Applying Machine Learning to Cancer Genomics

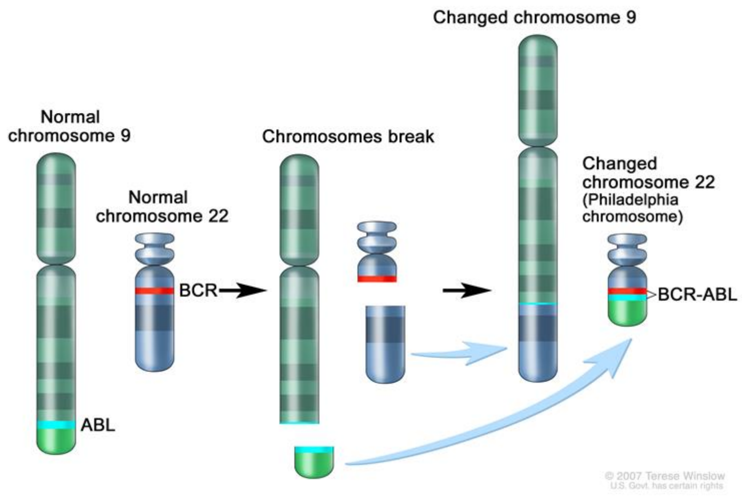
Melique Daley

1. Introduction

The Philadelphia Chromosome is a gene anomaly found in chromosome 22 where parts of chromosome 9 are transferred to it. Cells that contain the Philadelphia Chromosome are often found in Chronic myelogenous leukemia (CML) and sometimes found in Acute lymphocytic leukemia (ALL). Due to this, it is not sufficient to diagnose someone with CML or ALL since the Philadelphia Chromosome can be found in both. Consequently, this can lead to more general treatments for patients with these cancers instead of something more specific for that type of cancer. The purpose of this analysis is to gain insight if the mutation leads to ALL. If so, we can then provide better treatments for patients with ALL. Moreover, conduct further analysis to figure out what causes the mutation and find preventive actions.

2. Technical

The Philadelphia Chromosome is a defect that is caused by a reciprocal translocation (unusual rearrangement of the chromosome) of chromosomes 9 and 22. In doing so a fusion gene ABL1 is created in chromosome 9 that is attached to gene BCR (breakpoint cluster region on chromosome 22). This results in an elongated chromosome 9 and shortens chromosome 22.


Normal chromosome 9
Normal chromosome 22
BCR
ABL
Chromosomes break
Changed chromosome 9
Changed chromosome 22 (Philadelphia chromosome)
BCR-ABL
© 2007 Terese Winslow U.S. Govt. has certain rights

3. Data Filtering and Transformation

```
B_Cell <- grep("^B",as.character(ALL$BT))
gene_Phila <- which(as.character(ALL$mol.biol) %in% c("BCR/ABL","NEG"))
ALL_Phila <- ALL[,intersect(gene_Phila,B_Cell)]
```

B_Cell is a vector of B-Cell cancer types. gene_Phila is a matrix for genes that contain the BCR/ABL mutation and genes without it. All_Phila contains the intersection of gene_Phila and B_Cell.

```
ALL_Phila <- nsFilter(ALL_Phila,var.cutoff = 0.99)$eset
```

The purpose of nsFilter is to remove features. We remove features with a variance less than the cutoff. The lower the cutoff, the more features removed. Thus, nsFilter can be computationally expensive. For the sake of simplicity and computational cost, we set variance cutoff at 99%.

```r
rowIQR <- function(eSet) {
  numSamp <- ncol(eSet)
  lowQ <- rowQ(eSet,floor(0.25*numSamp))
  upQ <- rowQ(eSet,ceiling(0.75*numSamp))
  upQ-lowQ
}

standardize <- function(x) (x-rowMedians(x))/rowIQR(x)
ALL_Phila_exprs <- standardize(exprs(ALL_Phila))
```

The features in the data are measured differently from each other and because of this, they don't influence our predicated values correctly. We standardize all features so they all have equal influence in the model. The function rowIQR does this and each feature will range from -1 to 1 (some values will be outside but not too far which in fine). ALL_Phila_exprs holds all the extracted gene profiles.

