l_1 regularization: extensions

Machine Learning: a probabilistic perspective: Kevin P Murphy (Chapter 13.5)

Group Lasso

- In standard $l_1$regularization, we assume that there is a 1:1 correspondence between parameters and variables, so that if $\widehat{w}_j$=0, we interpret this to mean that variable j is excluded.
- But in more complex models, there may be many parameters associated with a given variable.
- In particular, we may have a vector of weights for each input, w_j

Some examples

- Examples where group lasso is used:
  - **Multinomial logistic regression :** Each feature is associated with C different weights, one per class.
  - **Linear regression with categorical inputs :** Each scalar input is one-hot encoded into a vector of length C.
  - **Multi-task learning**: In multi-task learning, we have multiple related prediction problems. For example, we might have C separate regression or binary classification problems. Thus, each feature is associated with C different weights. We may want to use a feature for all of the tasks or none of the tasks, and thus select weights at the group level.

Group lasso

- If we use an l_1 regularizer of the form $||\boldsymbol{w}|| = \sum_j \sum_c |w_{jc}|$, we may end up with with some elements of $w_{j,:}$ being zero and some not.
- To prevent this kind of situation, we partition the parameter vector into G groups.
- We now minimize the following objective

$$J(\boldsymbol{w}) = NLL(\boldsymbol{w}) + \sum_{g=1}^{G} \lambda_g \left|\left|w_g\right|\right|_2$$

where $\left|\left|w_g\right|\right|_2 = \sqrt{\sum_{j\in \mathrm{g}} w_j^2}$

- If NLL (Negative Log Likelihood) is least squares, this method is called the **group lasso.**

Group lasso

- We often use a larger penalty for larger groups, by setting $\lambda_g = \lambda\sqrt{d_g}$, where d_g is the number of elements in group g.
- For example, if we have groups {1, 2} and {3, 4, 5}, the objective becomes

$$J(\boldsymbol{w})$$
$$= \mathrm{NLL}(\mathbf{w}) + \lambda[\sqrt{2}\sqrt{w_1^2 + w_2^2} + \sqrt{3}\sqrt{w_3^2 + w_4^2 + w_5^2}]$$

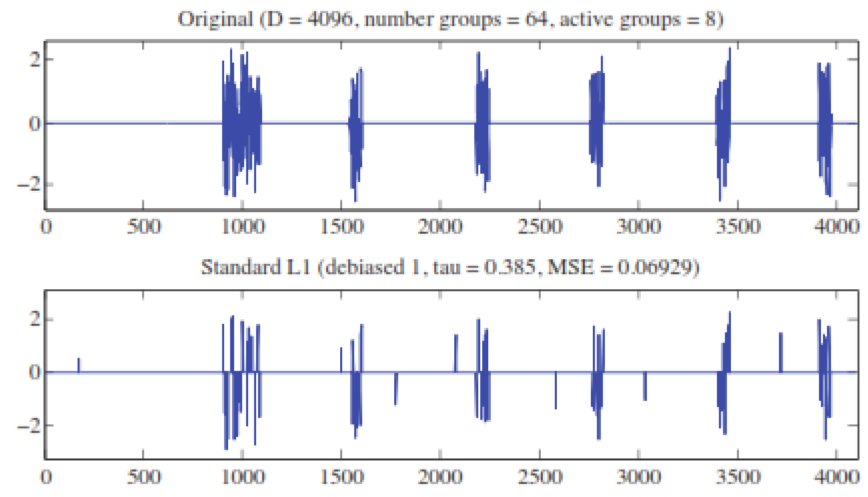
- Note that if we had used the square of the 2-norms, the model would become equivalent to ridge regression

