# Math 189: Final Exam Solution

## Introduction

In this final exam project for Math 189 the Swiss bank notes dataset is analyzed. We wish to predict whether a note is false or counterfeit using supervised learning. The dataset was provided in the course materials (https://github.com/tuckermcelroy/ma189), and is taken from "Multivariate Statistics: A practical approach", by Bernhard Flury and Hans Riedwyl, Chapman and Hall, 1988.

The data contains six variables measured on 100 genuine and 100 counterfeit old Swiss 1000-franc bank notes. In particular, BN1 to BN100 are genuine banknotes and BN101 to BN200 are counterfeit banknotes. The six features are:

1. Length of the note
2. Width of the Left-Hand side of the note
3. Width of the Right-Hand side of the note
4. Width of the Bottom Margin
5. Width of the Top Margin
6. Diagonal Length of Printed Area

There will be several elements to our analysis, utilizing many of the concepts from the course. In particular, we will use $K$-fold cross-validation to determine and fit the best models, and we will compare both Linear Discriminant Analysis and Logistic Regression. Also we will consider whether a factor model on the 6 explanatory variables can be used to refine the predictive models.

## Analysis

### Overview

The general approach we take is to first load and explore the data. Then we wish to compare Linear Discriminant Analysis (LDA) with Logistic Regression (LR), in regards to their ability to discern real versus fake bills. In order to assess which performs better, we will use concepts from cross-validation, examining which method performs better on a validation set. Thus, each method will be trained on a training set.

In order to guard against biases that might arise from a single split of the data, we will use the $K$-fold Cross Validation method, where we choose $K = 5$. This will allow us to essentially make $K = 5$ comparisons of LDA and LR, so that our assessment is not contingent on a single splitting. However, note that certain modeling features are particular to each split, and will be in essence "averaged out."

For instance, in LDA the determination of $\alpha$ and $\beta$ depends on estimates obtained from the particular split of the data; likewise in LR the choice of significant covariates may vary from fold to fold. These model parameters and settings are not determined outside of the cross-validation loop, and are incorporated into the overall comparison between LDA and LR.

A further element of our analysis involves examining the 6 covariates to see whether they can be culled via factor modeling. Note that the selection of a subset of variables is also associated to predictive ability (for the binary variable indicating counterfeit status), in the sense that a dimension reduction will only

be entertained if there is not substantial decrease in predictive performance. A nuance is that this factor analysis is also performed within the cross-validation loop, for each split of the data, and hence the selection of dimension (as well as determination of the factor scores) is contingent on each particular fold.