3.1. Similarities Metrics

In order to gain insight into the data and find predictions we need to choose ways on comparing data points. That's how do we determine the correlations between the data. We mainly use distance metrics such as Euclidean and Manhattan. Data points with a smaller distance mean they are closely related, while data points that are "far away" are less related to one another.

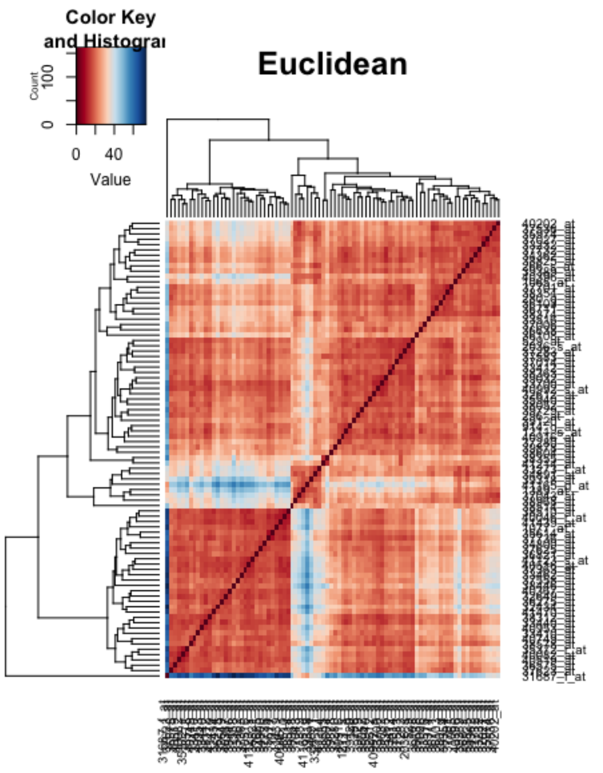
Euclidean: $f(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ **Manhattan**: $f(x,y) = \sum_{i=1}^{n}|x_i - y_i|$

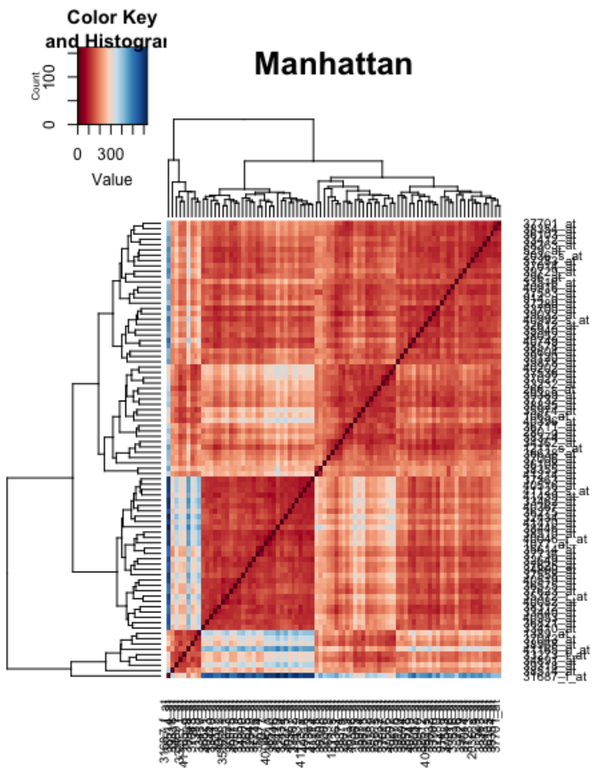
```r
eucD<- dist(exprs(ALL_Phila),method = "euclidean")
eucMat <- as.matrix(eucD)

manD<- dist(exprs(ALL_Phila),method = "manhattan")
manMat <- as.matrix(manD)
```

