Tim Paine
tkp2108
Stat ML Hw2

Problem 0.
1. We calculate the sign of the dot product minus the scaling factor for each point.
sign(<(-3,0),(1/√2,1/√2)> - 1/(2√2) * ||(-3,0)||^2) = -1 —> X1 in class (-1)
sign(<(1/2,1/2),(1/√2,1/√2)> - 1/(2√2) * ||(1/2,1/2)||^2) = 1 —> X2 in class (1)

2. No, assuming everything is done correctly, X1 and X2 will fall in the same classes. However, the hyperplane between the two points may be different from that of the perceptron, as the perceptron finds any hyperplane between the two, whereas the SVM finds a hyperplane with a given (max) margin between the convex sets. In this particular case, the hyperplane computed by the SVM will be further away from X2 (and closer to X1, as it should be equidistant from both X1 and X2), and at a steeper slope. However, X1 and X2 will fall in the same classes (-1 and 1 respectively).

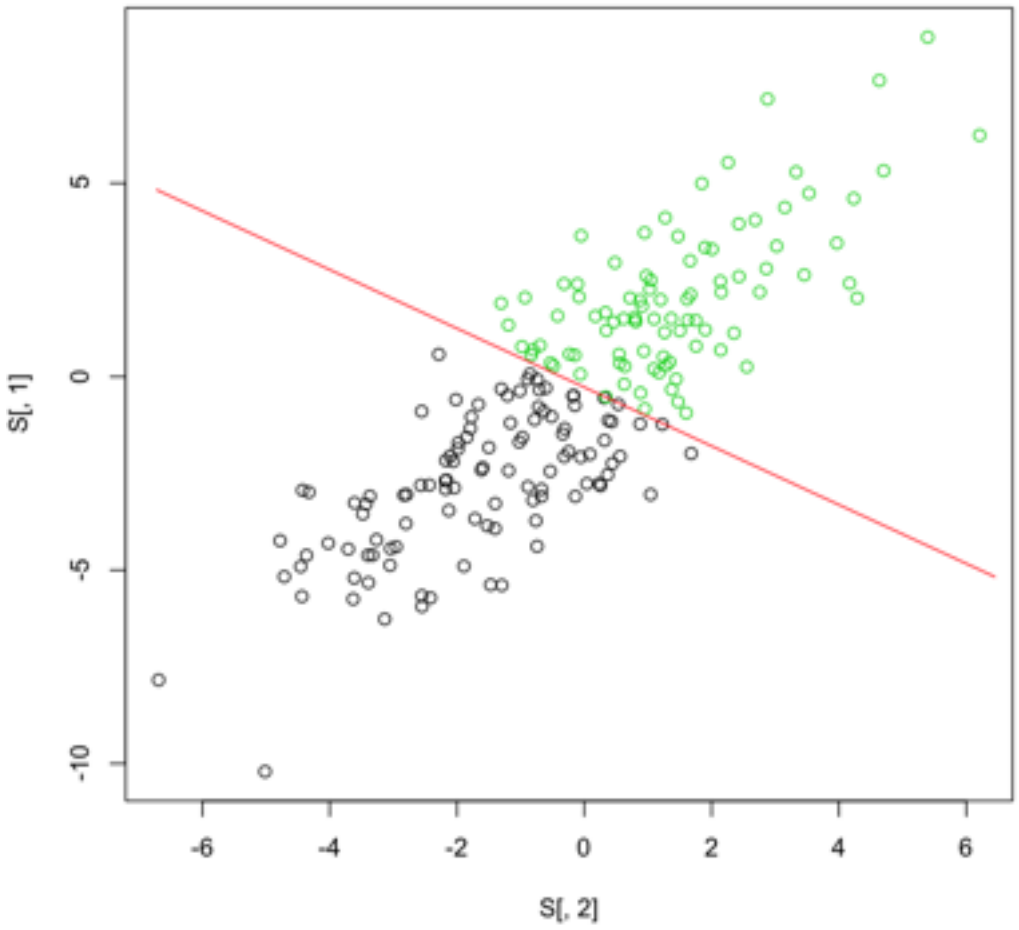
3. The cost function approximates the 0-1 Loss function (i.e. **step function**). However, since this is only piecewise constant, we cannot use it for gradient descent, and must instead approximate the step function by a **piecewise linear function** which we can then performa gradient descent on.

Problem 1.
1. (Code and full log attached)

