TP 2 Inverse Problem
Tikhonov and TV-L1 regularisation

Cabeza Gallucci, Manuel
mcabezag@fi.uba.ar

February 26, 2023

Contents

1 Introduction **1**

2 Preparation **2**
2.1 MNE algorithm . . . 2
2.2 SISSY algorithm . . . 2
2.2.1 Derivations for $s^{(k+1)}$ . . . 2
2.2.2 Derivations for $z^{(k+1)}$ . . . 3
2.2.3 Derivations for $y^{(k+1)}$ . . . 3
2.2.4 Derivations for $u^{(k+1)}$ . . . 3
2.2.5 Derivations for $v^{(k+1)}$ . . . 3

3 Algorithm implementations **4**
3.1 MNE algorithm . . . 4
3.1.1 Regularization criteria . . . 5
3.1.2 Conclusions . . . 6
3.2 SISSY algorithm . . . 6
3.2.1 Variations in λ . . . 7
3.2.2 Variations in α . . . 7
3.2.3 Regularization parameters estimation . . . 7

4 Conclusions **8**

5 Annex I. Codes **9**
5.1 MNE algorithm . . . 9
5.2 SISSY algorithm . . . 9

1 Introduction

The objective of this work is to solve the following inverse problem:

$$min_s||X - As||_2^2 + \lambda f(s) \tag{1}$$

Where f(s) is a regularization term. Two f(s) will be considered, the norm L2 ($||s||_2^2$) and the norm TV-L1 ($||Ts||_1 + \alpha||s||_1$). The former leads to the MNE (minimum norm estimate) algorithm and the latter to the SISSY (source imaging based on structured sparcity) algorithm (T is a linear operator).

2 Preparation

2.1 MNE algorithm

In this case an analytical expression can be derived from the equation (1) when f(s) is the L2 norm.

$$min_s||X - As||_2^2 + \lambda f(s)$$
$$min_s(X - As)^T(X - As) + \lambda s^T s$$

We can then differentiate this expression with respect to s and equate it to 0 to find the $\hat{s}$.

$$\frac{\partial}{\partial s}(X - As)^T(X - As) + \lambda s^T s 0$$
$$-2A^T X + 2(A^T A + \lambda I)\hat{s} 0$$
$$\hat{s} = (A^T A + \lambda I)^{-1} A^T X$$

This expression for $\hat{s}$ can be simplified by using the following inversion lemma with $B = A$, $R = I$ and $P^{-1} = \lambda I$.

$$(B^T R^{-1} B + P^{-1})^{-1} B^T R^{-1} = PB^T(BPB^T + R)^{-1}$$

Thus we get the expression for $\hat{s}$. Notably, since the matrix $A \in \mathbb{R}^{NxD}$, in the first case we would take an inverse of a DxD matrix, while in the second expression we have the inverse of an NxN matrix. As we know, in this type of problems: N « D, so the final result is much more computationally efficient.

$$\hat{s} = (A^T A + \lambda I)^{-1} A^T X$$
$$\hat{s} = \frac{1}{\lambda} A^T (\frac{1}{\lambda} AA^T + I)^{-1} X$$
$$\hat{s} = A^T (AA^T + \lambda I)^{-1} X$$

2.2 SISSY algorithm

In this case no analytical expression can be found, so we use the Alternating Direction Method of Multipliers (ADMM) to find an iterative solution using the augmented Lagrangian.

$$min_s||X - As||_2^2 + \lambda f(s)$$
$$min_s||X - As||_2^2 + \lambda(||z||_1 + \alpha||y||_1)$$
$$st. Z = Ts \; y = s$$

The solution for the ADMM algorithm is found via the Lagrangian $\mathcal{L}$ and the recursive equations for each of its parameters.

$$\mathcal{L}(s, z, y, u, v) = \frac{1}{2}||X - As||_2^2 + \lambda||z||_1 + \lambda\alpha||y||_1 + (z - Ts)^T u + (y - s)^T v + \frac{\rho}{2}||z - Ts||_2^2 + \frac{\rho}{2}||y - s||_2^2$$

