Logistic Regression - SIS

Statistics for Big Data,
MSc Data Science, AUEB

Kafritsas Nikolaos
Mouselinos Spyridon
Mpaltzi Sofia

Introduction

The purpose of this document is to report on the different approaches towards an effective variable screening on the genome dataset. More specifically, the dataset referred to 78 patients and part of their gene profile. The aim was to identify an effective subset of those genes, among the 24,482 that were given, that contribute to the patients' survival. The task is Binary Classification, modeled under a Logistic Regression approach. Thus, the dependent variable was treated as a binary indicator (0/1).

Variable screening is the process of filtering out irrelevant features in a dataset, with the aim to reduce the dimensionality from ultrahigh to high while retaining all important features[1]. The screening technique we used is Sure Independence Screening (SIS) as implemented in SIS R library. We must note here that the SIS technique seems to be implemented in various ways across literature[2] and amidst packages.

The SIS library chosen for the experiments, uses as a metric the absolute correlation between each independent variable and the dependent variable. On the other hand, Its iterative variant (ISIS) uses the absolute marginal regression coefficient towards that goal.

Data Preprocessing

The dataset initially consisted of 24,482 rows and 80 columns. The following steps were performed in order to get the final dataset:

1. Columns "Systematic name" and "Gene name" were disregarded, as they were not useful for the current analysis. A dictionary of row number to Systematic name was kept for reference.

2. The information of life expectancy of patients - either more or less than 5 years - was included in column labels (sample labels). Therefore, the binary variable used as the dependent variable "Y" in the Logistic Regression was extracted from the column labels, taking the value "1" where label included the string ">5" and 0 where label included the string "<5". Labels also included the age of the patients, which was extracted as well.

3. String "NaN" was present in the dataset. This was replaced by the object "NA". Values of genes were transformed into numeric values. They were also normalized.

4. Patients 54 and 53 were removed for having 10,896 and patient 2,397 missing genes respectively. Genes having "NA" values were removed as well.

The dataset was transposed in order to get the final version of it, which had 76 rows representing patients, 23,626 columns containing the gene and age information and 1 column containing the binary-survival variable.

Sure Independence Screening

Theoretical Approach

Consider the problem of estimating a p-vector of parameters β from the linear model $y = X\beta + \varepsilon$, where:

- $y = (Y_1, \cdots, Y_n)^{\mathrm{T}}$ is an n-vector of responses,
- $X = (x_1, \cdots, x_n)^T$ is an n × p random design matrix with i.i.d. $x_1, \cdots, x_n$
- $\beta = (\beta_1, \cdots, \beta_p)^T$ is a p-vector of parameters,
- and $\varepsilon = (\varepsilon_1, \cdots, \varepsilon_n)^T$ is an n-vector of i.i.d. random errors.

When dimension p is high, it is often assumed that only a small number of predictors among $X_1, \cdots, Xp$ contribute to the response, which amounts to assuming ideally that the parameter vector β is sparse.

With sparsity, variable selection can improve estimation accuracy by effectively identifying the subset of important predictors, and also enhance model interpretability with parsimonious representation.

Under that assumption, Sure Independence screening works as a hard filter towards this goal by using a metric score $\Phi(.)$ between each candidate predictor and the target variable in a two-step fashion:

- In the first step, the metric function is applied between each predictor and the target variable. Usually the following metric functions are used, that in literature:
    1) $\Phi(X_i, Y) = |cor(X_i, Y)|$
    2) $\Phi(X_i, Y) = |b|$, $from\ fitting\ a\ glm\ model\ \ Y = a + b X_i$
    3) $\Phi(X_i, Y) = |b / s.e(b)|$, $from\ fitting\ a\ glm\ model\ \ Y = a + b X_i$
       $where\ s.e\ is\ the\ Standard\ Error\ of\ the\ coefficient$
- In the second step, a decreasing ordering of the scores is performed, thus providing a ranking of quasi-importance of the predictors. By specifying a hyperparameter m, the user can obtain the m most important predictors of the screening method.

