# Assignment #4: Algorithms for feature selection

Yahel Golan ID 204133151

# Part 1

# Summary:

# This year's topic is training classification models on microarray data which are used to identify genetic mutations and changes between individuals and populations. Microarray data are characterized by a large number of features (= the genes) and relatively few samples for training (diseased/healthy) and therefore suffer from the "curse of dimensionality". The exercise aims to test methods for reducing the dimensionality of the data by feature selection.

# Paper 1: C. Kang, Y. Huo, L. Xin, B. Tian, B. Yu Feature selection and tumor classification for microarray data using relaxed lasso and generalized multi-class support vector machine J. Theor. Biol., 463 (2019), pp. 77-91

# Background:

Embedded methods for reducing feature size contain feature selection during training. For example, the regularization method proposed by Tibshirani (1996), the least absolute shrinkage selection operator (Lasso), uses the penalty term to shrink the coefficients of certain variables to zero to achieve feature

selection, which provides a reference to the development of related methods. However, when

processing high-dimensional and small-sample data with a lot of noisy information, Lasso performs the same degree of shrinkage on all variables. The redundancy variables are shrinking towards 0, meanwhile, the related variables are affected to the same extent, generating a biased estimate. Lasso also has a slow convergence rate and tends to choose more variables.

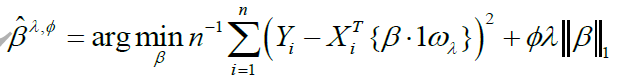
This article proposes using a relaxed Lasso. Relaxed Lasso uses two separate parameters to shrink the coefficients of the variables twice, providing a solution for continuous solutions including soft-threshold and hard-threshold estimators (Meinshausen, 2006). It uses the subset of variables chosen by Lasso for parameter estimation and variable selection for the second time.

# Relaxed Lasso:

Meinshausen (2007) proposed relaxed Lasso as a generalization of soft-thresholding and

hard-thresholding. The variable selection and shrinkage of coefficients is controlled by two

separate parameters  and :



where is the indicator function on the set of variables  . For all .  
Text

Description automatically generated

where thr variable in is selected by ordinary Lasso. Relaxed Lasso only uses elements of

for estimation. The parameter  controls variable selection as in ordinary Lasso. The

new parameter controls the shrinkage of coefficients. If  1, relaxed Lasso is identical to

Lasso. The shrinkage of relaxed Lasso with  1 compared to ordinary Lasso is reduced. We

define the relaxed Lasso for   0 as the limitation of the above definition for  0.

# Pseudo Code

1. Compute all ordinary Lasso solutions with the Lars algorithm in Efron et al. Let be the variable sets of results. Let be a sequence of penalty terms. When , if and only if .
2. For each , let . This is the direction of ordinary Lasso solutions. Let . If there is at least one component *l* so that , for compute all Lasso solutions on the set arying the penalty parameter between 0 and .  
   Otherwise, relaxed Lasso solutions for and are linear interpolation between and
3. The response variable is a linear combination of the predictor variables, , where ,.
4. the loss function of relaxed Lasso under parameter and is defined as following:

Meinshausen (2007) have shown that the convergence rate of relaxed Lasso is not affected by

the number of variables . As the gaining rate of increases, the convergence rate of Lasso

drops sharply. And if the redundancy variable increases too much, the ordinary Lasso's

convergence rate will also decrease. On the contrary, relaxed Lasso is more suitable for

sparse high-dimensional data.