R's dist() function returns a distance/similarity matrix between the rows or columns of the data set using the given methods.
We can now visualize eucMat and manMat by using heat maps given by,

```r
hmcol <- colorRampPalette(brewer.pal(10,"RdBu"))(256)
heatmap.2(eucMat,main="Euclidean",tracecol=NA,
          symm = TRUE,col=hmcol,
          distfun = as.dist)
heatmap.2(manMat,main="Manhattan",tracecol=NA,
          symm = TRUE,col=hmcol,
          distfun = as.dist)
```

The red colour indicates those data points are closely related to each other while the Blue indicated that data points are loosely related to each other. That is, the intensity of red means the distance between the two points is small while blue would indicate the distance between the points they are far.





We see using these distance measures the data is highly correlated.

4. Machine Learning

```
NEGS <- which(ALL_Phila$mol.biol=="NEG")
BCR <- which(ALL_Phila$mol.biol=="BCR/ABL")
sample_1 <- sample(NEGS,22,replace = FALSE)
sample_2 <- sample(BCR,22, replace = FALSE)
Training_set <- c(sample_1,sample_2)
Test_Set <- setdiff(1:80,Training_set)
```

We now split the data to apply machine learning methods to the data. NEGS contains the genes that don't have the BCR/ABL mutation and BCR contains the genes that do. Our training set contains 22 data points from NEGS and BCR and our test contains all the data points, not in the training set.

4.1. K-Nearest Neighbors (KNN)

Given a positive integer k and a test observation x_0, the KNN classifier first identifies the k points in the training data that are closet to x_0, represented by N_0. Then estimate the conditional probability for class j as the fraction of points in N_0 where response values equal j:

$$P(Y = j|X = x_0) = \tfrac{1}{k} \textstyle\sum_{i \in N_0} I(y_i = j)$$

Then KNN applies Bayes' Theorem,

$$P(Y = y|X = x) = \tfrac{P(Y=y,X=x)}{P(X=x)}$$

and classifies the test observation x_0 to the class with the largest probability. In other words, given the unknown class of observation x_0 we can classify it by seeing how close the unknown point is to known points.

```
knn1 <- MLearn(formula =mol.biol~.,
             data=ALL_Phila,
             .method = knnI(k=1,l=0),
             trainInd = Training_set)
confuMat(knn1)
error_rate_knn <- round(100-sum(diag(confuMat(knn1)))/sum(confuMat(knn1))*100,2)
```

In the above code segment, we see k = 1 so we are finding the closet point to x_0 and classify it as k. The confusion matrix we get is,

		predicted	
		BCR/ABL	*NEG*
given	*BCR/ABL*	11	4
	NEG	7	13

where the diagonal entries are the correct predictions and the off-diagonal entries are incorrect classification. So, the algorithm had $11 + 13 = 23$ correct classification and $7 + 4 = 11$ incorrect classifications. The error rate of KNN_1 is 31.43%.

4.2. Diagonal Linear Discriminant Analysis (DLDA)

Given a set of classes, LDA is used to classify an observation to one of the classes. LDA works by given A response Y, what is the distribution of X,

$$\hat{P}(X|Y)$$

Then we find how likely each category is,

$$\hat{P}(Y)$$

and then use Bayes' Theorem to get,

$$\hat{P}(Y = k, X = x) = \frac{\hat{P}(X=x|Y=k)\hat{P}(Y=k)}{\hat{P}(X=x)}$$

We actually use a Diagonal Linear Discriminant Analysis (DLDA), a special case of LDA where the diagonal entries in the covariance matrix are set to zero. The code is given below,

```
dlda1 <- MLearn(formula=mol.biol~.,
              data = ALL_Phila,
              .method = dldaI,
              trainInd = Training_set)
confuMat(dlda1)
error_rate_dlda1 <- round(100-sum(diag(confuMat(dlda1)))/sum(confuMat(dlda1))*100,2)
```

with confusion matrix,

		predicted	
		BCR/ABL	*NEG*
given	*BCR/ABL*	12	3
	NEG	6	14

with an error rate of 25.71%. We see DLDA outperforms KNN_1 which is expected since in KNN we classify our test observations to the closet known observation which can be very misleading.
We can improve the above models with feature selection.

