Clustering, classification and data mining

Eric Feigelson (Penn State) edf@astro.psu.edu

2nd East Asian Workshops in Astrostatistics Summer 2018

Adapted from R scripts in Appendix B, *Modern Statistical Methods for Astronomy With R Applications*, Eric D. Feigelson & G. Jogesh Babu 2012 http://astrostatistics.psu.edu/MSMA (http://astrostatistics.psu.edu/MSMA) Improved by Gabriel Caceres 2017.

Data mining and *machine learning* are ill-defined terms referring to a wide range of computationally intensive methods for finding patterns in large multivariate datasets. This encompasses (unsupervised) clustering, (supervised) classification, and regression. In some cases, the algorithms have a foundation in mathematical theorems, but often they are practical heuristic approaches to difficult problems. Some methods have optimize through iterative steps that *learn* the characteristics of the dataset. We start here with some simple clustering, and they proceed to a classification problem.

Comment: If you get an odd run-time error like "internal error -3 in R_decompress1", then click `Restart` under then `Kernel` tab of Jupyter.

In [ ]:
```
# Setup
setwd('/Users/ericfeigelson/Desktop/Rdir')

install.packages('fpc', repos='https://cloud.r-project.org')
# Flexible Procedures for Clustering
install.packages('class', repos='https://cloud.r-project.org')
# Functions for Classification
install.packages('randomForest', repos='https://cloud.r-project.org')
# Random Forests
install.packages('e1071', repos='https://cloud.r-project.org')
# SVM
```

I Color-magnitude diagram for low-redshift COMBO-17 galaxies

Contemporaneous with the early Sloan Digital Sky Survey, a major galaxy survey was called COMBO-17 that obtained photometry of thousands of galaxies in 17 spectral bands as faint as ~23 mag. One of the major results was investigation of a bifurcation in the color magnitude diagram of normal galaxies showing a `red sequence` and a `blue cloud` separated by a `green valley`. We study this here in a bivariate color-magnitude diagram using unsupervised methods.

In [ ]:
```
COMBO_loz=read.table('COMBO17_lowz.dat', header=T, fill=T)
str(COMBO_loz)
dim(COMBO_loz)
names(COMBO_loz)
names(COMBO_loz) <- c('MB', 'M280-MB')  ; names(COMBO_loz)
```

In [ ]:
```
plot(COMBO_loz, pch=20, cex=0.5, xlim=c(-22,-7), ylim=c(-2,2.5),
   xlab=expression(M[B]~~(mag)), ylab=expression(M[280] - M[B]~~(mag))
,
   main='')
```

In [ ]:
```
# Two-dimensional kernel-density estimator

library(MASS)
COMBO_loz_sm <- kde2d(COMBO_loz[,1], COMBO_loz[,2], h=c(1.6,0.4),
   lims = c(-22,-7,-2,2.5), n=500)
image(COMBO_loz_sm, col=grey(13:0/15), xlab=expression(M[B]~~(mag)),
   ylab=expression(M[280] - M[B]~~(mag)), xlim=c(-22,-7), ylim=c(-2,2.
5),
   xaxp=c(-20,-10,2))
text(-16.5, -1, "Blue cloud", col='darkblue', pos=4, cex=0.8)
text(-17,-0.7, 'Green valley', col='darkgreen', pos=4, cex=0.8)
text(-13, -0.2, 'Red sequence', col='red', pos=4, cex=0.8)
text(-18.5, 1.7, 'Bright cluster galaxies', col='deeppink3', pos=4,
cex=0.8)
dev.copy2pdf(file='COMBO17_CMD.pdf')
```

II Two nonparametric unsupervised clustering procedure

In [ ]:
```
# Standardize variables

Mag_std <- scale(COMBO_loz[,1])
Color_std <- scale(COMBO_loz[,2])
COMBO_std <- cbind(Mag_std,Color_std)

# Hierarchical clustering

COMBO_dist <- dist(COMBO_std)
COMBO_hc <- hclust(COMBO_dist, method='complete')

# Cutting the tree at k=5 clusters

plot(COMBO_hc, label=F)
COMBO_hc5a <- rect.hclust(COMBO_hc, k=5, border='red')
COMBO_hc5b <- cutree(COMBO_hc, k=5)
plot(COMBO_loz, pch=(COMBO_hc5b+18), cex=0.7, xlab=expression(M[B]~~(m
ag)),
   ylab=expression(M[280] - M[B]~~(mag)), main='')
```

Here we have run hierarchical clustering with `complete` linkage that produces compact groupings. Cutting the tree at k=2, 3, or 4 did not separate the red sequence and blue cloud; this occurred only at k=5. The plot here uses the vector of classifications from the clustering algorithm to determine the shape of the plotting symbol. **Exercise:** Read the help files associated with hierarchical clustering and try different parameters.