# Taxonomy

**Label status:** Supervised (SVM)  
**Search Strategy:** Wrapper

# Pros & Cons

The advantages:

1. It is computationally just as fast as forward selection.
2. It produces a full piecewise linear solution path, which is useful in [cross-validation](https://en.wikipedia.org/wiki/Cross-validation_(statistics)) or similar attempts to tune the model.
3. If two variables are almost equally correlated with the response, then their coefficients should increase at approximately the same rate. The algorithm thus behaves as intuition would expect, and also is more stable.
4. It is effective in contexts where  (i.e., when the number of predictors p is significantly greater than the number of points )

The disadvantages:

1. With any amount of noise in the dependent variable and with high dimensional [multicollinear](https://en.wikipedia.org/wiki/Multicollinearity) independent variables, there is no reason to believe that the selected variables will have a high probability of being the actual underlying causal variables. Because LARS is based upon an iterative refitting of the residuals, it would appear to be especially sensitive to the effects of noise.
2. Since almost all [high dimensional data](https://en.wikipedia.org/wiki/High_dimensional_data) in the real world will just by chance exhibit some fair degree of collinearity across at least some variables, the problem that LARS has with correlated variables may limit its application to high dimensional data.

# Paper 2: Wang, Fei, et al. "Unsupervised soft-label feature selection." Knowledge-Based Systems 219 (2021): 106847.

# Background:

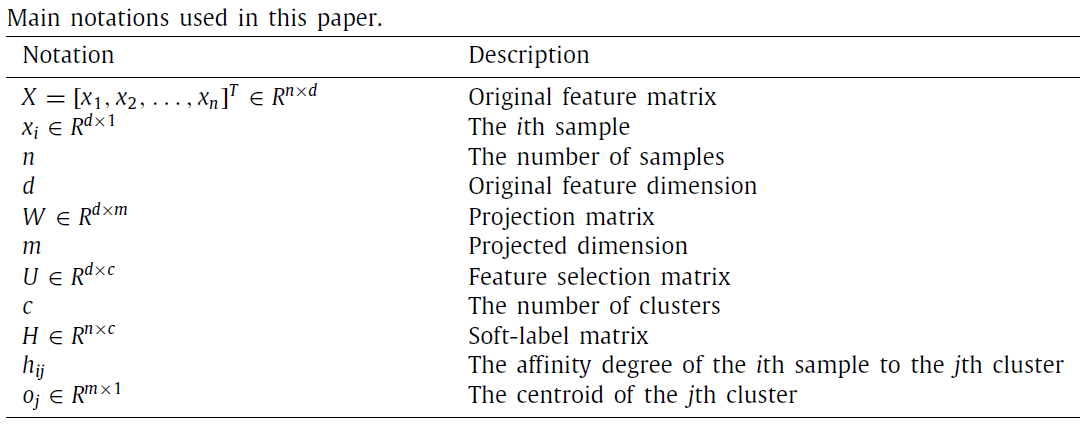
Oftentimes data labels are not directly available when performing feature selection, so unsupervised feature selection methods based on generated pseudo labels are preferred. However, most existing methods have the following issues: the learned pseudo labels are generated from the noisy original feature space which may degrade pseudo labels quality due to redundancies and outliers. Additionally, real world samples may belong to different clusters with certain probabilities (rather than only belonging to one cluster). Existing methods use hard-labels (binary) which ignore this issue, and in turn may lead to substantial information loss and hurt performance.

# Unsupervised Soft-label Feature Selection (USFS):

Wang et al. (2021) suggested a new model to tackle the issues mentioned above using soft-labels. The Unsupervised Soft-label Feature Selection (USFS) model first performs dimensionality reduction on the data to reduce the effects of noises, redundancies, and outliers. Then it uses the local distances in the low-dimensional subspace created to learn an affinity matrix with sparse constraint. This matrix is then determined as the soft-label matrix guiding the feature selection process.

# Pseudo Code

**Notations table:**



**Equations:**

The overall objective function of USFS:

Where are the regularization parameters.

Where and are diagonal matrices, with diagonal elements as and , respectively. The optimal solution of is formed by the eigenvectors of corresponding to the first smallest eigenvalues.