4.3. Improvements

```
training_ttets <- rowttests(ALL_Phila[,Training_set],"mol.biol")
order_training_ttest <- order(abs(training_ttets), decreasing = TRUE)
training_selected <- featureNames(ALL_Phila)[order_training_ttest[1:20]]
```

The above code does a t-test on each row on our training set to find which predictors (genes) are statistically significant. We then order the level of significance and take the top 20 predictors.

We run KNN again with the reduced genes,

```
knn2 <- MLearn(formula = mol.biol~.,
             data = ALL_Phila[training_selected,],
             .method = knnI(k=1,l=0),
             trainInd = Training_set)
confuMat(knn2)
error_rate_knn2 <- round(100-sum(diag(confuMat(knn2)))/sum(confuMat(knn2))*100,2)
```

with confusion matrix

		predicted	
		BCR/ABL	*NEG*
given	*BCR/ABL*	12	3
	NEG	5	15

with an error rate of 22.86%. As you can see KNN improved which is expected since we got rid of genes that are not statistically significant. Those predictors just added noise.

Now with DLDA,

```
dlda2 <- MLearn(formula=mol.biol~.,
              data = ALL_Phila[training_selected,],
              .method = dldaI,
              trainInd = Training_set)
confuMat(dlda2)
error_rate_dlda2 <- round(100-sum(diag(confuMat(dlda2)))/sum(confuMat(dlda2))*100,2)
```

with the same confusion matrix as KNN_2 and error rate. The reasoning is similar to how KNN_2 did better than KNN_1.

4.4. Cross Validation

Cross Validation randomly divides the available set of observations into two parts, a training set and a validation set/hold-out set. The model is fit on the training set, and the fitted model is used to predict the responses for the observation.

Leave-one-out cross-validation (LOOCV) - a single observation for the validation set and n-1 for the training data. Mean Square Error (MSE), $(y_1 - \hat{y_1})$ provides an approx. unbiased estimate for the test error. We repeat this process for the n training data. Each observation will be the validation set and produce n squared errors. The LOOCV estimate for the test MSE is the average of these n test error estimates,

$$CV_n = \frac{1}{n}\sum_{i=1}^{n} MSE_i$$

It is to be noted that, LOOCV can be expensive to implement since the model needs to be fit n times.

The code and confusion matrix is given below,

```
knn3 <- MLearn(formula =mol.biol~.,
             data=ALL_Phila,
             .method = knnI(k=1,l=0),
             xvalSpec("LOO"))
```

```
confuMat(knn3)
error_rate_knn3 <- round(100-sum(diag(confuMat(knn3)))/sum(confuMat(knn3))*100,2)
```

	ALL1/AF4	BCR/ABL	E2A/PBX1	NEG NUP-98	p15/p16	NA
ALL1/AF4	0	0	0	0	0	0
BCR/ABL	0	27	0	10	0	0
E2A/PBX1	0	0	0	0	0	0
NEG	0	8	0	34	0	0
NUP-98	0	0	0	0	0	0
p15/p16	0	0	0	0	0	0

with an error rate of 22.78%
Now we use Cross Validation with feature selection,

```
knn4 <- MLearn(formula =mol.biol~.,
             data=ALL_Phila,
             .method = knnI(k=1,l=0),
             xvalSpec("LOO",fsFun =fs.absT(20)))
confuMat(knn4)
error_rate_knn4 <- round(100-sum(diag(confuMat(knn4)))/sum(confuMat(knn4))*100,2)
```

