Wine Classification and Clustering

Funfay Jen

2019-12-13

Goal 1.

The covariance matrices of red and white wines are often at least an order of magnitude apart, so we chose to use Hotelling's T^2 test for unequal variances. The test statistic (2.943228×10^4) vastly exceeds the critical value at level 95% (19.68), therefore we reject that there are no difference in the mean vectors between red and white wines in favor the alternative that there are.

A table of simultaneous confidence intervals is shown below of the 11 attribues in the wine data.

	lo	hi
fixed.acidity	1.271	1.658
volatile.acidity	0.2297	0.2695
citric.acid	-0.0849	-0.0415
residual.sugar	-4.025	-3.68
chlorides	0.0365	0.0469
free.sulfur.dioxide	-20.62	-18.25
total.sulfur.dioxide	-95.59	-88.19
density	0.0025	0.0029
pH	0.1056	0.1401
sulphates	0.1494	0.1872
alcohol	-0.2108	0.0282

From the table, we cannot however, without further knowledge about the units in which variables were measured as well as knowledge about the scientifically important thresholds of differences for the relavant variables, say much about which variables seem to differ most between red and white wine. At face value though, it seems as though the total sulfur dioxide (and free sulfur dioxide) shows the most difference between the two with red wine possessinng much less of the chemical.

To classify wine types, we cannot apply Fisher's approach to classification with two populations, since his method implicitly assumes that the covariances of the two populations are equal. Therefore, we will go with ad hoc/alogirthmic approaches (KNN and CART) instead. After exploring with the performance of each of the two methods using a training/test split approach, we find that CART consistently predicts with a lower error rate (under 2%) than KNN (over 6%), regardless of the number of neighbors we set for KNN as well as the random assignment to the training/test sets (keeping the split ratio as 7:3). Therefore we recommend using CART for classification.

Table 2: KNN (k=5) confusion matrix for wine type

	RED	WHITE
RED	393	65
WHITE	50	1441

Table 3: CART confusion matrix for wine type

	RED	WHITE
RED	435	23
WHITE	15	1476

The conditional probability of correctly classifying each of the wine types is different than the overall correct rate, and for our test set we have

$$P(\text{classifying a wine as red} \mid \text{red wine}) = 0.95$$

$$P(\text{classifying a wine as white} \mid \text{white wine}) = 0.99$$

For clustering, we discard the true wine labels as it's an unsupervised learning method. When it comes to choosing the distance measure, it is important to realize that in unsupervised learing, standardizing the variables becomes imporant. We explored hierachical clustering and non-hierachical clustering (k-means, model-based) methods, Between the hierachical and the k-means clustering methods, we compare their performance in the following table. We noted that k-means clustering is able to assign observations to two clusters algorithmically, whereas the hierarchical methods using any of the three distance measures fail to do so (they allocate the vast majority of the observations to one group). For the model-based clustering method, allowing it to choose the number of clusters will be almost computationally intractable and also result in 9 clusters. Specifying the desired number of clusters results in very similar groupings as the k-means method but still computationally expensive for the size of our data, so we discourage its use. We also obtained the confusion matrices of the k-means clustering method with both unscaled and scaled inputs as the following:

Table 4: 2-means (unscaled) clustering compared to wine type

RED	WHITE
1514	1294
85	3604

Table 5: 2-means (scaled) clustering compared to wine type

RED	WHITE
1575	68
24	4830

As we can see, the correct classification rate using scaled data is pretty decent, with a 0.986 probability to correctly cluster wines according to their types (red wines and white wines). In comparison, if we do not standardize the data before applying the k-means algorithm, we end up with poor clustering.

Goal 2.

Treating each quality score as a separate group, the test statistic (951.28) exceeds the critical value at level 95% (73.31). If we group wines into Low (Quality 3-4), Medium (Quality 5-6), and High (Quality 7-8), the test statistic (554.75) also exceeds the critical value at level 95% (33.92). Thus in both cases we reject the null hypothesis that there are no differences in the mean vectors between wines with different quality scores in favor of the alternative that there are. One thing to note though that the test we are using assumes equal covariance matrices, an assumption that can be hard to verify. Grouping qualities into three categories can

possibly move the data closer to these assumptions.

