# Assignment 5

Gandhar Vaidya

16th November 2015

## Contents



## 1 Abstract

This document provides a report on a few methods of feature and model selection and comparison of the respective methods' accuracy. For the experiments with feature and model selection , we are using 2 datasets from UCI repository - 1. Arcene data set and 2. Leukemia data set (sparse). In first section , how to implement a golub score - a feature selection method is being shown. It is followed by comparison of results using L1 and L2 SVM methods for computing accuracies using cross validation. In the 3rd section of the document , golub score , RFE with SVM and the subsample feature selection technique are compared as a function of number of features selected. This technique is then compared to the nested cross validation technique to select the best model selection strategy.

## 2 Golub Score

Golub score is a feature selection method. The formula for the golub score is given in the assignment's problem statement. The golub score was implemented using the following code.

```
def golub(X,y):
    neg = []
    pos = []
    result = []
    for i in range((X.shape[1])):
        for j in range(len(y)):
            if y[j] < 0:
                neg.append(X[j,i])
            else:
```

```
                pos.append(X[j,i])
        p_mean = np.mean(pos)
        n_mean = np.mean(neg)
        p_sd = np.std(pos)
        n_sd = np.std(neg)
        Num = abs(p_mean - n_mean)
        Den = p_sd + n_sd
        result.append(Num/Den)
    return result,result
```

**The same function is used for both the data sets. From the main function 2 calls are made to this function - 1 using arcene data and other using leukemia data. This pattern is followed for all the codes implemented in this document**

# 3 Comparison of L1 chosen features with use of all features

This section has 4 sub sections to it as we implement various strategies and methods and discuss the best method to select features as well as compare accuracies between L1 and L2 classifiers for both the data sets. Also we implement a "k-subsample" method and use it for both the datasets.

## 3.1 L1 and L2 SVM trained on all features

The following snippet of code shows how the accuracies were computed for the entire data set.

```
def L1(X,y):
    classifier = LinearSVC(penalty='l1',dual=False)
    classifier.fit(X,y)
    return np.mean(cross_validation.cross_val_score(classifier, X, y, cv=5))
```

The above code shows a function designated for computing accuracy using L1 SVM for both data sets.The above code returned accuracy value of 72% for arcene data set and 97% for the leukemia data set.

```
def L2(X,y):
    classifier2 = LinearSVC(penalty='l2',dual=False)
    classifier2.fit(X,y)
    return np.mean(cross_validation.cross_val_score(classifier2, X, y, cv=5))
```

The above code shows a function designated for computing accuracy using L2 SVM for both data sets. The above code returned accuracy value of 86% for arcene data set and 80% for the leukemia data set.

## 3.2 Using L2 on features selected by L1 SVM

The following code is used to obtain average number of features which get selected after 10 iterations using L1 SVM on both the datasets.

```
def nonzero(X,y):
    for i in range(10):
        classifier = LinearSVC(penalty='l1',dual = False)
        classifier.fit(X,y)
        nz = classifier.coef_
        nz2 = np.transpose(nz)
        count[i] = np.count_nonzero(nz2)
    return np.mean(count)
```

The number of features selected in every iteration of the above code differs and hence an average of the number of features is taken to discuss the results. The average value of number of features ranges from 540-560 for arcene data set whereas for the leukemia(sparse) data set, it ranges from 35 to 50.

The fit method does not actually transform the given data. It is the fit_transform method which produces a new matrix with only the selected features. The syntax is given by

```
X_transformed = classifier.fit_transformed(X,y)
```

Now to obtain accuracy for a L2 SVM tranied on features selected by the L1 SVM , we make use of the make_pipeline class. We could also run L2 SVM on the transformed matrix. The following function achieves the above.

```
def accuracy(X,y):
    cv = cross_validation.StratifiedKFold(y, 5, shuffle=True, random_state=0)
    classifier2 = LinearSVC(penalty='l2',dual=False)
    selector = LinearSVC(penalty ='l1',dual=False)
    l1_l2= make_pipeline(selector, classifier2)
    return np.mean(cross_validation.cross_val_score(l1_l2, X, y, cv=cv))
```

We pass L1 SVM as the selector and L2 svm as the classifier for the make_pipeline class. The accuracy obtained using this method was in the range of 77-80% using arcene data and close to 95% for the leukemia data set.