We continue to use the LOOCV method but this time select the 20 most statistically significant predictors in our model. The error rate is 17.72%

The following code allows us to extract the top 20 genes from our model,

```
table(unlist(fsHistory(knn4)))
```

which are,

$X1211\_s\_at$	X32434_at	X32612_at
X33232_at	X33440_at	X33462_at
X33700_at	X34362_at	X36275_at
X36638_at	X37006_at	X37014_at
X37027_at	X37043_at	X37283_at
X37363_at	X38052_at	X38514_at
X38578_at	X40202_at	X40516_at
X40953_at	X41123_s_at	X41439_at

Now with 5 predictors,

```
knn5 <- MLearn(formula =mol.biol~.,
             data=ALL_Phila,
             .method = knnI(k=1,l=0),
             xvalSpec("LOO",fsFun =fs.absT(5)))
confuMat(knn5)
error_rate_knn5 <- round(100-sum(diag(confuMat(knn5)))/sum(confuMat(knn4))*100,2)
write.csv(table(unlist(fsHistory(knn5))), "top_5.csv")
```

	ALL1/AF4	BCR/ABL	E2A/PBX1	NEG NUP-98	p15/p16	NA
ALL1/AF4	0	0	0	0	0	0
BCR/ABL	0	30	0	7	0	0
E2A/PBX1	0	0	0	0	0	0
NEG	0	5	0	37	0	0
NUP-98	0	0	0	0	0	0
p15/p16	0	0	0	0	0	0

and the 5 most statistically significant predictors, X32434_at X33440_at X36275_at X37014_at X37027_at X37363_at X40202_at

with error rate 15.19%.

Obverse with our improvements of K-Nearest Neighbors and Diagonal Linear Discriminant Analysis we were able halved out initial error rate of 31.43% to 15.19%

4.5. Random Forrest

This classifier works by having a large group of individual uncorrelated decision trees. Each tree outputs a prediction for the observation. Whichever prediction has the highest count among all the trees, the classifiers classify the observation as that.

```
Rf1 <- MLearn(mol.biol~.,
              ALL_Phila,
              randomForestI,
              Training_set,
              ntree=1200,
              mtry=50,
              importance=TRUE)
confuMat(Rf1,"test")
error_rate_knn6 <- round(100-sum(diag(confuMat(Rf1,"test")))/sum(confuMat(Rf1,"test"))*100,2)
```

where $ntree$ is the amount of trees, $mtry$ is the number of predictors, and $importamce = TRUE$ are the relevant predictors. The confusion matrix is given by,

		predicted	
		BCR/ABL	*NEG*
given	*BCR/ABL*	13	2
	NEG	8	12

and an error rate of 28.57%. We apply Random Forrest again with only 5 predictors this time. That is the 5 most statistically significant predictors,

```
Rf2 <- MLearn(mol.biol~.,
              ALL_Phila,
              randomForestI,
              Training_set,
              ntree=1200,
              mtry=5,
              importance=TRUE)

confuMat(Rf2,"test")
error_rate_knn7 <- round(100-sum(diag(confuMat(Rf2,"test")))/sum(confuMat(Rf2,"test"))*100,2)
```