Group lasso

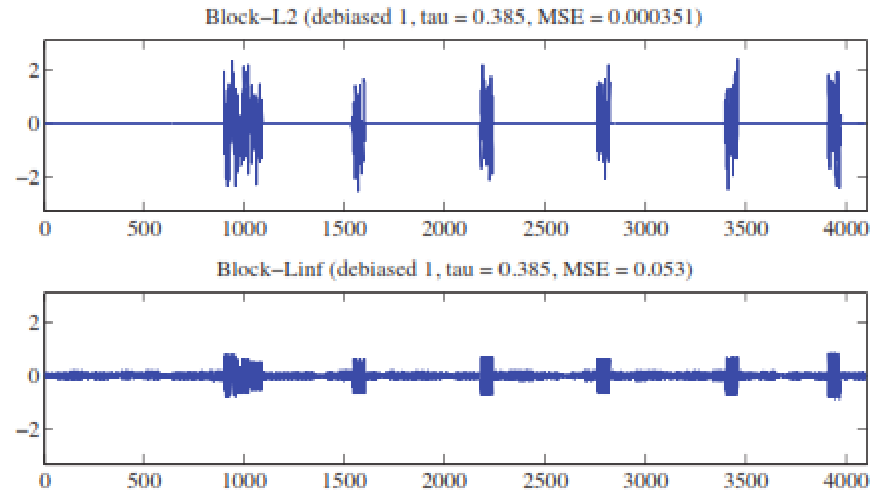
- By using the square root, we are penalizing the radius of a ball containing the group's weight vector: the only way for the radius to be small is if all elements are small. Thus the square root results in group sparsity.
- A variant of this technique replaces the 2-norm with the infinity-norm. This will also result in group sparsity.

i.e $\left|\left|w_g\right|\right|_\infty = \max_{j \in g} |w_j|$

- An illustration of the difference is shown in Figures 13.13 and 13.14. In both cases, we have a true signal $\boldsymbol{w}$ of size $D = 2^{12}$ = 4096, divided into 64 groups each of size 64.
- We randomly choose 8 groups of $\boldsymbol{w}$ and assign them non-zero values. In the first example, the values are drawn from a $\mathcal{N}$(0, 1).
- In the second example, the values are all set to 1. We then pick a random design matrix $\boldsymbol{X}$ of size $N \times D$, where $N = 2^{10}$=1024
- Finally, we generate $y = Xw + \epsilon$, where $\epsilon \sim \mathcal{N}(0,10^{-4}\boldsymbol{I}_N)$.
- Given this data, we estimate the support of $\boldsymbol{w}$ using l_1 or group $l_1$and then estimate the non-zero values using least squares.
- We see that group lasso does a much better job than vanilla lasso, since it respects the known group structure. We also see that the l_∞ norm has a tendency to make all the elements within a block to have similar magnitude.



(a)



(b)

Figure 13.13: Illustration of group lasso where the original signal is piecewise Gaussian. (a) top: original signal, bottom: vanilla lasso estimate.

(b) Top: group lasso estimate using a l_2 norm on the blocks, bottom: group lasso estimate using l_∞

