Group Sparse Regularization for Neural Nets

Yuze Zhou

Aug.2022

Theoretical Foundations

Cost Function and Regularization

- For a parameter matrix $W \in \mathbb{R}^{D*H}$, where $W = [w_1, \cdots, w_D]^\tau$ $w_j \in \mathbb{R}^H$, the group lasso penalty of W, $R_{l2,1}(W)$ is defined as:

$$R_{l2,1}(W) = \sum_{d=1}^{D} \sqrt{\|w_d\|_2^2}$$

- If W is a D-dimensional vector $W \in R^{1*D}$, the group lasso penalty would simply be transformed into the l_1 penalty such that:

$$R_{l2,1}(W) = R_{l1}(W) = \|W\|_1$$

Cost Function and Regularization

- Assume we would like to model the output y from input x and function $f(x; W_1, \cdots, W_T)$, where $W_1, \cdots, W_T$ are the parameters.
- Further assume the parameter we would like to regularize is W_j, by denoting the loss function l as well as the group lasso penalty $R_{l2,1}(\cdot)$, the corresponding cost function is:

$$C(W_1, \cdots, W_T) = \frac{1}{2N} \sum_{i=1}^{N} l(y_i, f(x_i; W_1, \cdots, W_T)) + \lambda R_{l2,1}(W_j)$$

- The best set of parameters $\hat{W}_1, \cdots, \hat{W}_T$ is obtained via minimizing the cost function C such that:

$$\hat{W}_1, \cdots, \hat{W}_T = \arg \min_{W_1, \cdots, W_T} C(W_1, \cdots, W_T)$$

Regularization on Linear Model

- The function of a linear model with input $x \in \mathbb{R}^d$ is formulated as:

$$f(x; \beta, \beta_0) = \beta^\tau x + \beta_0$$

- By choosing the loss function as square function and applying the group lasso penalty on β, the cost function is:

$$C(\beta, \beta_0) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - \beta^\tau x_i - \beta_0)^2 + \lambda \|\beta\|_1$$

- The cost function takes the form of a standard lasso problem.

Regularization on Multilayer Perceptron

- The function of a T-layer Multilayer Perceptron is:

$$f(x; W_1, \cdots, W_{T+1}, b_1, \cdots, b_{T+1}) = g_{T+1}(b_{T+1} + W_{T+1}^{\tau} g_T(b_T + W_T^{\tau} g_{T-1}(\cdots g_1(b_1 + W_1^{\tau} x)\cdots)))$$

- $b_1, \cdots, b_{T+1}$ and $W_1, \cdots, W_{T+1}$ are the biases and weights of each layer and $g_1, \cdots, g_{T+1}$ are the corresponding activation functions.
- Since the group lasso penalty serves the purpose of feature selection, we only apply it on the input layer:

$$C(W_1, \cdots, W_{T+1}, b_1, \cdots, b_{T+1}) = \frac{1}{2N} \sum_{i=1}^{N} l(y_i, f(x_i; W_1, \cdots, W_{T+1}, b_1, \cdots, b_{T+1})) + \lambda R_{l2,1}(W_1)$$

Regularization on Multilayer Perceptron

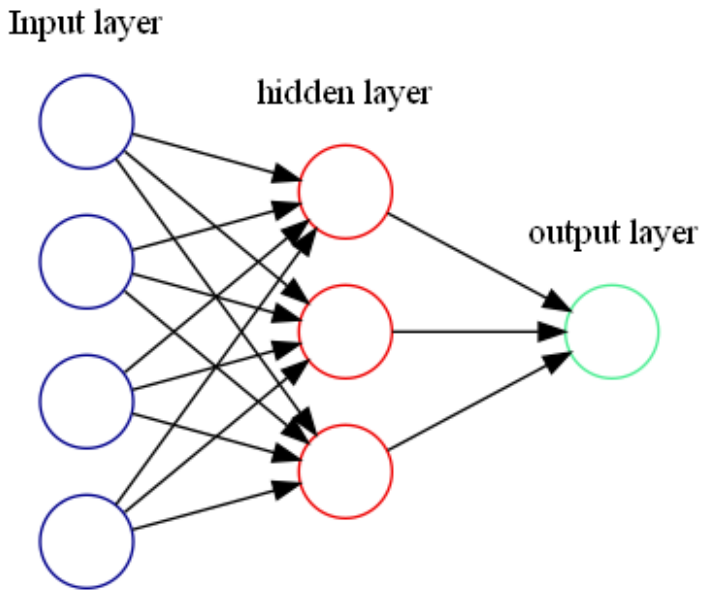


Figure 1: Multilayer Perceptron

Regularization on stacked RNN

- To simplify the illustration, we choose a multi-to-one, double-stacked Recurrent Neural Network with deep output, which takes in a sequential input of length T.
- By denoting the first and second hidden layer of RNN as $h^{(1)}$ and $h^{(2)}$, the RNN function goes as followed:

$$h_t^{(1)} = g_1(W_1^\tau h_{t-1}^{(1)} + U_1^\tau x_t + b_1)$$
$$h_t^{(2)} = g_1(W_2^\tau h_{t-1}^{(2)} + U_2^\tau h_t^{(1)} + b_2)$$
$$1 \leq t \leq T$$

  In the output:

$$y = h_2(a_2 + V_2^\tau h_1(a_1 + V_1^\tau h_T^{(2)}))$$

- b_1, b_2 and W_1, W_2 are parameters for the hidden layers in RNN; a_1, a_2 and V_1, V_2 are parameters for the output layers. g_1, g_2 and h_1, h_2 are the corresponding activation functions.

Regularization on stacked RNN

- The input $\mathbf{x}$ is a sequence of length T such that $\mathbf{x} = [x_1, \cdots x_T]$ and the final output function is denoted as $f(\mathbf{x}; b_1, b_2, W_1, W_2, U_1, U_2, a_1, a_2, V_1, V_2)$.
- The cost function of the corresponding RNN is:

$$C(b_1, b_2, W_1, W_2, U_1, U_2, a_1, a_2, V_1, V_2) =$$
$$\frac{1}{2N} \sum_{i=1}^{N} l(y_i, f(\mathbf{x}_i; b_1, b_2, W_1, W_2, U_1, U_2, a_1, a_2, V_1, V_2) + \lambda R_{l2,1}(U_1)$$

Regularization on stacked RNN

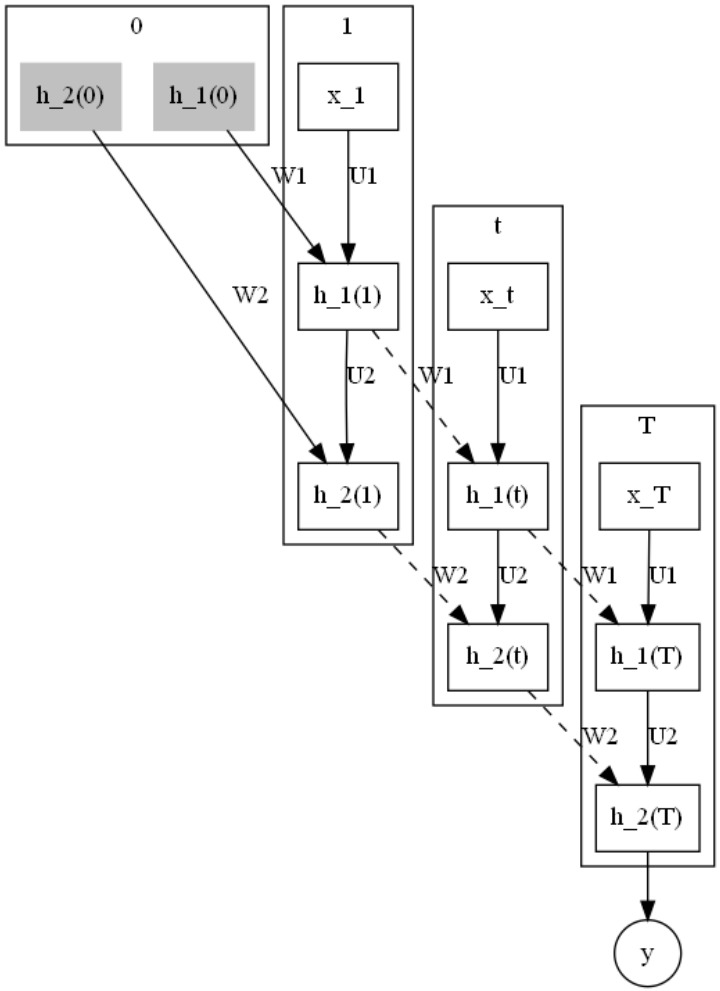


Figure 2: Stacked RNN

Model Details

Benchmark and Model Setup

- **Benchmark Evaluation** The benchmark to evaluate the model is the **correlation** between the predicted value $\hat{y}$ and the true label y.
- **Training set and Validation Set:** The training set and the validation set were splitted into proportion 5/7 and 2/7
- Two benchmarks were used for comparison: Linear Regression and Multilayer Perceptron without any regularizations.

