Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

BIOINF 525 Module 3

Lab #4

4/13/2017

Please complete the exercises below. Throughout the lab sessions for this module, we will use the following notation:

Plain text indicates actions that should be taken

*Italicized text indicates explanatory material*

**Bold text indicates a point where a written response is required**

We will pause at each point with a long horizontal line for discussion. If you reach those pause points but the rest of the class is still working, I encourage you to try out some variations on whatever step you were currently working on.

Before beginning, please be sure that the following libraries load without error:

library(Biobase)

library(GEOquery)

library(ROCR)

library(randomForest)

library(e1071)

library(tsne)

library(dbscan)

The following packages can be installed using the standard install.packages command:

randomForest

ROCR

e1071

tsne

dbscan

The remainder must be installed through a slightly different mechanism:

source("https://bioconductor.org/biocLite.R")

biocLite("Biobase")

biocLite(“GEOquery”)

***Exercise 1: Supervised learning on gene expression data sets***

*We will first apply a variety of machine learning methods to predict the incidence of recurrence in prostate cancer based on gene expression profiles from tissue removed during surgery. You can see the details on the GSE page at*

[*https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE10645*](https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE10645)

*In brief, the samples show gene expression profiles from microarray data, and are paired with information on whether the patients showed no signs of recurrence (NED), elevated PSA (PSA), or a full systematic recurrence with 5 years (Systemic). Having a reliable method to predict recurrence (preferably as cheaply as possible) would be highly valuable for informing therapeutic decisions.*

You can download the data into R using the GEOquery package

gset <- getGEO("GSE10645", GSEMatrix =TRUE, AnnotGPL=TRUE)[[1]]

*Then, we can gently process the data to get it in a suitable form for analysis. Use the following commands to set up the data set:*

expmat=exprs(gset)

exptab = as.data.frame(t(expmat))

pdat=pData(gset)

pdat$characteristics\_ch1

exptab$label = pdat$characteristics\_ch1

exptab.2=subset(exptab, label != "Case-Control Group: PSA")

exptab.2$label = droplevels(exptab.2$label)

attach(exptab.2)

*Here, we are making a data frame that includes the sample labels, and for the sake of argument restricting ourselves to only the two extreme classes (no recurrence and severe recurrence) so we can try binary classification.*

*First, we will try fitting an svm to predict which class each patient falls into. This can be done using all of the gene data in exptab.2 as follows:*

svm.mod=svm(label~.,data=exptab.2)

*You can then evaluate the performance by using the model to re-classify the input table, and seeing how the predictions correspond with the true labels.*

svm.pred=predict(svm.mod,exptab.2)

table(svm.pred,exptab.2$label)

**What are the precision and recall of the fitted model on the full data set, considering the Systemic Recurrence group as the positives?**

**Precision: 0.978**

**Recall: 0.92**

*Since we paid attention in lecture and know that SVMs are vulnerable to over-fitting, we can do a crude form of cross-validation to assess the model’s real world performance. For the sake of time, we will simply train the model on 70% of the data, and then test its performance on the other 30%.*

train.dat=exptab.2[1:277,]

test.dat=exptab.2[278:395,]

svm.lo = svm(label~.,data=train.dat)

pred.tr = predict(svm.lo, train.dat)

pred.lo = predict(svm.lo, test.dat)

table(pred.tr, train.dat$label)

table(pred.lo, test.dat$label)

**What are the precision and recall on the training set data? On the testing set? Is the svm overtrained?**

Training: Precision 0.977, Recall 0.929

Testing: Precision 0.759, 0.683

Yes, the svm is overtrained, as indicated by the much lower performance upon cross validation

*For comparison, we will now attempt the same task using a random forest classifier.*

rf.mod=randomForest(label~.,data=exptab.2,ntree=1000,proximity=TRUE)

*We can optimize the number of trees by plotting OOB performance vs. number of trees; we find that performance is essentially flat after about 200 trees.*

plot(rf.mod$err.rate[1:1000,1],xlab="# of trees", ylab="OOB error")

*Thus, we re-fit the model using 200 trees.*

rf.mod.2 = randomForest(label~.,data=exptab.2,ntree=200,proximity=TRUE)

**What are the OOB error rates observed for rf.mod and rf.mod.2?**

**(Note that since random forests are non-deterministic, you may see somewhat different values -- this holds for several of the answers below, but they should at least be similar)**