In [ ]:
```
# Density-based clustering algorithm

install.packages('fpc') ; library(fpc)
COMBO_dbs <-  dbscan(COMBO_std, eps=0.1, MinPts=5, method='raw')
print.dbscan(COMBO_dbs) ; COMBO_dbs$cluster
plot(COMBO_loz[COMBO_dbs$cluster==0,], pch=20, cex=0.7, xlab='M_B (mag
)',
   ylab='M_280 - M_B (mag)')
points(COMBO_loz[COMBO_dbs$cluster==2,], pch=2, cex=1.0, col='blue2')
points(COMBO_loz[COMBO_dbs$cluster==1 | COMBO_dbs$cluster==3,], pch=1,
cex=1.0, col='orangered3')
```

Here we run the well-known density-based clustering algorithm `DBSCAN`. It starts with a peaks found from kernel density estimation, then extends outward with a `reach` distance (eps), adding a constraint that clusters must have more members than a specified minimum (MinPts). Unlike many other clustering procedures, DBSCAN allows clusters to reside in a background of unclustered objects. Here we set both the color and symbol shapes by the resulting cluster vector. But attempts to include more galaxies in the red sequence or blue cloud resulted in unifying them, as the algorithm reached over the green valley. **Exercise:** Read the help file of DBSCAN and try different parameters.

In conclusion, it was difficult to get standard nonparametric clustering algorithms to find members of the known galaxy groupings. Clustering parameters had to be carefully tuned, and even then the results were not very satisfactory. Simple kernel smoothing did a better job in showing the structure, although it does not define cluster membership.

III Classification of Sloan point sources

We next apply a number of important supervised classifiers to the problem of classifying point sources in the Sloan Digital Sky Survey into three groups: normal stars, white dwarf stars, and quasars that appear star-like but are actually distant extragalactic objects. We treat the problem in four dimensions of color indices between the five Sloan photometric bands. Note that we could add other variables, such as magnitude in a fiducial band and location in the sky.

In the lengthy scripts below, we first ingest a `test set` of 17,000 point sources from the Sloan Digital Sky Survey, and then three `training sets`: 2000 quasars, 2000 white dwarfs, and 5000 main sequence and giant stars. The training samples are based on tedious spectroscopic confirmation with other telescopes. We combine the three training sets into a single R data.frame with a new column giving the known cluster identifiers (1=quasar, 2=star, 3=white dwarf). We then divide the traing set into 80% for designing the classifiers and 20% for validation. Our science goal is to classify the test dataset based on the best classifiers.

In [ ]:
```
#   ************  CLASSIFICATION OF SLOAN POINT SOURCES ***********
#   ************************************************************
# R script for constructing SDSS test and training datasets is given
# in Appendix C of MSMA (Feigelson & Babu 2012).


# SDSS point sources test dataset, N=17,000 (mag<21, point sources, hi
-qual)

SDSS <- read.csv('SDSS_test.csv', h=T)
dim(SDSS) ; summary(SDSS)
SDSS_test <- data.frame(
    u_g = SDSS$u_mag-SDSS$g_mag,
    g_r = SDSS$g_mag-SDSS$r_mag,
    r_i = SDSS$r_mag-SDSS$i_mag,
    i_z = SDSS$i_mag-SDSS$z_mag
    )
names(SDSS_test) <- c('u_g', 'g_r', 'r_i', 'i_z')
str(SDSS_test)

par(mfrow=c(1,3))
plot(SDSS_test[,1], SDSS_test[,2], xlim=c(-0.7,3), ylim=c(-0.7,1.8),pc
h=20,
        cex=0.6, cex.lab=1.5, cex.axis=1.5, main='Test dataset', xlab=
'u-g (mag)', ylab='g-r (mag)')
plot(SDSS_test[,2], SDSS_test[,3], xlim=c(-0.7,1.8), ylim=c(-0.7,1.8),
pch=20,
        cex=0.6, cex.lab=1.5, cex.axis=1.5, main='', xlab='g-r (mag)',
ylab='r-i (mag)')
plot(SDSS_test[,3], SDSS_test[,4], xlim=c(-0.7,1.8), ylim=c(-1.1,1.3),
pch=20,
        cex=0.6, cex.lab=1.5, cex.axis=1.5, main='', xlab='r-i (mag)',
ylab='i-z (mag)')
par(mfrow=c(1,1))
```