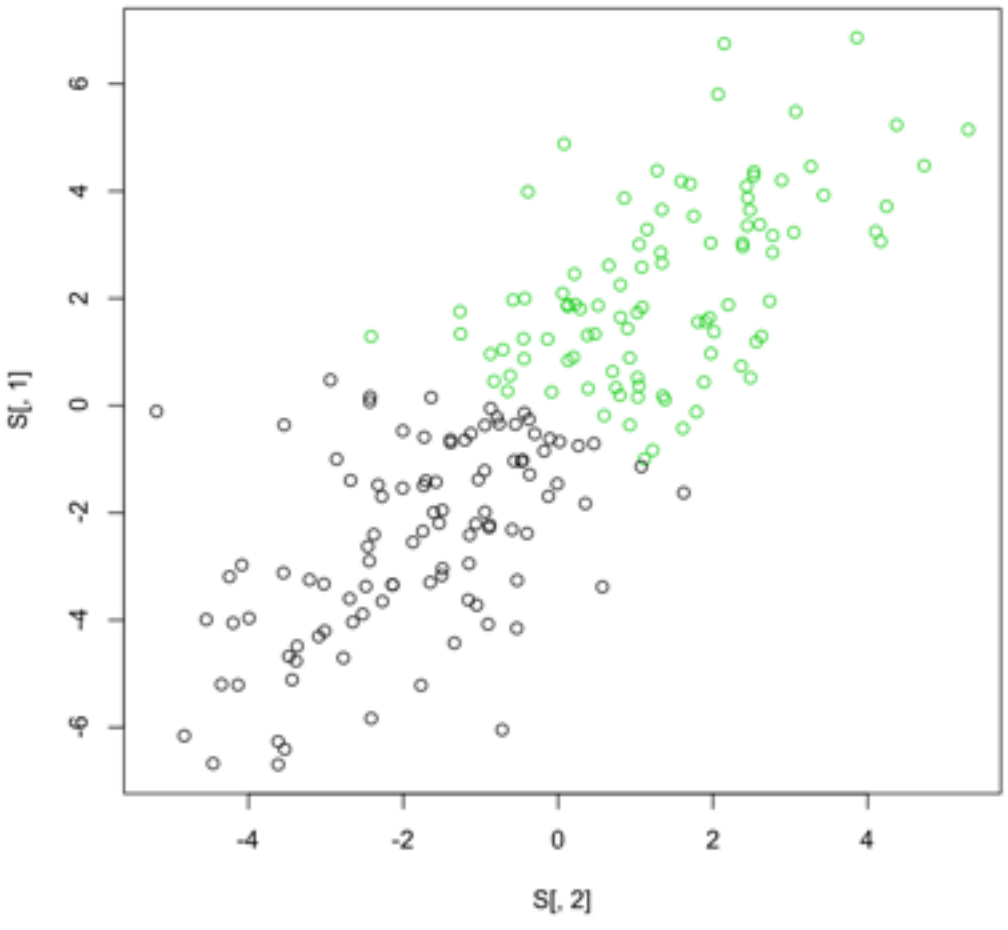
```
> s <- c(3,2,1)
> S <- fakedata(s, 200)
> y <- classify(S,s)
>
>
>
> S[2]
  [1]  1  1 -1 -1 -1  1 -1 -1  1 -1 -1  1 -1 -1 -1 -1 -1  1  1  1  1
 -1  1 -1 -1 -1  1  1  1  1 -1 -1  1 -1 -1  1  1 -1  1 -1 -1 -1  1  1
  1 -1 -1 -1  1 -1 -1 -1 -1 -1  1 -1 -1
 [58]  1 -1 -1 -1  1 -1  1 -1  1  1 -1 -1  1 -1  1  1  1  1 -1  1 -1
 -1  1 -1 -1 -1  1  1 -1  1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1 -1
 -1  1  1  1 -1  1 -1  1 -1  1  1  1  1
[115]  1  1  1  1  1 -1  1  1  1 -1 -1  1  1 -1  1 -1  1  1 -1 -1  1
 -1  1  1 -1 -1 -1 -1  1  1 -1  1 -1  1 -1 -1 -1 -1  1  1 -1 -1 -1 -1
 -1  1 -1 -1  1  1 -1 -1 -1  1 -1  1 -1
[172]  1 -1 -1 -1 -1 -1 -1 -1  1 -1  1  1 -1 -1  1  1 -1 -1  1 -1 -1
  1 -1  1 -1  1 -1 -1  1
>
>
>
>
>
> y
  [1]  1  1 -1 -1 -1  1 -1 -1  1 -1 -1  1 -1 -1 -1 -1 -1  1  1  1
  1 -1  1 -1 -1 -1  1  1  1  1 -1 -1  1 -1 -1  1  1 -1  1 -1 -1 -1
  1  1  1 -1 -1 -1  1 -1 -1 -1 -1 -1  1 -1 -1
 [58]  1 -1 -1 -1  1 -1  1 -1  1  1 -1 -1  1 -1  1  1  1  1 -1  1
 -1 -1  1 -1 -1 -1  1  1 -1  1  1 -1  1 -1  1  1 -1  1 -1  1 -1  1
 -1 -1 -1  1  1  1 -1  1 -1  1  1  1  1
[115]  1  1  1  1  1 -1  1  1  1 -1 -1  1  1 -1  1 -1  1  1 -1 -1
  1 -1  1  1 -1 -1 -1 -1  1  1 -1  1 -1  1 -1 -1 -1 -1  1  1 -1 -1
 -1 -1 -1  1 -1 -1  1  1 -1 -1 -1  1 -1  1 -1
[172]  1 -1 -1 -1 -1 -1 -1 -1  1 -1  1  1 -1 -1  1  1 -1 -1  1 -1
 -1  1 -1  1 -1  1 -1 -1  1
```

