Semester Project

Compressive Sensing with Applications to Near-Field to Far-Field Antenna Pattern Extrapolation

Jonas Thalmeier

30/04/2025

EURECOM
Sophia Antipolis

Contents

1 Introduction

This semester project is carried out within the framework of the Data Science track of the Master program and is supervised by Prof. Dirk Slock, Dr. Zilu Zhao, and Dr. Fangqing Xiao. The project focuses on the theory and application of compressive sensing, a signal processing technique that allows for the recovery of sparse signals from undersampled measurements.

In the first phase (100-hour scope), the project explores state-of-the-art methods in compressive sensing, with particular attention to Sparse Bayesian Learning (SBL) and the use of Stein's Unbiased Risk Estimate (SURE) for adaptive hyperparameter tuning in the SBL prior. Theoretical understanding and simulation-based validation on synthetic data are key objectives of this part.

In the extended 200-hour version, the project further investigates the practical application of compressive sensing techniques to antenna pattern extrapolation, specifically in transitioning from near-field to far-field measurements. This includes the construction of appropriate dictionaries or sensing matrices, initially based on simplified analytical models of antenna radiation patterns.

2 Synthetic Data Generation

To evaluate the performance of the implemented algorithms in a controlled environment, synthetic data was generated following a standard compressive sensing model. The measurement vector $\mathbf{t} \in \mathbb{R}^N$ is constructed as

$$\mathbf{t} = \Phi\mathbf{w} + \mathbf{e}, \tag{1}$$

where $\Phi \in \mathbb{R}^{N \times D}$ is a sensing matrix, $\mathbf{w} \in \mathbb{R}^D$ is a sparse signal, and $\mathbf{e} \in \mathbb{R}^N$ is additive Gaussian noise with standard deviation σ.

The sensing matrix Φ is constructed by selecting N rows at random from a normalized $D \times D$ discrete Fourier transform (DFT) matrix. Optionally, a random Gaussian matrix with variance $1/N$ can be used instead of the DFT for comparison purposes.

The sparse vector $\mathbf{w}$ contains a user-defined fraction $\rho \in (0, 1)$ of non-zero entries. These entries are assigned random values drawn from a standard normal distribution, and their indices are randomly selected. The noise vector $\mathbf{e}$ is sampled from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$.

This setup allows flexible control over the number of measurements N, signal dimensionality D, sparsity level ρ, and noise level σ, providing a reproducible and interpretable environment for algorithmic evaluation. The generation process is deterministic under a fixed random seed, ensuring consistency across multiple runs.

3 Choice of Algorithm

In the early phase of this project, a central task was to select a suitable inference algorithm for sparse signal recovery. The field of *Approximate Message Passing* (AMP) algorithms has seen considerable research activity in recent years, offering powerful and efficient solutions for compressive sensing problems. A concise yet comprehensive overview is provided in the tutorial by Zou and Yang [1], which summarizes key variants and their respective strengths.

However, AMP methods rely heavily on assumptions about the structure of the measurement matrix. As noted by Zou and Yang [1], AMP algorithms often fail to converge when the measurement matrix deviates from the independent and identically distributed (i.i.d.) sub-Gaussian regime. To address this limitation, modified approaches such as Orthogonal AMP (OAMP) have been proposed [1], extending AMP's applicability to certain classes of unitarily-invariant matrices.

In the context of this project, the eventual goal is to apply compressive sensing to near-field to far-field antenna pattern extrapolation. The underlying sparse representation may involve FFT bases or modal decompositions of the near-field, which are expected to yield structured, non-random measurement matrices. To maintain flexibility in handling such structure, methods constrained to i.i.d. random matrices—such as classical AMP—appear less suitable.

As a result, I chose to explore an alternative inference approach based on the Expectation Maximization (EM) algorithm, as introduced by Wipf and Rao [2] first. The EM-based SBL framework is known for its robustness and generality, making fewer assumptions about the measurement matrix. While it may not offer the same computational efficiency as AMP in ideal settings, it provides a more versatile and stable foundation for the anticipated applications of this project.