Updating : when and are fixed, becomes as follows:

Updating W: when and are fixed, becomes as follows:

Updating H: when and are fixed, can be derived as:

Where is the th column of

**Input:** Original feature matrix , regularization parameters , projected dimension , number of clusters .

**Output:** Feature selection matrix

* + 1. Randomly initialize and . To ensure deal with each by .
    2. While not converged:
       1. Fix update Calculate by , is formed by the eigenvectors of corresponding to the first smallest eigenvalues.
       2. Fix update by .
       3. Fix update by .
       4. Fix update . For each , theth row of is a vector with the th element as , and can be calculated by .
    3. Calculate and rank them in descending order. The features with the top rank order are the selected features.
    4. Return .

# Taxonomy

**Label status:** unsupervised  
**Search Strategy:** embedded

# Pros & Cons

The advantages:

* + - 1. USFS solves the intrinsic limitations of previous feature selection methods which do not use label guidance or use hard-label guidance, thanks to the consistency provided by its soft-labels.
      2. Soft-labels discriminative capability is improved by the learned projection matrix which reduces the impact of adverse noises, redundancies, and outliers.
      3. USFS has a simple yet efficient optimization method designed to solve the proposed objective function.

The disadvantages:

1. The number of selected features must be set manually. While this is good for our assignment (where we control the in SelectKBest) - generally t would be better to also have an option of adaptive feature selection automatically determining the optimal .

# Toy example

I’ve performed 100 iterations on the toy example dataset, which can be found in the folder “FINAL RESULTS”.

# Part 2

First algorithm – Relaxed Lasso: The paper shared their source code (<https://www.github.com/QUST-AIBBDRC/rL-GenSVM/)>, however as it is in R and I work with python, I had to reimplement some of the code in python, while making changes to make the output fit the pipeline required by the assignment. I was however able to access the R package “relaxo” through python, which did most of the heavy lifting.

Second algorithm – USFS: The paper shared their source code ([GitHub - wang-feifei/USFS-code](https://github.com/wang-feifei/USFS-code)), however it is in Matlab. After spending **well over 20** **hours** trying to convert the Matlab code to Python, I had to give up as there were just too many conflicts with Matlab counting arrays and matrices from 1 instead of 0, while at the same time the authors of the paper used the value “0” to refer to the end of a list in a matrix, which I couldn’t do because they relied on the fact that 0 cannot be an index in a vector/matrix. It was all too impossible to implement as the article is very complex. I then spent another several hours trying to access the Matlab code from python but without success as all the methods I found required Matlab version of at most 2014 while I have 2019. So after all the time and thought invested in this paper, I unfortunately did not manage to get it to work at all.

## B.

After running the pipeline for a few days on some databases I noticed there were oftentimes errors in the validation scores, because of the small amount of samples in the validation set due to the CV requirements (leave-pair-out/leave-one-out) which simply lead to errors which gave me NANs in all four test measures in the validation set (but not the train set, where I checked the pipeline to make sure it’s all good before letting it do a full run…). So after fixing the errors by choosing folds = 2 if under 100 samples (that was the only way for me), I had very little time before the submission date (and I’m still probably going to be late, as I’m typing this at 23:00). Therefore, I only used 6 databases:  bladderbatch cll sorlie subramanian khan leukemia

Instead of the 20 I planned to use: ['all','bladderbatch','breastcancervdx','cll','curatedovariandata',

                'bp','cbh','cs','css','pdx',

                'khan', 'sorlie', 'subramanian', 'sun', 'yeoh',

                'gli-85','leukemia','smk-can-187','tox-171','usps']

Additionally, I had to drop the mrmr filtering method as it took so much longer than everything else. I also had to drop a couple of k values and be left with k = [1,2,3,4,5,10,20,50,100].

**I wish I could run the pipeline on all the databases with all the filtering methods, but after spending hundreds of hours during the last few weeks coding, debugging, running, reading and trying to implement the articles, I simply ran out of time.**

***Results tables can be found in the folder “Final Results”.***

# Part 3

A disadvantage of lasso is that variable selection is conditional, which means that it does not work similarly to an oracle estimator. The irrepresentable condition is a necessary and sufficient criterion for lasso to satisfy consistency. However, relaxed lasso has been proved to be consistent without satisfying the strict condition.