Next, we attempt to apply this technique to our data.

Our Approach

As mentioned before, we are given a dataset of 76 rows (observations) by 23,626 columns (features). In order to reduce the features we apply the SIS technique. That gives us the m features with the largest absolute correlation to the dependent variable.

```
# Run SIS
Xsis = SIS::SIS(x=X, y=Y, family="binomial", nsis=m, iter=FALSE, standardize=FALSE)
```

Then, we fit a glm model of the binomial family, using those m features and the dependent variable.

```
# Fit glm model
model1= glm(Y~., data = X, family = "binomial")
```

For all models, we performed an additional 5-fold cross validation with a hand-crafted accuracy function as the cost function.

```
# Fit cv glm
cvmodel1 = cv.glm(cbind(Y,X), model1, K=5, cost=costfunc)
```

Attempt 1: m = 100

For the first attempt, we set m=100 and fit the glm model to the top 100 features. That results in an unstable model, with p-values for all features being either equal to 1 or NA (see example in next image). That means that with all other features present, no feature is of statistical importance for our model.

```
Coefficients: (25 not defined because of singularities)
             Estimate    Std. Error z value Pr(>|z|)
(Intercept)    4.1946    40850.2044       0        1
X89         -171.2999  7851597.2209       0        1
X512         166.7957  8588198.1724       0        1
X1558        -60.0087  2142159.9560       0        1
X1719        140.9224  4703499.2086       0        1
X2100       -226.0385  9909587.1491       0        1
X3224        195.2288  8710506.4452       0        1
X3232         59.2597  2259864.6224       0        1
X3251        -81.0763  4005455.2000       0        1
X3261       -166.8391  7856985.9064       0        1
X3461        192.5872  9481150.7401       0        1
X3492        -21.5306  1080087.5011       0        1
```

That observation is further verified by the cross validation accuracy being about 58% (varying for different seed values).

Attempt 2: m = 15

For the second attempt, we reduce m to 15 and fit the glm model to the top 15 features. That, again, results in an unstable model, with p-values for all features being close to 1 (see next image). That means that with all other features present, no feature is of statistical importance for our model.

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    99.854  20326.716   0.005    0.996
X512          -33.010  22298.735  -0.001    0.999
X3786         -78.493  18446.183  -0.004    0.997
X5377         -54.642  25986.617  -0.002    0.998
X6541          19.983  25506.290   0.001    0.999
X8576          80.287  18362.050   0.004    0.997
X9730         142.730  25326.492   0.006    0.996
X10889        -94.064  21397.985  -0.004    0.996
X12259       -161.597  35612.440  -0.005    0.996
X12781         -4.046  14447.695   0.000    1.000
X13800        -77.149  19551.319  -0.004    0.997
X15157       -209.543  36829.587  -0.006    0.995
X15390        -58.202  16372.241  -0.004    0.997
X16474         62.916  16268.748   0.004    0.997
X16523         -8.623  15520.442  -0.001    1.000
X19044       -170.757  30238.413  -0.006    0.995
```

Cross validation accuracy is now about 78% (varying for different seed values), which means that we have made steps towards a better feature selection.

Attempt 3: m = 10

For the third attempt, we further reduce m to 10 and fit the glm model to the top 10 features. We now have some important features, in the presence of all the other features in our model (see next image).

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   2.4905     1.2117   2.055   0.0399 *
X512          0.2736     1.2316   0.222   0.8242
X5377         0.7105     1.0791   0.658   0.5103
X8576         1.4180     1.1262   1.259   0.2080
X9730         1.7278     0.8412   2.054   0.0400 *
X10889       -3.0905     2.0029  -1.543   0.1228
X12259       -2.8540     1.3298  -2.146   0.0319 *
X15157       -3.8285     2.0613  -1.857   0.0633 .
X16474        0.9785     1.6547   0.591   0.5543
X16523       -1.4473     0.9030  -1.603   0.1090
X19044       -2.7894     1.1542  -2.417   0.0157 *
```

Cross validation accuracy is now about 80% (varying for different seed values). The features used in this model are translated to the following gene names:

D42044, Contig54742_RC, AJ011306, **Contig48328_RC**, AL080059, **NM_006544**, **Contig11065_RC**, Contig14882_RC, Contig20217_RC, **Contig63649_RC**

* the colored ones are the significant ones from glm model

Observations

By observing the outcomes of our experiments we can spot potential issues that arise with this approach:

1. Some unimportant predictors that are highly correlated with the important predictors can have higher priority to be selected by SIS than other important predictors that are relatively weakly related to the response.