4 EM-Based Sparse Bayesian Learning

To estimate sparse weights from an overcomplete basis, the Expectation-Maximization (EM) algorithm proposed in [2] was implemented. This method belongs to the broader Sparse Bayesian Learning (SBL) framework, where sparsity is induced not by fixed priors, but through evidence maximization over a parameterized Gaussian prior.

4.1 Model Overview

Given an observation model

$$\mathbf{t} = \Phi\mathbf{w} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2\mathbf{I}), \tag{2}$$

where $\mathbf{t} \in \mathbb{R}^N$ is the observed signal, $\Phi \in \mathbb{R}^{N\times D}$ is the (possibly overcomplete) measurement matrix, and $\mathbf{w} \in \mathbb{R}^D$ is the weight vector to be estimated, SBL assumes a zero-mean Gaussian prior with individual variances for each weight:

$$p(\mathbf{w}; \boldsymbol{\gamma}) = \prod_{i=1}^{M} (2\pi\gamma_i)^{-\frac{1}{2}} \exp(-\frac{w_i^2}{2\gamma_i}). \tag{3}$$

The hyperparameters $\boldsymbol{\gamma}$ (the prior variances of the weights) are inferred via Type-II maximum likelihood by marginalizing out $\mathbf{w}$ and maximizing the resulting evidence.

4.2 Algorithm Description

The EM algorithm alternates between computing the posterior distribution over weights (E-step) and updating the hyperparameters (M-step). Specifically:

E-step: The posterior $p(\mathbf{w} \mid \mathbf{t}, \boldsymbol{\gamma}, \beta)$ is Gaussian:

$$\boldsymbol{\Sigma}_w = \left(\beta\Phi^\top\Phi + \mathrm{diag}(\boldsymbol{\gamma}^{-1})\right)^{-1}, \tag{4}$$

$$\boldsymbol{\mu}_w = \beta\boldsymbol{\Sigma}_w\Phi^\top\mathbf{t}, \tag{5}$$

where $\beta = 1/\sigma^2$ is the precision of the Gaussian likelihood.

M-step: The prior variances are updated based on the posterior moments:

$$\gamma_i = \mu_{w,i}^2 + (\Sigma_w)_{ii}, \tag{6}$$

and the noise variance is updated as:

$$\sigma^2 = \frac{1}{N}\|\mathbf{t} - \Phi\boldsymbol{\mu}_w\|^2 + \frac{1}{N}\sum_i (\Phi\Sigma_w\Phi^\top)_{ii}. \tag{7}$$

In my implementation, the update of σ^2 was simplified or deferred for stability, and a fixed β was used initially.

4.3 Implementation Remarks

The algorithm was implemented in Python based directly on the formulation in [2]. A simplified version of the EM loop is shown below:

```
mu, Sigma = estimate_posterior()
gamma = mu**2 + np.diag(Sigma)
```

The evolution of weights was tracked during training for diagnostic purposes, and convergence is monitored using the maximum change in μ and the relative mean squared error with respect to $\mathbf{t}$.

The main practical limitation of this approach lies in the need to invert a $D \times D$ matrix at every iteration to compute Σ_w, making it computationally expensive for high-dimensional problems. This is noted in the original paper as well [2, p. 2155], and future work may include reformulating the algorithm to reduce this complexity using the Woodbury identity or low-rank approximations.

4.4 Scalability via Covariance-Free Expectation Maximization (CoFEM)

While the EM-based Sparse Bayesian Learning implementation performs well for small-scale problems, it becomes prohibitively slow as the problem size increases. This is due to the need to invert a dense $D \times D$ matrix in each iteration to compute the posterior covariance. To address this computational bottleneck, the *Covariance-Free EM (CoFEM)* algorithm as proposed by Lin et al. [3] was implemented.

CoFEM avoids explicitly computing or storing the posterior covariance matrix. Instead, it estimates the required posterior statistics—the mean vector $\boldsymbol{\mu}_w$ and the diagonal elements of the covariance matrix Σ_{jj}—by solving a set of linear systems involving the precision matrix. These estimates are obtained using Rademacher probe vectors and linear solvers such as conjugate gradient, allowing for significant computational and memory savings.