Table 6: KNN (k=5) confusion matrix for wine quality

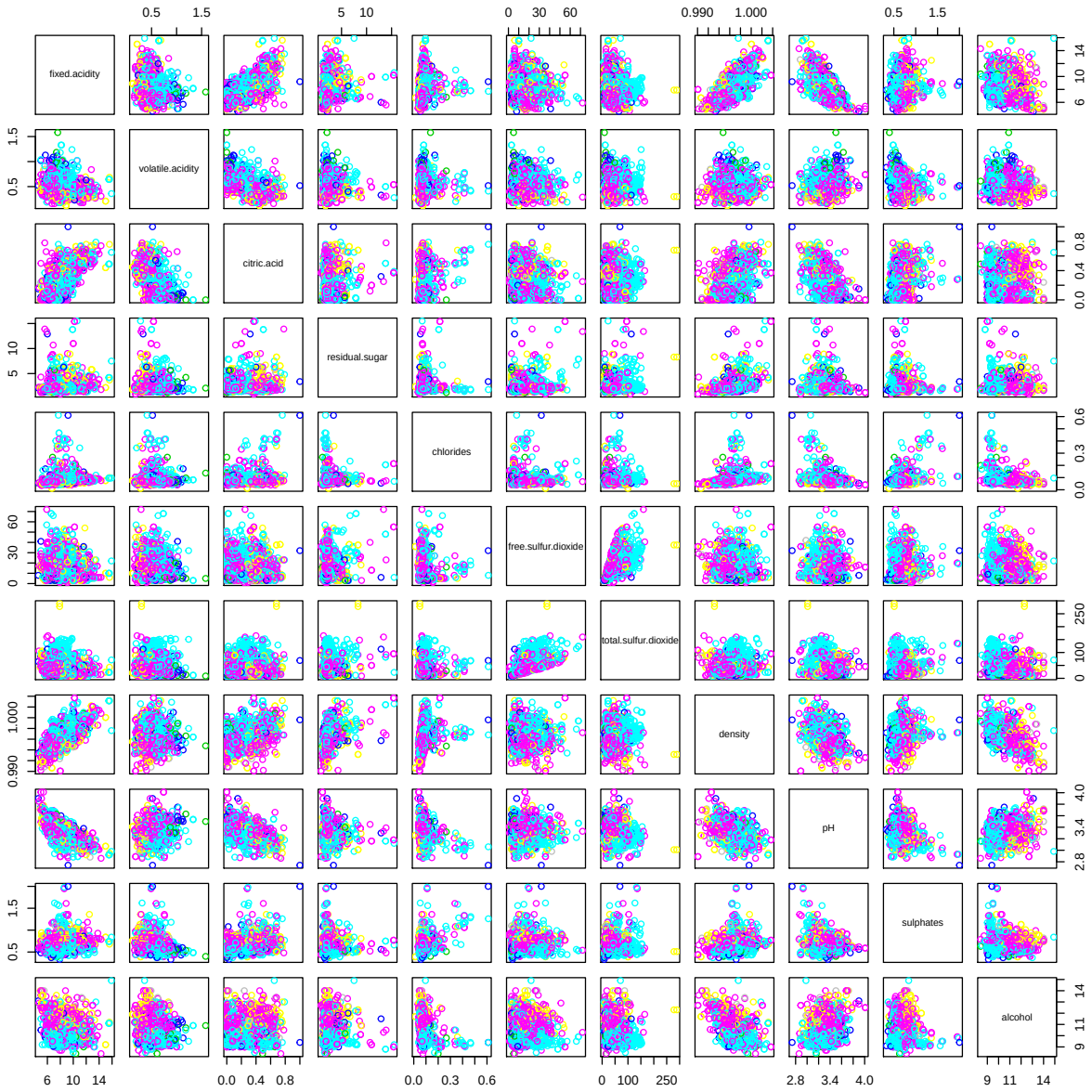
	3	4	5	6	7	8
3	0	0	0	2	1	0
4	0	0	8	6	0	0
5	0	0	126	69	9	0
6	0	0	76	93	26	0
7	0	0	16	21	21	0
8	0	0	2	3	0	0

