1 Classifier

1.1 proof

Dot product $\vec{a}\vec{b} = a_x b_x + a_y b_y = |\vec{a}||\vec{b}|\cos(\theta)$
Exercise:
$g(x) =< C_+ - C_-, X - C >=< C_+, X > - < C_-, X > - < C_+, C > + < C_-, C >;$
$< C_+, X >=< \frac{1}{n_+} \sum_{l \in I_+}^{n} x_i, x >;$
$< C_-, X >=< \frac{1}{n_-} \sum_{l \in I_-}^{n} x_i, x >;$
$< C_+, C >=< C_+, \frac{1}{2}C_+ > + < C_+, \frac{1}{2}C_- >= \frac{1}{2n_+^2} \sum_{(i,j) \in I_+} < x_i, x_j > + \frac{1}{2} < C_+, C_- >$
$< C_-, C >=< C_-, \frac{1}{2}C_+ > + < C_-, \frac{1}{2}C_- >= \frac{1}{2} < C_+, C_- > + \frac{1}{2n_-^2} \sum_{(i,j) \in I_-} < x_i, x_j >$
$g(x) = \sum_{l=1}^{n} \alpha_i < x_i, x > + b,$
$b = \frac{1}{2}\left[\frac{1}{n_-^2} \sum_{(i,j) \in I_-} < x_i, x_j > - \frac{1}{n_+^2} \sum_{(i,j) \in I_+} < x_i, x_j >\right]$
$\alpha_i = \begin{cases} \frac{1}{n_+} & y_i = +1 \\ -\frac{1}{n_-} & y_i = -1 \end{cases}$

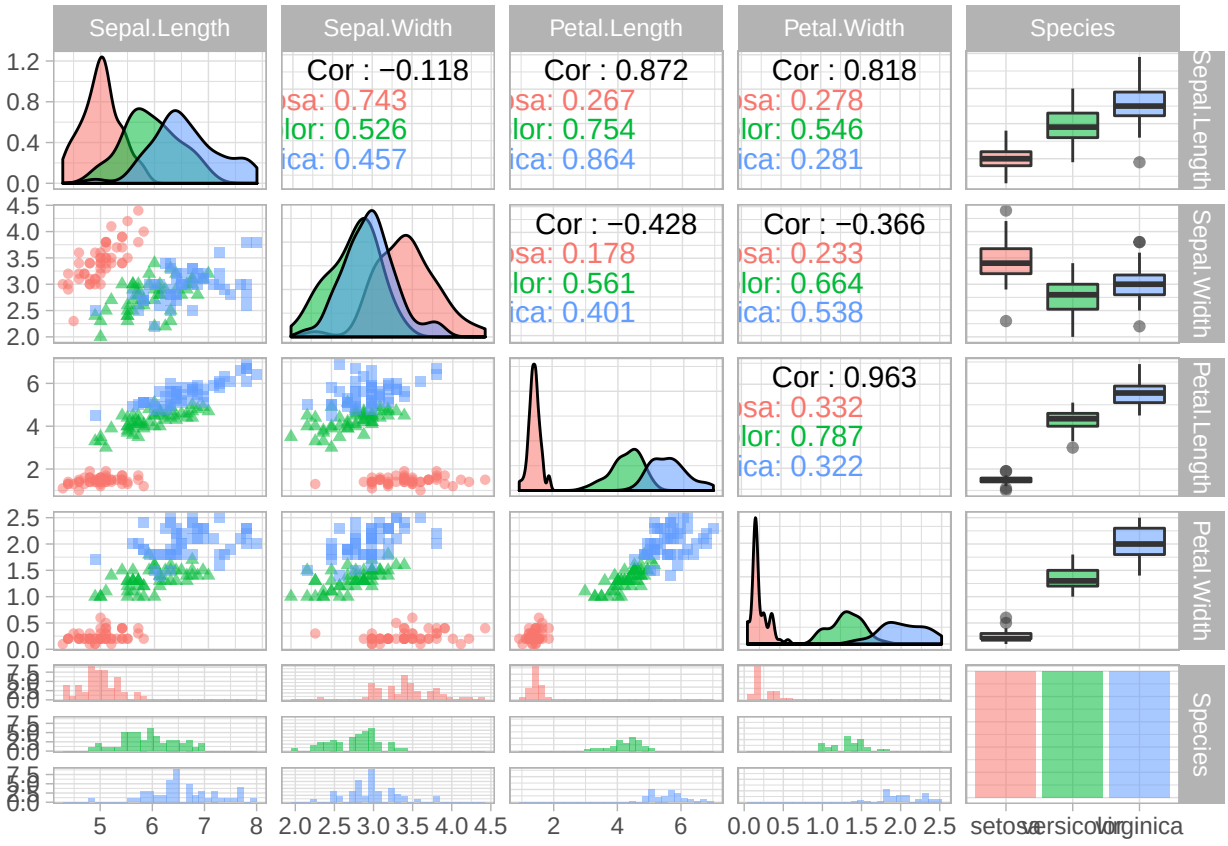
2 Classifier

2.1 iris data clasification

- Import the iris data

```
rm(list=ls())
library(datasets)
data(iris)
```

```
library(ggplot2)
GGally::ggpairs(iris,mapping=aes(color =Species,shape=Species,alpha=0.3),
        columns=c("Sepal.Length","Sepal.Width","Petal.Length","Petal.Width","Species"))+theme_light()
```