In the E-step, the algorithm solves one linear system to compute the posterior mean and K additional systems to estimate the posterior variances using the diagonal estimation rule. The M-step then updates the hyperparameters α_j using the standard SBL update:

$$\alpha_j = \frac{1}{\mu_j^2 + s_j}, \tag{8}$$

where μ_j is the j-th component of the posterior mean and s_j is the estimated variance of w_j obtained from the Rademacher probe-based approximation.

Although the CoFEM algorithm is designed to benefit from conjugate gradient (CG) solvers for efficiently handling large-scale linear systems, the custom CG implementation used in this project proved to be slower than expected. In practice, it exhibited slower convergence and higher runtime compared to NumPy's built-in direct solver.. As a result, the final implementation resorts to NumPy's `np.linalg.solve` for solving the linear systems in the E-step, which yielded higher performance for the problem sizes considered in this study.

To assess the scalability, the runtime of the EM and CoFEM algorithm was measured as a function of the number of measurements N, with parameters set as follows: $D = 3N$, sparsity level $\rho = 0.1$, and noise variance $\sigma^2 = 10^{-6}$. The algorithms iterated until the relative residual dropped below 1e-2:

$$\frac{|\mathbf{t} - \Phi\boldsymbol{\mu}|_2}{|\mathbf{t}|_2} < 10^{-2} \tag{9}$$

Both algorithms required roughly the same numbers of iterations.

As shown in Figure 1, the CoFEM implementation significantly reduces runtime compared to the standard EM approach, especially as the dimensionality increases.

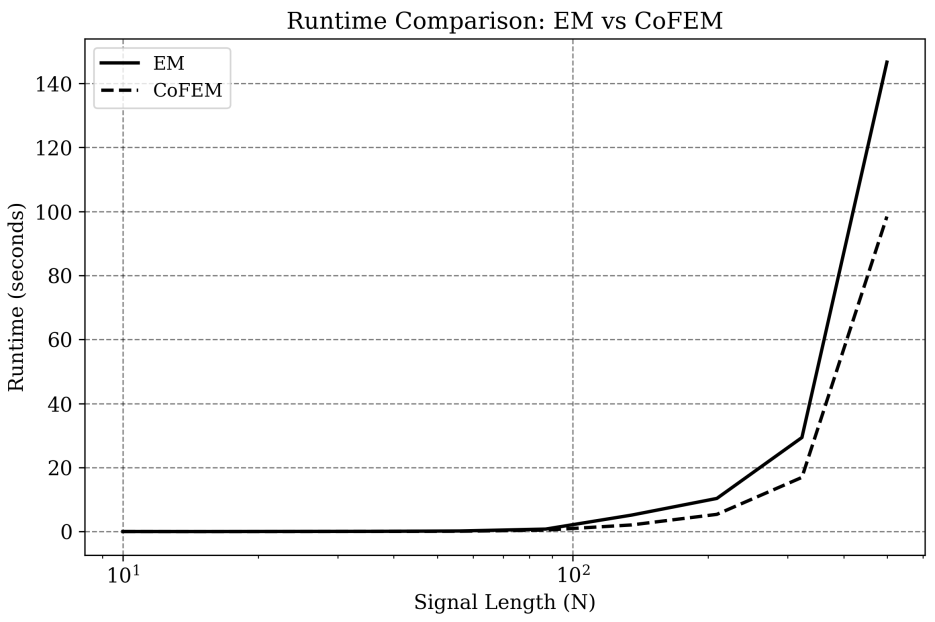

Figure 1: Runtime of the EM and CoFEM algorithms for increasing number of measurements N. The dictionary size is set to $D = 3N$.

While the CoFEM algorithm achieves a noticeable speedup over the standard EM algorithm—98.4 seconds compared to 146.6 seconds for N=500—the improvement falls short of the expected at least twofold acceleration.

