COMP3212 Major Assignment 2020

Using Machine Learning Techniques to Understand and Classify Lung, Breast and Skin Cancers

[redacted]

Carl Richardson - [redacted]

[redacted]

(Three Persons Project)

A report presented for the module of
Computational Biology

Department of Electronics and Computer Science
University of Southampton
28/05/2020

1 Introduction

The purpose of this project was to find a subset of genes for classifying which cancer- known to be skin, lung or breast, a sample cell was from. The identified genes were used to establish a more computationally efficient classifier; as well as focus the scope of study for cancer researchers. To achieve this goal, three data sets were identified which contained samples of each cancer type. These were used to train clustering, feature selection, dimensionality reduction and classification models.

The clustering model was used to prove the data was sufficiently separated to allow for good classification. The feature selection model filtered out the genes which contained redundant information whereas, the dimensionality reduction model transformed the data set into an optimal, lower dimensional, space for separating the classes. Feature selection and dimensionality reduction models were used as pre-processing steps for improving efficiency and performance of the classification model. A variety of classification algorithms were used for validating the performance. The classification performance with and without the pre-processing steps were compared against each other. From this, a good classification algorithm was found, and the effectiveness of each pre-processing step was evaluated.

2 The Data

The datasets were selected from the UCSC archive (1). Each dataset took the form of a gene expression matrix with the same 21,816 genes. Each column represented the gene expression levels from a single experiment (cancer cell sample) and each row represented the expression of a gene across all experiments. Each element of the matrix was a base 2 log of the gene expression ratio denoted by equation 1.

$$\begin{aligned} y &= \log_2 (\text{Gene Expression Ratio}) \\ \text{Gene Expression Ratio} &= \frac{T}{R} \end{aligned} \tag{1}$$

The numerator, T, is the gene expression level of the testing sample (cancer cell) and the denominator, R, is the gene expression level of the reference sample (normal cell). The value of y was interpreted as equation 2 shows. A gene expression can be regulated by a wide range of mechanisms used, by cells, to increase or decrease the production of specific gene products (protein or RNA). When a gene was up-regulated, its production increased, with respect to the reference, while the opposite was true for down-regulation.

$$\begin{aligned} y = 0 &: \text{represented no change i.e. } T = R \\ y > 0 &: \text{represented upregulation i.e. } T > R \\ y < 0 &: \text{represented downregulation i.e. } T < R \end{aligned} \tag{2}$$

To form the input data matrix, 95 skin, 130 lung and 51 breast cancer samples were obtained from (2) (3) (4). Each sample was labelled by its cancer type and contained the log of the gene expression ratio for 21,816 genes. Hence, the input data matrix had 276 rows and 21,816 columns. When set up in this way, the genes were the features of the data matrix as opposed to the samples.

3 Clustering

Clustering is an unsupervised process that allows for patterns to be found within a dataset. The aim was to be able to find a distinction between lung, skin and breast cancer. Collating each cancer dataset into one, a range of clustering algorithms (5) can be used to see if there was a separation between the three types of cancers. Table 1 shows the clustering analysis.

Clustering algorithms that use a euclidean approach such as Mini Batch K Means, Agglomerative and Birch were able to find three clusters, each relating to one of the cancers. Algorithms that use a hierarchical approach such as DBSCAN and OPTICS, struggle to find a separation. This shows that the dataset will allow for classification of a cancerous cell into one of three classes.

Clustering Algorithm	**Number of Clusters Found**
Mean Shift	1
Mini Batch K Means	3
Agglomerative	3
DBSCAN	1
OPTICS	1
Affinity Propagation	24
Birch	3

Table 1: Number of clusters found using each clustering algorithm

Even though the dataset shows separation between the cancers, feature selection and feature reduction performed on the dataset would further aid the classification of cancerous cells.

4 Feature Selection

A linear support vector classification (SVC) model (5), penalised by its L1- norm, was used as the feature selection model. This classification model penalises weight vectors with larger norms; hence, it tends to find sparse solutions. For feature selection, the features corresponding to the non-zero elements of the solution were interpreted as the features important for classifying the labelled data whereas the features corresponding to zero elements of the solution were considered redundant.