**rf.mod: 30.63%**

**rf.mod.2: 30.63%**

**What are the precision and recall for rf.mod.2?**

**Precision: 0.71**

**Recall: 0.69**

*Now, let’s compare the OOB error estimate with our crude cross-validation and see how well the error rate is estimated.*

detach(exptab.2)

attach(train.dat)

rf.mod.cv = randomForest(label~.,data=train.dat,ntree=200,proximity=TRUE)

pred.tr = predict(rf.mod.cv, train.dat)

pred.lo = predict(rf.mod.cv, test.dat)

table(pred.tr, train.dat$label)

table(pred.lo, test.dat$label)

**Does the RF model show signs of overtraining? How does the performance observed from cross-validation compare with what you expect from the OOB error estimate?**

The RF does show some signs of over-training, because the performance is better on the training set than on the testing set. However, the OOB error is almost exactly the same as the actual performance on the testing set, showing that the OOB error provides a reasonable estimate for out-of-set errors.

*As a classic evaluation of performance, we will plot the ROC curve for the fully fitted random forest model. To do this, we must figure out how the predictions would look at a variety of thresholds. We also calculate the AUC in the process.*

predfreq = as.vector(rf.mod.2$votes[,2])

pred=prediction(predfreq,exptab.2$label)

perf\_ROC = performance(pred,"tpr","fpr")

perf\_AUC = performance(pred,"auc")

AUC=perf\_AUC@y.values[[1]]

plot(perf\_ROC,main="ROC plot")

abline(0,1,col='red',lty=2)

text(0.2,0.8,paste("AUC = ",format(AUC, digits=5, scientific=FALSE)))

**Does the random forest model show better than random performance? What is the AUC?**

**Yes. The AUC is 0.75; any AUC above 0.5 indicates better than random performance.**

*RNA-seq and microarrays are still fairly expensive – we might wish to develop a qpcr-based assay to predict disease progression. Or, we might be biologists interested in what aspects make prostate cancer more likely to recur after surgical resection. Either way, it is useful to know which features contribute most strongly to the predictions. To understand this, we begin by looking at the decreases in Gini coefficients when each feature is removed from the model. Note that the feature names are all gene names, with the –S or –A at the end indicating whether sense or antisense expression is listed.*

varImpPlot(rf.mod.2)

**If you had to pick three qpcr targets to try to distinguish recurrent from non-recurrent prostate cancer, what would they be? Give the annotation for one of them (google is your friend).**

**(note that the answers here are highly dependent on your random model, since the variable importances are pretty uniform)**

**GI\_21361115-S – This is the mRNA for versican, an anti-adhesion proteoglycan that has been widely noted in the literature as being overexpressed in cancer (particularly metastatic cancer)**

**GI\_4502144-S**

**GI\_31652260-S**

**What antisense RNA had the highest Gini importance?**

**GI\_27886643-A – antisense to CDC2 mRNA**

*Consider the very simplest case where you could only look at the two strongest contributors. You can plot these with*

plot(exptab.2$"GI\_4507170-S", exptab.2$"GI\_20127489-S", col=exptab.2$label)

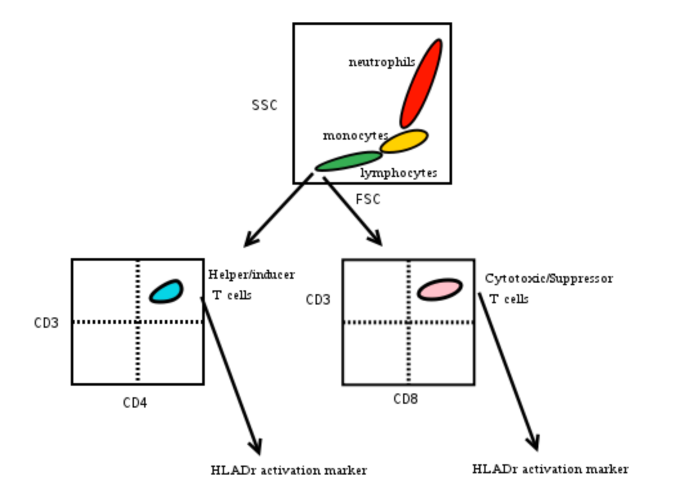
*In the resulting plot, black and red points indicate no disease progression and systemic recurrence, respectively.*

**What rule would you make if you had to classify patients just by these two expression levels? How well do you think that classifier would perform?**

The plot is not very informative, but the best we could do would be to predict metastasis for cases showing higher expression of GI\_4507170 and lower expression of GI\_20127489. Performance would be quite bad because the distributions overlap heavily.