2. An important predictor that is marginally uncorrelated but jointly correlated with the response can not be picked by SIS and thus will not enter the estimated model

3. The issue of collinearity between predictors adds difficulty to the problem of variable selection.

Iterative Sure Independence Screening

Theoretical Approach

The three issues spotted in the SIS approach will be addressed in the ISIS method, which allows us to use more fully the joint information of the covariates rather than just the marginal information in variable selection.

The ISIS works in two step loop:

- In the first step, we select a subset of $k1$ variables $A1 = \{Xi1, \cdots, Xik1\}$ using a SIS based model selection method such as the SIS-SCAD or SIS-LASSO.
  Then, we have an n-vector of the residuals from regressing the response Y over $\{Xi1, \cdots, Xik1\}$.

- In the second step, we treat those residuals as the new responses and apply the same method as in the previous step to the remaining $p - k1$ variables, which results in a subset of $k2$ variables $A2 = \{Xj1, \cdots, Xjk2\}$.

Fitting the residuals from the previous step on $\{X1, \cdots, Xp\} \setminus A1$ can significantly weaken the priority of those unimportant variables that are highly correlated with the response through their associations with $\{Xi1, \cdots, Xik1\}$, *since the residuals are uncorrelated with those selected variables in A1.* This helps solve the first issue.

It also *makes those important predictors that are missed in the previous step possible to survive*, which addresses the second issue above.

We can keep on doing this until we get disjoint subsets $\{A1, \cdots, An\}$ whose union has a size d (hyperparameter), which is less than n (original predictors).

Our Approach

For the problem of ultra-high dimensional variable selection, we now have the ISIS based model selection methods which are extensions of SIS based model selection methods. They are equipped with penalizing marginal regression methods, like SCAD and LASSO, and thus will reduce the number of predictors from ultra-high to model that is very close to the true sparse model.

Now, in order to perform the iterative alternative, we just have to pass the parameter iter as "TRUE". Note here that the parameter penalty is also complementary to the iter=TRUE option, since a penalty needs to be passed during the marginal regression iterations. Those iterations will be tuned based on a 5-fold cross validation schema as shown in the parameters tune="cv" and nfolds=5, on the accuracy metric - or as shown here type.measure="class". In our experiments we opted for the non-permutation variation of ISIS (perm=FALSE), namely "vanilla-ISIS" as mentioned above in the theoretical approach.

```
# Run ISIS with max N=20
Xisis = SIS::SIS(x=X, y=Y,
                 family="binomial",
                 penalty='lasso',
                 nsis=20,
                 iter=TRUE,
                 standardize=FALSE,
                 perm=FALSE,
                 tune="cv",
                 nfolds=5,
                 type.measure="class",
                 seed=666)
```

Attempt 1: m = 20

For the first attempt, we set m=20 and let the ISIS procedure to opt for up to 20 maximum predictors.

By plotting the results of the screened and regularized lasso model the ISIS produced, we observe that, we end up with only 16 predictors and with a small L1 penalty of ~ $7 * 10^{-5}$.

```
    Df  %Dev   Lambda
...
83 16 99.84 0.000127
84 16 99.85 0.000116
85 16 99.87 0.000106
86 16 99.88 0.000096
87 16 99.89 0.000088
88 16 99.90 0.000080
89 16 99.91 0.000073
```

