

1 Solution – Assignment 1

This document is produced with Sweave. For that reason all uses of ggplot2 functions for plotting need to be done inside a `print` – otherwise there will be no figures.

```
> require(MASS)
> require(ggplot2)
> setwd("~/courses/statlearn/statlearn_2009/R/")
> load("Assignment1.RData")
```

1.1 Question 1

```
> X <- as.matrix(Assignment1Train[, -16])
> y <- Assignment1Train[, 16]
```

It seems that 230 has some outlier values and it is exclude.

```
> X <- X[-230, ]
> y <- y[-230]
```

Computing group means and the estimate of the covariance matrix.

```
> groupMeans <- apply(X, 2, function(x) tapply(x, y, mean))
> SigmaHat <- t(X - groupMeans[y, ]) %*% (X - groupMeans[y, ])/(dim(X)[1] -
+ 3)
```

	D8S1179	D21S11	D7S820	CSF1PO	D3S1358	TH01	D13S317	D16S539	D2S1338	D19S433	vWA	TPOX	D18S51	D5S818	FGA
African American	27.17	59.57	19.70	21.28	31.70	15.12	23.28	21.68	21.68	41.74	26.91	32.87	17.76	23.24	45.82
Caucasian	25.57	59.79	20.02	22.70	32.18	15.95	22.17	22.80	22.80	41.43	27.84	33.40	18.37	23.02	43.93
Hispanic	25.99	60.20	20.59	22.30	32.01	15.58	21.68	22.41	22.41	40.16	27.56	33.04	18.81	22.61	45.30

Table 1: Group Means

	D8S1179	D21S11	D7S820	CSF1PO	D3S1358	TH01	D13S317	D16S539	D2S1338	D19S433	vWA	TPOX	D18S51	D5S818	FGA
D8S1179	5.11	0.01	0.06	-0.10	-0.06	0.29	0.26	0.02	-0.75	0.17	-0.17	0.30	-0.34	0.13	-0.01
D21S11	0.01	6.07	0.30	-0.16	-0.22	-0.05	-0.29	-0.20	0.10	-0.16	0.07	-0.04	0.07	-0.12	-0.15
D7S820	0.06	0.30	3.63	0.31	-0.39	-0.15	0.17	0.01	-0.15	0.15	0.02	0.13	0.48	-0.04	0.18
CSF1PO	-0.10	-0.16	0.31	3.22	-0.37	0.05	0.13	0.06	-0.35	-0.15	0.04	0.05	0.17	0.22	0.39
D3S1358	-0.06	-0.22	-0.39	-0.37	2.55	-0.03	-0.11	-0.18	0.29	0.01	-0.19	-0.02	0.35	-0.03	-0.37
TH01	0.29	-0.05	-0.15	0.05	-0.03	3.39	-0.10	0.15	-0.51	-0.15	-0.10	0.24	0.05	0.04	-0.04
D13S317	0.26	-0.29	0.17	0.13	-0.11	-0.10	4.88	0.29	-0.14	-0.07	-0.17	-0.06	0.04	0.12	-0.07
D16S539	0.02	-0.20	0.01	0.06	-0.18	0.15	0.29	4.09	0.08	0.22	0.22	-0.04	-0.00	0.17	-0.06
D2S1338	-0.75	0.10	-0.15	-0.35	0.29	-0.51	-0.14	0.08	16.48	0.14	0.14	-1.00	0.27	-0.11	0.15
D19S433	0.17	-0.16	0.15	-0.15	0.01	-0.15	-0.07	0.22	0.14	2.85	-0.28	-0.09	0.17	-0.09	-0.25
vWA	-0.17	0.07	0.02	0.04	-0.19	-0.10	-0.17	0.22	0.14	-0.28	3.96	-0.08	-0.30	-0.17	0.32
TPOX	0.30	-0.04	0.13	0.05	-0.02	0.24	-0.06	-0.04	-1.00	-0.09	-0.08	5.15	0.19	-0.07	-0.03
D18S51	-0.34	0.07	0.48	0.17	0.35	0.05	0.04	-0.00	0.27	0.17	-0.30	0.19	11.09	0.18	0.14
D5S818	0.13	-0.12	-0.04	0.22	-0.03	0.04	0.12	0.17	-0.11	-0.09	-0.17	-0.07	0.18	3.07	0.01
FGA	-0.01	-0.15	0.18	0.39	-0.37	-0.04	-0.07	-0.06	0.15	-0.25	0.32	-0.03	0.14	0.01	9.01