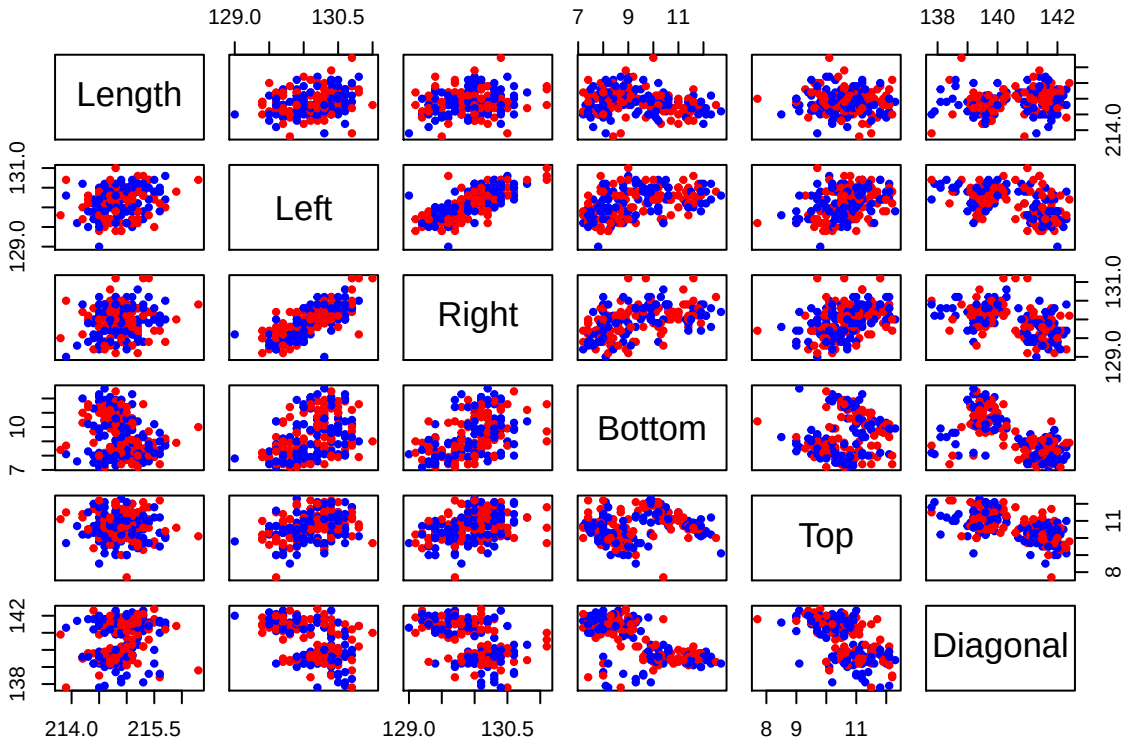
## Loading and Exploring the Data

- We begin by loading the data, and examining the relationships via scatterplot.

```
swiss <- read.table("SBN.txt")
n_real <- 100
n_fake <- 100
swiss$real <- c(rep(1,n_real),rep(0,n_fake))
```

- The red dots are real francs, and blue dots are fake francs.
- There seems to be a fair amount of mingling.

```
pairs(rbind(swiss[swiss$real==1,1:6],swiss[swiss$real==0,1:6]),
      pch=20,col=c(2,4))
```




## Spliting the Data

Our purpose now is to split the data into $K = 5$ folds. This means taking 20% of the real and fake notes as a validation set, and the other 80% (consisting of the other 4 folds) as the training set. Since there is

no reason to think the data has any ordering, we deterministically split the data. Our first fold is obtained from the first 20 records for both real and fake bills, and this will be the *validation* (or test) set. We walk through this analysis explicitly for the first fold; later in the report, all the code is repeated for all 5 folds.

```
fold1.ind <- seq(1,20)
fold1.ind <- c(fold1.ind,100+fold1.ind)
swiss.train <- swiss[-fold1.ind,]
swiss.test <- swiss[fold1.ind,]
```

## Linear Discriminant Analysis

We take a straightforward implementation of LDA. The prior probabilities are set to the respective proportions of real and fake bills, or $p = .5$ for both. The following code estimates the parameters on the training set, and computes estimates of $\alpha$ and $\beta$.

### Train

We fit the LDA procedure to the training data, outputting the $\alpha$ (intercept) and $\beta$ (slope) coefficients.

```
# set priors
p_real <- 1/2
p_fake <- 1/2

# mean vectors
Mean_real <- colMeans(swiss.train[swiss.train$real == 1,1:6])
Mean_fake <- colMeans(swiss.train[swiss.train$real == 0,1:6])
#rbind(Mean_real,Mean_fake)

# pooled covariance
S_real <- cov(swiss.train[swiss.train$real == 1,1:6])
S_fake <- cov(swiss.train[swiss.train$real == 0,1:6])
S_pooled <- ((n_real-1)*S_real +(n_fake-1)*S_fake)/(n_real+n_fake-2)
#S_pooled

# intercepts
S_inv <- solve(S_pooled)
alpha_real <- -0.5* t(Mean_real) %*% S_inv %*% Mean_real + log(p_real)
alpha_fake <- -0.5* t(Mean_fake) %*% S_inv %*% Mean_fake + log(p_fake)
alpha <- c(alpha_real,alpha_fake)
alpha
```

```
## [1] -244210.3 -242467.8
```

```
# slopes
beta_real <- S_inv %*% Mean_real
beta_fake <- S_inv %*% Mean_fake
beta <- cbind(beta_real,beta_fake)
beta
```

```
##                [,1]       [,2]
## Length    1083.4886 1082.34223
```

```
## Left      866.9856  861.59440
## Right     348.1310  353.95531
## Bottom   -191.1723 -182.18510
## Top      -107.9467  -99.29522
## Diagonal  709.5336  697.22391
```