In the next sections the expressions for all ADMM iterations are found. To that extent, the following function, which has a known analytical solution, is defined.

$$prox_\beta(y) = argmin_x \frac{1}{2}||x - y||_2^2 + \beta||x||_1 \tag{2}$$

2.2.1 Derivations for $s^{(k+1)}$

$$s^{(k+1)} = argmin_s \mathcal{L}(s, z^{(k)}, y^{(k)}, u^{(k)}, v^{(k)})$$

Since this derivative exists, we can derivative $\mathcal{L}$ with respect to s and make the expression to 0.

$$\frac{\partial \mathcal{L}}{\partial s} = -A^T X + A^T As - T^T u + \rho T^T(Ts - z) + \rho(s - y) - v = 0$$
$$(A^T A + \rho(T^T T + I))s = A^T X + T^T u + \rho T^T z + \rho y + v$$
$$s = (A^T A + \rho(T^T T + I))^{-1}(A^T X + T^T u + \rho T^T z + \rho y + v)$$

We then evaluate the former expression in $z^{(k)}, y^{(k)}, u^{(k)}, v^{(k)}$ to get $s^{(k+1)}$.

2.2.2 Derivations for $z^{(k+1)}$

$$z^{(k+1)} = argmin_z \mathcal{L}(s^{(k+1)}, z, y^{(k)}, u^{(k)}, v^{(k)})$$

In this case since we have the L1 norm we use the prox function (2) to derive the result. The terms in the Lagrangian which do not depend on z are excluded when finding the argmin.

$$\begin{gathered}
z^{(k+1)} = argmin_z \lambda||z||_1 + (z - Ts)^T u + \frac{\rho}{2}||z - Ts||_2^2 \\
argmin_z \frac{\rho}{2}(z - Ts)^T(z - Ts) + (z - Ts)^T u + \lambda||z||_1 \\
argmin_z \frac{1}{2}(z - Ts + \frac{1}{\rho}u)^T(z - Ts + \frac{1}{\rho}u) - \frac{1}{2\rho^2}||u||_2^2 + \frac{\lambda}{\rho}||z||_1 \\
argmin_z \frac{1}{2}||z - Ts + \frac{1}{\rho}u||_2^2 + \frac{\lambda}{\rho}||z||_1 \\
z^{(k+1)} = prox_{\lambda/\rho}(Ts^{(k+1)} - \frac{1}{\rho}u^{(k)})
\end{gathered}$$

2.2.3 Derivations for $y^{(k+1)}$

$$y^{(k+1)} = argmin_z \mathcal{L}(s^{(k+1)}, z^{(k+1)}, y, u^{(k)}, v^{(k)})$$

This is solved similarly to the expression for $z^{(k+1)}$.

$$y^{(k+1)} = argmin_y \frac{\rho}{2}||y - s||_2^2 + (y - s)^T v + \alpha\lambda||y||_1$$

We see that the problem is exactly the same as the one posed for $z^{(k+1)}$ if we perform the following changes of variable: s $Ts \to s$, $u \to v$, $\lambda \to \lambda\alpha$. Thus we get the corresponding solution.

$$y = prox_{\alpha\lambda/\rho}(s^{(k+1)} - \frac{1}{\rho}v^{(k)})$$

2.2.4 Derivations for $u^{(k+1)}$

$$\begin{gathered}
u^{(k+1)} = argmax_u \mathcal{L}(s^{(k+1)}, z^{(k+1)}, y^{(k+1)}, u, v^{(k)}) \\
u^{(k+1)} = argmax_u (z - Ts)^T u
\end{gathered}$$

Thus we can use an algorithm of the gradient descent type with a fixed step to find $u^{(k+1)}$.

$$u^{(k+1)} = u^{(k)} + \rho(z^{(k+1)} - Ts^{(k+1)})$$

2.2.5 Derivations for $v^{(k+1)}$

The derivation of the equation for $v^{(k+1)}$ is similar to the one for $u^{(k+1)}$, substituting $Ts \to s$, $z \to y$.