Table 2: Sigmahat

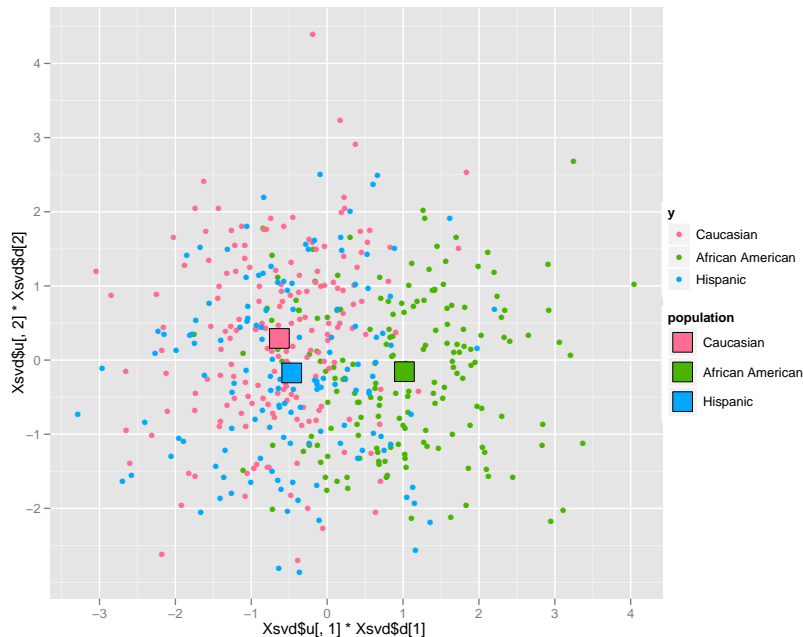
For later use in subsequent plots we compute the group means after a centering and scaling.

```
> groupMeans <- apply(scale(X), 2, function(x) tapply(x, y, mean))
```

1.2 Question 2

We make this plot after centering (compulsory) and scaling (optional, but generally recommended).

```
> Xsvd <- svd(scale(X))
> population <- levels(Assignment1Train[, 16])[c(2, 1, 3)]
> print(qplot(Xsvd$u[, 1] * Xsvd$d[1], Xsvd$u[, 2] * Xsvd$d[2],
+   colour = y) + geom_point(aes(groupMeans[c(2, 1, 3), ] %*%
+   Xsvd$v[, 1], groupMeans[c(2, 1, 3), ] %*% Xsvd$v[, 2], fill = population),
+   colour = I("black"), shape = 22, size = 8))
```