In [ ]:
```
# Quasar training set, N=2000 (Class 1)

qso1 <- read.table('SDSS_QSO.dat', h=T)
dim(qso1) ; summary(qso1)
bad_phot_qso <- which(qso1[,c(3,5,7,9,11)] > 21.0 | qso1[,3]==0)
qso2 <- qso1[1:2000,-bad_phot_qso,]
qso3 <- cbind((qso2[,3]-qso2[,5]), (qso2[,5]-qso2[,7]), (qso2[,7]-qso2
[,9]), (qso2[,9]-qso2[,11]))
qso_train <- data.frame(cbind(qso3, rep(1, length(qso3[,1]))))
names(qso_train) <- c('u_g', 'g_r', 'r_i', 'i_z', 'Class')
dim(qso_train) ; summary(qso_train)

# Star training set, N=5000 (Class 2)
```

```
temp2 <- read.csv('SDSS_stars.csv', h=T)
dim(temp2) ; summary(temp2)
star <- cbind((temp2[,1]-temp2[,2]), (temp2[,2]-temp2[,3]), (temp2[,3]
-temp2[,4]),
        (temp2[,4]-temp2[,5]))
star_train <- data.frame(cbind(star, rep(2, length(star[,1]))))
names(star_train) <- c('u_g','g_r','r_i','i_z','Class')
dim(star_train) ; summary(star_train)

# White dwarf training set, N=2000 (Class 3)

temp3 <- read.csv('SDSS_wd.csv', h=T)
dim(temp3) ; summary(temp3)
temp3 <- na.omit(temp3)
wd <- cbind((temp3[1:2000,2]-temp3[1:2000,3]), (temp3[1:2000,3]-temp3[
1:2000,4]),
        (temp3[1:2000,4]-temp3[1:2000,5]), (temp3[1:2000,5]-temp3[1:20
00,6]))
wd_train <- data.frame(cbind(wd, rep(3, length(wd[,1]))))
names(wd_train) <- c('u_g', 'g_r', 'r_i', 'i_z', 'Class')
dim(wd_train) ; summary(wd_train)

# Combine and plot the training set (9000 objects)

SDSS_train <- data.frame(rbind(qso_train, star_train, wd_train))
names(SDSS_train) <- c('u_g', 'g_r', 'r_i', 'i_z', 'Class')
str(SDSS_train)

par(mfrow=c(1,3))
plot(SDSS_train[,1], SDSS_train[,2], xlim=c(-0.7,3), ylim=c(-0.7,1.8),
pch=20,
        col=SDSS_train[,5], cex=0.6, cex.lab=1.6, cex.axis=1.6, main='
Training dataset', xlab='u-g (mag)',
        ylab='g-r (mag)')
legend(-0.5, 1.7, c('QSO','MS + RG','WD'), pch=20, col=c('black','red'
,'green'),
        cex=0.8)
plot(SDSS_train[,2], SDSS_train[,3], xlim=c(-0.7,1.8), ylim=c(-0.7,1.8
), pch=20,
        col=SDSS_train[,5], cex=0.6, cex.lab=1.6, cex.axis=1.6, main='
', xlab='g-r (mag)',
        ylab='r-i (mag)')
plot(SDSS_train[,3], SDSS_train[,4], xlim=c(-0.7,1.8), ylim=c(-1.1,1.3
), pch=20,
        col=SDSS_train[,5], cex=0.6, cex.lab=1.6, cex.axis=1.6, main='
', xlab='r-i (mag)',
        ylab='i-z (mag)')
par(mfrow=c(1,1))

# Save 20% of training set for cross-validation
```

```
ran9000 <- runif(9000)
SDSS_train80 <- SDSS_train[(ran9000 < 0.80),]
SDSS_train20 <- SDSS_train[(ran9000 > 0.80),]
```

To assist with evaluation of different classifiers, let's make a function that prints and plots the `confusion matrix` showing the number of objects correctly and incorrectly classified. The `accuracy` value summarizes the classifier's performance ... we want it to be as close to 1.00 as possible.