4.5 Performance Comparison of EM Algorithms

Gaussian Matrix To compare the performance of different EM-based Sparse Bayesian Learning algorithms, experiments were conducted using synthetic data generated from Gaussian dictionaries, varying both the sparsity level ρ and the undersampling factor $\delta = N/D$. Four variants were considered: standard EM and CoFEM, each with known and unknown noise variance. The normalized root mean squared error (NRMSE) was used to evaluate reconstruction quality. Results are averaged over multiple random trials to ensure statistical robustness.

The signal dimension was fixed to $D = 1024$. For the sparsity analysis the undersampling factor was fixed at $\delta = 0.25$, corresponding to $N = \lfloor \delta D \rfloor$ measurements. For the undersampling analysis, the sparsity level was fixed at $\rho = 0.06$. The measurement matrix was drawn from a standard Gaussian distribution. Additive Gaussian noise with standard deviation $\sigma = 0.01$ was added to the measurements. For each setting, the reported results represent the average over five random trials to account for variability due to random initialization and noise.

As shown in Figure 2 and Figure 3, all four algorithms exhibit remarkably similar performance across the full range of tested sparsity and undersampling settings. In particular, the CoFEM algorithm with unknown noise variance performs on par with the variants that assume knowledge of the noise level. This observation is consistent with the findings reported in [3], where CoFEM and EM methods were shown to yield nearly identical reconstruction accuracy under equivalent settings.

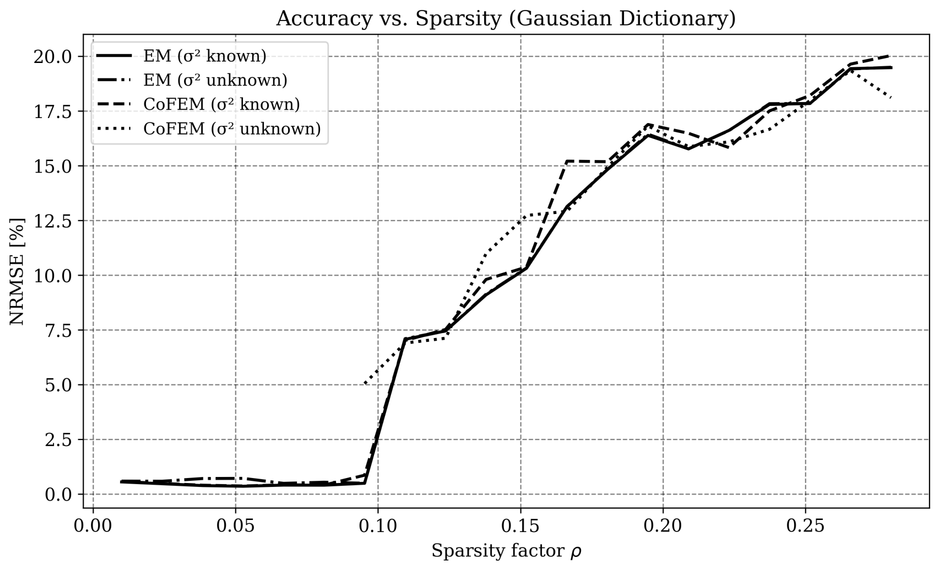

Figure 2: Reconstruction accuracy (NRMSE) versus sparsity level ρ. All four EM variants show near-identical behavior.