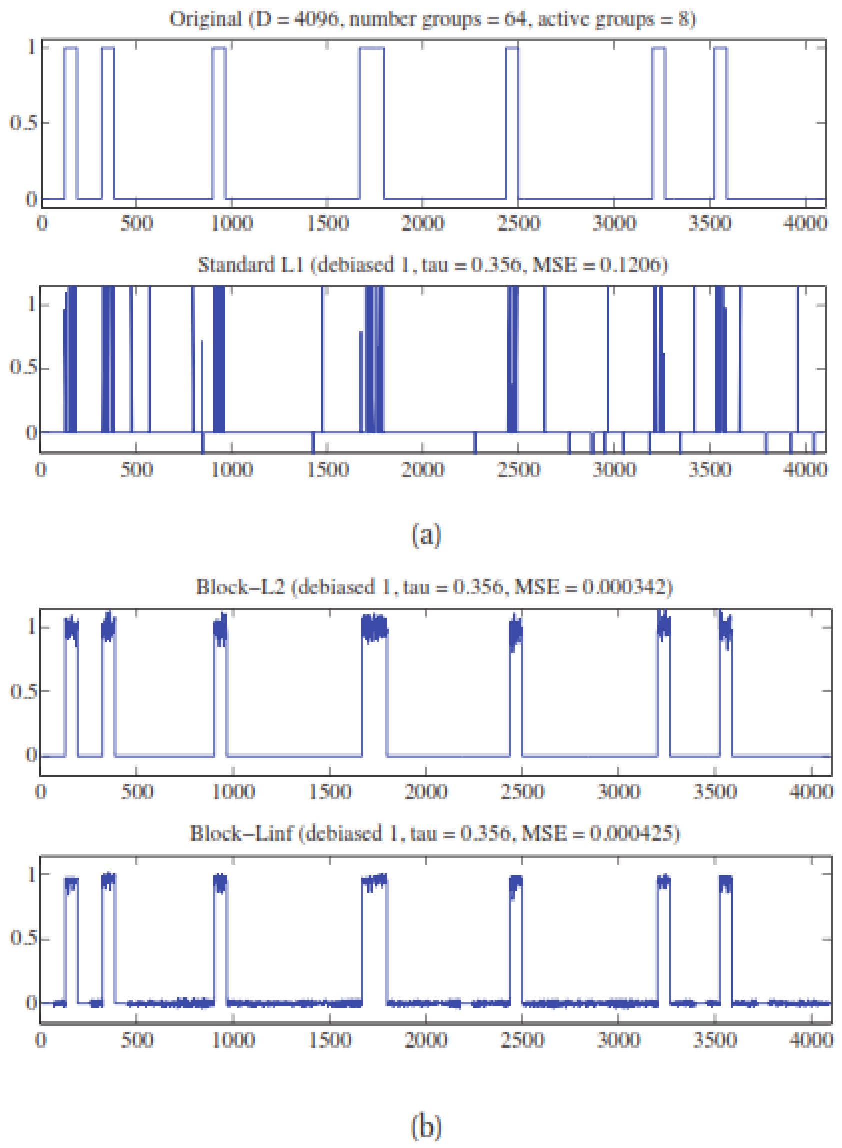


Figure 13.14: Illustration of group lasso where the original signal is piecewise constant. (a) top: original signal, bottom: vanilla lasso estimate.

(b) Top: group lasso estimate using a l_2 norm on the blocks, bottom: group lasso estimate using l_∞