In [ ]:
```
## Function to evaluate classification performance
class_eval <- function(pred, act, plot=TRUE, ...){
  iact <- as.integer(act)
  ipred <- as.integer(pred)
  acc <- sum(ipred==iact)/length(iact)  # accuracy
  if (isTRUE(plot)){
    plot(jitter(ipred), jitter(iact), pch=20, cex=0.5, xlab='Predicted
Class', ylab='True class',lab=c(3,3,1), ...)
    mtext(paste("Accuracy =", round(acc, 3)))
  }
  return(list("Confusion Table"=table("True Class"=iact, "Predicted Cl
ass"=ipred), Accuracy=acc))
}
```

Also, let us use 80% of the training set for classification, and save 20% of the training set for classifier validation (evaluation)

In [ ]:
```
## Copy original full dataset before splitting
SDSS_train_full <- SDSS_train

set.seed(456)
## Save 20% of training set for validation
val_set <- sample(nrow(SDSS_train), round(nrow(SDSS_train)*0.2))
SDSS_val <- SDSS_train_full[val_set,]
SDSS_train <- SDSS_train_full[-val_set,]
```

Before supervised classification, we make two tests. First, we show that random classification assignments will give an accuracy of 0.33 for three classes. This is obvious. Second, we try an unsupervised clustering algorithm, k-means partitioning. Since clustering algorithms performed poorly for the simpler `red sequence` vs. `blue cloud` galaxy groups above, we do not expect them to do well discriminating these overlapping, multivariate distributions with very non-Gaussian morphologies. The k-means procedure did not begin to separate the groups until k>5, but then it created false groupings amoung the normal stasr. Altogether, k-means partitioning does a terrible job separating the three classes.

In [ ]:
```
## Test of classification evaluation:  random class assignment
set.seed(123)
SDSS_rand_train_pred <- sample(1:3, nrow(SDSS_train), replace=T)
par(mfcol=c(1, 1))
class_eval(SDSS_rand_train_pred, SDSS_train$Class, main="Random Assign
ment")
```

In [ ]:
```
# Unsupervised k-means partitioning
# A terrible solution

SDSS.kmean <- kmeans(SDSS_test,6)
print(SDSS.kmean$centers)
plot(SDSS_test[,1], SDSS_test[,2], pch=20, cex=0.3, col=gray(SDSS.kmea
n$cluster/7),
   xlab='u-g (mag)', ylab='g-r (mag)', xlim=c(-0.5,3), ylim=c(-0.6,1.5
))
```

We now proceed with six multivariate classifiers commonly used in data mining: linear discriminant analysis; k-nearest neighbor classification; a simple neural network; Classification And Regression Trees; CART with Random Forests; and Support Vector Machine.

The R script is similar for each of the classification methods. We first bring the classifier into our R session, run it on the 80% training dataset, and save the results in an R object. We then produce two plots: one of the Sloan color-color diagrams with colors based on the new classifications; and a confusion matrix plot showing the predicted classes vs. the known classes for the 20% validation dataset. In some cases, we use R's `predict` function to apply the classifier to new datasets. **Exercise: Read the help files and run `str` on these objects to understand the outputs.**

In [ ]:
```
# Linear discriminant analysis

library(MASS)
SDSS_lda <- lda(SDSS_train[,1:4], as.factor(SDSS_train[,5]))
SDSS_lda_train_pred <- predict(SDSS_lda)$class
SDSS_lda_val_pred <- predict(SDSS_lda, SDSS_val[,1:4])$class
SDSS_lda_test_pred <- predict(SDSS_lda, SDSS_test[,1:4])$class


plot(SDSS_val[,1],SDSS_val[,2], xlim=c(-0.7,3), ylim=c(-0.7,1.8), pch=
20,
     col=SDSS_lda_val_pred, cex=0.5, main='', xlab='u-g (mag)', ylab='
g-r (mag)')

class_eval(SDSS_lda_val_pred, SDSS_val$Class, main="LDA Classification
")
```

Surprisingly, linear discriminant analysis did not do poorly, even though it is limited to single 4-dimensional a hyperplane that optimally separates the classes assuming multivariate normal distributions. Even though the shapes are far from normal, and the separators are not linear, the classification accuracy is 93%.