Table 7: CART confusion matrix for wine quality

	3	4	5	6	7	8
3	0	0	3	0	0	0
4	0	0	10	4	0	0
5	0	0	162	40	2	0
6	0	0	88	85	22	0
7	0	0	8	26	24	0
8	0	0	0	3	2	0

Using the original 11 variables (on their original scales), we again see that CART (error rate = 0.434) outperforms KNN (error rate = 0.499). In this case however, neither is any good since they both predict over 40% of the red wine qualities incorrectly. We explored scaling the data and observed no real improvements in CART performance.

We also made the pairs plot color-coded by wine quality, and we observed that certain sets of variables were able to separate the wines into different quality groups.



We performed PCA on the 11 chemical attributes and used the first two principal component scores as predictors. We also explored the effect of scaling the variables. The findings were surprising: without scaling the input variables, both KNN (error rate = 0.317) and CART (error rate = 0.501) performed similarly to when inputs were not scaled, although KNN performs better. When inputs were scaled, KNN (error rate = 0.033) performed drammatically better than CART (error rate = 0.478). This might be due to that the original variables were measured on very different scales, so scaling the varaibles make them much better predictors. We also give the confusion matrix for KNN using the first two PCs with scaled input:

Table 8: confusion matrix for KNN using the first two PCs with scaled input predicting wine quality