### Evaluate

In order to assess performance, we obtain the confusion matrix, or the tally of correct and incorrect classifications. The off-diagonal entries in the confusion matrix correspond to tallies of incorrect classifications, so by summing these cells we get a measure of performance, lower values being better. We see that classification is perfect, which may be a fluke due to the splitting.

```
predictions <- NULL

for(i in 1:nrow(swiss.test)){
    #Read an ouservation in test data
    x <- t(swiss.test[i,1:6])

    #Calculate linear discriminant functions for each
    d_real <- alpha_real + t(beta_real) %*% x
    d_fake <- alpha_fake + t(beta_fake) %*% x

    prediction <- 0
    if(d_real >= d_fake) { prediction <- 1 }
    predictions <- c(predictions,prediction)
}
#predictions

# Check the prediction accuracy
table(predict=predictions, truth=swiss.test$real)
```

```
##        truth
## predict  0  1
##       0 20  0
##       1  0 20
```

### Factor Analysis

We now have a LDA model that works quite well on the first fold, and we could consider refining it. Are all 6 variables really necessary? That is perhaps some linear combination of the 6 variables is really sufficient to get good classification. Reflecting on this a bit, we realize that if a factor model can be fitted to the 6 variables, then the factor scores can then be passed into the LDA framework. If the dimension is reduced from 6 to some $m < 6$, then the factor scores will be a new $n \times m$-dimensional data set (here $n = 160$, due to the splitting) with the associated binary variable indicating counterfeit status.

```
pca_result <- eigen(cor(swiss.train[,1:6]))
pca_var <- pca_result$values
pve <- pca_var/sum(pca_var)
out2 <- cbind(pca_var,pve,cumsum(pve))
colnames(out2) <- c("Eigenvalue","Proportion","Cumulative")
rownames(out2) <- c("PC1","PC2","PC3","PC4","PC5","PC6")
out2
```

```
##     Eigenvalue Proportion Cumulative
## PC1  3.0090496 0.50150827  0.5015083
## PC2  1.2715900 0.21193167  0.7134399
## PC3  0.8445859 0.14076432  0.8542043
## PC4  0.4263202 0.07105337  0.9252576
## PC5  0.2666051 0.04443418  0.9696918
## PC6  0.1818492 0.03030820  1.0000000
```

We use PCA to do the estimation and analysis, since this is more timely than running a full MLE routine. We see above that using $m = 4$ components still explains 92% of the variation. Hence we pursue a factor analysis with 4 factors.

```
factor.dim <- 4
load.mat <- pca_result$vector[,seq(1,factor.dim)]
factor.scores <- solve(t(load.mat) %*% load.mat) %*% t(load.mat) %*% t(swiss.train[,1:6])
```