***Exercise 2: Unsupervised clustering and visualization on a flow cytometry dataset***

*In this exercise, we will consider a flow cytometry data set using four surface markers to distinguish between different types of lymphocytes. The gating scheme used in expert analysis of the data sets, as shown in* [*https://www.bioconductor.org/help/workflows/highthroughputassays/flowWorkFlow.pdf*](https://www.bioconductor.org/help/workflows/highthroughputassays/flowWorkFlow.pdf)*, is:*

**

*We would like to see how well an unsupervised analysis can reproduce these distinctions. Note that because the data set is quite small, and contains fewer than 100 CD4+ and CD8+ lymphocytes out of 4,000 cells, efforts to automatically identify those populations will be especially challenging.*

*We will begin by seeing how various data transformations can help us to visualize the differences between populations along a reasonable number of dimensions, since the input data are in seven dimensions.*

Download and uncompress the needed data files from the class website.

Load the flow cytometry data:

detach(train.dat)

flowdat=read.csv("flowdat.csv",row.names=1)

And inspect the table to see what types of data are present.

Then, plot the data set along various axes to compare with the diagram shown above:

plot(flowdat$FSC, flowdat$SSC,col=as.factor(flowdat$class))

plot(flowdat$CD4, flowdat$CD3,col=as.factor(flowdat$class))

plot(flowdat$CD3, flowdat$CD8,col=as.factor(flowdat$class))

flowdat$class=as.factor(flowdat$class)

*Note the color map used for all of the analysis here:*

*Black = CD4+ T cell*

*Red = CD8+ T cell*

*Green = Other lymphocyte*

*Blue = Monocyte*

*Cyan = Neutrophil*

*The focus of the panel of markers used here was to distinguish, and subsequently characterize, CD4+ and CD8+ T cells from the remainder of the population.*

Now, apply PCA-based dimension reduction with:

exp.pc = prcomp(flowdat[,1:7],center=TRUE,scale.=TRUE)

Examine the relative contributions of the components to representing the variance in the data with

plot(exp.pc)

summary(exp.pc)

**How many components would you need to include to capture at least 75% of the variance?**

**At least 4**

*Now we will observe how our picture of the data set changes when plotting along the principal components.*

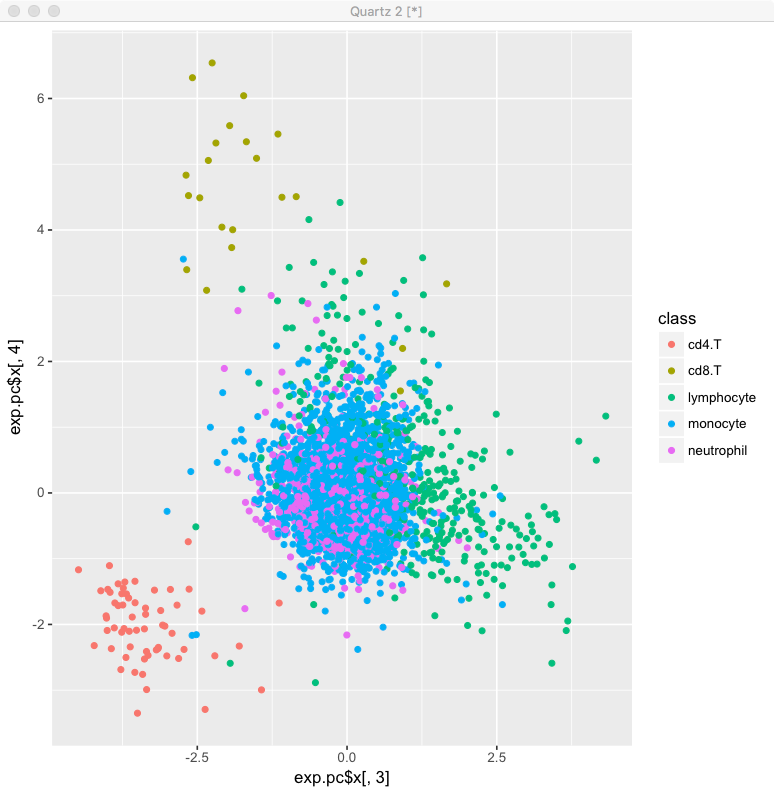
plot(exp.pc$x[,1], exp.pc$x[,2],col=flowdat$class)

plot(exp.pc$x[,3], exp.pc$x[,4],col=flowdat$class)

*Be sure to pause and inspect the plots in between issuing these commands, or else the second plot will simply replace the first.*