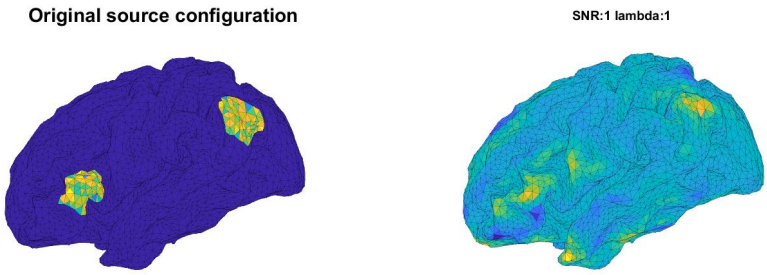
$$\begin{gathered}
v^{(k+1)} = argmax_v \mathcal{L}(s^{(k+1)}, z^{(k+1)}, y^{(k+1)}, u^{(k+1)}, v) \\
v^{(k+1)} = v^{(k)} + \rho(y^{(k+1)} - s^{(k+1)})
\end{gathered}$$

3 Algorithm implementations

Both algorithms were implemented in Matlab and the codes can be found in Annex I. The results are based on the active dipoles problem using brain scan data. The vector s represents the sources from all possible points and X the measured points in the brain. The objective is to find the correct source configuration for an epileptic strike. For this problem we have $N = 91$ and $D = 19626$.

3.1 MNE algorithm

We now will take different values of λ and SNR to compare the performance of the algorithm.



(a) Origianl source. (b) Results.

Figure 1: Example for SNR=1, λ=1.

We observe that the algorithm can potentially yield good results, the next figure contains the plotted results for an SNR in [0.1 , 1, 10] and λ in [0.1, 1, 10, 100].

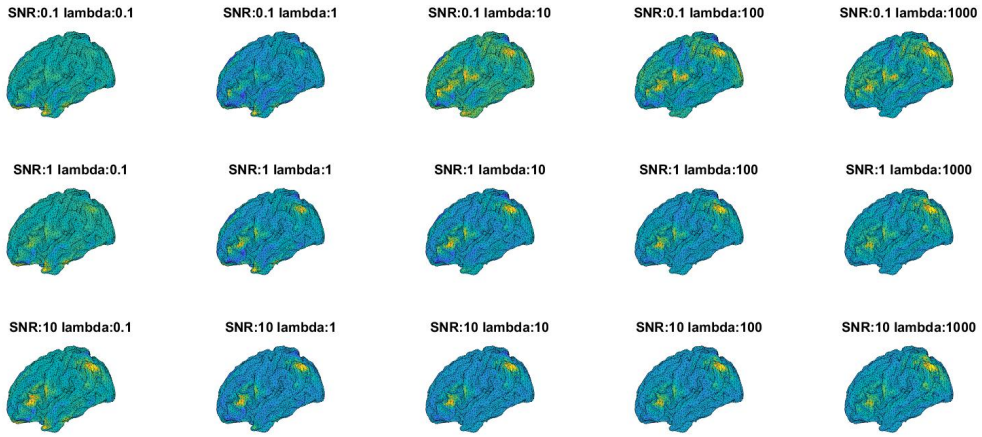


Figure 2: Comparison for different SNR and λ.

From the precedent chart we observe that for small values of λ the solution reassembles the MSE solution, which is more or less constant. If λ is too big then the algorithm favors one solution source over the correct two source solution.

If the value of the SNR goes up, then a smaller λ is allowed to keep the solution closer to the correct result. Conversely, if the SNR goes down then a bigger λ is needed. This is because a lower SNR introduces more noise into the system so the first term in the minimization equation becomes bigger and to counter this, λ should go up. From the empirical observations, λ around 10 to 100 performs well when the SNR is not too small (1 or 10).

3.1.1 Regularization criteria

In this section different criteria to define the value of λ are presented for an SNR=1. We define the reconstruction error as $e = ||X - As||_2$.

L-curve criterion In this case we graph the reconstruction error e as a function of the L2 norm of the solution s.

