1 Regression

1.1 Maximum Likelihood Solution

First we load the dataset, give it reasonable column names, and split it up into a training and a testing subset.

```
> rm(list=ls())
> setwd('~/github/StatML/Assignment2')
> load('./Data/bodyfat.RData')
> require(MASS)
> colnames(data)<-c('density','bodyFatPercentage','Age','Weight',
+                   'Height','Neck','Chest','Abdomen','Hip','Thigh',
+                   'Knee','Ankle','Biceps','Forearm','Wrist')
> ridx <- sample( 1:dim( data )[1], dim( data )[1] )
> trainIdx<-ridx[1:200]
> testIdx<-ridx[201:length( ridx )]
> train <- data[trainIdx,]
> test <- data[testIdx,]
```

We attempt to predict body fat using two models. The first one is based on weight, and chest, abdomen, and hip circumferences, and the second is based solely on abdominal circumference.

```
> selection1Train<-train[,c(4,7,8,9)]
> selection2Train<-train[,8]
> selection1Test<-test[,c(4,7,8,9)]
> selection2Test<-test[,8]
> targetTrain<-train[,2]
> targetTest<-test[,2]
```

We make a design matrix for every selection, and then find $\mathbf{w}_{\mathrm{ML}}$ by $\mathbf{w}_{\mathrm{ML}} = (\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-\mathbf{1}}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}$ where $(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-\mathbf{1}}\mathbf{\Phi}^{\mathrm{T}}$ is the pseudo-inverse of $\mathbf{\Phi}$.

```
> design1Train<-as.matrix(cbind(seq(1,1,length.out=nrow(selection1Train)),
+                           selection1Train))
> design2Train<-as.matrix(cbind(seq(1,1,length.out=length(selection2Train)),
+                           selection2Train))
> wML1<-ginv(design1Train)%*%targetTrain
> wML2<-ginv(design2Train)%*%targetTrain
```

Once e have found $\mathbf{w_{ML}}$ we can apply it to the test set, via $y(\mathbf{x}, \mathbf{w}) = \mathbf{w^T}\phi(\mathbf{x})$, and we can calculate the RMS via

$$\mathrm{RMS} = \sqrt{\frac{1}{N} = \sum_{n=1}^{N} (t_n - y(\mathbf{x_n}, \mathbf{w}))^{\mathbf{2}}}$$

```
> design1Test<-as.matrix(cbind(seq(1,1,length.out=nrow(selection1Test)),
+                           selection1Test))
> design2Test<-as.matrix(cbind(seq(1,1,length.out=length(selection2Test)),
```

```
+                          selection2Test))
> y1<-t(wML1)%*%t(design1Test)
> y2<-t(wML2)%*%t(design2Test)
> RMS1<-sqrt(sum((targetTest-y1)^2)/length(targetTest))
> RMS2<-sqrt(sum((targetTest-y2)^2)/length(targetTest))
> print(RMS1)

[1] 5.105047

> print(RMS2)

[1] 5.245014
```

Thus we get an RMS of 4.07 using the first model, and an RMS of 4.4 for the second model, showing the first model providing more accurate predictions.

```
> y1Train<-t(wML1)%*%t(design1Train)
> y2Train<-t(wML2)%*%t(design2Train)
> RMS1Train<-sqrt(sum((targetTrain-y1Train)^2)/length(targetTrain))
> RMS2Train<-sqrt(sum((targetTrain-y2Train)^2)/length(targetTrain))
> print(RMS1Train)

[1] 4.257557

> print(RMS2Train)

[1] 4.773962
```

For posterity, we have also calculated the error on the training set. For the two models, we get RMS of 4.52, and 4.81. The fact that the using the train error is not smaller than the test error, likely points to the fact that we are not over-fitting.