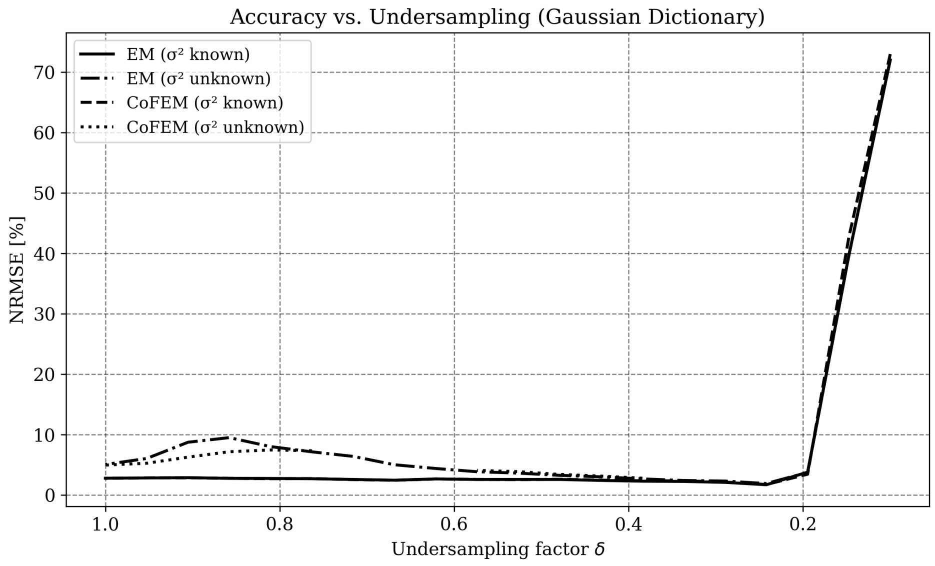

Figure 3: Reconstruction accuracy (NRMSE) versus undersampling factor δ. Similar performance is observed across all algorithms.

Given these results, and in order to simplify subsequent analysis, I choose to focus on the CoFEM algorithm with unknown noise variance in the remainder of this report. This choice is motivated by its computational scalability, the practical advantage of not requiring prior knowledge of the noise level, and the observation that its reconstruction performance closely matches that of other EM-based approaches in the tested settings. While this similarity may not generalize to all scenarios, it provides a reasonable basis for concentrating on a single representative method in this context.

Fourier Matrix As soon as the measurement matrix Φ gets changed from Gaussian to a matrix representing a Fourier Transform, thinks change rapidly. The diagonal Estimation of the CoFEM algorithm had to be modified so that both

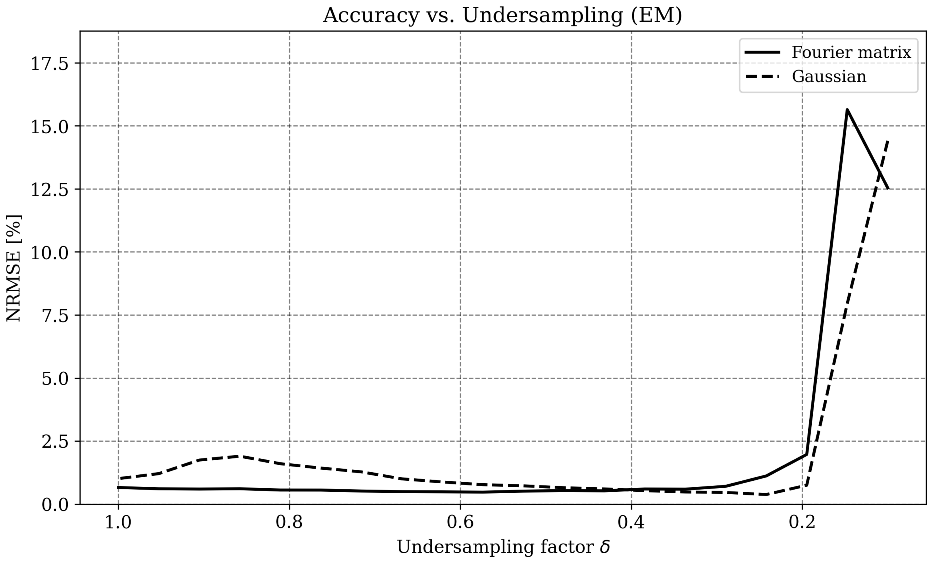

Figure 4: Reconstruction accuracy (NRMSE) versus undersampling factor δ.