	3	4	5	6	7	8
3	1	2	0	0	0	0
4	0	9	5	0	0	0

	3	4	5	6	7	8
5	0	0	204	0	0	0
6	0	0	4	191	0	0
7	0	0	0	0	58	0
8	0	0	0	0	5	0

Appendix

All the codes are appended here:

```
# 1 a) Hotelling's T2 test
winequality.red <- read.csv('winequality-red.csv')
winequality.white <- read.csv('winequality-white.csv')

n1 <- nrow(winequality.red)
n2 <- nrow(winequality.white)

p <- ncol(winequality.red[,-c(12)])

X1.bar <- apply(winequality.red[,-c(12)], 2, mean)
X2.bar <- apply(winequality.white[,-c(12)], 2, mean)

S1 <- cov(winequality.red[,-c(12)])
S2 <- cov(winequality.white[,-c(12)])

alpha <- 0.05

Tu2 <- t(X1.bar - X2.bar) %*% solve(1/n1*S1 + 1/n2*S2) %*% (X1.bar - X2.bar)
crit <- qchisq(alpha, p, lower.tail=F)

# simultaneous CIs
lo <- X1.bar - X2.bar - sqrt(qchisq(alpha, p, lower.tail=F) * 1/n1*diag(S1) + 1/n2*diag(S2))
hi <- X1.bar - X2.bar + sqrt(qchisq(alpha, p, lower.tail=F) * 1/n1*diag(S1) + 1/n2*diag(S2))
CI <- cbind(lo, hi)
pander(round(CI,4))

# 1 b)

# k-Nearest Neighbors
set.seed(1234567)

winequality <- rbind(winequality.red, winequality.white)[,-c(12)]
n <- nrow(winequality)
ntrain <- ceiling(n * 0.7)

train.rows <- sample(1:n, ntrain, replace=F)

wine.ClassLabel <- data.frame(rep(c("RED", "WHITE"), c(n1, n2)))
colnames(wine.ClassLabel) <- c("ClassLabel")
wine <- cbind(winequality, wine.ClassLabel)
wine.train <- wine[train.rows,]
wine.test <- wine[-train.rows,]

# Fit knn model with k = 5
```

```
knnRslt.5 <- knn(wine.train[,-c(12)], wine.test[,-c(12)], cl=wine.train$ClassLabel, k=5)

# Compare true labels to predicted labels
pander(table(wine.test$ClassLabel, knnRslt.5), caption='KNN (k=5) confusion matrix for wine type')

# Calculate classification error rate on test set
pe.knn.wine.type <- mean(wine.test$ClassLabel != knnRslt.5)

# CART

wine.tree <- rpart(ClassLabel ~ ., control=rpart.control(minsplit=30, minbucket=5), data=wine.train)

# prp(wine.tree, type=1, digits=4, extra=1, varlen=0)

wine.tree.testPredCl <- predict(wine.tree, wine.test, type="class")

pander(table(wine.test$ClassLabel, wine.tree.testPredCl), caption='CART confusion matrix for wine type'

p.e.cart.wine.type <- mean(wine.test$ClassLabel != wine.tree.testPredCl)

# calculate the conditional probability of P(classify a wine to be red | red wine)
p.red <- sum((wine.test$ClassLabel == wine.tree.testPredCl) & wine.test$ClassLabel == "RED") / sum(wine

# calculate the conditional probability of P(classify a wine to be white | white wine)
p.white <- sum((wine.test$ClassLabel == wine.tree.testPredCl) & wine.test$ClassLabel == "WHITE") / sum(w

# 1c)
# unscaled
wine.unsc <- wine[,-c(12)]
wine.distEuc <- dist(wine.unsc)

# wine.hcEuc <- hclust(wine.distEuc, method="complete")
# wine.haEuc <- hclust(wine.distEuc, method="average")
# wine.hsEuc <- hclust(wine.distEuc, method="single")


wine.km2.unsc <- kmeans(wine.unsc[,-c(12)], centers=2, nstart=10)


# Compare the k-means clustering to the complete-linkage hierarchical
# clustering with Euclidean distance cut to produce 3 clusters:

# table(wine.km2.unsc$clus, cutree(wine.hcEuc, k=2))
# table(wine.km2.unsc$clus, cutree(wine.haEuc, k=2))
# table(wine.km2.unsc$clus, cutree(wine.hsEuc, k=2))

pander(table(wine.km2.unsc$clus, wine[, 12]), caption='2-means (unscaled) clustering compared to wine ty

# scaled
wine.sc <- scale(wine[,-c(12)])
wine.distEucSc <- dist(wine.sc)

# wine.hcEucSc <- hclust(wine.distEucSc, method="complete")
```

```
# wine.haEucSc <- hclust(wine.distEucSc, method="average")
# wine.hsEucSc <- hclust(wine.distEucSc, method="single")


wine.km2 <- kmeans(wine.sc[,-c(12)], centers=2, nstart=10)