*A nicer version of this plot can be made using the ggplot2 package, with the command*

*ggplot(flowdat, aes(x=exp.pc$x[,3],y=exp.pc$x[,4],col=class)) + geom.point()*

**

**Are the PC plots informative? What do each of the first four components appear to represent?**

**Yes, the PCs are informative.**

**PC1: Mostly appears to separate monocytes (low) from everything else**

**PC2: In order from high to low, partially separates cd4+ t cells, cd8+ t cells, lymphocytes, monocytes, and neutrophils**

**PC3: low for CD4 T cells, high for other lymphocytes**

**PC4: high for cd8 T cells, low for cd4 T cells**

*As an alternative, we can see how the data would be split up using a nonlinear dimensionality reduction method. This could be done using t-SNE with the commands:*

*flow.tsne=tsne(flowdat[,1:7])*

*However, this takes about an hour to run, so you can just load the results from the downloaded zip archive above with:*

load("flow\_tsne.rdat")

And then visualize the data in the mapped space with

plot(flow.tsne[,1],flow.tsne[,2], col=flowdat$class)

**Does t-SNE allow reasonable identification of the lymphocyte populations?**

**Yes; both the CD4+ and CD8+ t cells form mostly self-contained clusters distinct from the main lymphocyte population**

Having tried various methods for visualizing the data, now we will apply two methods for unsupervised clustering to attempt to divide the cells .

First, we apply k-means clustering, under the assumption that we at least know ahead of time that we are trying to split the cells into 5 classes. This can be done with:

km = kmeans(flowdat[,1:7],5)

table(km$cluster,flowdat$class)

**What do you think of the performance? On which parts of our stated aims did the k-means clustering perform well or poorly?**

**(note that because k-means clustering is non-deterministic, you may see different results. Here is the table that I got when I ran it:**

**> table(km$cluster,flowdat$class)**

**cd4.T cd8.T lymphocyte monocyte neutrophil**

**1 0 0 0 883 0**

**2 0 0 0 362 686**

**3 0 0 0 1276 0**

**4 0 1 2 56 198**

**5 61 22 411 42 0 )**

**The clustering does a good job of separating lymphocytes from other cell types, but does not allow detection of the CD4 and CD8 sub-populations.**

For comparison, we will also apply DBSCAN. There is guidance in the DBSCAN documentation for setting the key parameters eps and minPts; here I have followed that advice but omitted the actual process of determining them.

We can get an initial look at DBSCAN’s performance with

orig.db = dbscan(flowdat[,1:7],eps=25,minPts=8)

table(orig.db$cluster, flowdat$class)

However, methods like both k-means clustering and DBSCAN can perform poorly when the distance scales across different dimensions of the data are non-uniform. We can improve that situation simply by log-transforming the FSC and SSC data, and dropping the irrelevant HLADr feature:

newdat=data.frame(FSC=log(flowdat$FSC),SSC=log(flowdat$SSC),CD8=flowdat$CD8,CD69=flowdat$CD69,CD4=flowdat$CD4,CD3=flowdat$CD3,class=flowdat$class)

new.db = dbscan(newdat[,1:6],eps=0.8,minPts=7)

table(new.db$cluster, newdat$class)

plot(flowdat$CD3, flowdat$CD4,col=new.db$cluster+1)

plot(flowdat$CD3, flowdat$CD8,col=new.db$cluster+1)

**How does the performance of the model on the newly normalized data compare with the original data?**

**The original model is a mess, and has 9 clusters that do now have strong biological meaning. The new model has a clear cd4+ cluster, but stil lumps the cd8+ cells along with half of the lymphocytes as “noise”.**

*In some cases, dimensionality reduction methods can also provide effective feature selection/normalization for classifiers. Consider again the t-SNE visualization of this flow cytometry data set:*

plot(flow.tsne[,1],flow.tsne[,2], col=newdat$class)

Now try repeating DBSCAN clustering in the t-SNE space:

tsne.db=dbscan(flow.tsne,eps=10,minPts=10)

table(newdat$class,tsne.db$cluster)

plot(flow.tsne[,1],flow.tsne[,2], col=as.factor(tsne.db$cluster))

**How does the performance of DBSCAN in t-SNE normalized space compare with the original performance in identifying CD4+/CD8+ lymphocytes?**

**Unlike the other cases, here we observe properly defined clusters for both the CD4 and CD8 t cells separately, along with one cluster that is mostly otherlymphocytes, and one background cluster of mostly monocytes and neutrophils. Thus, this clustering scheme accomplishes our objectives well.**