sparse solution of multiple-measurement vectors. 48th Asilomar Conf. of Sig., Syst., Compt; 2014:1962-1966
8. Yu L, Wei C, Jia J, Sun H. Compressive sensing for cluster structured sparse signals: Variational Bayes approach. *IET Sig Proc*. 2016; 10(7):770–779.
9. Beal, M. PhD dissertation University College London; 2003 Variational Algorithms for Approximate Bayesian Inference.

Figure 1.

From top row to the bottom row, we illustrate the results when the sampling ratio is 0.85, 0.7, and 0.65, respectively.

Figure 2.
Results of G-OSBL(VB) algorithm when the sampling ratio is 0.65.

Table 1

Investigating the performance of O-SBL(VB).

Sampling ratio	Detection rate	False alarm rate	NMSE (dB)
$M/N = 0.85$	0.8	0.00	-13.4015
$M/N = 0.7$	0.68	0.0133	-10.6168
$M/N = 0.65$	1.00	1.00	-0.1031

Table 2

Investigating the performance of G-OSBL(VB).

Sampling ratio	Detection rate	False alarm rate	NMSE (dB)
$M/N = 0.85$	0.84	0.00	-20.8901
$M/N = 0.7$	0.76	0.0133	-11.5602
$M/N = 0.65$	0.76	0.0133	-8.2291

Table 3

Comparing update rules of the noise precision.

O-SBL	$\varepsilon = \text{Gamma}(\theta_0 + \frac{M}{2}, \theta_1 + \frac{1}{2} \ \mathbf{y} - A(\mathbf{s} \circ \mathbf{x})\ _2^2)$
O-SBL(EM)	$\tilde{\varepsilon} = \frac{\theta_0 + \frac{M}{2} - 1}{\theta_1 + \text{trace}\left(\mathbf{y}\mathbf{y}^T - A(\tilde{\mathbf{s}} \circ \boldsymbol{\mu}_x)\mathbf{y}^T + \frac{1}{2} A(\tilde{\mathbf{s}} \circ (\boldsymbol{\Sigma}_x + \boldsymbol{\mu}_x \boldsymbol{\mu}_x^T))A^T\right)}$
O-SBL(VB)	$\varepsilon \sim \text{Gamma}(\theta_0 + \frac{M}{2}, \theta_1 + \frac{1}{2} \tilde{\Psi}), \quad \tilde{\varepsilon} = \frac{\theta_0 + \frac{M}{2}}{\theta_1 + \frac{1}{2} \tilde{\Psi}},$ $\tilde{\Psi} = \left(\mathbf{y}^T \mathbf{y} - 2(\tilde{\mathbf{x}} \circ \tilde{\mathbf{s}})^T A^T \mathbf{y} + \text{tr}((\tilde{\mathbf{x}} \tilde{\mathbf{x}} + \boldsymbol{\Sigma}_{\tilde{\mathbf{x}}}) \tilde{\Phi}) \right)$

Table 4

Comparing update rules of the solution precision.

O-SBL	$\tau = \text{Gamma}(a_0 + \frac{N}{2}, b_0 + \frac{1}{2}\ \mathbf{x}\ _2^2)$
O-SBL(EM)	$\tilde{\tau} = \frac{a_0 + \frac{N}{2} - 1}{b_0 + \frac{1}{2}\text{trace}(\Sigma_x + \mu_x \mu_x^T)}$
O-SBL(VB)	$\tau \sim \text{Gamma}\left(a_0 + \frac{N}{2}, b_0 + \frac{1}{2}(\ \tilde{\mathbf{x}}\ _2^2 + \text{trace}(\Sigma_{\tilde{\mathbf{x}}}))\right)$ $\tilde{\tau} = \frac{a_0 + \frac{N}{2}}{b_0 + \frac{1}{2}(\ \tilde{\mathbf{x}}\ _2^2 + \text{trace}(\Sigma_{\tilde{\mathbf{x}}}))}$

Table 5

Comparing update rules of the mixing coefficients.

O-SBL	$\gamma_n \sim \text{Beta}(a_0 + s_n, b_0 + 1 - s_n)$
O-SBL(EM)	$\tilde{\gamma}_n = \frac{\alpha_0 + \tilde{s}_n - 1}{\alpha_0 + \beta_0}$
O-SBL(VB)	$\gamma_n \sim \text{Beta}(a_0 + \tilde{s}_n, b_0 + 1 - \tilde{s}_n), \tilde{\gamma}_n = \frac{\alpha_0 + \tilde{s}_n}{\alpha_0 + \beta_0 + 1}$