# Compare the k-means clustering to the complete-linkage hierarchical
# clustering with Euclidean distance cut to produce 3 clusters:

# table(wine.km2$clus, cutree(wine.hcEucSc, k=2))
# table(wine.km2$clus, cutree(wine.haEucSc, k=2))
# table(wine.km2$clus, cutree(wine.hsEucSc, k=2))

# not much different from unscaled clustering
pander(table(wine.km2$clus, wine[, 12]), caption='2-means (scaled) clustering compared to wine type')
p.s.km2.sc <- (1575 + 4830) / (1575 + 4830 + 68 + 24)

# Fit a model-based clustering allowing the function to select
# the best number of clusters according to the BIC.

# takes forever
# wine.mc <- Mclust(wine[,-c(12)])


# Fit a model-based clustering with a specified number of clusters
# (in this case, 3 clusters)

# computationally expensive, so omit
# wine.mc2 <- Mclust(wine[,-c(12)], G=2)


# Compare the results from k-means to the results from model-based
# clustering, both with 3 clusters:

# table(wine.mc$clas, wine.km2$clus)
# pander(table(wine.mc2$clas, wine.km2$clus))

# 2 a)
q3 <- subset(winequality.red, quality==3)[,-c(12)]
q4 <- subset(winequality.red, quality==4)[,-c(12)]
q5 <- subset(winequality.red, quality==5)[,-c(12)]
q6 <- subset(winequality.red, quality==6)[,-c(12)]
q7 <- subset(winequality.red, quality==7)[,-c(12)]
q8 <- subset(winequality.red, quality==8)[,-c(12)]

S3 <- cov(q3)
S4 <- cov(q4)
S5 <- cov(q5)
S6 <- cov(q6)
S7 <- cov(q7)
S8 <- cov(q8)

n3 <- nrow(q3)
```

```
n4 <- nrow(q4)
n5 <- nrow(q5)
n6 <- nrow(q6)
n7 <- nrow(q7)
n8 <- nrow(q8)


S <- cov(winequality.red[,-c(12)])
n <- nrow(winequality.red)

W <- (n3-1)*S3 + (n4-1)*S4 + (n5-1)*S5 + (n6-1)*S6 + (n7-1)*S7 + (n8-1)*S8
T <- (n-1)*S


alpha <- 0.05
lambda.star <- det(W) / det(T)
p <- ncol(q3)
k <- 6

# perform the test using an asympotic approximation
manova.stat1 <- -(n-1-(p+k)/2)*log(lambda.star)
crit1 <- qchisq(alpha, p*(k-1), lower.tail=F)


# group the wines into Low, Medium, and High qualities
q34 <- subset(winequality.red, quality==3 | quality==4)[,-c(12)]
q56 <- subset(winequality.red, quality==5 | quality==6)[,-c(12)]
q78 <- subset(winequality.red, quality==7 | quality==8)[,-c(12)]

S34 <- cov(q34)
S56 <- cov(q56)
S78 <- cov(q78)

n34 <- nrow(q34)
n56 <- nrow(q56)
n78 <- nrow(q78)


W <- (n34-1)*S34 + (n56-1)*S56 + (n78-1)*S78


alpha <- 0.05
lambda.star <- det(W) / det(T)
p <- ncol(q34)
k <- 3

# perform the test using an asympotic approximation
manova.stat2 <- -(n-1-(p+k)/2)*log(lambda.star)
crit2 <- qchisq(alpha, p*(k-1), lower.tail=F)

# 2 b)
# k-Nearest Neighbors
set.seed(1234567)
```

```
n.q <- nrow(winequality.red)
ntrain.q <- ceiling(n.q * 0.7)

train.rows.q <- sample(1:n.q, ntrain.q, replace=F)

wine.train.q <- winequality.red[train.rows.q,]
wine.test.q <- winequality.red[-train.rows.q,]

# Fit knn model with k = 5

knnRslt.5.q <- knn(wine.train.q[,-c(12)], wine.test.q[,-c(12)], cl=wine.train.q[,12], k=5)

# Compare true labels to predicted labels
pander(table(wine.test.q[,12], knnRslt.5.q), caption='KNN (k=5) confusion matrix for wine quality')

# Calculate classification error rate on test set
p.e.knn.wine.quality <- mean(wine.test.q[,12] != knnRslt.5.q)


# CART

wine.tree.q <- rpart(factor(quality) ~ ., control=rpart.control(minsplit=30, minbucket=5), data=wine.tra

# prp(wine.tree.q, type=1, digits=4, extra=1, varlen=0)

wine.tree.testPredCl.q <- predict(wine.tree.q, wine.test.q, type="class")

pander(table(wine.test.q[,12], wine.tree.testPredCl.q), caption='CART confusion matrix for wine quality

p.e.cart.wine.quality <- mean(wine.test.q[,12] != wine.tree.testPredCl.q)

# scaled
winequality.red.sc <- scale(winequality.red[,-12])