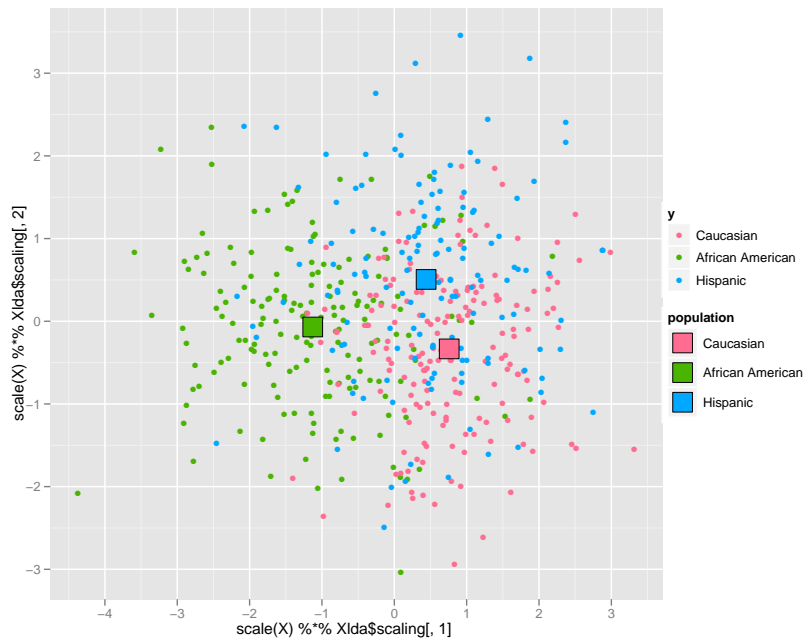


1.3 Question 3

We do this using centered and scaled X 's. The actual lda classifier is not affected by this (just remember that new observations should be centered and scaled using the same estimated means and variances) but the resulting plots are then in a standardized form, which is common.

```
> Xlda <- lda(scale(X), y)
> print(qplot(scale(X) %*% Xlda$scaling[, 1], scale(X) %*% Xlda$scaling[,
+   2], colour = y) + geom_point(aes(groupMeans[c(2, 1, 3), ] %*%
+   Xlda$scaling[, 1], groupMeans[c(2, 1, 3), ] %*%
+   Xlda$scaling[, 2], fill = population),
+   colour = I("black"), shape = 22, size = 8))
```

```
+ Xlda$scaling[, 1], groupMeans[c(2, 1, 3), ] %*% Xlda$scaling[,
+ 2], fill = population), colour = I("black"), shape = 22,
+ size = 8))
```



1.4 Question 4

```
> X <- X[y != "Hispanic", ]
> y <- y[y != "Hispanic", drop = TRUE]
> Xlda <- lda(X, y)
```

Computing misclassification tables.

```
> pred <- table(predict(Xlda)$class, y)
> print(pred)
```

	y	
	African American	Caucasian
African American	143	22
Caucasian	30	154

```
> print(pred/sum(pred), digits = 3)
```

	y	
	African American	Caucasian
African American	0.410	0.063
Caucasian	0.086	0.441

Then computing the number of misclassifications and the relative number of misclassifications.

```
> print(pred[1, 2] + pred[2, 1])  
[1] 52  
  
> print((pred[1, 2] + pred[2, 1])/sum(pred), digits = 3)  
[1] 0.149
```

1.5 Question 5

```
> Xglm <- glm(population ~ ., data = cbind(X, data.frame(population = y)),  
+ family = binomial)
```

Computing misclassification tables.

```
> pred <- table(levels(y)[round(predict(Xglm, type = "response")) +  
+ 1], y)  
> print(pred)
```

	y	
	African American	Caucasian
African American	144	24
Caucasian	29	152

```
> print(pred/sum(pred), digits = 3)
```

	y	
	African American	Caucasian
African American	0.4126	0.0688
Caucasian	0.0831	0.4355

Then computing the number of misclassifications and the relative number of misclassifications.

```
> print(pred[1, 2] + pred[2, 1])  
[1] 53  
  
> print((pred[1, 2] + pred[2, 1])/sum(pred), digits = 3)  
[1] 0.152
```

1.6 Question 6

```
> mu1Hat <- Xlda$means[2, ]
> mu0Hat <- Xlda$means[1, ]
> residuals <- X - Xlda$means[y, ]
> SigmaHat <- t(residuals) %*% residuals/(dim(X)[1] - 2)
> tauHatBeta0 <- log(Xlda$prior[2]/Xlda$prior[1]) + (t(mu0Hat) %*%
+   solve(SigmaHat, mu0Hat) - t(mu1Hat) %*% solve(SigmaHat, mu1Hat))/2
> tauHatBeta <- solve(SigmaHat, mu1Hat - mu0Hat)
> hat <- data.frame(tauHat = c(tauHatBeta0, tauHatBeta), Hat = Xglm$coef)
> print(hat)
```

	tauHat	Hat
	-18.3628	-20.9147
D8S1179	-0.3760	-0.3948
D21S11	0.0473	0.0897
D7S820	0.1326	0.1534
CSF1P0	0.5268	0.5846
D3S1358	0.3123	0.2789
TH01	0.2931	0.2979
D13S317	-0.2494	-0.2969
D16S539	0.2356	0.2106
D2S1338	-0.0151	-0.0171
D19S433	0.4000	0.4427
vWA	0.1815	0.2094
TP0X	0.0762	0.0921
D18S51	-0.2497	-0.2392
D5S818	-0.1165	-0.1152
FGA	-0.2238	-0.2539