### Repeat LDA with Factor Scores

We now repeat the training and evaluation of the LDA method, using this new dataset.

```
swiss.subtrain <- cbind(t(factor.scores),swiss.train$real)

# set priors
p_real <- 1/2
p_fake <- 1/2

# mean vectors
Mean_real <- colMeans(swiss.subtrain[swiss.subtrain[,factor.dim+1] == 1,1:factor.dim])
Mean_fake <- colMeans(swiss.subtrain[swiss.subtrain[,factor.dim+1] == 0,1:factor.dim])
#rbind(Mean_real,Mean_fake)

# pooled covariance
S_real <- cov(swiss.subtrain[swiss.subtrain[,factor.dim+1] == 1,1:factor.dim])
S_fake <- cov(swiss.subtrain[swiss.subtrain[,factor.dim+1] == 0,1:factor.dim])
S_pooled <- ((n_real-1)*S_real +(n_fake-1)*S_fake)/(n_real+n_fake-2)
#S_pooled

# intercepts
S_inv <- solve(S_pooled)
alpha_real <- -0.5* t(Mean_real) %*% S_inv %*% Mean_real + log(p_real)
alpha_fake <- -0.5* t(Mean_fake) %*% S_inv %*% Mean_fake + log(p_fake)
alpha <- c(alpha_real,alpha_fake)
#alpha

# slopes
beta_real <- S_inv %*% Mean_real
beta_fake <- S_inv %*% Mean_fake
beta <- cbind(beta_real,beta_fake)
#beta

predictions <- NULL

for(i in 1:nrow(swiss.test)){
```

```
    #Read an ovservation in test data
    x <- t(swiss.test[i,1:6])
    z <- solve(t(load.mat) %*% load.mat) %*% t(load.mat) %*% x

    #Calculate linear discriminant functions for each
    d_real <- alpha_real + t(beta_real) %*% z
    d_fake <- alpha_fake + t(beta_fake) %*% z

    prediction <- 0
    if(d_real >= d_fake) { prediction <- 1 }
    predictions <- c(predictions,prediction)
}
#predictions

# Check the prediction accuracy
table(predict=predictions, truth=swiss.test$real)
```

```
##        truth
## predict  0  1
##       0 20  0
##       1  0 20
```

On this first split, LDA works perfectly, and we can reduce the dimension of the covariates from 6 to 4 without any loss to our results.

### Combining LDA Results Over $K = 5$ Splits

Now we combine all this analysis over the splits by writing a small loop. We will automatically select a factor dimension in the following way: we want to explain at least 90% of the variation.

```
mspe.lda <- 0
for (fold in 1:5)
{

fold.ind <- matrix(seq(1,100),ncol=5)[,fold]
fold.ind <- c(fold.ind,100+fold.ind)
swiss.train <- swiss[-fold.ind,]
swiss.test <- swiss[fold.ind,]

pca_result <- eigen(cor(swiss.train[,1:6]))
pca_var <- pca_result$values
pve <- pca_var/sum(pca_var)
factor.dim <- min(seq(1,6)[cumsum(pve) >= .90])
load.mat <- pca_result$vector[,seq(1,factor.dim),drop=FALSE]
factor.scores <- solve(t(load.mat) %*% load.mat) %*% t(load.mat) %*% t(swiss.train[,1:6])

swiss.subtrain <- cbind(t(factor.scores),swiss.train$real)

# set priors
p_real <- 1/2
p_fake <- 1/2
```

```
# mean vectors
Mean_real <- colMeans(swiss.subtrain[swiss.subtrain[,factor.dim+1] == 1,1:factor.dim])
Mean_fake <- colMeans(swiss.subtrain[swiss.subtrain[,factor.dim+1] == 0,1:factor.dim])
#rbind(Mean_real,Mean_fake)

# pooled covariance
S_real <- cov(swiss.subtrain[swiss.subtrain[,factor.dim+1] == 1,1:factor.dim])
S_fake <- cov(swiss.subtrain[swiss.subtrain[,factor.dim+1] == 0,1:factor.dim])
S_pooled <- ((n_real-1)*S_real +(n_fake-1)*S_fake)/(n_real+n_fake-2)
#S_pooled

# intercepts
S_inv <- solve(S_pooled)
alpha_real <- -0.5* t(Mean_real) %*% S_inv %*% Mean_real + log(p_real)
alpha_fake <- -0.5* t(Mean_fake) %*% S_inv %*% Mean_fake + log(p_fake)
alpha <- c(alpha_real,alpha_fake)
#alpha

# slopes
beta_real <- S_inv %*% Mean_real
beta_fake <- S_inv %*% Mean_fake
beta <- cbind(beta_real,beta_fake)
#beta

predictions <- NULL

for(i in 1:nrow(swiss.test)){
    #Read an ovservation in test data
    x <- t(swiss.test[i,1:6])
    z <- solve(t(load.mat) %*% load.mat) %*% t(load.mat) %*% x

    #Calculate linear discriminant functions for each
    d_real <- alpha_real + t(beta_real) %*% z
    d_fake <- alpha_fake + t(beta_fake) %*% z

    prediction <- 0
    if(d_real >= d_fake) { prediction <- 1 }
    predictions <- c(predictions,prediction)
}
#predictions

# Check the prediction accuracy
out <- table(predict=predictions, truth=swiss.test$real)
#print(out)
mspe.lda <- mspe.lda + out[1,2] + out[2,1]

}
mspe.lda <- mspe.lda/5
```