winequality.red.quality <-  data.frame(winequality.red$quality)
colnames(winequality.red.quality) <- c("quality")
winequality.red.sc <- cbind(winequality.red.sc, winequality.red.quality)
n.q.sc <- nrow(winequality.red.sc)
ntrain.q.sc <- ceiling(n.q.sc * 0.7)

train.rows.q.sc <- sample(1:n.q.sc, ntrain.q.sc, replace=F)

wine.train.q.sc <- winequality.red.sc[train.rows.q.sc,]
wine.test.q.sc <- winequality.red.sc[-train.rows.q.sc,]

# Fit knn model with k = 5

knnRslt.5.q.sc <- knn(wine.train.q.sc[,-c(12)], wine.test.q.sc[,-c(12)], cl=wine.train.q.sc[,12], k=5)

# Compare true labels to predicted labels
# table(wine.test.q.sc[,12], knnRslt.5.q.sc)

# Calculate classification error rate on test set
# mean(wine.test.q.sc[,12] != knnRslt.5.q.sc)
```

```
# CART

wine.tree.q.sc <- rpart(factor(quality) ~ ., control=rpart.control(minsplit=30, minbucket=5), data=data

# prp(wine.tree.q.sc, type=1, digits=4, extra=1, varlen=0)

wine.tree.testPredCl.q.sc <- predict(wine.tree.q.sc, wine.test.q.sc, type="class")

# table(wine.test.q.sc[,12], wine.tree.testPredCl.q.sc)

# mean(wine.test.q.sc[,12] != wine.tree.testPredCl.q.sc)

# 2 c)

# PCA
pairs(winequality.red[,-12], col=winequality.red[,12])

wine.pc.unsc <- prcomp(winequality.red[,-12])
wine.pc.2.unsc <-  wine.pc.unsc$x[,c(1,2)]
qual <- data.frame(winequality.red$quality)
colnames(qual) <- c("quality")
wine.pc.2.unsc <- cbind(wine.pc.2.unsc, qual)

wine.pc.2.train.q.unsc <- wine.pc.2.unsc[train.rows.q,]
wine.pc.2.test.q.unsc <- wine.pc.2.unsc[-train.rows.q,]

# Fit knn model with k = 5

knnRslt.5.q.unsc <- knn(wine.pc.2.train.q.unsc, wine.pc.2.test.q.unsc, cl=wine.pc.2.train.q.unsc[,3], k=

# Compare true labels to predicted labels
# table(wine.pc.2.test.q.unsc[,3], knnRslt.5.q.unsc)

# Calculate classification error rate on test set
p.e.knn.wine.quality.pc2.unsc <- mean(wine.pc.2.test.q.unsc[,3] != knnRslt.5.q.unsc)


# CART
wine.tree.pc.2.q.unsc <- rpart(factor(quality) ~ ., control=rpart.control(minsplit=30, minbucket=5), da

# prp(wine.tree.pc.2.q.unsc, type=1, digits=4, extra=1, varlen=0)

wine.tree.testPredCl.pc.2.q.unsc <- predict(wine.tree.pc.2.q.unsc, wine.pc.2.test.q.unsc, type="class")

# table(wine.pc.2.test.q.unsc[,3], wine.tree.testPredCl.pc.2.q.unsc)

p.e.cart.wine.quality.pc2.unsc <- mean(wine.pc.2.test.q.unsc[,3] != wine.tree.testPredCl.pc.2.q.unsc)


# scaled
winequality.red.sc <- scale(winequality.red[,-12])
wine.pc.sc <- prcomp(winequality.red.sc)
wine.pc.2.sc <-  wine.pc.sc$x[,c(1,2)]
qual <- data.frame(winequality.red$quality)
```

```
colnames(qual) <- c("quality")
wine.pc.2.sc <- cbind(wine.pc.2.sc, qual)

wine.pc.2.train.q.sc <- wine.pc.2.sc[train.rows.q,]
wine.pc.2.test.q.sc <- wine.pc.2.sc[-train.rows.q,]

# Fit knn model with k = 5

knnRslt.5.q.sc <- knn(wine.pc.2.train.q.sc, wine.pc.2.test.q.sc, cl=wine.pc.2.train.q.sc[,3], k=5)


# Calculate classification error rate on test set
p.e.knn.wine.quality.pc2.sc <- mean(wine.pc.2.test.q.sc[,3] != knnRslt.5.q.sc)

# CART

wine.tree.pc.2.q.sc <- rpart(factor(quality) ~ ., control=rpart.control(minsplit=30, minbucket=5), data=

# prp(wine.tree.pc.2.q.sc, type=1, digits=4, extra=1, varlen=0)

wine.tree.testPredCl.pc.2.q.sc <- predict(wine.tree.pc.2.q.sc, wine.pc.2.test.q.sc, type="class")

# table(wine.pc.2.test.q.sc[,3], wine.tree.testPredCl.pc.2.q.sc)

p.e.cart.wine.quality.pc2.sc <- mean(wine.pc.2.test.q.sc[,3] != wine.tree.testPredCl.pc.2.q.sc)

# Compare true labels to predicted labels
pander(table(wine.pc.2.test.q.sc[,3], knnRslt.5.q.sc), caption='confusion matrix for KNN using the first
```