In [ ]:
```
# k-nn classification

library(class)
SDSS_knn_test_pred <- knn(SDSS_train[,1:4], SDSS_test, as.factor(SDSS_
train[,5]), k=5, prob=T)
SDSS_knn_val_pred <- knn(SDSS_train[,1:4], SDSS_val[,1:4], as.factor(S
DSS_train[,5]), k=5, prob=T)

plot(SDSS_val[,1], SDSS_val[,2], xlim=c(-0.7,3), ylim=c(-0.7,1.8), pch
=20,
     col=SDSS_knn_val_pred, cex=0.5, main='', xlab='u-g (mag)', ylab='
g-r (mag)')

class_eval(SDSS_knn_val_pred, SDSS_val$Class, main="kNN Classification
")
```

Here we see that the k-NN algorithm -- a simple voting procedure among neighbors in 4-dimensional color space -- performed very well for this classification problem with only 2% misclassifications. Note however, that k-NN (like LDA and some other classifiers) depend on a distance matrix. This makes sense when (as in this case) all variables have the same units and similar range, but can be problematic when variables of very different units and ranges are combined. Also, there is a worry that the k-NN voting will depend on the artificial choice made of the sizes of the training sets (2000 for quasars, 2000 for white dwarfs, and 5000 for stars).

In [ ]:
```
# Neural network

library(nnet)
SDSS_nnet <- nnet(as.factor(Class) ~ u_g + g_r + r_i + i_z, SDSS_train
,size=5)

SDSS_nnet_train_pred <- predict(SDSS_nnet, type="class")
SDSS_nnet_val_pred <- predict(SDSS_nnet, SDSS_val, type="class")
SDSS_nnet_test_pred <- predict(SDSS_nnet, SDSS_test, type="class")

plot(SDSS_val[,1], SDSS_val[,2], xlim=c(-0.7,3), ylim=c(-0.7,1.8), pch
=20,
     col=SDSS_nnet_val_pred, cex=0.5, main='', xlab='u-g (mag)', ylab=
'g-r (mag)')

class_eval(SDSS_nnet_val_pred, SDSS_val$Class, main="Neural Net Classi
fication")
```

Here we find a classification accuracy of 96%. Some quasars are incorrectly classified as horizontal branch giant stars. Some investigation shows that the performance is improved with a larger and deeper network, and more computing time.

In [ ]:
```
# Classification And Regression Tree

library(rpart)
SDSS_rpart <- rpart(SDSS_train[,5] ~., data=SDSS_train[,1:4], method="
class")
summary(SDSS_rpart)
str(SDSS_rpart)

SDSS_rpart_train_pred <- predict(SDSS_rpart, type="class")
SDSS_rpart_val_pred <- predict(SDSS_rpart, SDSS_val, type="class")
SDSS_rpart_test_pred <- predict(SDSS_rpart, SDSS_test, type="class")

plot(SDSS_val[,1], SDSS_val[,2], xlim=c(-0.7,3), ylim=c(-0.7,1.8), pch
=20,
   col=SDSS_rpart_val_pred, cex=0.5,
     main='', xlab='u-g (mag)', ylab='g-r (mag)')

class_eval(SDSS_rpart_val_pred, SDSS_val$Class, main="Tree Classificat
ion")
```

This CART classification gave 95% accuracy. There are many options to CART, and performance may improve with tuning operating parameters. Below are two diagnostic diagrams, one showing the tree splits and the other the criterion for pruning the tree to ~12 nodes.