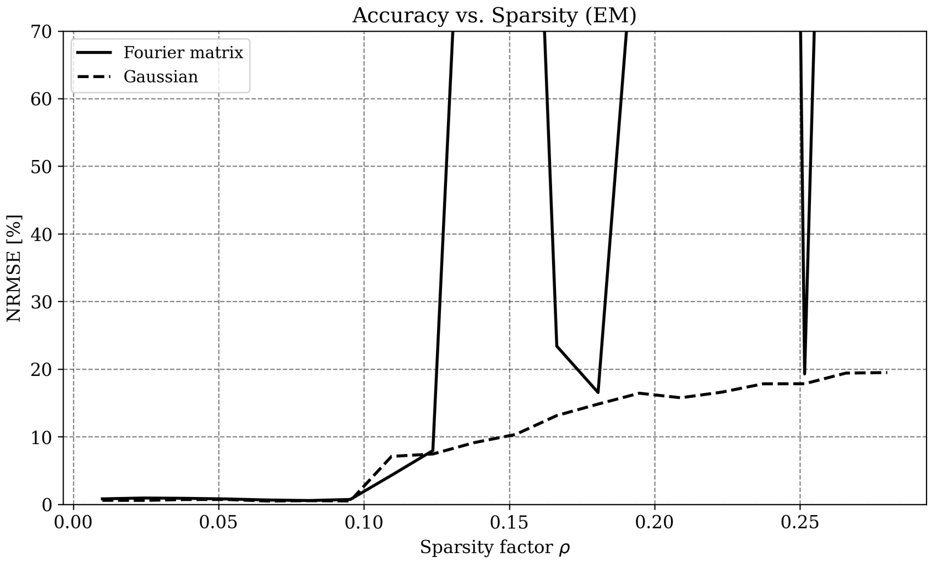

Figure 5: Reconstruction accuracy (NRMSE) versus undersampling factor δ.

5 Sparse Baysian Learning with Stein's Unbiased Risk Estimate

5.1 Stein's Lemma

Stein's Lemma provides a fundamental relationship between the expectation of a function of a Gaussian random variable and its derivative. For the univariate case, let $Z \sim \mathcal{N}(0,1)$ and $f : \mathbb{R} \to \mathbb{R}$ be an absolutely continuous function with derivative f' satisfying $\mathbb{E}|f'(Z)| < \infty$. Then, Stein's Lemma states:

$$\mathbb{E}[Zf(Z)] = \mathbb{E}[f'(Z)] \tag{10}$$

The proof follows from integration by parts and leverages the property of the standard normal density $\phi(z)$, where $\phi'(z) = -z\phi(z)$ [4]. This result extends to $X \sim \mathcal{N}(\mu, \sigma^2)$ through standardization:

$$\frac{1}{\sigma^2}\mathbb{E}[(X-\mu)f(X)] = \mathbb{E}[f'(X)] \tag{11}$$

For the multivariate case, let $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2\mathbf{I}_n)$ and let $f : \mathbb{R}^n \to \mathbb{R}$ be an almost differentiable function with gradient ∇f. Stein's Multivariate Lemma states:

$$\frac{1}{\sigma^2}\mathbb{E}[(\mathbf{X}-\boldsymbol{\mu})f(\mathbf{X})] = \mathbb{E}[\nabla f(\mathbf{X})] \tag{12}$$

This is proved by applying the univariate lemma component-wise and using Fubini's theorem under the almost differentiability condition [4]. The lemma enables covariance estimation without explicit knowledge of $\boldsymbol{\mu}$, replacing it with computable partial derivatives.

5.2 Stein's Unbiased Risk Estimate (SURE)

For a Gaussian model $\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2\mathbf{I}_n)$ and estimator $\hat{\boldsymbol{\mu}}(\mathbf{y})$, the risk $R = \mathbb{E}\|\boldsymbol{\mu} - \hat{\boldsymbol{\mu}}\|_2^2$ can be decomposed as:

$$R = -n\sigma^2 + \mathbb{E}\|\mathbf{y} - \hat{\boldsymbol{\mu}}\|_2^2 + 2\sigma^2\sum_{i=1}^{n}\mathrm{Cov}(y_i, \hat{\mu}_i) \tag{13}$$