The linear SVC was trained for a range of regularisation parameter values. The number of features selected by each model is shown in figure 1. This figure shows that as the regularisation parameter was increased, so was the number of features selected.

The mean classification accuracy was then computed for each model. This is shown by figure 1, where the number of features was plotted against the mean classification accuracy. The minimum number of components, which achieved the maximum classification accuracy of 0.82, was 64. The mean accuracy decreased gradually between 64 and 14 features where the classification accuracy was 0.72. When the number of features was less than 14, the classification accuracy decreased rapidly.

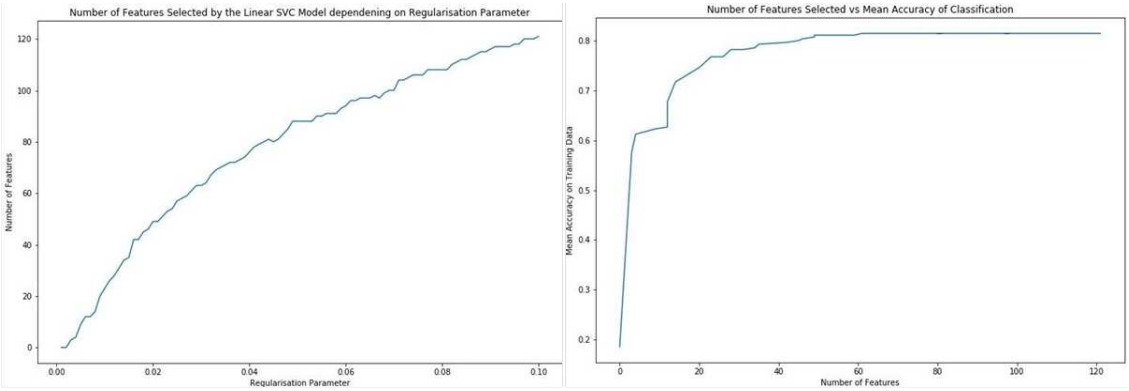

Figure 1: Number of features selected dependant on the regularisation parameter, Number of features selected vs classification accuracy

The 23 most informative genes corresponding to the solution found using a regularisation parameter value of 0.01 was used to reduce the number of features used during classification of the cancers. This model had a classification accuracy of 0.77. The names of the 23 most informative genes found by feature selection were:

201820_at, 206153_at, 206276_at, 206376_at, 209125_at, 209278_s_at, 209810_at, 210437_at, 21173_x_at, 213240_s_at, 214877_at, 215145_s_at, 215621_s_at, 216623_x_at, 217626_at, 219140_s_at, 219554_at, 220057_at, 220414_at, 220779_at, 221728_x_at, 221854_at, 37004_at.

5 Feature Reduction

Unlike feature selection, where the best N features are selected, dimensional reduction aims to reduce the size of the feature set whilst preserving the information within data. Reducing the number of features has many benefits. These include, removing redundant correlated features that do not help describe the data and speeding up computations.

Principal Component Analysis (PCA) is a technique where features, are transformed into a new set of features, which are linear combination of original features. These new set of features are known as principle components. These become the new set of axis for the data in a lower dimension.

PCA uses the idea that by maximising the variance of the new features, the amount of information retained from the original data set is maximised. Principle components are obtained in such a way that first principle component accounts for most of the possible variation of original data after which each succeeding component has the next highest possible variance. Figure 2 shows that the first 250 principle components can explain 99% of the original 21816 features.