Model	Training Correlation	Validation Correlation
Linear Regression	0.1416	0.0701
Multilayer Perceptron	0.1921	0.0752

Model Details of Multilayer Perceptron

- **Hidden Layer:** Three hidden layers are incorporated into the model, each with size 600, 300 and 200.
- **Activation Function:** For each hidden layer, the activation functions $g_1, \cdots, g_{T+1}$ are all sigmoid functions.
- **Loss Function:** Let $\hat{y}_i = f(x_i; W_1, \cdots, W_{T+1}, b_1, \cdots, b_{T+1})$, the loss function used here is the square loss function:

$$l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$$

Model Details of RNN

- **Hidden Layer in RNN:** The size of each of hidden layer in the stacked RNNs is 350
- **Hidden Layer in Output:** The size of the intermediate layer in the output is 200.
- **Activation Functions:** The activation functions in the stacked RNN part, g_1 and g_2 are both *tanh*; The activation functions in the output part, h_1 and h_2 are both *sigmoid*.
- **Loss Function:** Let $\hat{y}_i = f(\mathbf{x}_i; b_1, b_2, W_1, W_2, U_1, U_2, a_1, a_2, V_1, V_2)$, the loss function is the square loss function:

$$l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$$

Model Details and Performance

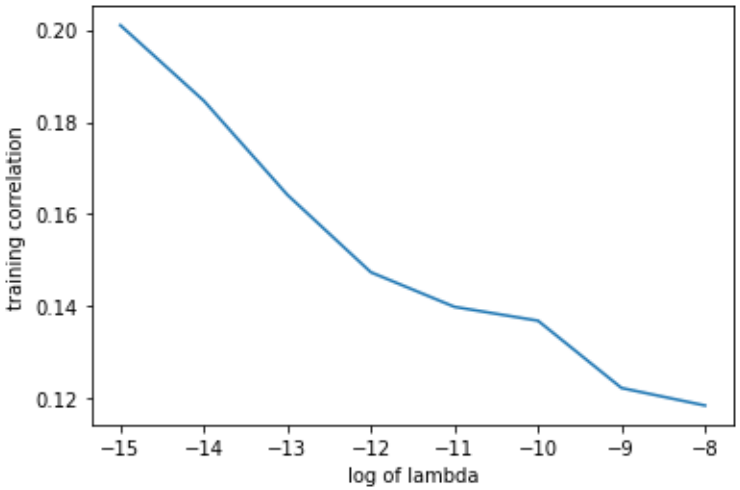
- **Warm Start:** Training with the group lasso penalty enables the use of warm start techniques. When tuning a new value of hyper parameter λ, we can use the previously trained value of the parameters to initialize.
- **Early Stop:** Since the benchmark for evaluating models is correlation, we will stop the training of current λ when validation correlation is at its highest. In practice, we stop training at the epoch when the correlation dropped by more than $3e-4$ compared to the current highest.

Model Details and Performance

- **Correlation against λ:** One important and significant feature of our model is the relationship between group lasso penalty hyper-parameter and correlation. Training correlation keeps increases when λ decreases, while the validation correlation first increases then drops as λ decreases.
- Example of group lasso penalty applied to Multilayer Perceptron:

Model Details and Performance

λ	best epoch	training correlation	validation correlation
1e-8	12	0.1185	0.0784
1e-9	3	0.1223	0.0816
1e-10	4	0.1369	0.0843
1e-11	4	0.1399	0.0804
1e-12	2	0.1474	0.0739
1e-13	5	0.1642	0.0716
1e-14	4	0.1846	0.0696
1e-15	7	0.2010	0.0684