When we do two class classification the scalings from lda and the $\hat{\beta}$ coefficient from logistic regression are not directly comparable, but tauHatBeta above, which is comparable with $\hat{\beta}$, is in fact proportional to the scaling. The constant of proportionality comes from two facts. First, there is a different "centering" used for the scaling in the affine space spanned by $\hat{\mu}_1$ and $\hat{\mu}_2$, and second, the scaling vector is by definition of unit length when we use the inner product given by $\hat{\Sigma}$.

```
> tildeBeta <- solve(SigmaHat, mu1Hat - (Nk[2] * mu1Hat + Nk[1] *
+   mu0Hat)/sum(Nk))
> tildeBeta <- tildeBeta/sqrt(tildeBeta %*% SigmaHat %*% tildeBeta)
> lda(X, y)$scaling/tildeBeta
```

	LD1
D8S1179	1
D21S11	1
D7S820	1
CSF1P0	1
D3S1358	1
TH01	1

D13S317	1
D16S539	1
D2S1338	1
D19S433	1
vWA	1
TPOX	1
D18S51	1
D5S818	1
FGA	1

If you turn to classification problems with the number of groups $K \geq 3$ the $\hat{\beta}_1, \dots, \hat{\beta}_{K-1}$ vectors from multinomial regression are in general not comparable in any direct way to the scalings computed from lda. In this situation the only way to get something out of lda that is comparable to the estimates from the multinomial regression is to compute the plug-in estimates of the parameter function τ .

```
> trainPosterior <- data.frame(logis = predict(Xglm, type = "response"),
+   lda = predict(Xlda)$posterior[, 2])[order(y), ]
> print(ggplot(data = melt(cbind(data.frame(index = 1:(dim(X)[1])),
+   diff = trainPosterior[, 1] - trainPosterior[, 2] + 0.5),
+   trainPosterior), id = "index"), aes(x = index, y = value,
+   col = variable)) + geom_point())
```



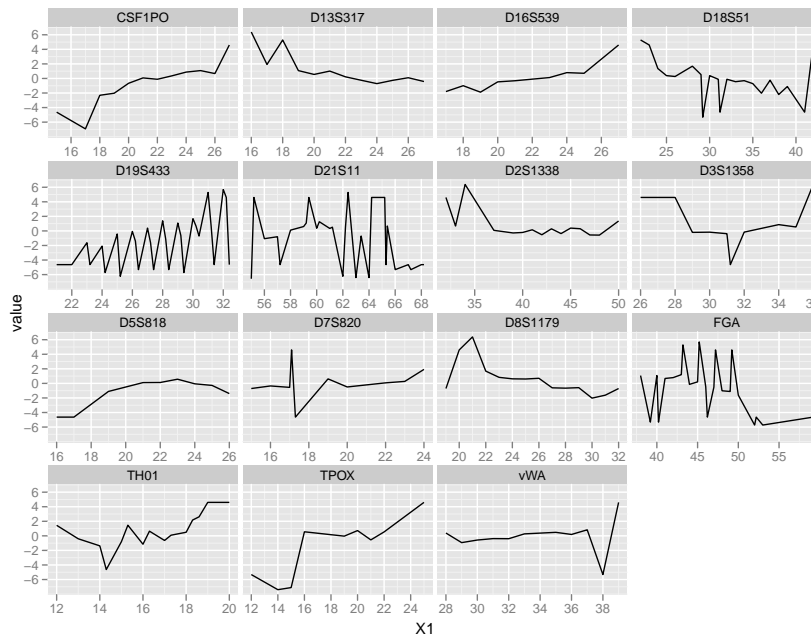
1.7 Question 7

```
> allLevels <- levels(factor(c(unlist(Assignment1Train[, -16]),
+   unlist(Assignment1Test[, -16])))))
```

```
> Assignment1TrainMod <- data.frame(lapply(as.data.frame(X), function(x) factor(x,
+   levels = allLevels))))
```