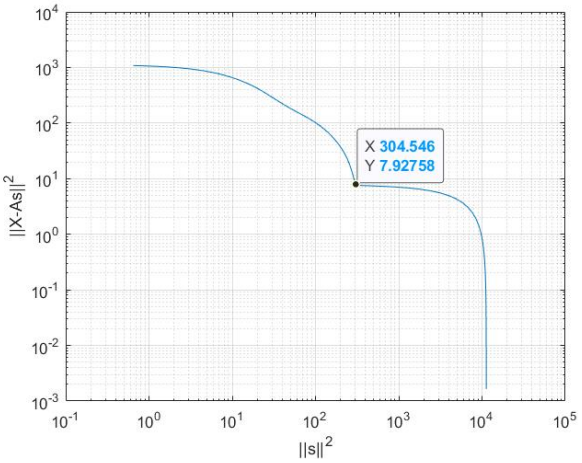


Figure 3: L-curve (logarithmic plot in both axes).

From the point in the graph we get that the regularization parameter is $\lambda \approx 0.003$.

Discrepancy principle In this case we graph the reconstruction error but as a function of λ and search for the place where the energy of the reconstruction error is equal to the energy of the noise. Since the SNR=1, then the square L2 norm is $1e^6$.

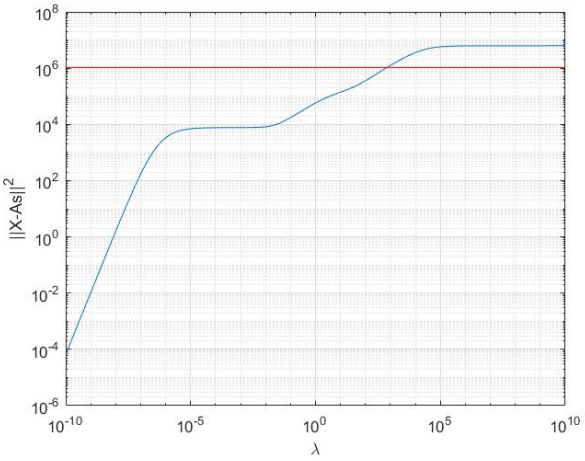


Figure 4: Noise (red) and e (blue) as a function of λ (logarithmic plot in both axes).

We observe that the crossing is at $\lambda \approx 1000$.

Generalized cross validation We now plot the GCV as a function of λ and search for the minimum of the function. We observe that it is found at around $\lambda \approx 100$.

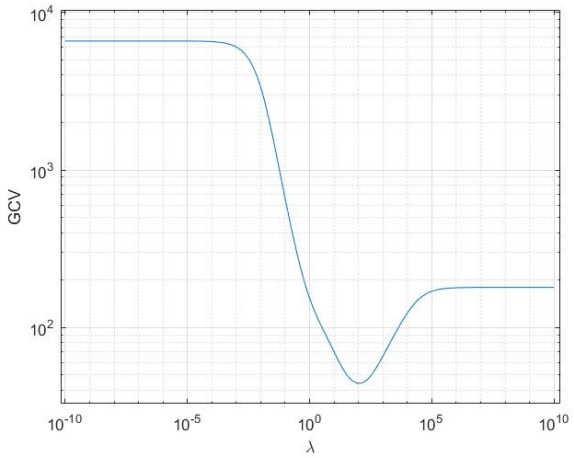


Figure 5: GCV as a function of λ (logarithmic plot in both axes).

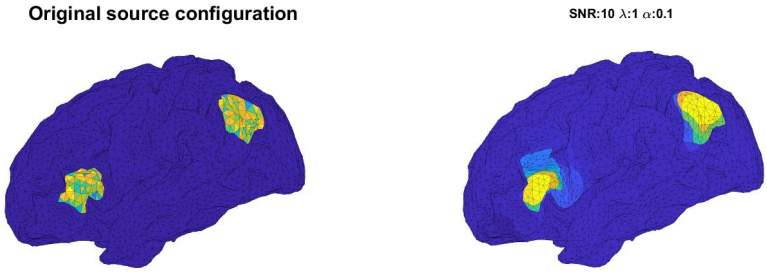
3.1.2 Conclusions

We find that using L2 as a regularization term can yield good results and a very fast solution (computationally) since the analytical expression exists. However, the solution is very dependent on the regularization parameter which depends on level of the noise.