### 3.3 Using class RFECV to select number of features automatically

For this part , RFECV class is used along with L2 SVM as the classifier. The following code is used to implement RFECV using L2.

```
def RFE_CV(X,y):
    classifier = LinearSVC(penalty='l2',dual=False)
    classifier.fit(X,y)
    filter_selector = RFECV(classifier, step = 0.1)
    filter_svm = make_pipeline(filter_selector, classifier)

    return np.mean(cross_validation.cross_val_score(filter_svm, X, y, cv=cv))
```

The step size was taken to be 0.1,0.01 and accuracy values of 83-86% were recorded for arcene dataset and for the leukemia data set the accuracy was in the range of 80-83% . The result showed an increase to 90% when step was 50 for the leukemia data whereas remained in the range mentioned above for arcene data set.

### 3.4 K-subsample method

The k-subsample method is implemented using 30 subsample values. The code for implementing the k-subsample method is given below

```
def ksubsample(X,y):
    i = 0.8*X.shape[0]
    j = 0.8*y.shape[0]
    i = int(i)
    j = int(j)
    score = np.zeros(X.shape[1])
    for k in range(30):
        X , y = shuffle (X,y,random_state=0)
        X = X[:i]
        y = y[:j]
        classifier = LinearSVC(penalty='l1',dual=False)
        classifier.fit(X,y)
        nz = classifier.coef_
        nz2 = np.transpose(nz)
        ind = np.nonzero(nz2)
        l = list(ind[0])
        for k in range(X.shape[1]):
            for m in range(len(l)):
                if l[m] == k:
                    score[k]=score[k]+1
    return score,score
```

### 3.5 Discussion of the results

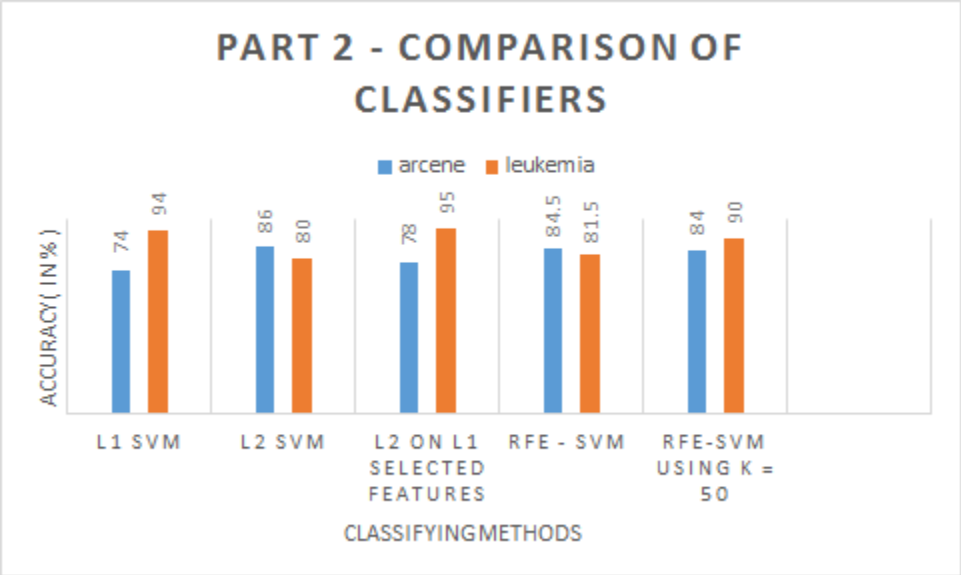


Figure 1: Part2-Classifier Comparison

As you can see from the chart above L1 gives a good result for the leukemia data but performs poorly on the arcene data whereas L2 does the exact opposite. The classifiers giving good results for both the data set are RFE SVM and L2 trained on features selected by L1. They provide reasonable results considering both the data sets.

## 4 Part 3

In this section , we compare 3 feature selection methods and based on the accuracy values , we discuss the best feature selection technique. We also compare these results with nested cross validation by varying the soft margin parameter "C".

### 4.1 Method Comparison

**1.Golub Score:** The following code snippet shows the implementation of L2 SVM for both the data sets using Golub score for feature selection. The function "part3" takes in 2 arrays - examples(X),labels(y) and returns mean value of accuracy. The code was tested using k = 30% of data and 50% of data. The accuracies were found in the range of 72-75% for the arcene dataset and for the leukemia data set the accuracy was found in the range of 78-81%. The value of k used for above experiment is varied in the range of 10 - 100% of data . The accuracy goes on increasing as we increase the number of features for the arcene data set but for the leukemia data set the accuracy stays constant for a while and then decreases as the number of features increase.

```
def part3(X,y):
    for i in range(10):
        j = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]
        cv = cross_validation.StratifiedKFold(y, 5, shuffle=True, random_state=0)
        filter_selector = SelectKBest(golub,k=int(X.shape[1]*j[i]))
        classifier = LinearSVC(penalty='l2',dual=False)
        filter_svm = make_pipeline(filter_selector, classifier)
        result.append (np.mean(cross_validation.cross_val_score(filter_svm, X, y, cv=cv)))

    return result
```