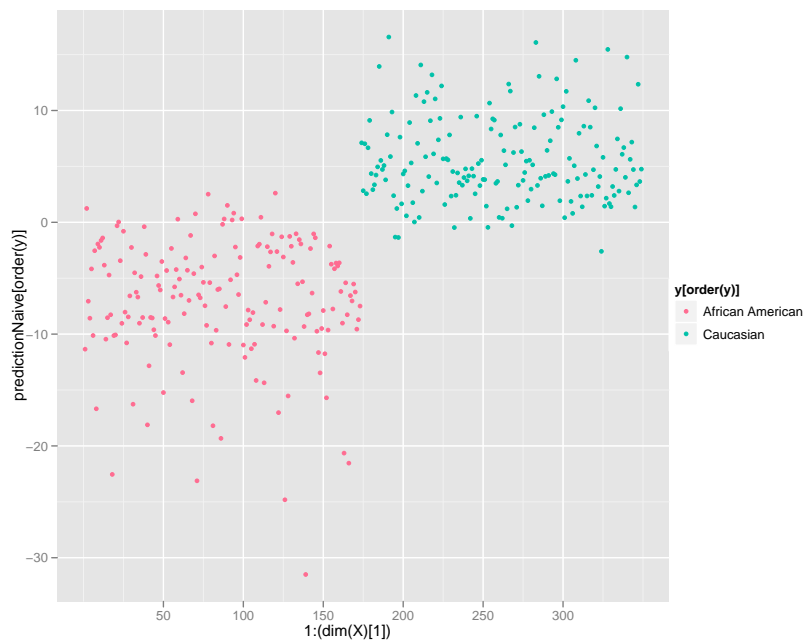
We compute the estimates of the marginal distributions.

```
> counts <- lapply(Assignment1TrainMod, function(x) tapply(x, y,
+   table))
> epsilon <- 0.01
> h <- lapply(counts, function(x) lapply(x, function(x) (x + epsilon)/sum(x +
+   epsilon)))
> logit <- sapply(h, function(x) log((x[[2]])/(x[[1]])))
> logitMelt <- melt(logit)
> print(qplot(x = X1, y = value, data = logitMelt[abs(logitMelt$value) >
+   0.02, ], geom = "line") + facet_wrap(~X2, scale = "free_x"))
```



Computing the training error.

```
> Nk <- table(y)
> intercept <- log(Nk[2]/Nk[1])
> predictionNaive <- apply(X, 1, function(x) intercept + sum(diag(logit[as.character(x),
+   ])))
> print(qplot(1:(dim(X)[1]), predictionNaive[order(y)], shape = I(20),
+   colour = y[order(y)]))
```

Computing the misclassification tables.

```
> pred <- table(levels(y)[(predictionNaive > 0) + 1], y)
> print(pred)
```

	y	
	African American	Caucasian
African American	161	6
Caucasian	12	170

```
> print(pred/sum(pred), digits = 3)
```

	y	
	African American	Caucasian
African American	0.4613	0.0172
Caucasian	0.0344	0.4871

Then computing the number of misclassifications and the relative number of misclassifications.

```
> print(pred[1, 2] + pred[2, 1])
```

```
[1] 18
```

```
> print((pred[1, 2] + pred[2, 1])/sum(pred), digits = 3)
```

```
[1] 0.0516
```

1.8 Question 8

```
> yTest <- as.factor(as.vector(Assignment1Test[, 16]))
> XTest <- Assignment1Test[, -16]
> predictionNaive <- apply(XTest, 1, function(x) intercept + sum(diag(logit[as.character(x),
+      ])))
```

Naive test error:

```
> pred <- table(levels(yTest)[(predictionNaive > 0) + 1], yTest)
> print(pred)
```

	yTest	
	African American	Caucasian
African American	65	12
Caucasian	17	75

```
> print(pred/sum(pred), digits = 3)
```

	yTest	
	African American	Caucasian
African American	0.385	0.071
Caucasian	0.101	0.444