Here are the coefficients of the final model.

```
> Xisis$coef.est
(Intercept)        X806       X3335       X6269       X8193       X9286      X10299      X10386      X11707
  0.9787601   0.4372242  -0.2295793   0.2207299   0.8339836   0.7815498  -0.4467524  -1.2293736  -1.4156262
     X12212      X13271      X14519      X15799      X15852      X17149      X18311      X22002
 -0.8443173   0.6259788  -1.2552013  -0.1308191  -0.4083422   0.1157921  -1.2187024  -0.5308333
```

This model gives 100% accuracy. Model features (X806, X3335 etc) are translated to the following gene names:

Contig35657, Contig31033_RC, NM_003751, **AJ011306**, **Contig48328_RC**, Contig51456_RC, **AL080059**, **NM_006544**, NM_016017, NM_016184, **Contig11065_RC**, Contig46443_RC, **Contig20217_RC**, Contig24138_RC, **Contig63649_RC**, NM_000605

* the colored ones are present in the SIS models

Attempt 2: m = 10

For the second attempt, we reduce m to 10 and let the algorithm decide again. By plotting the results of the screened and regularized lasso model the ISIS produced, we observe that, we end up with only 10 predictors and a L1 penalty of ~ $3 * 10^{-4}$.

```
    Df  %Dev   Lambda
...
64 10 97.96 0.000745
65 10 98.13 0.000679
66 10 98.30 0.000619
67 10 98.45 0.000564
68 10 98.58 0.000514
69 10 98.71 0.000468
70 10 98.83 0.000426
71 10 98.93 0.000389
```

Here are the coefficients of the final model.

```
> Xisis2$coef.est
(Intercept)       X3189       X5146       X7007       X8602      X10386      X11707      X14519      X17497      X18311      X23383
  0.6234809   0.6474836  -0.8588988  -0.1131186  -0.5462393  -1.2259110  -1.0418130  -1.3397160   0.3219593  -1.2244821  -0.7028451
```

This model gives again 100% accuracy. Model features (X3189, X5146 etc) are translated to the following gene names:

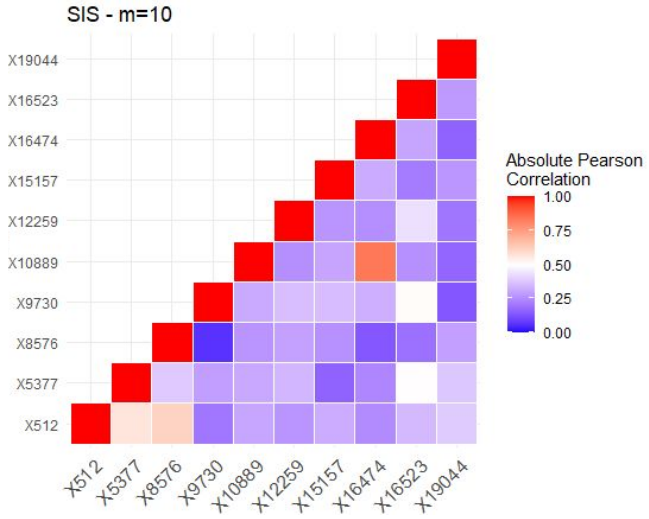
Contig8077_RC, **Contig54742_RC**, Contig23447_RC, NM_006168, **AL080059**, **NM_006544**, **Contig11065_RC**, Contig10670_RC, **Contig63649_RC**, NM_001553

* the colored ones are present in the SIS models. The underlined are present in the previous ISIS model.