From this results we conclude that either the Discrepancy principle or the GCV can give good estimates for the values of λ while the L-curve yields a parameter that does lead to the real solutions. This is seen in the picture (2), for an SNR=1 either λ=100 or λ=1000 work fine, while a smaller λ tends to homogenize the solution.

3.2 SISSY algorithm

The SISSY algorithm was run for an SNR=10 and 60 iterations of ADMM. Both λ and α were changed to find 'good' solutions (qualitatively).



(a) Origianl source. (b) Results.

Figure 6: Example for SNR=10, λ=1, α=0.1.

We observe that the results of the algorithm are quite good for this initial parameters choice and the algorithm seems to perform better than MNE.

3.2.1 Variations in λ

Now we fix $\alpha = 0.1$ and see the influence of different values of λ in the range [0.01, 10000].

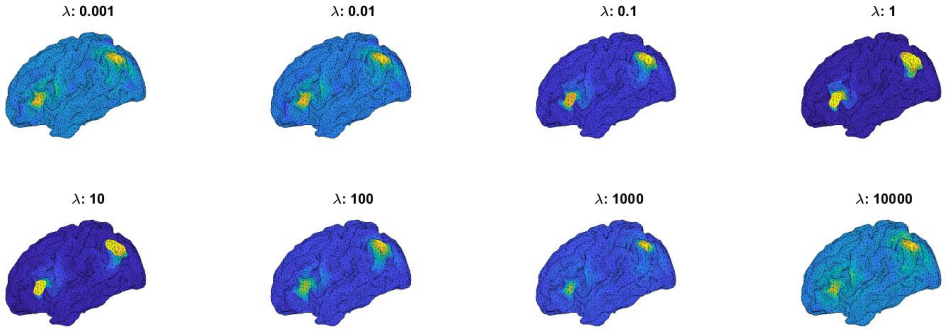


Figure 7: Results for SNR=10, α=0.1 and 60 iterations.

From the figure we observe that if the value of λ is too small then the results become noisier, inversely, if λ is too big, the solution becomes homogenized and there is little sparsity. Empirically a value of $\lambda \approx 10$ seems correct.

3.2.2 Variations in α

Now $\lambda = 10$ and we change the values of α in the range ([0, 1.0]).

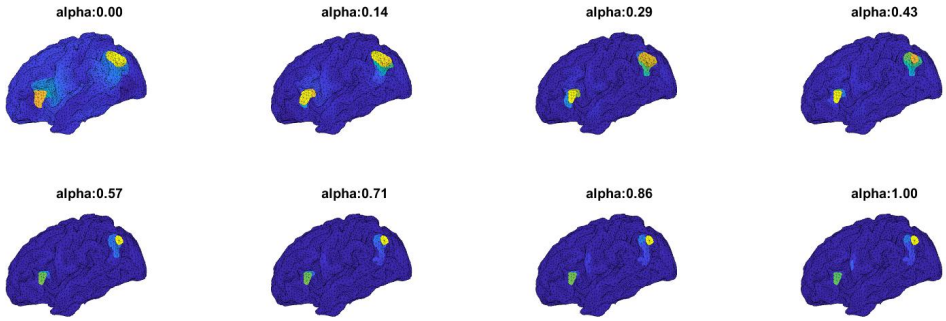


Figure 8: Results for SNR=10, λ=1.0 and 60 iterations.

We observe that a value of α between 0.14 and 0.29 seems correct. As smaller α introduces more noise while a bigger one makes the sources smaller.

3.2.3 Regularization parameters estimation

A way to automatically find λ is by minimizing the following function with respect to λ. In this case we fix α to be 0.2.

$$f(\lambda) = ||Ts||_0 + \alpha||s||_0 \ st.\ s \equiv s(\lambda)$$

The equations graph is seen below. We find that the value of $\lambda \approx 1$ is obtained. This correlates well with the observations, as from figure (10) λ between 1 and 10 seems to be correct.