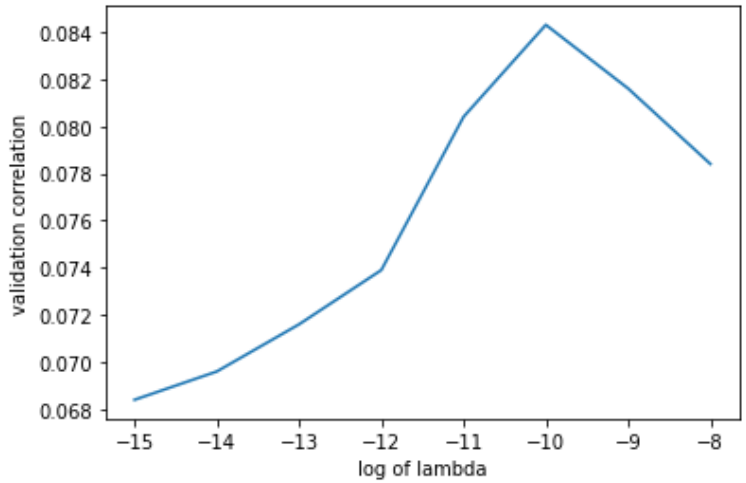


Figure 3: Correlation vs λ

Model Details and Performance

- **Distribution of norms of weights:** The other observation we have is the distribution of the norms of the weights. Consider the input parameter matrix (i.e., β in Linear Model, W_1 in Multilayer Perceptron, U_1 in stacked RNN), where $W = [w_1, \cdots, w_D]^\tau$, w_j corresponds to all the weights linking to feature j and the $2-norm$ of which, $\|w_j\|_2^2$ would somehow affect the importance of feature j to the model.
- We observed that the distribution of $\|w_j\|_2^2$ is different for features that are considered signals and those considered as noises.

Parameter Tuning and Model Selection

- In the high **signal-to-noise ratio (SNR)** setting, we have 240 true signal features and 100 random noise features, the distribution histograms are shown below, where $\lambda = 1e-7, 1e-8, \cdots, 1e-15$