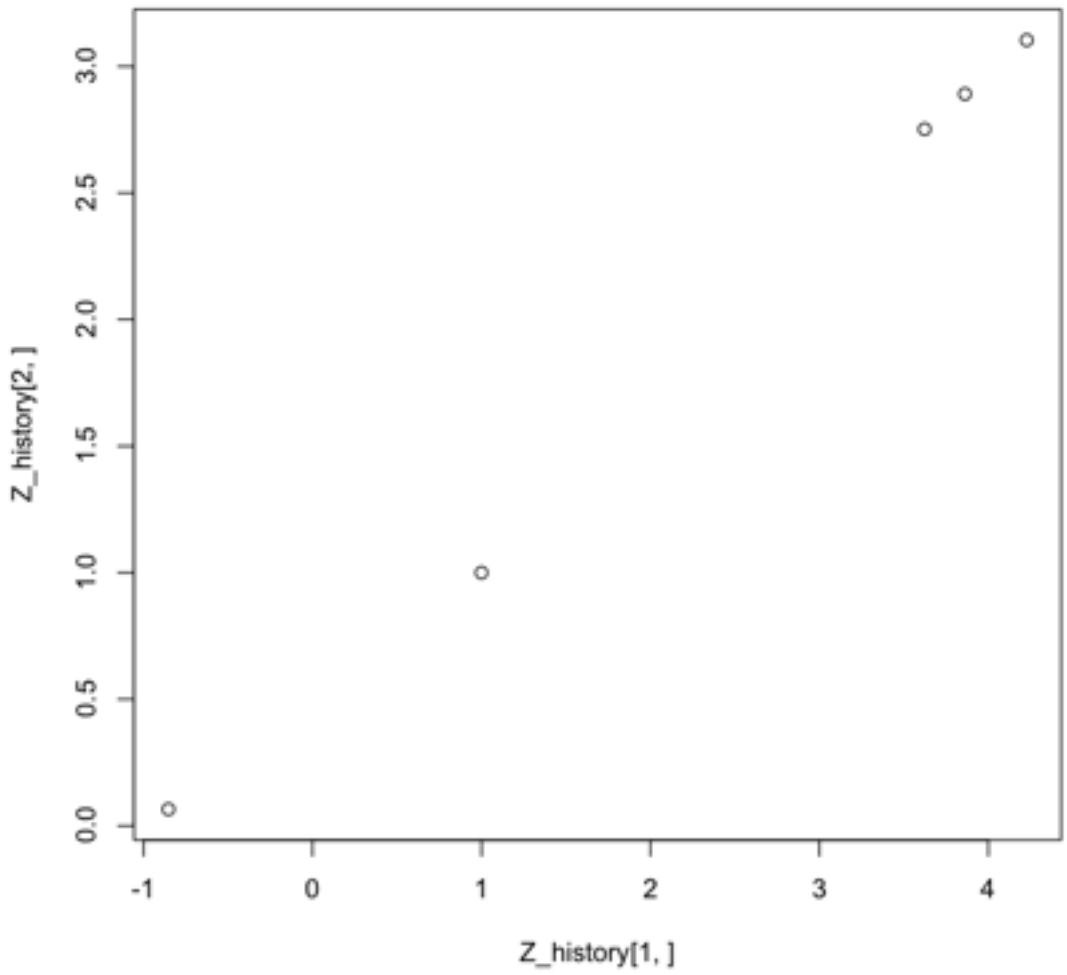
2. (code attached)
3. (code attached)
4. (code attached)

Plots
Classified test data



Train data





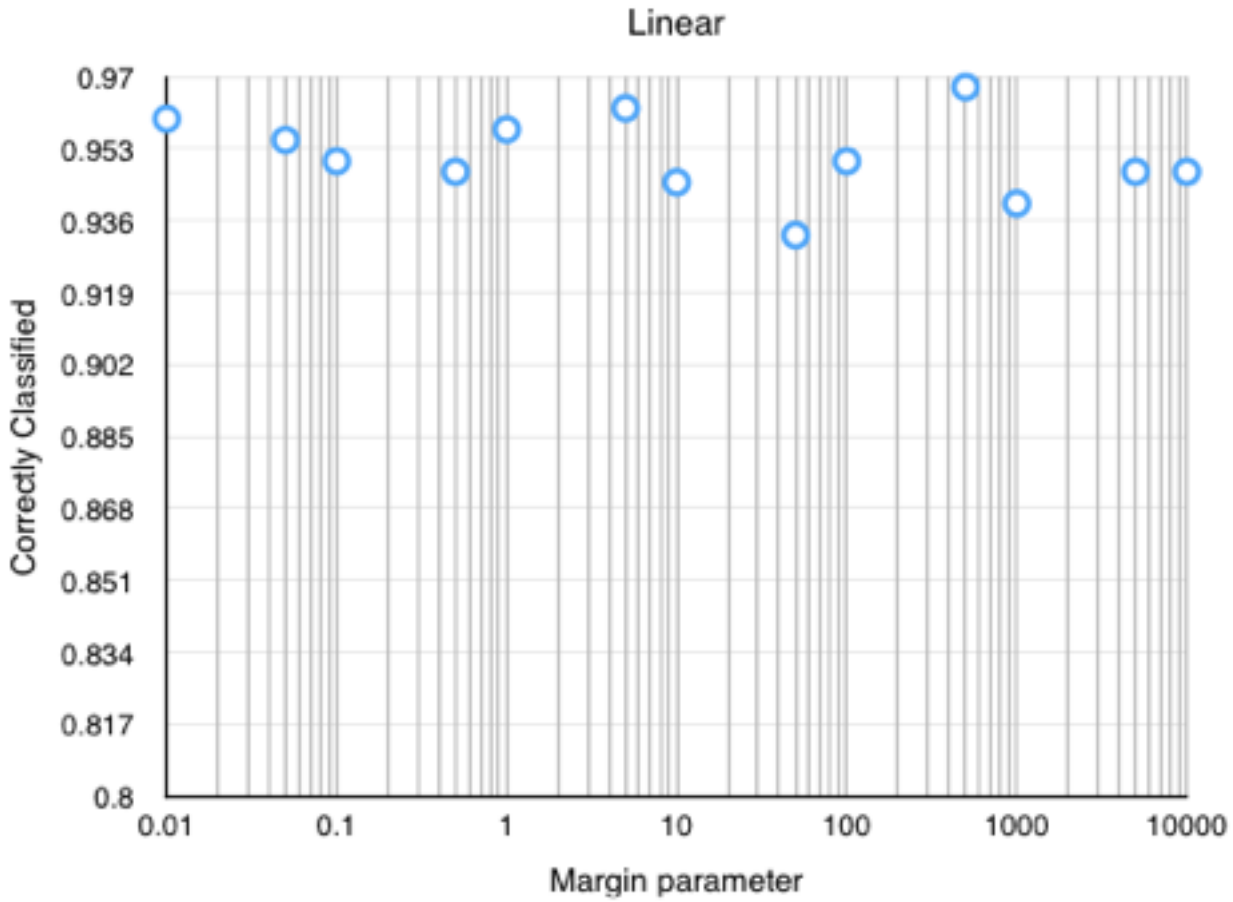
Z_history trajectory —> starting point (1,1),

Problem 2.