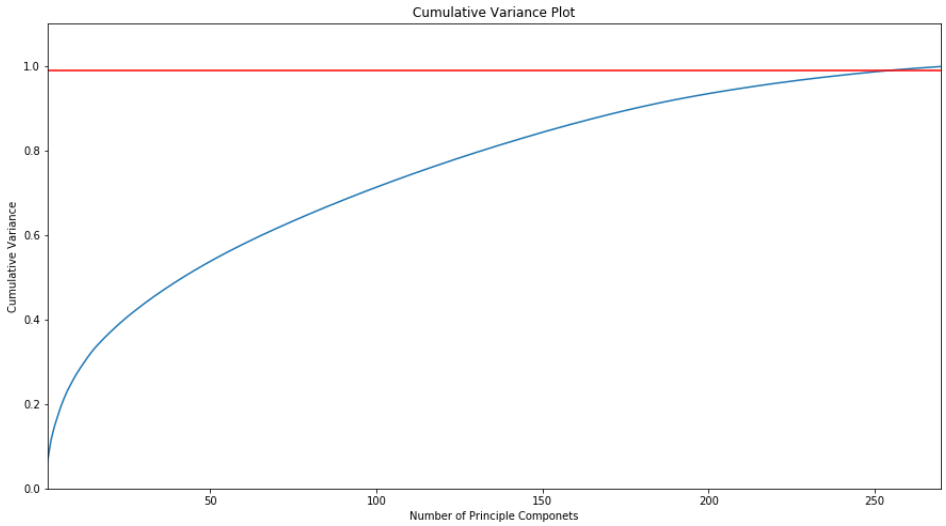

Figure 2: Number of principle components selected vs data accuracy

6 Classification

Classification is necessary to be able to assign labels to unknown samples. The goal of the classification is to assign labels of the type of cancer they most closely describe, hence the labels will be “skin”, “breast” and “lung”.

By splitting the dataset into training and testing sets, the classifiers can be initially trained to provide a more accurate result. The percentage of training data is set to 40% of the total dataset, this can be changed however was found to be a good middle ground between good results and a good number of usable results, as too much training tends to lead to less results.

To this end, a good supervised classifier was needed. A suite of classifiers (5) and their respective accuracy scores was used to determine the most successful classifier for this task. Each classifier returns a percentage score of how accurate it classifies the samples. The suite was run on the entire original dataset of all three cancers, the feature selection dataset and the feature reduced dataset, providing the accuracy table 2.

Classifier Algorithm	**Original Dataset Accuracy**	**Feature Selection Accuracy**	**Feature Reduction Accuracy**
Nearest Neighbor	54.1%	84.7%	32.4%
Linear SVM	38.7%	80.2%	19.8%
RBF SVM	44.1%	62.2%	45.1%
Gaussian Process	44.1%	82.9%	36.0%
Decision Tree	91.0%	85.6%	68.5%
Random Forest	82.9%	96.4%	68.5%
Neural Net	47.7%	91.0%	44.1%
AdaBoost	81.1%	94.6%	72.1%
Naive Bayes	99.1%	98.2%	92.8%
QDA	29.7%	86.5%	21.6%

Table 2: The accuracy of each classifier algorithm using different dataset formations

Much like the clustering predicted, the original dataset could be used to classify a cancerous cell. It performed averagely, with some classifier algorithms such as AdaBoost, Decision Tree, Random Forest and Naive Bayes performing admirably.

The feature selection dataset, which only included the 23 informative genes, performed extremely well when being used by all but one classifier algorithms with the highest consistent accuracy rates. This shows that there are features within the original dataset that hinder the classification of cancerous cells.

The feature reduction dataset, consisting of the 250 largest principle components, performed much worse than expected. Using PCA was supposed to remove unwanted and useless features, however performed worse than the original dataset.

One possible reason for a poor result could be the assumption that features with high variance equals more information is wrong. Another reason could be the assumption of the statistical importance of the mean and covariance of the data. PCA uses the eigenvectors of the covariance matrix which finds the independent axes of the data under a Gaussian assumption. In the case of non-Gaussian data, PCA simply de-correlates the axes. There is no guarantee that the directions of maximum variance will contain good features for the separation of cancers (6).

The best classifier algorithm was found to be Naive Bayes. It had an accuracy within the 90% range for each dataset. Naive Bayes classifiers have worked quite well in many complex real-world situations. “An analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of Naive Bayes classifiers” (7). An advantage of Naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification (8).

7 Conclusion

In conclusion, the informative genes important for classifying lung, skin and breast cancer were found using feature selection techniques. The identified genes were successfully used to establish a more computationally efficient classifier; as well as focus the scope of study for cancer researchers. Before continuing with this line of research, these results should be evaluated from a biological perspective. This will determine whether the interpretation of the results fits with established work in the field.