**2.K-subsamples method:** For the k - sub samples method we use the above code but instead of making use of golub as selector , we use k-sub sample method we implemented above. The difference from the above code lies just in this line of code. The accuracy obtained is in the range of 82-90% when values of k were changed from 10 - 100% of the data for leukemia. The accuracy goes on increasing as we increase the number of features for the arcene data set but for the leukemia data set the accuracy stays constant for a while and then decreases as the number of features increase. The value of arcene data set varies in the range 76-85% over the same range of k. The number of subsamples used for this method is 30% of the data.

```
filter_selector = SelectKBest(ksubsample,k=int(j[i]*X.shape[1]))
```

**3.RFE-SVM:** Here also,the only change is the selector line in the code. The accuracy using RFE SVM was found in the range of 52-54% for arcene data set and 94% for the leukemia data set. The RFECV class gives a better accuracy of around 80% for arcene data and for leukemia data set it gives an accuracy of 90%.The value of k used for above experiment is varied in the range of 10 - 100% of the data set . The accuracy goes on increasing as we increase the number of features for the arcene data set but for the leukemia data set the accuracy stays constant for a while and then decreases as the number of features increase.

```
filter_selector = RFE(classifier , step=0.1, n_features_to_select=int(j[i]*X.shape[1]))
```

## 4.2 Internal Cross validation using GridSearch

**1.Golub Score:** The following code snippet shows the implementation of L2 SVM for both the data sets using Golub score for feature selection. The function "chooseC" takes in 2 arrays - examples(X),labels(y) and returns mean value of accuracy.The values of C as you can see from the code are in the range 0.01 to 1000. The accuracies were found in the range of 75-81% for the arcene data set and for the leukemia data set the accuracies were in the range of 83-86%. The value of k used for above experiment is varied in the range of 10 - 100% of the data set . The accuracy goes on increasing as we increase the number of features for the arcene data set but for the leukemia data set the accuracy stays constant for a while and then decreases as the number of features increase.

```

def chooseC(X,y):
    for i in range(10):
        j = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]
        param_grid = [
    {'linearsvc__C': [0.01,0.1,1,10,100]},]
        cv = cross_validation.StratifiedKFold(y, 5, shuffle=True, random_state=0)
        classifier = LinearSVC(penalty='l2',dual=False)
        filter_selector = SelectKBest(golub,k=int(X.shape[1]*j[i]))
        filter_svm = make_pipeline(filter_selector,classifier)
        grid_search = GridSearchCV(filter_svm, param_grid=param_grid)
        result.append (np.mean(cross_validation.cross_val_score(grid_search, X, y, cv=cv)))

    return result
```

**2.K-subsamples method:** For the k - sub samples method we use the above code but instead of making use of golub as selector , we use k-sub sample method we implemented above. The difference from the above code lies just in this line of code. The accuracy obtained is in the range of 88-90% for the leukemia data set and that for the arcene data set is around 83-85%. The value of k used for above experiment is varied in the range of 10 - 100% of the data set . The accuracy goes on increasing as we increase the number of features for the arcene data set but for the leukemia data set the accuracy stays constant for a while and then decreases as the number of features increase. The number of subsamples used for this method is 30% of the data.

```
filter_selector = SelectKBest(ksubsample,k=int(j[i]*X.shape[1]))
```

**3.RFE-SVM:** Here also we just change the selector line in the code and change it to the following. The accuracy using RFE SVM was found in the range of 57% for arcene data set and 97% for the leukemia data set. If, RFECV class is used instead of the normal RFE a significant increase in accuracy is seen for arcene data. The accuracy rises to 84% whereas for the leukemia data , it comes down to 92%. The value of k used for above experiment is varied in the range of 10 - 100% of the data set . The accuracy goes on increasing as

we increase the number of features for the arcene data set but for the leukemia data set the accuracy stays constant for a while or decreases as the number of features increase.

```
filter_selector = RFE(classifier, step=0.1, n_features_to_select=int(j[i]*X.shape[1]))
```

## 4.3 Comparison of results

**To obtain the below charts , average values of the accuracies over a range of number of features obtained from above experiments are being used. Since there was no mention of specific plots against number of features , those plots are not provided in the report. I have used the average of all the accuracy values as a means to decide the best feature selection method and also the model selection method. The general trend seen from plotting accuracy vs number of features is explained above.**