1. I constructed my solution in the following way (data generation code attached):

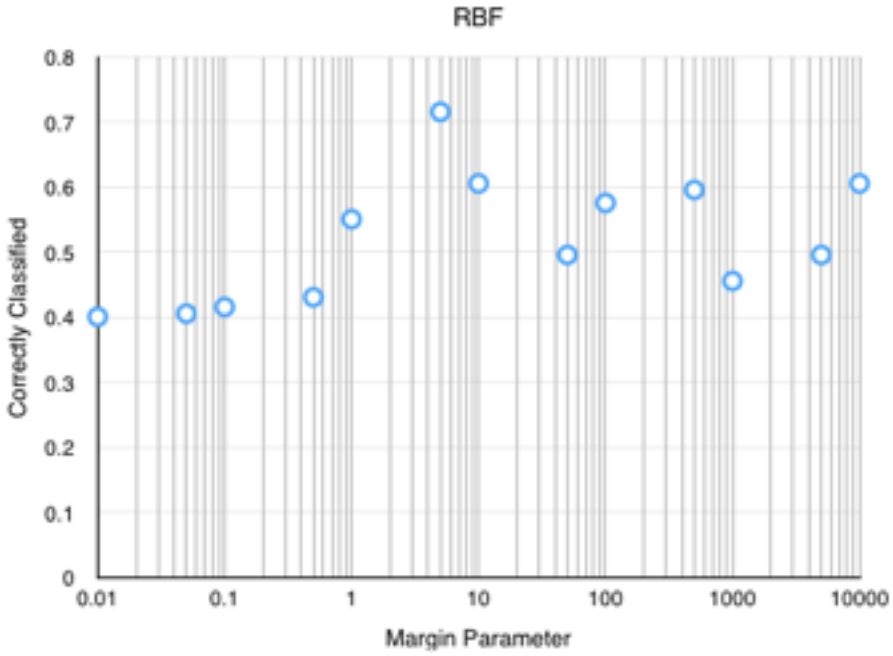
a. Margin parameter was varied between 0.01 and 10000. 20% of the data was randomly removed as test data each iteration, and 10 iterations were averaged for each point, with 10 fold cross validation. Data was entered into excel and graphed.

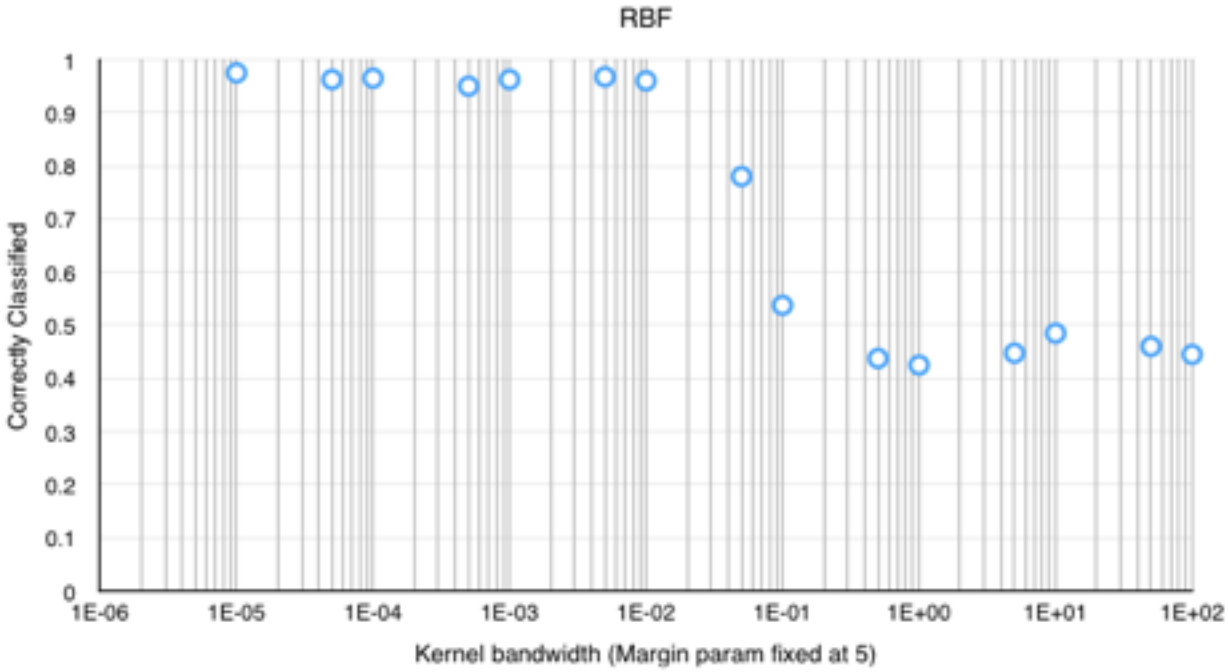
b. Margin parameter was kept constant at 5 while bandwidth was varied between 0.00001 and 100. Then bandwidth was kept constant at and margin varied between 0.01 and 10000. 20% of the data was randomly removed as test data each iteration, and 10 iterations were averaged for each point, with 10 fold cross validation.



a. We see an expected amount of noise, but in general the linear case classifies well, with misclassification rates between 3.5-7%. In general, since the data seems to be linearly separable, varying the margin parameter does little to change the overall accuracy of the model.

b. Picking a suboptimal bandwidth, we are able to see some good variation in the accuracy of the model based on the cost.