Algorithms for group lasso

- Proximal Gradient

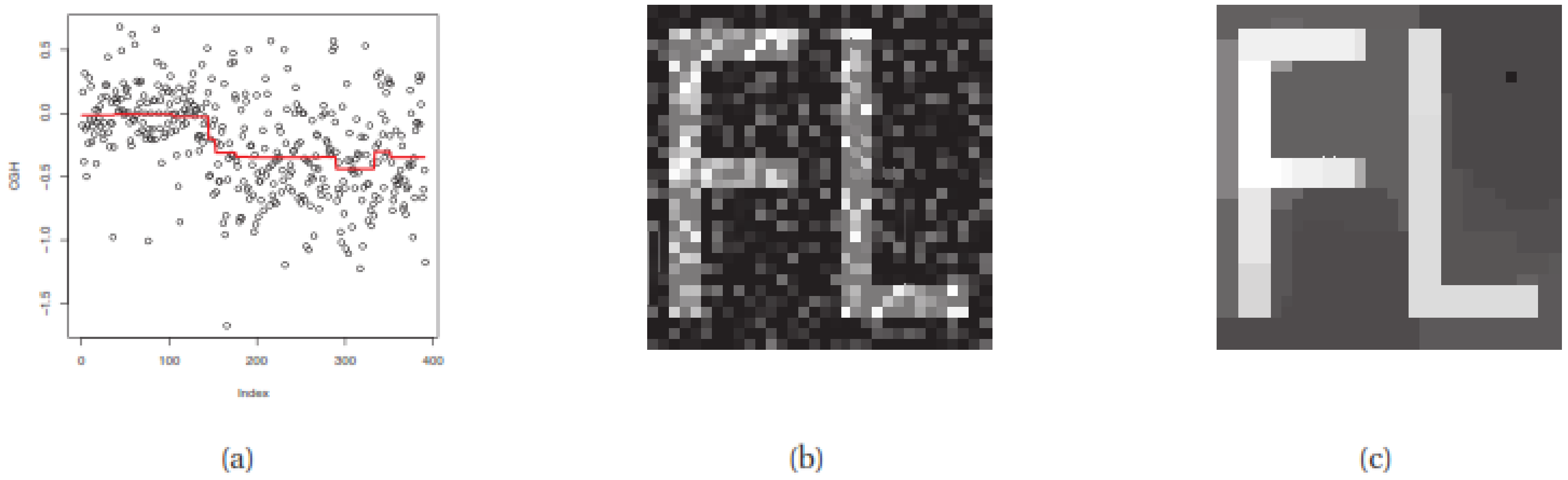
- EM algorithm

Fused Lasso

- In some problem settings (e.g., functional data analysis), we want neighboring coefficients to be similar to each other, in addition to being sparse. An example is given in Figure 13.16(a), where we want to fit a signal that is mostly “off”, but in addition has the property that neighboring locations are typically similar in value. We can model this by using a prior of the form

$$p(\boldsymbol{w}|\sigma^2) \propto \exp(-\frac{\lambda_1}{\sigma}\sum_{j=1}^{D}|w_j| - \frac{\lambda_2}{\sigma}\sum_{j=1}^{D-1}|w_{j+1} - w_j|)$$

- This is known as the fused lasso penalty



- **Figure 13.16** (a) Example of the fused lasso. The vertical axis represents array CGH (chromosomal genome hybridization) intensity, and the horizontal axis represents location along a genome. (b) Noisy image. (c) Fused lasso estimate using 2d lattice prior

Fused Lasso

- This is known as the **fused lasso penalty.** In the context of functional data analysis, we often use $X = I$, so there is one coefficient for each location in the signal . In this case, the overall objective has the form

$$J(\boldsymbol{w}, \lambda_1, \lambda_2) = \sum_{i=1}^{N} (y_i - w_i)^2 + \lambda_1 \sum_{i=1}^{N} |w_i|$$
$$+\lambda_2 \sum_{i=1}^{N-1} |w_{i+1} - w_i|)$$

Fused Lasso

- It is possible to generalize this idea beyond chains, and to consider other graph structures, using a penalty of the form

$$J(\boldsymbol{w}, \lambda_1, \lambda_2) = \sum_{s \in V} (y_s - w_s)^2 + \lambda_1 \sum_{s \in V} |w_s|$$
$$+ \lambda_2 \sum_{(s,t) \in E} |w_s - w_t|)$$

- Here, V are the set of vertices and E are the set of edges
- This is called **graph-guided fused lasso** The graph might come from some prior knowledge, e.g., from a database of known biological pathways. In the example shown in Figure 13.16(b-c), the graph structure is a 2d lattice.

Algorithms for fused lasso

- It is possible to generalize the EM algorithm to fit the fused lasso model, by exploiting the Markov structure of the Gaussian prior for efficiency.
- Direct solvers (which don’t use the latent variable trick) can also be
- However, this model is undeniably more expensive to fit than the other variants we have considered.

Elastic Net

- Disadvantages of LASSO
  - If there is a group of variables that are highly correlated (e.g., genes that are in the same pathway), then the lasso tends to select only one of them, chosen rather arbitrarily. It is usually better to select all the relevant variables in a group. If we know the grouping structure, we can use group lasso, but often we don’t know the grouping structure.
  - In the D>N case, lasso can select at most N variables before it saturates.
  - If N>D, but the variables are correlated, it has been empirically observed that the prediction performance of ridge is better than that of lasso.