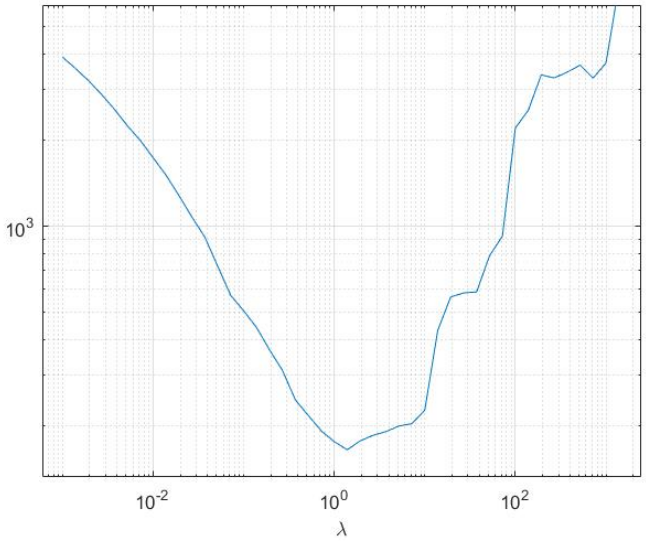


Figure 9: Regularization function for α=0.2 (logarithmic axes).

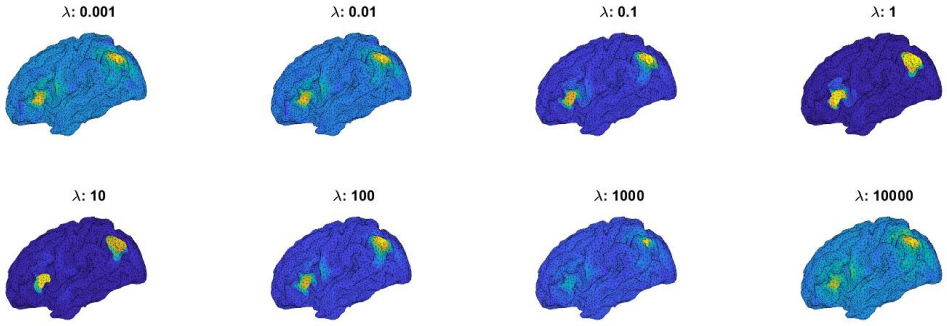


Figure 10: Results for SNR=10, α=0.2 and 60 iterations.

4 Conclusions

In this work both minimum norm estimate [MNE] and source imaging based on structured sparsity [SISSY] algorithms where analyzed and mathematical models for computing each were developed as well as code implementations. Both were used to find the sparse solution of the active double dipole problem using brain scan data.

We conclude that the SISSY algorithm performs best in practice and while the MNE algorithm can give a good estimation, the solution was generally quite noisy.

Generally, the results depended heavily on the regularization parameters used, so different methods of estimation were analyzed for each algorithm. In the case of MNE either the discrepancy principle or the generalized cross validation gave good estimates and in the case of SISSY the minimization of the regularization function was used to find one of the parameters.

This two algorithms as well as the one presented in the previous work (Gibbs Sampler) give the basic techniques to analyse and solve inverse problems.

5 Annex I. Codes

5.1 MNE algorithm

```
function s = MNE_algo(X, A, lambda)
    [N, ~] = size(A);
    s = A' * ((A * A' + lambda * eye(N)) \ X);
end
```

5.2 SISSY algorithm

```
function  s = SISSY_algo(X, A, T,lambda, alpha, niter)
    rho = 1;
    [~,D] = size(A);

    y = zeros(D,1);
    z = zeros(size(T,1),1);
    u = zeros(size(T,1),1);
    v = zeros(D,1);

    P=sparse(rho*(T.'*T+speye(size(T,2))));
    APi=A/P;
    L=chol(eye(size(A,1))+APi*A.',"lower");
    s1=A.'*X;

    for i = 1:niter
        b=s1+rho*(T.'*(z+u/rho)+y+v/rho);
        s=P\b-APi'*(L'\(L\(APi*b)));

        z = prox_op( T*s - u / rho ,'L1' , (lambda)/rho);
        y = prox_op( s - v / rho,'L1' , (lambda*alpha)/rho);
        u = u + rho*(z - T*s);
        v = v + rho*(y - s);
    end
end
```