Based on the choice of the bandwidth, we either see terrible performance (misclassification +50%), or very good classification (<3% misclassification)

2. Comparing the two models, we see that the linear case gives us good average performance (3.5-7% error) for little work tuning the margin parameter (in this case, best performance was at cost=500, misclassification averaged to 3.25%). However, if we work out finding a good margin parameter and bandwidth, the RBF gave even better average performance

(bandwidth=0.00001, cost = 5, misclassification averaged to 2.5%). Because of this, I think a well tuned non-linear makes a better choice.
Code:

fakedata.R

```
#Inputs
#w:  w[1:d] is the normal vector of a hyperplane,
#    w[d+1] = -c is the negative offset parameter.
#n: sample size

#Outputs
#S: n by (d+1) sample matrix with last col 1
#y: vector of the associated class labels

fakedata <- function(w, n){

if(! require(MASS))
{
        install.packages("MASS")
}
if(! require(mvtnorm))
{
        install.packages("mvtnorm")
}

require(MASS)
require(mvtnorm)

# obtain dimension
d <- length(w)-1

# compute the offset vector and a Basis consisting of w and its nullspace
offset <- -w[length(w)] * w[1:d] / sum(w[1:d]^2)
Basis <- cbind(w[1:d],Null(w[1:d]))


# Create samples, correct for offset, and extend
# rmvnorm(n,mean,sigme) ~ generate n samples from N(0,I) distribution
S <- rmvnorm(n, mean=rep(0,d),sigma = diag(1,d)) %*%  t(Basis)
S <- S + matrix(rep(offset,n),n,d,byrow=T)
S <- cbind(S,1)

# compute the class assignments
w<- t(as.matrix(w))

y <- as.vector(sign( (S[,-3]  %*%  w[,-3]) + w[,3] ))


# add corrective factors to points that lie on the hyperplane.
S[y==0,1:d] <- S[y==0,1:d] + runif(1,-0.5,0.5)*10^(-4)

y = as.vector(sign( (S[,-3]  %*%  w[,-3]) + w[,3] ))
return(list(S=S, y=y))

} # end function fakedata
```

classify.R

```
#inputs
#S: set of vectors to be classified
#z: z[1:d] is vector to defining hyperplane

#Outputs
#y: class label

classify <- function(S,z){

SS <- do.call(rbind, S[1])

z <- t(as.matrix(z))

y = as.vector(sign((  SS[,-3]  %*%  z[,-3]  ) + z[,3] ))
return(y=y)
}
```

perceptrain.R

```
#inputs
#S: set of vectors to be trained around
#y: class labels

#Outputs
#z: hyperplane defining classifier
#Z_history: history of transformation of Z over iterations

perceptrain <- function(S){
   y <- do.call(rbind, S[2])
   S <- do.call(rbind, S[1])

   num <- nrow(S)
   dim <- ncol(S)

   #current hyperplane
   Z <- as.matrix(rep(1, dim), ncol=1)

   #history list
   Z_history <- list()
   Z_history <- cbind(Z_history,Z)


   #iteration
   k <- 1


   while(abs(cost(S,y,Z,dim,num)) != 0){
      #print(cost(S,y,Z,dim,num))
      a <- 1/k
      c <- grad(S,y,Z,dim,num)
```

```
        #get new z
        Z <- Z - t(a*c)

        #add Z to Z_history
        Z_history <- cbind(Z_history,Z)

        k<-k+1
    }
    return(list(Z=Z, Z_history=Z_history))
}

cost <- function(x, y, z, dim, num){
    cc <- 0

    for(i in 1:num){
        #get ith row of x
        #x(,i)
        xx <- t(x[i,])

        #xx <- t(rbind(1,xx))
        #f(x) = sign((1,x), z)
        #fx <- sign(t(z) %*% xx)

        #print( (xx[,-3] %*% z[-3,]) - z[3,])


        fx <- sign( (xx[,-3] %*% z[-3,]) + z[3,] )

        dot <- (xx[,-3] %*% z[-3,]) + z[3,]

        id <- 1*(fx != y[i])
        cc <- cc + id*dot
    }
    return (cc=cc)
}

grad <- function(x, y, z, dim, num){
    g <- t(as.matrix(rep(0, dim)))
   #sum indic*(-y)(1,x)
    for(i in 1:num){

        xx <- t(as.matrix(x[i,]))

        fx <- as.integer(sign( (xx[,-3] %*% z[-3,]) + z[3,] ))

        id <- as.integer(1*(fx != y[i]))

        negy <- -1*y[i]

        g <- g + id*negy*xx
    }

    return (g=g)
}
```