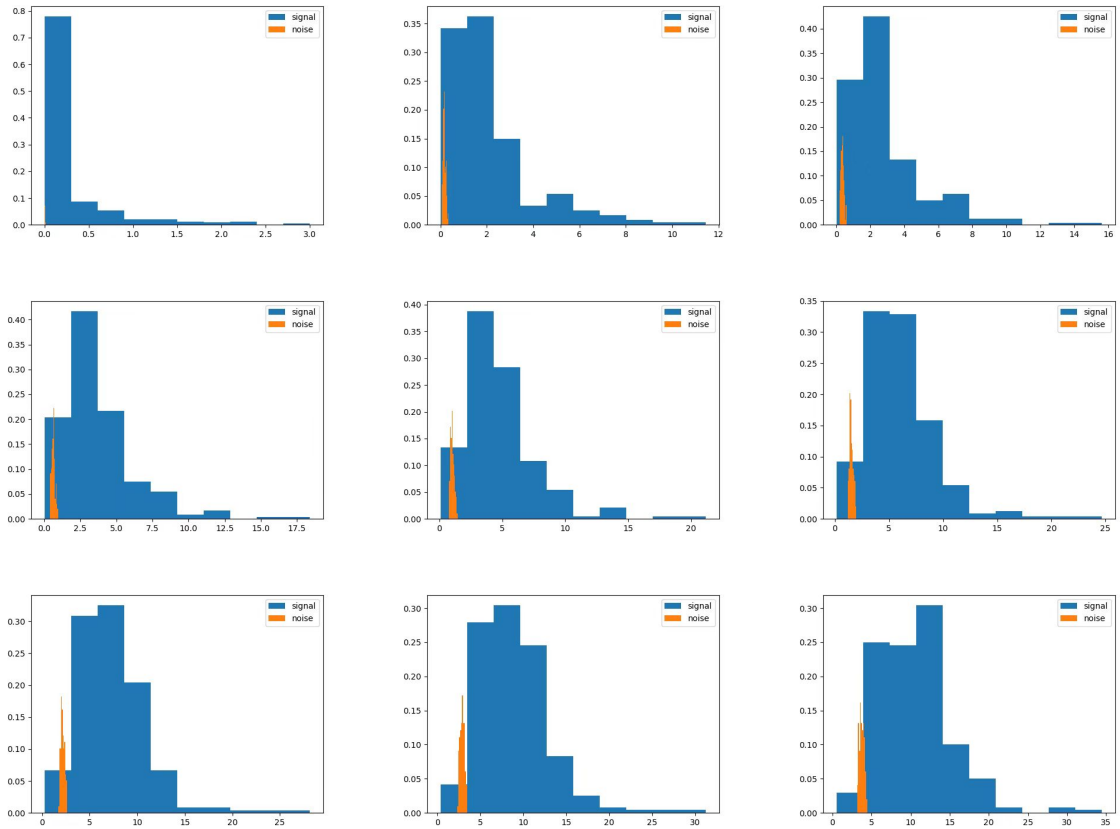


Figure 6: Distribution of high SNR

Model Details and Performance

- In the low **signal-to-noise ratio** setting, we have 240 true signal features and 700 random noise features, the distributions are shown below, where $\lambda = 1e-7, 1e-8, \cdots, 1e-15$

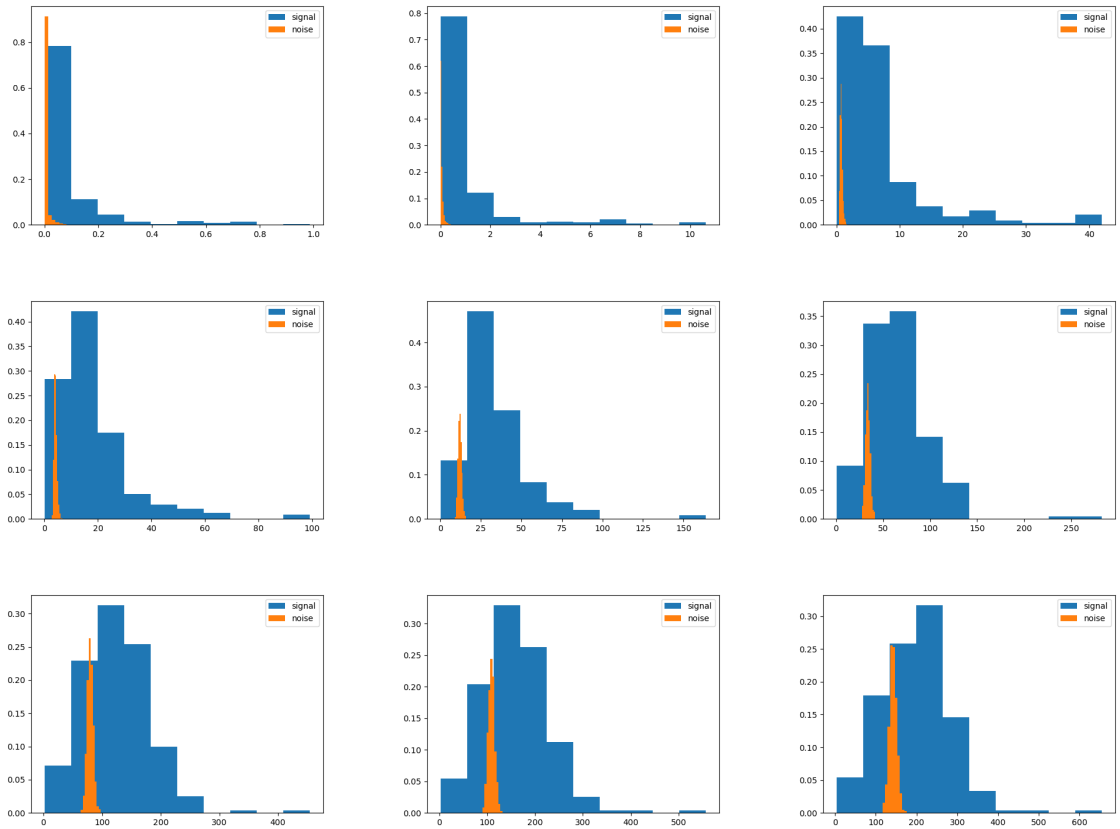


Figure 9: Distribution of low SNR

Model Details and Performance

- From the histogram, it is easy to observe that the distribution for signal features differs significantly from that for noise features; the distribution of the weight norms of noises are centered at a much smaller value and has much smaller variance.
- As λ becomes larger, the distribution of weights norms of both signals and noises are more likely to be suppressed to the level of 0.
- Compared to the low **SNR** setting, in the high **SNR** setting, it is much easier to distinguish the histogram for signal features from that for noise features. The weights of signal features are more likely to be buried within the weights of noise features when SNR is low.
- The difference between the distribution might not only be the result of different **SNR**, a different network structure might improve the distribution result.