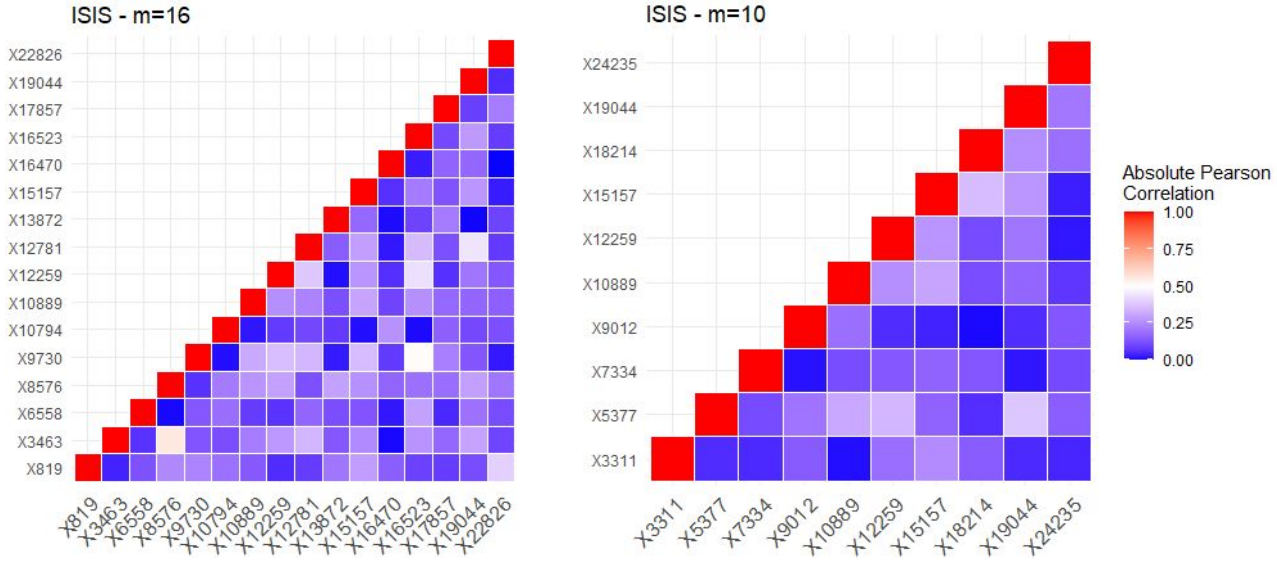
Observations

In the following diagrams, for the various models, we plot the Absolute Pearson Correlation of each feature with the rest of the features. Red color symbolizes a strong linear correlation between two features, while blue symbolizes no correlation between two features.

For the SIS model with m=10, we see that there are features that are highly correlated (e.g. X16474 - X10889).



In the ISIS model with m=16 (Attempt 1 result) there are only two pairs of features with absolute correlation larger than 0.5 (X3463-X8576 about 0.56 and X9730-X16523 about 0.51). Absolute correlation is below 0.5 for the rest of the pairs. In the ISIS model with m=10 (Attempt 2 result) absolute correlations for all the pairs of features are less than 0.5.



Conclusions

To sum up, our attempts to effectively screen the 25.000 predictors of the survival target variable came down to 2 solutions. By applying the SIS technique we managed to eliminate a vast amount of predictors, shrinking our ultra-high initial model (m=25.000) to models ranging between high(m=100) and low(m=10) complexity. However, due to the inherent issues bound with the SIS technique the predictive power of the logistic regression models on those predictors was of low and dubious quality. Afterwards, the Iterative SIS technique was implemented, that simultaneously reduced the predictors and fitted a Lasso based model on the final predictors. To this end, the lambdas used by the model were small and all of the predictors had non-zero coefficients in both of our experiments (m= 20,10). The predictive power of those models was significantly higher than before, while the magnitude of the final coefficients was smaller, signifying a more parsimonious model that its SIS counterparts (where the coefficients were high given that our features were Zeta Scaled). As a final sanity-check the pearson-correlation between the final predictors was measured both in SIS and ISIS techniques, confirming our initial hypothesis that part of the poor performance of the SIS screening was due to collinear features. The ISIS procedure produced far less collinear features and thus the small coefficients and better final performance is justified.

References

[1] Q. Huang, "Model-Free Variable Screening, Sparse Regression Analysis and Other Applications with Optimal Transformations," Open Access Diss., Aug. 2016, [Online]. Available: https://docs.lib.purdue.edu/open_access_dissertations/774.

[2] Jianqing Fan, Rui Song "Sure independence screening in generalized linear models with NP-dimensionality" , The Annals of Statistics Dec 2010, Pages 3567–3604