The best classifier algorithm for the cancer dataset was found to be Naive Bayes. It had an accuracy of 96.7%, due to the small amount of training it requires. If more training data was available the other classifiers such as Random Forest, Decision Trees and AdaBoost may have been alternative options. Future work could look into how the classifier treats other cancerous cells, outside its scope. This could expose the weaknesses of Naive Bayes.

Using PCA as the feature reduction technique was found to a poor decision. This was due to its reliance on statistical assumptions, that might not manifest themselves in biological systems. Future work could look into using supervised approaches such as Fishers LDA, that do not rely so much on statistical inference.

References

[1] "Cancer database," 2020. [Online]. Available: https://xenabrowser.net/datapages/

[2] "Breast cancer dataset," 2020. [Online]. Available: https://xenabrowser.net/datapages/?dataset=ucsfNeve_public%2FucsfNeveExp_genomicMatrix&host=https%3A%2F%2Fucscpublic.xenahubs.net&removeHub=https%3A%2F%2Fxena.treehouse.gi.ucsc.edu%3A443

[3] "Skin cancer dataset," 2020. [Online]. Available: https://xenabrowser.net/datapages/?dataset=lin2008_public%2Flin2008Exp_genomicMatrix&host=https%3A%2F%2Fucscpublic.xenahubs.net&removeHub=https%3A%2F%2Fxena.treehouse.gi.ucsc.edu%3A443

[4] "Lung cancer dataset," 2020. [Online]. Available: https://xenabrowser.net/datapages/?dataset=raponi2006_public%2Fraponi2006_genomicMatrix&host=https%3A%2F%2Fucscpublic.xenahubs.net&removeHub=https%3A%2F%2Fxena.treehouse.gi.ucsc.edu%3A443

[5] "Sklearn library," 2020. [Online]. Available: https://scikit-learn.org/stable/index.html

[6] "Why is principal component analysis a bust for genetics?" 2020. [Online]. Available: https://amethix.com/why-is-principal-component-analysis-a-bust-for-genetics/?fbclid=IwAR1Nv-17cy2PrB2ZKHeK8AMQG90DJT8DvINQz9UW7-qT2QqHQCEWoBnHfJ4

[7] "The optimality of naive bayes," 2020. [Online]. Available: http://www.cs.unb.ca/~hzhang/publications/FLAIRS04ZhangH.pdf

[8] "Naive bayes classifier 2020," 2020. [Online]. Available: https://en.wikipedia.org/wiki/Naive_Bayes_classifier