The $K$-fold cross-validation MSPE is 0.2. This is quite low, and could be even better if we didn't do dimension reduction. We note that another approach would be to do factor analysis at the outset, before splitting, in which case we would obtain a best set of factor scores for LDA. This is an appealing alternative analysis to what we have done here, which in a sense incorporates dimension reduction within the LDA

analysis.

## Logistic Regression

We now consider the Logistic Regression (LR) approach to the classification problem. The basic setting involves using all 6 variables as covariates in the regression equation, and we mimic the discussion of LDA by first considering a single split.

```
fold1.ind <- seq(1,20)
fold1.ind <- c(fold1.ind,100+fold1.ind)
swiss.train <- swiss[-fold1.ind,]
swiss.test <- swiss[fold1.ind,]
```

### Train

First we examine all the variables, but the algorithm does not converge. This is indicative of either redundancy among covariates (quasi-linear dependence) or a close association of some covariates with the dependent variables; ironically, if the binary is highly predictable from some of the covariates, this can cause numerical problems.

```
library(ISLR)
y <- swiss.train$real
x <- cbind(swiss.train$Length,swiss.train$Left,swiss.train$Right,
           swiss.train$Bottom,swiss.train$Top,swiss.train$Diagonal)
all.fit <- glm(y~x,family=binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(all.fit)
```

```
##
## Call:
## glm(formula = y ~ x, family = binomial)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -3.668e-05  -2.100e-08   0.000e+00   2.100e-08   5.618e-05
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.377e+03  3.707e+07   0.000    1.000
## x1          -2.015e+01  1.427e+05   0.000    1.000
## x2          -1.863e+01  1.402e+05   0.000    1.000
## x3          -7.146e+00  1.805e+05   0.000    1.000
## x4          -3.526e+01  2.122e+04  -0.002    0.999
## x5          -1.328e+01  3.957e+04   0.000    1.000
## x6           3.409e+01  2.007e+04   0.002    0.999
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.2181e+02  on 159  degrees of freedom
## Residual deviance: 8.5228e-09  on 153  degrees of freedom
## AIC: 14
##
## Number of Fisher Scoring iterations: 25
```

Examining the correlation matrix, it seems the Diagonal variable has a tight association with the categorical.

```
cor(swiss.train)
```

```
##              Length       Left      Right     Bottom        Top   Diagonal
## Length    1.0000000  0.2210606  0.1263706 -0.1877629 -0.1154988  0.2161980
## Left      0.2210606  1.0000000  0.7437577  0.4060249  0.3694353 -0.5067844
## Right     0.1263706  0.7437577  1.0000000  0.5135590  0.4453479 -0.5582099
## Bottom   -0.1877629  0.4060249  0.5135590  1.0000000  0.1631362 -0.6168099
## Top      -0.1154988  0.3694353  0.4453479  0.1631362  1.0000000 -0.6188772
## Diagonal  0.2161980 -0.5067844 -0.5582099 -0.6168099 -0.6188772  1.0000000
## real      0.2308273 -0.4952913 -0.6211084 -0.7891710 -0.6014796  0.9032663
##                real
## Length    0.2308273
## Left     -0.4952913
## Right    -0.6211084
## Bottom   -0.7891710
## Top      -0.6014796
## Diagonal  0.9032663
## real      1.0000000
```