**Relaxed lasso** can still retain a high rate of convergence and can lead to consistent variable selection no matter what the asymptotic result is. With huge quantities of data, **adaptive lasso** can efficiently filter out the model’s sparse solution while retaining oracle features. To combine the advantages of the preceding two models, I propose a relaxed adaptive lasso and demonstrate that it has the same asymptotic properties and excellent convergence rate as the relaxed lasso. Adaptive lasso estimation improves the shrinkage force to equalize the coefficients in lasso by applying a weight vector.

**Source for Part 3:** Zhang, R., Zhao, T., Lu, Y., & Xu, X. (2022). Relaxed Adaptive Lasso and Its Asymptotic Results. *Symmetry*, *14*(7), 1422.

# Part 4

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dataset Name | Number of samples | Original Number of features | Filtering Algorithm | Learning algorithm | Number of features selected (K) | Fold | Measure Type | Measure Value |
| khan | 63 | 2308 | f\_classif | SVC(C=0.1, kernel='linear', probability=True, random\_state=0) | 10 | 2 | AUC | 1 |
| khan | 63 | 2308 | f\_classif | LogisticRegression() | 10 | 2 | AUC | 1 |
| leukemia | 72 | 7070 | relieff | LogisticRegression() | 4 | 2 | AUC | 0.979930888 |
| cll | 22 | 12625 | relieff | GaussianNB() | 2 | 2 | AUC | 0.8 |
| bladderbatch | 57 | 22283 | relaxed\_lasso | SVC() | 10 | 2 | AUC | 0.987809073 |
| sorlie | 85 | 456 | relieff | LogisticRegression(C=0.1, random\_state=0) | 100 | 2 | AUC | 0.979212454 |
| subramanian | 50 | 10100 | f\_classif | GaussianNB() | 50 | 2 | AUC | 0.749375 |

As we can see, except for RFE, every filtering algorithm had a database where it got to excel, But RFE might have been the best at some point if I had time to go over more databases. We can also note that f\_classif got a perfect score both with a support vector machine and logistic regression in the khan database.

# Part 5

Friedman chi square Result: statistic=261.1057367829023, p-value=2.591861363181662e-56

As Friedman’s statistic was extremely significant, I continued to perform the nemenyi post-hoc test:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | f\_classif | relaxed\_lasso | RFE | relieff |
| f\_classif | 1 | 0.001236 | 0.001 | 0.001 |
| relaxed\_lasso |  | 1 | 0.001 | 0.050315 |
| RFE |  |  | 1 | 0.001 |
| relieff |  |  |  | 1 |

As we can see, the four filtering methods are all significantly different from each other in terms of AUC, except for relieff vs relaxed lasso – albeit the p-value is very close to 0.05, so perhaps with more data we’d get a significant result here as well.

These results indicate that the different filtering methods indeed lead to different results. As such, one must take care in choosing the best filtering method for one’s current task. Otherwise, results might suffer.

# Part 6

In LOOCV protocol, in order to obtain the AUC score, one must pool the predictions made in each CV round into one set (known as *pooling*). The problem is that pooling assumes that classifiers produced on different CV round all come from the same population. However, when computing AUC, some of the positive-negative pairs are constructed using data instances from different folds, thus the assumption might not hold. Therefore, LOOCV might lead to pessimistic/negative bias, i.e., underestimate classifiers performance.

Airola et al. (2011) suggest using leave-pair-out CV (LPOCV) to solve this bias while maximizing the use of available training data. In LPOCV, each positive-negative instance pair constitutes its own fold. This assures that only instances from the same round of CV are compared (thus guaranteeing the aforementioned assumption always holds). To obtain the AUC score, the CV estimate is calculated by averaging over all the positive-negative pairs.