Then computing the number of misclassifications and the relative number of misclassifications.

```
> print(pred[1, 2] + pred[2, 1])
```

```
[1] 29
```

```
> print((pred[1, 2] + pred[2, 1])/sum(pred), digits = 3)
```

```
[1] 0.172
```

LDA test error:

```
> pred <- table(predict(Xlda, XTest)$class, yTest)
> print(pred)
```

	yTest	
	African American	Caucasian
African American	61	19
Caucasian	21	68

```
> print(pred/sum(pred), digits = 3)
```

	yTest	
	African American	Caucasian
African American	0.361	0.112
Caucasian	0.124	0.402

Then computing the number of misclassifications and the relative number of misclassifications.

```
> print(pred[1, 2] + pred[2, 1])
```

```
[1] 40
```

```
> print((pred[1, 2] + pred[2, 1])/sum(pred), digits = 3)
```

```
[1] 0.237
```

Glm test error:

```
> pred <- table(levels(y)[round(predict(Xglm, XTest, type = "response")) +
+ 1], yTest)
> print(pred)
```

	yTest	
	African American	Caucasian
African American	65	21
Caucasian	17	66

```
> print(pred/sum(pred), digits = 3)
```

	yTest	
	African American	Caucasian
African American	0.385	0.124
Caucasian	0.101	0.391

Then computing the number of misclassifications and the relative number of misclassifications.

```
> print(pred[1, 2] + pred[2, 1])
```

```
[1] 38
```

```
> print((pred[1, 2] + pred[2, 1])/sum(pred), digits = 3)
```

```
[1] 0.225
```

The Naive Bayes method seems to be able to make the best predictions - also on the test data. The method does adapt a lot to the concrete dataset and we see a considerable increase in the estimated expected prediction error on the test data compared to misclassification rate on the training data for the Naive Bayes procedure. However, it also seems that the Naive Bayes procedure captures some important, non-linear effects that improves on the prediction.

Some have noted that there is an implementation of “naive Bayes” in the `e1071` library. In effect this is a diagonal QDA – a quadratic discriminant analysis but where the covariance matrices are assumed diagonal. It turns out that this method performs surprisingly well and seems to have the smallest generalization error. It does not suffer from the many free parameters that are in my hand-crafted naive Bayes procedure above, but it incorporates quadratic terms, which seem to be very important.

1.9 Generalizations

If we think of include quadratic terms in a more direct way, we can do so, either explicitly using a logistic regression model or through more general non-linear effects in a generalized additive model. Using logistic regression with quadratic terms included:

```
> form <- as.formula(paste("population~", paste(colnames(X), collapse = "+"),
+      "+", paste("I(", colnames(X), "^2)", sep = "", collapse = "+")))
> XQglm <- glm(form, data = cbind(X, data.frame(population = y)),
+      family = binomial)

> pred <- table(levels(y)[round(predict(XQglm, XTest, type = "response")) +
+      1], yTest)
> print(pred)
```

	yTest	
	African American	Caucasian
African American	66	18
Caucasian	16	69

```
> print(pred/sum(pred), digits = 3)

yTest
African American Caucasian
African American 0.3905 0.1065
Caucasian        0.0947 0.4083

> print(pred[1, 2] + pred[2, 1])

[1] 34

> print((pred[1, 2] + pred[2, 1])/sum(pred), digits = 3)

[1] 0.201
```

Using a generalized additive model

```
> require(gam)
> form <- as.formula(paste("population~", paste("s(", colnames(X),
+      ",df=3)", sep = "", collapse = "+")))
> Xgam <- gam(form, data = cbind(X, data.frame(population = y)),
+      family = binomial)

> pred <- table(levels(y)[round(predict(Xgam, XTest, type = "response")) +
+      1], yTest)
> print(pred)
```

	yTest	
	African American	Caucasian
African American	66	15
Caucasian	16	72

```
> print(pred/sum(pred), digits = 3)
```

	yTest	
	African American	Caucasian
African American	0.3905	0.0888
Caucasian	0.0947	0.4260

```
> print(pred[1, 2] + pred[2, 1])
```

```
[1] 31
```

```
> print((pred[1, 2] + pred[2, 1])/sum(pred), digits = 3)
```

```
[1] 0.183
```