  (The design matrix is of size $N \text{ X D}$)

Elastic Net: Vanilla Version

- The vanilla version of the model defines the following objective function:

$$J(\boldsymbol{w}, \lambda_1, \lambda_2) = \left|\left|\mathbf{y} - \mathbf{X}\mathbf{w}\right|\right|^2 + \lambda_2 \left|\left|\boldsymbol{w}\right|\right|_2^2 + \lambda_1 \left|\left|\boldsymbol{w}\right|\right|_1$$

- Notice that this penalty function is *strictly convex (assuming* λ_2> 0) so there is a unique global minimum, even if X is not full rank.
- Any strictly convex penalty on **w** will exhibit a **grouping effect,** which means that the regression coefficients of highly correlated variables tend to be equal (up to a change of sign if they are negatively correlated).
- For example, if two features are equal, so $X_{:,j} = X_{:,k}$, one can show that their estimates are also equal, $\hat{w}_j = \hat{w}_k$
- By contrast, with lasso, we may have that $\hat{w}_j = 0$ and $\hat{w}_k \neq 0$ or vice versa

Algorithms for Vanilla Elastic Net

- The elastic net problem can be reduced to a lasso problem on modified data. In particular (Exercise 13.5),

$$\widetilde{\boldsymbol{X}} = c\begin{pmatrix}\boldsymbol{X}\\ \sqrt{\lambda_2}\boldsymbol{I}_D\end{pmatrix}, \widetilde{\boldsymbol{y}} = \begin{pmatrix}\boldsymbol{y}\\ \boldsymbol{0}_{D\times 1}\end{pmatrix},$$

Where $c = (1+\lambda_2)^{-\frac{1}{2}}$. Then we solve,

$$\widetilde{\boldsymbol{w}} = argmin_{\widetilde{\boldsymbol{w}}} \left|\left|\widetilde{\boldsymbol{y}} - \widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{w}}\right|\right|^2 + c\lambda_1 \left|\left|\widetilde{\boldsymbol{w}}\right|\right|_1$$

and set $\mathbf{w} = c\widetilde{\boldsymbol{w}}$

- We can use LARS to solve this subproblem; this is known as the LARS-EN algorithm. When using LARS-EN (or other solvers), one typically uses cross-validation to select λ_1 and λ_2

Improved version of Elastic Net

- Unfortunately it turns out that the “vanilla” elastic net does not produce functions that predict very accurately, unless it is very close to either pure ridge or pure lasso.
- Intuitively the reason is that it performs shrinkage twice: once due to the l_2 penalty and again due to the l_1 penalty.
- The solution is simple: undo the l_2 shrinkage by scaling up the estimates from the vanilla version.
- In other words, a better estimate (corrected estimate) for elastic net is

$$\widehat{\boldsymbol{w}} = \sqrt{1 + \lambda_2}\,\widetilde{\boldsymbol{w}}$$

Improved version of ElasticNet

- One can show that the corrected estimates are given as:

$$\hat{\boldsymbol{w}} = argmin_{\boldsymbol{w}} \boldsymbol{w}^T \left(\frac{\boldsymbol{X}^T \boldsymbol{X} + \lambda_2 \boldsymbol{I}}{1 + \lambda_2} \right) \boldsymbol{w} - 2 \boldsymbol{y}^T \boldsymbol{X} \boldsymbol{w} + \lambda_1 \left|\left| \boldsymbol{w} \right|\right|_1$$

Now,

$$\frac{\boldsymbol{X}^T \boldsymbol{X} + \lambda_2 \boldsymbol{I}}{1+\lambda_2} = (1-\rho)\widehat{\boldsymbol{\Sigma}} + \rho \boldsymbol{I}, \quad \text{where } \rho = \frac{\lambda_2}{1+\lambda_2}$$

- So the elastic net is like lasso but where we use a version of $\widehat{\boldsymbol{\Sigma}}$ (covariance matrix) that is shrunk towards $\boldsymbol{I}$.