1.2 Maximum a posteriori Solution

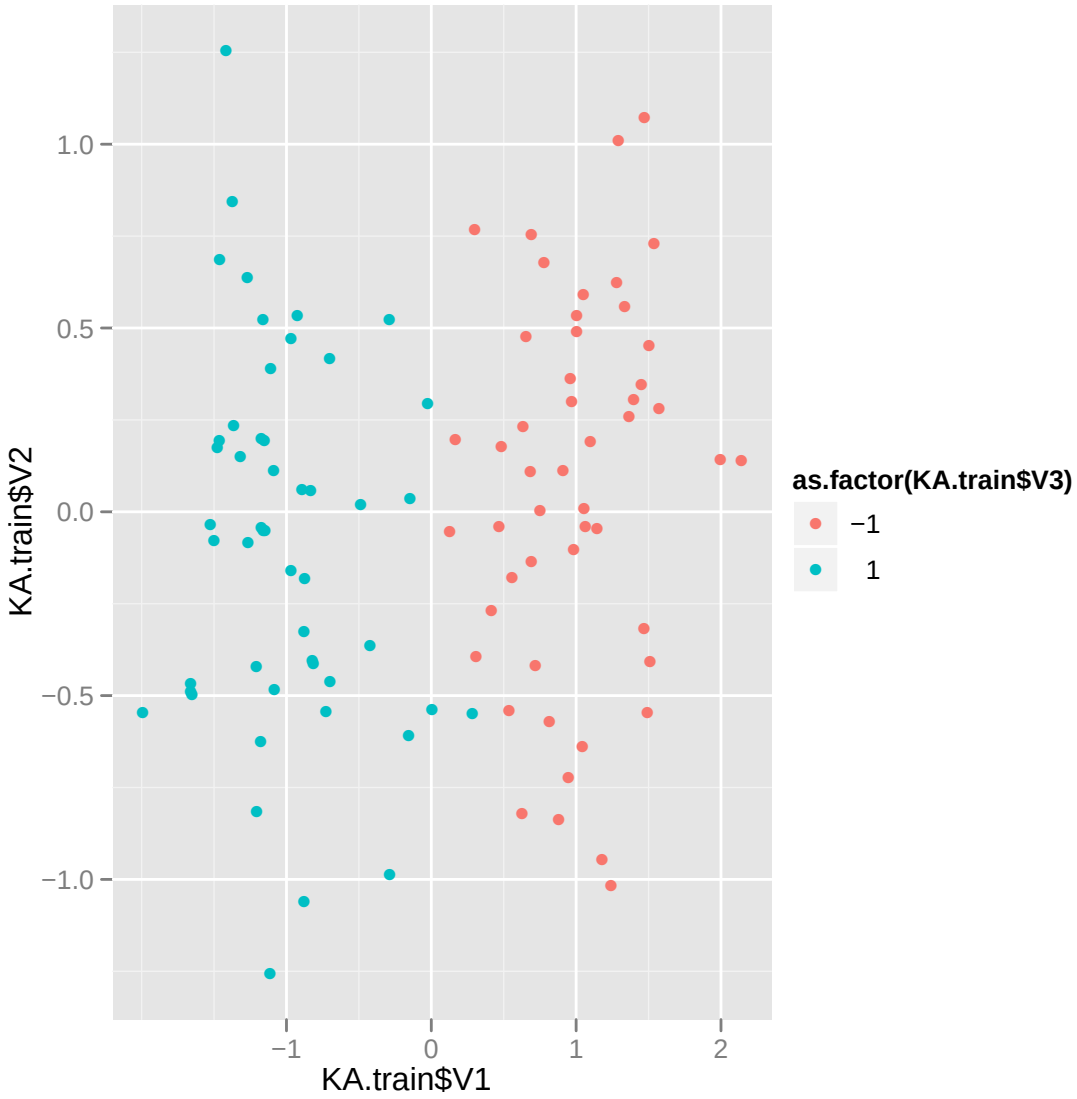
In order to evaluat how different values of α influence our prediction, we vary α from 0.1 to 10. We keep $\beta = 1.$ constant

```
> Sn<-function(alpha,beta=1,Phi){
+   tmp<-beta*t(Phi)%*%Phi
+   return(alpha*diag(ncol(tmp))+tmp)
+ }
> Mn<-function(beta=1,Sn,Phi,t){
+   beta*Sn%*%t(Phi)%*%t
+ }
> Sn3<-Sn(0.5,Phi=design1Train)
> Mn3<-Mn(Sn=Sn3,Phi=design1Train,t=targetTrain)
> yMAP<-t(Mn3)%*%t(design1Test)
> RMS.MAP<-function(targetTest=targetTest,yMAP=yMAP){sqrt(sum((targetTest-yMAP)^2)/length(targetTe
```

2 Linear Discriminant Analysis

We load the data KnollA, KnollB, and KnollC, and create 2-dimensional plots, while coloring by index. KnollA and KnollC

```
> KA.test<-read.table('./Data/KnollA-test.dt')
> KA.train<-read.table('./Data/KnollA-train.dt')
> KB.test<-read.table('./Data/KnollB-test.dt')
> KB.train<-read.table('./Data/KnollB-train.dt')
> KC.test<-read.table('./Data/KnollC-test.dt')
> KC.train<-read.table('./Data/KnollC-train.dt')
> require(ggplot2)
> par(mfrow=c(1,3))
> print(qplot(KA.train$V1,KA.train$V2,color=as.factor(KA.train$V3)))
> print(qplot(KB.train$V1,KB.train$V2,color=as.factor(KB.train$V3)))
> print(qplot(KC.train$V1,KC.train$V2,color=as.factor(KC.train$V3)))
> #KA.train and KC.train are very similar;
> # KA.train$V1-KC.train$V1*100 = 0
> # KA.train$V2-KC.train$V2 = 0
```


1.0
0.5
0.0
−0.5
−1.0
KA.train$V2
−1
0
1
2
KA.train$V1
as.factor(KA.train$V3)
−1
1