plotpercep.R

```
#inputs
#S: set of vectors
#Z: normal vector
#Z_history: history of normal vector
#y: classes
#Outputs

plotpercep <- function(S,Z){
    y <- do.call(rbind, S[2])
    S <- do.call(rbind, S[1])
    S <- S[,-3]
    S <- cbind(S,t(y))

    plot(S[,2], S[,1], col = as.integer(S[,3]+2))

    #normalize and plot Z
    Z_history <- do.call(rbind, Z[2])
    Z <- do.call(rbind, Z[1])
    Z_history <- matrix(unlist(Z_history), nr=nrow(Z) )

    Zx <- Z[1,]
    Zy <- Z[2,]
    Zc <- -1*Z[3,] *(sqrt(Zx^2 + Zy^2))

    Zr <- sqrt(Zx^2 + Zy^2)
    Zr_normalized <- Zr/Zc

    Zx_normalized <- (Zr_normalized)*(Zx/Zr)
    Zy_normalized <-  (Zr_normalized)*(Zy/Zr)

    Z_final <- c(Zx_normalized,Zy_normalized)


    normal_slope <- -Zx/Zy
    Z_other <- c(Zx_normalized + 5, Zy_normalized + 5*normal_slope )
    Z_other2 <- c(Zx_normalized - 5, Zy_normalized - 5*normal_slope )

    #plot(Zx_normalized, Zy_normalized)
    p <- cbind(Z_final, Z_other, Z_other2)
    lines(p[2,], p[1,], col = "red")


    #plot history of Z
    Z_history <-cbind(Z_history)

    for(i in 1:ncol(Z_history)){
        Zz <- t(t(Z_history[,i]))
        Zx <- Zz[1,]
        Zy <- Zz[2,]
        Zc <- Zz[3,]

        Zr <- sqrt(Zx^2 + Zy^2)
        Zr_normalized <- Zr/Zc

        Zx_normalized <- (Zr_normalized)*(Zx/Zr)
        Zy_normalized <-  (Zr_normalized)*(Zy/Zr)

        Z_final <- c(Zx_normalized,Zy_normalized, -1)
        Z_history[,i] <- Z_final
```

```
    }
    plot(Z_history[1,], Z_history[2,])
}
```