Therefore we could consider a LR model with only this covariate. With a bit of experimentation, we get convergence with all variables except *Bottom* included. However, only *Diagonal* is significant, and even the constant can be dropped. (However, we retain the constant because this is important for prediction.)

```
y <- swiss.train$real
x <- cbind(swiss.train$Length,swiss.train$Top,swiss.train$Right,
           swiss.train$Left,swiss.train$Diagonal)
sub.fit <- glm(y~x,family=binomial)
summary(sub.fit)
```

```
##
## Call:
## glm(formula = y ~ x, family = binomial)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.95738  -0.08989   0.00045   0.01539   3.06075
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -483.1675   583.2679  -0.828  0.40746
## x1             1.6939     3.1660   0.535  0.59263
## x2            -0.7237     1.7448  -0.415  0.67832
```

```
## x3            -2.8916     3.6386  -0.795  0.42679
## x4            -2.0886     4.7823  -0.437  0.66231
## x5             5.5229     1.7194   3.212  0.00132 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.807  on 159  degrees of freedom
## Residual deviance:  12.763  on 154  degrees of freedom
## AIC: 24.763
##
## Number of Fisher Scoring iterations: 9
```

So we finish this step with a refined model, with just the single covariate plus a constant.

```
y <- swiss.train$real
x <- swiss.train$Diagonal
best.fit <- glm(y~x,family=binomial)
summary(best.fit)
```

```
##
## Call:
## glm(formula = y ~ x, family = binomial)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9723  -0.0582   0.0006   0.0318   3.2338
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -946.488    235.480  -4.019 5.83e-05 ***
## x              6.743      1.679   4.017 5.91e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 221.807  on 159  degrees of freedom
## Residual deviance:  15.341  on 158  degrees of freedom
## AIC: 19.341
##
## Number of Fisher Scoring iterations: 9
```

### Evaluate

Like LDA, we assess performance through tallies drawn from the confusion matrix. We will consider results both from the larger model (with 5 covariates and a constant) as well as the refined model. The larger model generates predictions that each bill is genuine, regardless of the values of covariates, and hence its performance is terrible (50% success). The refined model performs quite well.

```
pred_lr <- function(x,coefs){
  pred <- as.numeric(as.numeric(x %*% coefs))
  pred <- 1/(1+exp(-pred))
  return(pred)
}

predictions.sub <- NULL
predictions.best <- NULL

for(i in 1:nrow(swiss.test)){
    #Read an ovservation in test data
    x <- t(swiss.test[i,c(1,2,3,5,6)])
    pred.sub <- pred_lr(c(1,x),sub.fit$coefficients)
    x <- t(swiss.test[i,6])
    pred.best <- pred_lr(c(1,x),best.fit$coefficients)
    prediction <- 0
    if(pred.sub > .5) { prediction <- 1 }
    predictions.sub <- c(predictions.sub,prediction)
    prediction <- 0
    if(pred.best > .5) { prediction <- 1 }
    predictions.best <- c(predictions.best,prediction)
}
#predictions.sub
#predictions.best

# Check the prediction accuracy
table(predict=predictions.sub, truth=swiss.test$real)
```

```
##        truth
## predict  0  1
##       1 20 20
```

```
table(predict=predictions.best, truth=swiss.test$real)
```

```
##        truth
## predict  0  1
##       0 19  0
##       1  1 20
```