Feature Selection

Feature Selection

- From the comparison of the two distributions, the feature selection rule could be based upon thresholding the two norm $\|w_j\|_2^2$.
- Given a fixed level of threshold, if the l_2 norm for weights linking feature j in the inputs layer is larger than the threshold value, the feature would be selected; otherwise the feature is neglected.

Feature Selection

- We would like to illustrate the power of thresholding by Multilayer Perceptron Model with $\lambda = 1e - 10$, where the validation correlation is at the highest.
- The number of features selected and the correlation after selection is presented in the following table:
- We used the weights from the previously trained model, but only use the selected features to get predicted values and obtain the correlation of the validation set. Compared with re-training the model with selected features, fitting with trained weights obtained previously would not much harm the correlation.

threshold	signal features	noise features	correlation
0.8	238/240	71/100	0.0807
0.9	236/240	27/100	0.0811
1.0	235/240	8/100	0.0812
1.1	235/240	0/100	0.0811

Automated Feature Selection

- In the previous data set, since the signal and noise features are already known, it is easy to manually tune the threshold of the weights for better selection performance.
- To automatically find the threshold, we add noises to the original data set and train the neural net with newly-added noise features.
- The automatic threshold is therefore calculated from the weights of the artificial noise features, using certain metric.
- The optimizer for each model is **Adam** with hyper-parameters set as $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1e-6$.
- Selection of the tuning parameter λ is based on the validation correlation.

Automated Feature Selection

- The corresponding automatic selection algorithm goes as followed:

Algorithm 1 Automated Feature Selection with Group Lasso Penalty

---

Given: Training set x_{train}, y_{train}; Validation set x_{valid}, y_{valid}; Sequence of decreasing parameter for group lasso penalty $\lambda_1, \cdots, \lambda_m$

- Generate random noise features, the number of which is approximately 5% the number of features of the original data set
- Train the model with penalty $\lambda_1, \cdots, \lambda_m$ using the augmented data set
- Select best $\hat{\lambda}$ where the validation correlation is highest
- Calculate the median value of the weights of generated noise features as threshold. Features in the original data set with weights larger than the threshold are selected.

Automated Feature Selection

- We perform features selection using **Algorithm 1** for all three models: **Linear Model**, **Multilayer Perceptron** and **Stacked RNN**.
- **High SNR:** The selection result is for the original data set with high signal to noise ratio, which has a total of 240 signal features and 100 noise features.
- In the following plot, the green circle represents results by Multilayer Perceptron, the orange circle represents results by Linear Model while the red circle represents results by stacked RNN.
- Selection results by by MLP and RNN are very much similar, while both of them outperform the result by linear model.

Automated Feature Selection

Figure 10: Selection with High SNR

- The corresponding best value of λ and correlations are:

Model	λ	Training Correlation	Validation Correlation
Linear	1e-12	0.1210	0.0713
Multilayer Perceptron	1e-10	0.1366	0.0812
Stacked RNN	1e-9	0.2154	0.0775

Automated Feature Selection

- **Low SNR:** The selection result is for data set with low signal to noise ratio, which has a total of 240 signal features and 400 noise features.
- In the following plot, the red circle is the selection results by Multilayer Perceptron while the black circle is the results by the Linear Model.

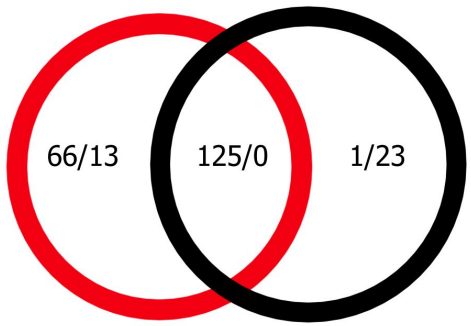


Figure 11: Selection results with Low SNR

Automated Feature Selection

- The corresponding best value of λ and correlations are:

Model	λ	Training Correlation	Validation Correlation
Linear	1e-13	0.1277	0.0718
Multilayer Perceptron	1e-10	0.1441	0.0807

Comparison with Input Decay

- Both methods need to specify a threshold value for the $l2$ norm of the input weights to do feature selection.
- Group Lasso penalty selects the features slightly better than input decay.
- Warm start could be adopted for group lasso penalty, greatly reducing the computation time.