Log of a trial run

```
> source("/Volumes/Macintosh HD/Users/theocean154/Documents/School_files/College/Semester 8/STAT 4400/HW/stat4400/hw2/part1/classify.R")
> source("/Volumes/Macintosh HD/Users/theocean154/Documents/School_files/College/Semester 8/STAT 4400/HW/stat4400/hw2/part1/fakedata.R")
> source("/Volumes/Macintosh HD/Users/theocean154/Documents/School_files/College/Semester 8/STAT 4400/HW/stat4400/hw2/part1/perceptrain.R")
> source("/Volumes/Macintosh HD/Users/theocean154/Documents/School_files/College/Semester 8/STAT 4400/HW/stat4400/hw2/part1/plotpercep.R")
> s <- c(3,2,1)
> S <- fakedata(s, 200)
> y <- classify(S,s)
>
>
>
> S
$S
             [,1]        [,2] [,3]
  [1,]  0.34996241  0.55792272    1
  [2,]  0.27020363  0.63639241    1
  [3,] -4.30372167 -4.02196859    1
  [4,] -2.80608409 -2.56829678    1
  [5,] -2.06797199 -0.31392660    1
  [6,]  2.39446759 -0.10728576    1
  [7,] -3.72217522 -0.76131576    1
  [8,] -0.78115738 -0.71997213    1
  [9,]  0.55825120 -0.83338372    1
 [10,] -0.34201888 -0.70367855    1
 [11,] -5.67977887 -4.43612848    1
 [12,]  3.64200153 -0.05013296    1
 [13,] -5.39110129 -1.29622338    1
 [14,] -2.90331570 -2.17801979    1
 [15,] -2.81517808  0.25567151    1
 [16,] -1.10999089 -0.78140682    1
 [17,] -4.87337366 -3.05169481    1
 [18,]  0.68758700  2.14203727    1
 [19,]  2.79173419  2.85865447    1
 [20,]  1.56211860  0.17405049    1
 [21,]  1.82512922  0.90929117    1
 [22,] -1.16291204  0.42973973    1
 [23,]  0.35416100 -0.53009713    1
 [24,] -3.67151051 -1.71757579    1
 [25,] -0.89554030 -2.55523215    1
 [26,] -0.49972748 -0.17824257    1
 [27,]  2.00366736  1.61667984    1
 [28,]  7.17621295  2.87853568    1
 [29,]  8.77310597  5.39422348    1
 [30,]  1.14514559  1.26023020    1
 [31,] -2.67212068 -2.16551911    1
 [32,] -2.93437353 -4.42617172    1
 [33,]  0.25409067  2.55135893    1
 [34,]  0.06999895 -0.85144428    1
 [35,] -3.09258035 -0.67425033    1
 [36,]  1.89697484 -1.30761622    1
 [37,]  7.65603930  4.63494489    1
 [38,] -2.79975905 -2.43439356    1
 [39,]  2.60452455  0.97384503    1
 [40,] -1.47927188 -0.33830523    1
 [41,] -4.44151662 -3.04192704    1
 [42,] -1.03899964 -1.77522261    1
```

```
 [43,]   1.98810585  1.19978417    1
 [44,]   0.78813718  1.75501689    1
 [45,]   0.51171498  1.24237715    1
 [46,]  -0.36705435 -1.01198514    1
 [47,]  -1.56752052 -1.84714725    1
 [48,] -10.20492403 -5.01560974    1
 [49,]  -0.33055608  1.37215913    1
 [50,]  -4.39568532 -2.95604000    1
 [51,]  -2.41309068 -1.61559944    1
 [52,]  -0.29413722 -0.60417387    1
 [53,]  -4.61383910 -3.32629150    1
 [54,]  -4.61802745 -4.36454242    1
 [55,]   2.18339593  2.15068325    1
 [56,]  -1.64642908  0.31750594    1
 [57,]  -3.08733302 -3.36888062    1
 [58,]   0.19866247  1.09055692    1
 [59,]  -2.07831948 -0.05660766    1
 [60,]  -3.26178331 -3.60948916    1
 [61,]  -5.74766654 -3.63095079    1
 [62,]   1.41352132  0.46136990    1
 [63,]  -1.86994618 -1.98615835    1
 [64,]   3.37411536  3.02116285    1
 [65,]  -1.70740353 -1.97832796    1
 [66,]   0.66204003  0.93535205    1
 [67,]   2.18678299  2.75594876    1
 [68,]  -3.05935839 -2.83472646    1
 [69,]  -0.59403954 -2.01745214    1
 [70,]   2.58735059  2.42892180    1
 [71,]  -0.54700212  0.31234790    1
 [72,]   2.12326768  1.66879518    1
 [73,]   4.73835121  3.53001244    1
 [74,]  -0.42202792  0.88252457    1
 [75,]   4.10723355  1.27170355    1
 [76,]  -1.33661935 -1.78983800    1
 [77,]   5.28897222  3.32732700    1
 [78,]  -0.07428572 -0.73878873    1
 [79,]  -5.71592708 -2.41751612    1
 [80,]  -0.19399068  0.62677501    1
 [81,]  -6.26372801 -3.13538303    1
 [82,]  -0.48638396 -0.16901537    1
 [83,]  -3.09338429 -0.14105611    1
 [84,]   2.94678815  0.48039183    1
 [85,]   0.69069128 -0.80347851    1
 [86,]  -4.89621944 -1.89141635    1
 [87,]   0.27749520 -0.48464247    1
 [88,]   2.06142388 -0.08266562    1
 [89,]  -1.34240700 -0.30956634    1
 [90,]   0.57661977  0.54708753    1
 [91,]  -1.69042230 -1.02192999    1
 [92,]   1.57144315 -0.41991960    1
 [93,]   3.62756785  1.47234762    1
 [94,]  -1.02991877 -0.51579070    1
 [95,]   2.62971920  3.45533132    1
 [96,]  -3.85346280 -1.52412772    1
 [97,]   1.19313239  0.34609872    1
 [98,]  -1.83108799 -1.49913576    1
 [99,]   0.55554509 -0.14802104    1
[100,]  -3.04525298  1.04168685    1
[101,]  -4.90615620 -4.45595992    1
[102,]  -3.28020189 -1.39997909    1
[103,]  -0.52145077  0.34049536    1
[104,]   1.12708361  2.35122624    1
[105,]   2.26276478  1.02996412    1
[106,]  -1.56998068 -0.97352916    1
```

```
[107,]  3.33973057  1.89547098    1
[108,] -2.68995087 -2.18318511    1
[109,]  2.03431829  0.71821587    1
[110,] -3.92620221 -1.40240587    1
[111,]  2.99568145  1.66204190    1
[112,]  2.46521890  2.13660020    1
[113,]  0.80821009 -0.69609175    1
[114,]  5.32316179  4.70129424    1
[115,]  0.58302051 -0.23203382    1
[116,]  1.49718541  0.61781024    1
[117,]  2.39888457 -0.31910955    1
[118,]  1.53112304  0.79293138    1
[119,] -0.06403280  1.44282851    1
[120,] -2.84327956 -0.88642015    1
[121,]  4.37247124  3.14955757    1
[122,]  3.29545072  2.00581464    1
[123,]  0.06367855 -0.06194153    1
[124,] -5.32491649 -3.39146736    1
[125,] -4.23859427 -4.77908846    1
[126,]  0.10135971  1.17381129    1
[127,]  4.60490333  4.23192893    1
[128,] -2.00539959  0.08769905    1
[129,]  1.97494945  0.87676641    1
[130,] -4.45665753 -3.70947104    1
[131,]  0.28326827  1.28227590    1
[132,]  1.42886805  0.80918415    1
[133,] -2.87285171 -2.04265758    1
[134,] -5.37872627 -1.47187864    1
[135,]  2.04503885 -0.93209059    1
[136,]  0.57617571 -2.28595883    1
[137,]  3.95033399  2.42709693    1
[138,]  0.77642301 -0.97857347    1
[139,] -2.76494232  0.23363770    1
[140,] -2.24307356  0.44111542    1
[141,] -5.20231076 -3.61375203    1
[142,] -3.54858698 -3.47999972    1
[143,]  1.50038668  1.36561219    1
[144,]  5.53397688  2.25905721    1
[145,] -0.31729232 -1.30393191    1
[146,]  2.03191206  4.28646295    1
[147,] -5.16796052 -4.71356141    1
[148,]  1.46230505  1.62642438    1
[149,] -1.19725916 -1.16012373    1
[150,] -0.89651298 -0.64230432    1
[151,] -3.19342356 -0.80634945    1
[152,] -1.93606184 -0.24274975    1
[153,] -0.93722104  1.59966919    1
[154,]  1.18717551  1.50019163    1
[155,] -2.98664768 -4.32621562    1
[156,] -3.29082995 -3.42833263    1
[157,] -0.48902345 -1.20613351    1
[158,] -4.61659864 -3.39896658    1
[159,] -2.42837402 -1.18929111    1
[160,] -0.64958302  1.47287469    1
[161,] -4.21545185 -3.26442345    1
[162,] -2.53228663  0.36662781    1
[163,]  2.49102524  1.04369928    1
[164,]  1.45561725  1.75171493    1
[165,] -0.06823481 -0.88998432    1
[166,] -3.04494907 -2.79076763    1
[167,] -5.93755045 -2.55019348    1
[168,]  1.44139949  0.80099881    1
[169,] -2.18915765 -2.05947858    1
[170,]  3.71959959  0.94651680    1
```

```
[171,]  -0.74416290 -0.14869010    1
[172,]   3.45103477  3.96743131    1
[173,]  -4.38418389 -0.74186095    1
[174,]  -2.34713798 -1.59535056    1
[175,]  -0.71632113 -1.66345091    1
[176,]  -0.71400207  0.53392914    1
[177,]  -2.16561714 -2.17466782    1
[178,]  -1.12738315  0.37506205    1
[179,]  -2.44573393 -0.53548867    1
[180,]   4.04732990  2.68412692    1
[181,]  -7.83884201 -6.68717451    1
[182,]   1.33428565 -1.19273327    1
[183,]  -0.83116322  0.94986802    1
[184,]  -2.05921781  0.56188900    1
[185,]  -1.23157550  1.22506300    1
[186,]   2.41230013  4.16688201    1
[187,]   1.49252352  1.09565782    1
[188,]  -3.45869005 -2.12865257    1
[189,]  -3.79545246 -2.80207294    1
[190,]   6.23784389  6.21068912    1
[191,]  -2.04815320 -2.10461317    1
[192,]  -1.98536555  1.67869103    1
[193,]   1.65506288  0.33361305    1
[194,]  -1.22102064  0.87623548    1
[195,]   4.99318794  1.84582576    1
[196,]  -2.76562144  0.04089825    1
[197,]   1.21004161  1.90095926    1
[198,]  -2.91190036 -0.67123294    1
[199,]  -5.66085407 -2.55222334    1
[200,]   0.37686696  1.33953644    1

$y
  [1]  1  1 -1 -1 -1  1 -1 -1  1 -1 -1  1 -1 -1 -1 -1 -1  1  1  1  1 -1  1 -1 -1 -1  1  1  1 -1 -1  1 -1 -1 -1  1  1  1 -1 -1 -1  1
 -1 -1 -1 -1 -1  1 -1 -1
 [58]  1 -1 -1 -1  1 -1  1 -1  1  1 -1 -1  1 -1  1  1  1  1 -1  1 -1 -1  1 -1 -1 -1  1  1 -1  1 -1  1 -1  1 -1  1 -1 -1 -1  1 -1  1 -1
 1 -1  1 -1  1  1  1  1
[115]  1  1  1  1  1 -1  1  1  1 -1 -1  1  1 -1  1 -1  1  1 -1 -1  1 -1  1  1 -1 -1 -1 -1  1  1 -1  1 -1  1 -1 -1 -1  1  1 -1 -1 -1 -1 -1  1 -1 -1  1
 1 -1 -1 -1  1 -1  1 -1
[172]  1 -1 -1 -1 -1 -1 -1 -1  1 -1  1  1 -1 -1  1  1 -1 -1  1 -1 -1  1 -1  1 -1  1 -1 -1  1
>
>
>
>
> y
  [1]  1  1 -1 -1 -1  1 -1 -1  1 -1 -1  1 -1 -1 -1 -1 -1  1  1  1  1 -1  1 -1 -1 -1  1  1  1 -1 -1  1 -1 -1 -1  1  1  1 -1 -1 -1  1
 -1 -1 -1 -1 -1  1 -1 -1
 [58]  1 -1 -1 -1  1 -1  1 -1  1  1 -1 -1  1 -1  1  1  1  1 -1  1 -1 -1  1 -1 -1 -1  1  1 -1  1 -1  1 -1  1 -1  1 -1 -1 -1  1 -1  1 -1
 1 -1  1 -1  1  1  1  1
[115]  1  1  1  1  1 -1  1  1  1 -1 -1  1  1 -1  1 -1  1  1 -1 -1  1 -1  1  1 -1 -1 -1 -1  1  1 -1  1 -1  1 -1 -1 -1  1  1 -1 -1 -1 -1 -1  1 -1 -1  1
 1 -1 -1 -1  1 -1  1 -1
[172]  1 -1 -1 -1 -1 -1 -1 -1  1 -1  1  1 -1 -1  1  1 -1 -1  1 -1 -1  1 -1  1 -1  1 -1 -1  1
>
>
>
>
> Z <- perceptrain(S)
> Z
$Z
         [,1]
[1,] 16.602286
[2,] 12.613575
[3,]  4.583333

$Z_history
```

```
     [,1] [,2]       [,3]     [,4]     [,5]
[1,] 1    11.08619   16.90647 16.73265 16.60229
[2,] 1    -0.8507291 12.41495 12.52845 12.61357
[3,] 1    -13        4        4.333333 4.583333

>
```