In [ ]:
```
# Additional plots for decision tree
plot(SDSS_rpart, branch=0.5, margin=0.05)
text(SDSS_rpart, digits=3, use.n=T, cex=0.8)
plotcp(SDSS_rpart, lwd=2, cex.axis=1.3, cex.lab=1.3)
```

In [ ]:
```
# CART with Random Forests

## Random Forests
library(randomForest)
SDSS_rf <- randomForest(as.factor(Class) ~ u_g + g_r + r_i + i_z, data
=SDSS_train, mtry=2, importance=TRUE, do.trace=TRUE, ntree=100)

print(SDSS_rf)

SDSS_rf_train_pred <- predict(SDSS_rf, SDSS_train)
SDSS_rf_oob_pred <- predict(SDSS_rf)
SDSS_rf_val_pred <- predict(SDSS_rf, SDSS_val)
SDSS_rf_test_pred <- predict(SDSS_rf, SDSS_test)

plot(SDSS_val[,1], SDSS_val[,2], xlim=c(-0.7,3), ylim=c(-0.7,1.8), pch
=20,
   col=SDSS_rf_val_pred, cex=0.5,
     main='', xlab='u-g (mag)', ylab='g-r (mag)')

class_eval(SDSS_rf_train_pred, SDSS_train$Class, main="Random Forest C
lassification")
class_eval(SDSS_rf_oob_pred, SDSS_train$Class, main="Random Forest Cla
ssification")
class_eval(SDSS_rf_val_pred, SDSS_val$Class, main="Random Forest Class
ification")
```

Here have run Random Forest, where many decision trees are constructed and averaged using strategies of boosting and bagging. We run the classification evaluation on three subsets of the training data: the 80% training data (the 100% accuracy is not interesting here), the `out of bag` (oob) subset representing a validation set internal to the Random Forest, and our 20% validation subset. These give 98% accuracy. Random Forest has many options, and the performance may be improvable.

In [ ]:
```
# Support Vector Machine model, prediction and validation

library(e1071)
SDSS_svm <- svm((SDSS_train[,5]) ~.,data=SDSS_train[,1:4],cost=100,
gamma=1)
SDSS_svm <- svm(as.factor(SDSS_train[,5]) ~.,data=SDSS_train[,1:4],cos
t=100, gamma=1)

summary(SDSS_svm)

SDSS_svm_train_pred <- predict(SDSS_svm)
SDSS_svm_val_pred <- predict(SDSS_svm, SDSS_val)
SDSS_svm_test_pred <- predict(SDSS_svm, SDSS_test)

plot(SDSS_test[,1], SDSS_test[,2], xlim=c(-0.7,3), ylim=c(-0.7,1.8),
pch=20,
   col=round(as.numeric(SDSS_svm_test_pred)), cex=0.5, main='',
   xlab='u-g (mag)', ylab='g-r (mag)')

class_eval(SDSS_svm_val_pred, SDSS_val$Class, main="Random Forest Clas
sification")
```

The Support Vector Machine calculation is computer intensive, as it performs many operations, some in higher dimensions. But the performance is, like k-NN, exceppent with 98% recovery of true classes in the 20% validation training set.

Exercise: Take one or more of the sophisticated classifiers -- neural nets, Random Forests, SVM -- and study its performance carefully for the problem here. Read the help files, and appropriate chapter in a book on machine learning. Examine the output R objects with `str` and plot various quantities. Tweak various parameters of the method, and study the effect on the classifier performance.

We are now ready to stand back from our statistical analysis and examine the situation from a scientific (astronomical) viewpoint. The k-NN and SVM classifiers both showed excellent performance on the 20% validation dataset, with scientifically reasonable results on the test dataset. But there were some methodology questions about the k-NN procedure (e.g. effect of training set size on voting, value of k). So, it is reasonable to decide that the SVM classifier is the `best`.

Exercise: Produce the 2x2 contingency matrix -- True Positive, False Positive, True Negative, False Negative -- for the SVM classifier. Calculate (and try to interpret) various scalar measures of its performance such as precision, recall and F1 score. See https://en.wikipedia.org/wiki/Precision_and_recall (https://en.wikipedia.org/wiki/Precision_and_recall) and related Wikipedia pages.

We are now ready to bring the results out of R for use in other languages, by colleagues, and for publication in a journal. We need both publication-quality graphics and a table of the Sloan test set (17,000 lines) with a new column showing the derived classification of each object. Note that we could, using `library(xtable)` produce the table in LaTeX format.

In [ ]:
```
SDSS_test_svm_out <- cbind(SDSS[,6], SDSS[,7], SDSS_test, round(as.num
eric(SDSS_svm_test_pred)))
names(SDSS_test_svm_out)[c(1,2,7)] <- c('R.A.', 'Dec', 'SVM Class')
write.table(format(SDSS_test_svm_out), file='SDSS_test_svm.out',sep='\
t',quote=F)
```