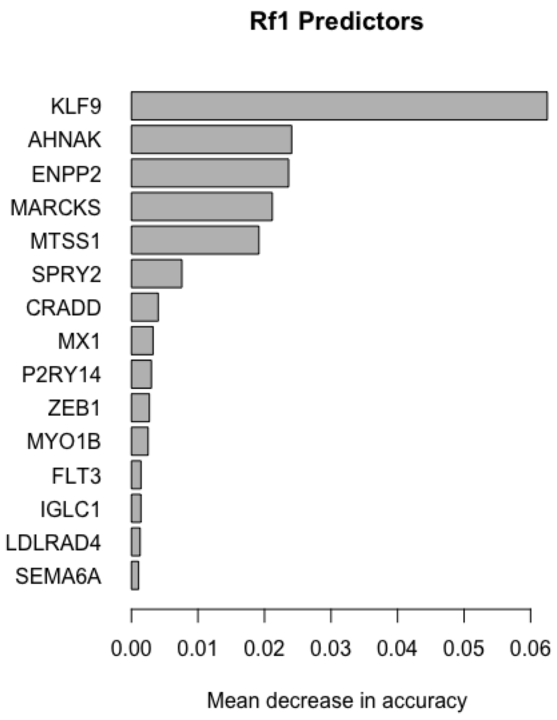
with confusion matrix,

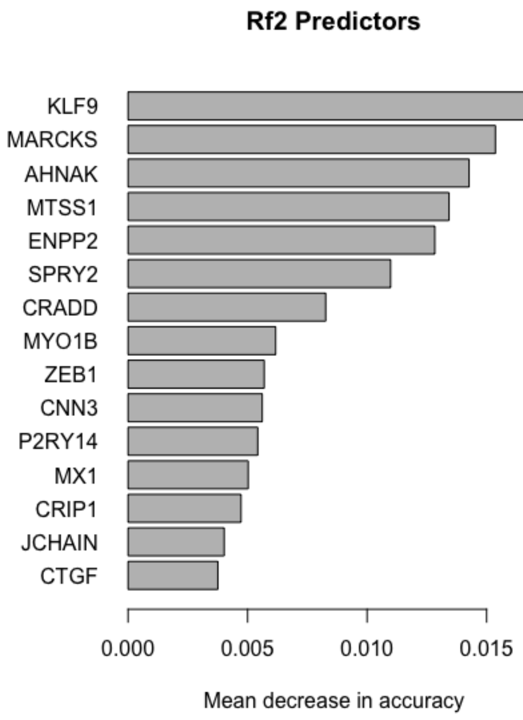
		predicted	
		BCR/ABL	*NEG*
given	*BCR/ABL*	13	2
	NEG	7	13

and error rate of 14.29%.

Obverse that $Rf2$ with 5 predictors is slightly better than $Rf1$ with 50 predictors. If we had to choose between the two models above we would choose $Rf2$ as it outperforms $Rf1$ and is less complicated (fewer predictors).

We can see which predictors are statistically important in the Random Forrest models,





We can interpret Mean decrease in accuracy as to how much the model fit's accuracy decreases when you drop a variable. The greater the drop the more significant the variable.

5. Conclusion

Our objective of this analysis was to try to identify genes that contain the BCR/ABL mutation and see if they contribute to ALL. After filtering formatting the data, we visualize the data to see if the data points are correlated with the Euclidean and Manhattan distance metric. Indeed, they are so we continue our analysis. Next, we apply 3 Machine Learning models, K-Nearest Neighbours, Diagonal Linear Discriminant Analysis (DLDA), and Random Forrest. Each model gave reasonable error rates with the best being $Rf25$ with 14.29% error rate. So, with this information, the genes that contain the BCR/ABL mutation and have ALL are KLF9, AHNAK, ENPP2, and MARCKS. Preventative treatments should focus on these genes to try to prevent the mutation. It's clear that is a hard task as mutations are mostly random. So, for people already diagnosed with ALL treatments can now focus specifically on the above genes.

6. What's Next

To find similarities and correlations in the data we simply used Euclidean and Manhattan distance. These metrics are vague and board and can be used on any type of data. We can get a better insight into the data by taking advantage that the data is genes and use metrics that are more suitable for that type of data. Moreover, we can rigorously check that our models are adequate, follow a Normal distribution, and heteroscedasticity and then applying data transformation techniques (WLS, Box-Cox transformation, log transformation, etc) if necessary. The above suggestions would allow for stronger results which can lead to better treatments and solutions for people with ALL.