### Factor Analysis

Given the preceding results, it is unlikely that factor analysis will be helpful at all. For one thing, the best model seems to involve a single covariate, for which no dimension reduction is possible. However, we proceed by considering various possible reductions of dimension, noting that the cumulative variability results have already been discussed above.

```
pca_result <- eigen(cor(swiss.train[,1:6]))
factor.dim <- 4
load.mat <- pca_result$vector[,seq(1,factor.dim)]
factor.scores <- solve(t(load.mat) %*% load.mat) %*% t(load.mat) %*% t(swiss.train[,1:6])
```

```
y <- swiss.train$real
x <- t(factor.scores)
factor.fit <- glm(y~x,family=binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(factor.fit)
```

```
##
## Call:
## glm(formula = y ~ x, family = binomial)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -3.952e-05  -2.100e-08   0.000e+00   2.100e-08   6.213e-05
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.112e+03  1.430e+07   0.000    1.000
## x1           4.863e+01  3.264e+04   0.001    0.999
## x2          -8.304e+00  4.475e+04   0.000    1.000
## x3          -9.597e+00  1.197e+04  -0.001    0.999
## x4          -3.164e+01  3.341e+04  -0.001    0.999
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.2181e+02  on 159  degrees of freedom
## Residual deviance: 8.9060e-09  on 155  degrees of freedom
## AIC: 10
##
## Number of Fisher Scoring iterations: 25
```

Having also tried other factor dimensions (from 5 down to 1), we find that the LR model does not fit the factor scores well. This approach shall be abandoned henceforth.

### Combining LR Results Over $K = 5$ Splits

Like the LDA, we now combine the LR analysis over the splits by writing a small loop. There will be no factor analysis, and we instead use the single covariate *Diagonal*.

```
mspe.lr <- 0
for (fold in 1:5)
{

fold.ind <- matrix(seq(1,100),ncol=5)[,fold]
fold.ind <- c(fold.ind,100+fold.ind)
swiss.train <- swiss[-fold.ind,]
swiss.test <- swiss[fold.ind,]
```

```
y <- swiss.train$real
x <- swiss.train$Diagonal
best.fit <- glm(y~x,family=binomial)

predictions.best <- NULL

for(i in 1:nrow(swiss.test)){
    #Read an ouservation in test data
    x <- t(swiss.test[i,6])
    pred.best <- pred_lr(c(1,x),best.fit$coefficients)
    prediction <- 0
    if(pred.best > .5) { prediction <- 1 }
    predictions.best <- c(predictions.best,prediction)
}

# Check the prediction accuracy
out <- table(predict=predictions.best, truth=swiss.test$real)
#print(out)
mspe.lr <- mspe.lr + out[1,2] + out[2,1]

}
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
mspe.lr <- mspe.lr/5
```

The $K$-fold cross-validation MSPE is 0.4. In the third split there was a problem with the fitting, although the predictions came out perfect anyways. In fact, there was a single error in 2 of the 5 splits, so performance is quite good overall.

## Summary

We have proposed two different approaches for categorizing counterfeit status. First, LDA with factor modeling (to dimension reduce the covariates) is explored, and secondly we examine LR with only using the sixth variable (*Diagonal*) along with a constant. Because there is a high degree of predictability on the basis of different characteristics, the error rate for both methods is extremely low. We have assessed both methods by using $K$-fold cross-validation (with $K = 5$), essentially making certain modeling decisions within the loop over the splits. It seems that the *Diagonal* variable is really driving most of the differentiation between real and fake notes, in the sense that once the *Diagonal* variable is included in the LR model, the other covariates become irrelevant. However, such a finding is not readily apparent from the LDA approach, which in some sense is more robust as an algorithm, since there is no issue of failed convergence of nonlinear optimization as occurred with LR.

Both approaches achieve excellent results on the validation set, which estimates a test set by doing out-of-sample prediction. So, either method could be reliably used. If a choice is to be made, we note that the LR method had more error and also can run into numerical problems. The LDA in contrast is more numerically stable, but does rely on tuning parameters such as the prior probabilities, and the threshold variability for the factor dimension. Regarding the dimension reduction due to factor analysis, there seems to be no pressing need to utilize this in the LDA approach, and it is actually unhelpful in LR. These considerations may lead us to prefer LDA, while noting the insights available from LR (such as the predicted probabilities of group membership) are an asset of that method.