8 Appendix

This appendix contains the relevant code and results for proof of the work in this report.

```
# To support both python 2 and python 3
from __future__ import division, print_function, unicode_literals
%matplotlib inline

import numpy as np # numerical computation packages in python
import pandas as pd # pandas is a library for handling datasets
import matplotlib.pyplot as plt # plotting routines

#feature analysis
from sklearn.svm import LinearSVC
from sklearn.feature_selection import SelectFromModel


#importing data and transposing matrix- features are now the columns
location = 'C:/Users/Carl/Documents/Year3/CompBio/CancerAnalysis/'
df_breast = pd.read_csv(location+'Breast_ucsfNeveExp_genomicMatrix', delimiter= '\s+', index_col='probe').T
df_lung = pd.read_csv(location+'Lung_raponi2006_genomicMatrix', delimiter= '\s+', index_col='probe').T
df_skin = pd.read_csv(location+'Melanoma_lin2008Exp_genomicMatrix', delimiter= '\s+', index_col='probe').T


#sorting each dataframe into the same order
df_breast.sort_index(axis=1, inplace=True)
df_lung.sort_index(axis=1, inplace=True)
df_skin.sort_index(axis=1, inplace=True)

#checking each data frame has the same full set of features
print("Number of common features:")
print(len(set(df_breast.columns).intersection(df_lung.columns, df_skin.columns)))
if(len(set(df_breast.columns)) == len(set(df_lung.columns)) == len(set(df_skin.columns))):
    print("Each dataset has same features")


#concatenating into a single dataframe
frames = [df_breast, df_lung, df_skin]
df_data = pd.concat(frames, sort=False)

#display(df_data)

#array of feature names for referencing once key features have been identified
feat_names = df_data.columns;
feat_names = feat_names.to_numpy();

#defining combined input data matrix, X, as numpy array
#X[i] = sample i
#X[i][j] = gene expression j of sample i
#X.T[i] = expression of gene i across all samples

X = df_data.to_numpy();


#defining combined output vector, y, as numpy array
y = [];

for i in range(len(df_breast)):
    y.append(0.0)

for i in range(len(df_lung)):
    y.append(1.0)

for i in range(len(df_skin)):
    y.append(2.0)

y = np.asarray(y)


#training Linear SVC- Linear models penalized with the L1 norm have sparse solutions: many of their estimated coefficients
#are zero. The non-zero coeffs correspond to the important features
```

```
lsvc = LinearSVC(C=0.01, penalty="l1", dual=False).fit(X, y)
#C controls the sparsity: the smaller C the fewer features selected
#dual=false: algorithm solves the primal optimization problem


#measuring performance (classification accuracy)
lsvc.score(X,y)

#When the goal is to reduce the dimensionality of the data to use with another classifier, they can be used along with
#feature_selection.SelectFromModel to select the non-zero coefficients.

model = SelectFromModel(lsvc, prefit=True) #Meta-transformer for selecting features based on importance weights


#finding index of the genes corresponding to the most important features
#the index corresponds to the index of X.T
#thus, X_new.T[0] == X.T[i] where i was the lowest index

index = []
for i in range(len(model.get_support())): #vector of true/false values.. true if feature is used in reduced model
    if(model.get_support()[i]==True):
        index.append(i)

index = np.asarray(index)


#This is the new matrix where the number of genes being expressed has been reduced
X_new = model.transform(X)

#training Linear SVC with reduced number of features
lsvc2 = LinearSVC(C=0.01, penalty="l1", dual=False).fit(X_new, y) #Linear Support Vector Classification
#C controls the sparsity: the smaller C the fewer features selected
#dual=false: algorithm solves the primal optimization problem


#measuring performance (classification accuracy) of model with reduced features
lsvc2.score(X_new,y)

#index of the features found by linear SVC
print("Number of important features: ", len(index))
print(index)

#storing the names of the important genes
important_genes = []
for i in range(len(index)):
    important_genes.append(feat_names[index[i]])

important_genes = np.asarray(important_genes)


print("The important genes are:")
print(important_genes)


#defining array of C values
C_vals = []
c = 0.10
for i in range(100):
    C_vals.append(c)
    c = c - 0.001

C_vals = np.asarray(C_vals)
#C_vals
#len(C_vals)

#finding number of components and corresponding scores for each C value
no_feats = []
scores = []
```

```
136
137 for i in range(len(C_vals)):
138     lsvc = LinearSVC(C=C_vals[i], penalty="l1", dual=False).fit(X, y)
139     model = SelectFromModel(lsvc, prefit=True)
140     X_new = model.transform(X)
141     no_feats.append(len(X_new[0]))
142     scores.append(lsvc.score(X,y))
143
144 no_feats = np.asarray(no_feats)
145 scores = np.asarray(scores)
146
147 #Number of features vs mean accuracy
148 plt.figure(figsize=(12,8))
149 plt.plot(no_feats, scores)
150 plt.xlabel('Number of Features')
151 plt.ylabel('Mean Accuracy on Training Data')
152 plt.title('Number of Features Selected vs Mean Accuracy of Classification')
153 plt.savefig('Scores.jpg')
154 plt.show()
155
156
157 #Regularisation Parameter vs the number of features
158 plt.figure(figsize=(12,8))
159 plt.plot(C_vals, no_feats)
160 plt.xlabel('Regularisation Parameter')
161 plt.ylabel('Number of Features')
162 plt.title('Number of Features Selected by the Linear SVC Model dependening on
    Regularisation Parameter')
163 plt.savefig('No_feats.jpg')
164 plt.show()
165
166
167
```