runsvm.R

```
runsvm <- function(data,class){
   if(! require(e1071))
   {
      install.packages("e1071")
   }

   library(e1071)
   library(rpart)


   d <- read.table(data, header = FALSE, sep = "", skip = 0)
   c <- read.table(class, header = FALSE, skip = 0)

   d <- as.matrix(d)
   class <- as.matrix(c)
   colnames(class) <- "class"
   d <- cbind(d,class)
   d <- data.frame(d)

   #f <- as.formula(paste(tail(names(d), 1), "~ ."))

   index <- 1:nrow(d)
   testindex <- sample(index, trunc(length(index)/5))
   testset <- d[testindex,]
   trainset <- d[-testindex,]

   x <- as.matrix(trainset[,-ncol(d)])
   y <- as.matrix(trainset[,ncol(d)])

   #gamma is margin parameter
   #cost is kernel bandwidth

   #linear
   #svm.model <- svm(class ~ ., data=trainset, cost=500, kernel = "linear", type="C-classification", cross=10)
   #rbf
   svm.model <- svm(class ~ ., data=trainset, cost=10000, gamma = .1, type="C-classification", cross=10)

   #model <- svm(data=x, x, y, cost=70, gamma = 0.003, cross=5)
   #print(summary(svm.model))

   svm.pred <- predict(svm.model, testset[,-ncol(d)], decision.values = TRUE)

   #print(summary(pred))

   tab <- table(pred = svm.pred, true = testset[,ncol(d)])

   #print(tab)
   print(classAgreement(tab))
}
```