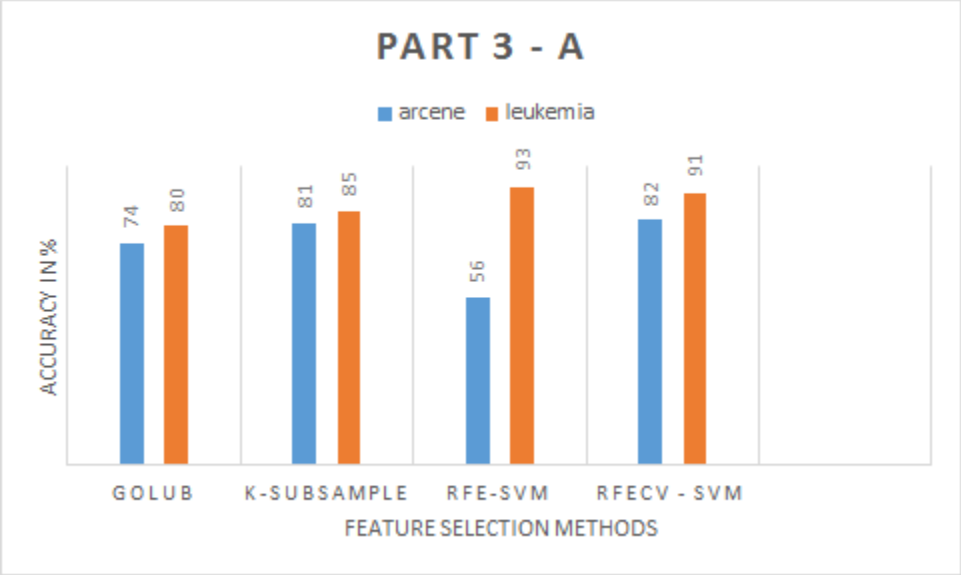


Figure 2: Part3-Feature Selection Methods Comparison

As you can see from the chart above , that the RFECV with SVM feature selection gives the best results. It is followed by k-subsample method and then Golub Score method. The RFE-SVM returns a very poor accuracy for arcene data set but performs very well for the leukemia data set.

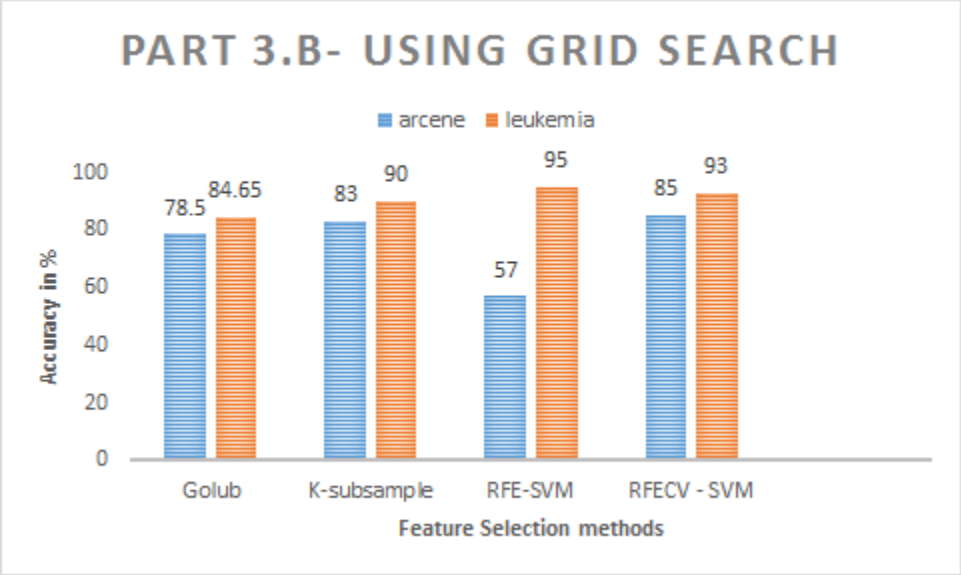


Figure 3: Part3- Using Grid Search

As you can see from both the charts above it is clear that the internal cross validation using a grid of values yields a better accuracy than just the normal cross validation without utilizing the soft margin parameter "C" Hence, the internal cross validation model is a better option when it comes to model selection.

# 5 References

1. Code provided in the schedule page featur_selection.py
2. Slides provided in the class
3. http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html
4. http://scikit-learn.org/stable/modules/model_evaluation.html