Using Stein's Lemma, the covariance term is replaced with partial derivatives when $\hat{\boldsymbol{\mu}}$ is almost differentiable. This yields Stein's Unbiased Risk Estimate (SURE):

$$\widehat{R} = -n\sigma^2 + \|\mathbf{y} - \hat{\boldsymbol{\mu}}\|_2^2 + 2\sigma^2\sum_{i=1}^{n}\frac{\partial\hat{\mu}_i}{\partial y_i}(\mathbf{y}) \tag{14}$$

SURE provides an unbiased risk estimator ($\mathbb{E}[\widehat{R}] = R$) without requiring knowledge of $\boldsymbol{\mu}$ [4]. For estimators depending on tuning parameters λ, SURE enables model selection through:

$$\hat{\lambda} = \arg\min_{\lambda\in\Lambda}\left(\|\mathbf{y} - \hat{\boldsymbol{\mu}}_\lambda\|_2^2 + 2\sigma^2\sum_{i=1}^{n}\frac{\partial\hat{\mu}_{\lambda,i}}{\partial y_i}(\mathbf{y})\right) \tag{15}$$

This framework has become instrumental for risk estimation and hyperparameter optimization in high-dimensional statistics.

5.3 Algorithm Design for SBL with SURE

The implementation of Sparse Bayesian Learning (SBL) with Stein's Unbiased Risk Estimator (SURE) was planned based on the theoretical framework presented in [5]. In this setting, we consider the linear model:

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{v}, \quad \mathbf{v} \sim \mathcal{N}(\mathbf{v}; \mathbf{0}, \sigma^2\mathbf{I}), \quad \mathbf{x} \sim \mathcal{N}(\mathbf{x}; \mathbf{0}, \mathbf{P}), \tag{16}$$

where $\mathbf{y}$ represents the observed data, $\mathbf{A}$ is the measurement matrix, $\mathbf{x}$ is the sparse signal of interest, and $\mathbf{v}$ is additive Gaussian noise. The prior on $\mathbf{x}$ is Gaussian with covariance $\mathbf{P}$, which is typically diagonal to enforce sparsity. The posterior distribution of $\mathbf{x}$ given $\mathbf{y}$ is derived as:

$$\mathbf{x}|\mathbf{y} \sim \mathcal{N}\left(\mathbf{x}; \mathbf{P}\mathbf{A}^T\mathbf{R}^{-1}\mathbf{y}, \mathbf{P} - \mathbf{P}\mathbf{A}^T\mathbf{R}^{-1}\mathbf{A}\mathbf{P}\right), \tag{17}$$

where $\mathbf{R} = \mathbf{A}\mathbf{P}\mathbf{A}^T + \sigma^2\mathbf{I}$. Alternatively, this can be rewritten as:

$$\mathbf{x}|\mathbf{y} \sim \mathcal{N}\left(\mathbf{x}; \mathbf{S}^{-1}\mathbf{A}^T\mathbf{y}, \sigma^2\mathbf{S}^{-1}\right), \tag{18}$$

with $\mathbf{S} = \mathbf{A}^T\mathbf{A} + \sigma^2\mathbf{P}^{-1}$.

5.4 Connection to Tipping's Fast Marginal Likelihood Algorithm

The posterior expressions derived above align with the sparse Bayesian framework described in [6]. By adopting the notation used in Tipping's paper, where $\mathbf{A} \to \mathbf{\Phi}$, $\mathbf{y} \to \mathbf{t}$, $\mathbf{x} \to \mathbf{w}$, and $\mathbf{P} \to \mathbf{A}^{-1}$, the posterior mean and covariance become:

$$\boldsymbol{\mu} = \sigma^{-2}\mathbf{\Sigma}\mathbf{\Phi}^T\mathbf{t}, \quad \mathbf{\Sigma} = \left(\mathbf{A} + \sigma^{-2}\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}. \tag{19}$$

These expressions are identical to those derived in [?] for the posterior distribution of the weights $\mathbf{w}$ in the sparse Bayesian learning framework. This equivalence confirms that the Fast Marginal Likelihood Maximisation algorithm proposed by Tipping can be directly applied to implement SBL with SURE in our setting.

The algorithm proceeds as follows:

1. Initialize the hyperparameters $\boldsymbol{\alpha}$ and noise variance σ^2.
2. Compute the posterior statistics $\boldsymbol{\mu}$ and $\mathbf{\Sigma}$ using the current hyperparameters.
3. Evaluate the marginal likelihood and update $\boldsymbol{\alpha}$ by sequentially adding, deleting, or re-estimating basis functions to maximize the marginal likelihood.
4. Iterate until convergence, ensuring that the hyperparameters and noise variance are optimized for minimal mean squared error (MSE) as guided by SURE.

This approach leverages the efficiency of Tipping's algorithm while incorporating the risk estimation principles of SURE to optimize hyperparameters. The result is a computationally tractable implementation of SBL that maintains sparsity and generalizes well, as demonstrated in the experiments reported in [?].

5.5 Implementation of the Fast Marginal Likelihood Algorithm

The Fast Marginal Likelihood Algorithm is implemented through an efficient basis selection and pruning strategy. Unlike traditional SBL approaches that update all basis functions simultaneously, this implementation sequentially evaluates individual basis functions to maximize the marginal likelihood more efficiently.

The algorithm begins by initializing a single most relevant basis function, selected by evaluating the correlation between each basis vector and the target signal. All other basis functions are initially deactivated by setting their corresponding α values to infinity. This sparse initialization provides a computationally efficient starting point.

During each iteration, the algorithm considers three possible actions for each basis function:

- Adding a currently inactive basis ($\alpha = \infty$)
- Deleting an active basis from the model
- Re-estimating the α value of an active basis

For each potential action, the algorithm computes the change in marginal likelihood (ΔL) that would result. This computation involves two key quantities for each basis i:

- The sparsity factor S_i, measuring the potential sparsity contribution
- The quality factor Q_i, indicating the basis function's alignment with the target

These factors are efficiently computed using the current model state without requiring full matrix inversions. The algorithm then selects the action that yields the largest positive change in marginal likelihood. If no action provides sufficient improvement ($\Delta L <$ threshold), the algorithm terminates.

The posterior statistics ($\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$) are maintained and updated efficiently after each action. For addition and re-estimation, only the affected elements of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ need to be recomputed. When deleting a basis, the algorithm efficiently downdates the model parameters.

This sequential approach provides several advantages:

- Computational efficiency through selective updates
- Natural sparsity promotion through explicit basis selection
- Robust convergence through guaranteed marginal likelihood improvements
- Automatic determination of model complexity

The implementation handles both real and complex-valued inputs, with appropriate conjugate operations for complex matrices. The noise precision β $(1/\sigma^2)$ can either be specified or estimated from the data during optimization.

This efficient implementation enables the algorithm to handle high-dimensional problems while maintaining sparsity and achieving fast convergence, typically requiring fewer iterations than traditional SBL approaches.

References

[1] Q. Zou and H. Yang, "A concise tutorial on approximate message passing," 2022.

[2] D. Wipf and B. Rao, "Sparse bayesian learning for basis selection," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2153–2164, 2004.

[3] A. Lin, A. H. Song, B. Bilgic, and D. Ba, "Covariance-free sparse bayesian learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 3818–3831, 2022.

[4] R. J. Tibshirani and L. A. Wasserman, "Stein ' s unbiased risk estimate statistical machine learning , spring 2015," 2015.

[5] D. Slock, "Sparse bayesian learning with stein's unbiased risk estimator based hyperparameter optimization," in *2022 56th Asilomar Conference on Signals, Systems, and Computers*, pp. 857–861, 2022.

[6] M. E. Tipping and A. C. Faul, "Fast marginal likelihood maximisation for sparse bayesian models," in *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics* (C. M. Bishop and B. J. Frey, eds.), vol. R4 of *Proceedings of Machine Learning Research*, pp. 276–283, PMLR, 03–06 Jan 2003. Reissued by PMLR on 01 April 2021.