- Define the train and test sets

```
iris$class <- NA
iris_setosa <- iris[iris$Species=="setosa",]
iris_versicolor <- iris[iris$Species=="versicolor",]
iris_virginica <- iris[iris$Species=="virginica",]

iris_train_se<- iris_setosa[1:40,]
iris_train_ve<- iris_versicolor[1:40,]
iris_train_vi<- iris_virginica[1:40,]

iris_test_se<- iris_setosa[41:50,]
iris_test_ve<- iris_versicolor[41:50,]
iris_test_vi<- iris_virginica[41:50,]

iris_train_se.ve<- rbind(iris_train_se,iris_train_ve)
iris_train_ve.vi<- rbind(iris_train_ve,iris_train_vi)
iris_test_se.ve<- rbind(iris_test_se,iris_test_ve)
iris_test_ve.vi<- rbind(iris_test_ve,iris_test_vi)
```

- Define the kernel function and Computing the classifier

```
k = function(x,y)  return(sum(x*y))
k.pp=outer(1:40,1:40,Vectorize(function(i,j) k(iris_train_se[i,1:4],iris_train_se[j,1:4])))
k.mm=outer(1:40,1:40,Vectorize(function(i,j) k(iris_train_ve[i,1:4],iris_train_ve[j,1:4])))
b=(sum(k.mm)/(40^2)-sum(k.pp)/(40^2))/2
alpha=ifelse(iris_train_se.ve$Species=="setosa",1/40,-1/40)

k.x=outer(1:80,1:20,Vectorize(function(i,j) k(iris_train_se.ve[i,1:4],iris_test_se.ve[j,1:4])))
iris_test_se.ve[,6]=(t(k.x)%*%alpha+b)
```

```
k = function(x,y)  return(sum(x*y))
k.pp=outer(1:40,1:40,Vectorize(function(i,j) k(iris_train_vi[i,1:4],iris_train_vi[j,1:4])))
k.mm=outer(1:40,1:40,Vectorize(function(i,j) k(iris_train_ve[i,1:4],iris_train_ve[j,1:4])))
b=(sum(k.mm)/(40^2)-sum(k.pp)/(40^2))/2
alpha=ifelse(iris_train_ve.vi$Species=="virginica",1/40,-1/40)

k.x=outer(1:80,1:20,Vectorize(function(i,j) k(iris_train_ve.vi[i,1:4],iris_test_ve.vi[j,1:4])))
iris_test_ve.vi[,6]=(t(k.x)%*%alpha+b)
```

- Evaluate the classifier

```
iris_test_se.ve$evaluate=ifelse(iris_test_se.ve$class>0,"setosa","versicolor")
iris_test_se.ve$evaluate=ifelse(iris_test_se.ve$Species==iris_test_se.ve$evaluate,"Rigt","Wrong")
length(which(iris_test_se.ve$evaluate=="Wrong"))/20
## [1] 0

iris_test_ve.vi$evaluate=ifelse(iris_test_ve.vi$class>0,"virginica","versicolor")
iris_test_ve.vi$evaluate=ifelse(iris_test_ve.vi$Species==iris_test_ve.vi$evaluate,"Rigt","Wrong")
length(which(iris_test_ve.vi$evaluate=="Wrong"))/20
## [1] 0.05
```

Table 1: Confusion matrix

	Actural Species					
	test 1	Setosa	Versicolor	test 2	Virginica	Versicolor
Test Species	Setosa	10	0	Virginica	9	0
	Versicolor	0	10	Versicolor	1	10

Error rate = 0% and 5% in two tests respectively.

Table 2: setosa v.s.versicolor/ /virginica v.s.versicolor

Species	class	evaluate	Species1	class1	evaluate1
setosa	5.856	Rigt	versicolor	-1.575	Rigt
setosa	5.536	Rigt	versicolor	-0.7495	Rigt
setosa	6.35	Rigt	versicolor	-1.907	Rigt
setosa	4.665	Rigt	versicolor	-3.482	Rigt
setosa	4.135	Rigt	versicolor	-1.69	Rigt
setosa	5.429	Rigt	versicolor	-1.638	Rigt
setosa	5.215	Rigt	versicolor	-1.592	Rigt
setosa	5.869	Rigt	versicolor	-1.157	Rigt
setosa	5.239	Rigt	versicolor	-3.708	Rigt
setosa	5.548	Rigt	versicolor	-1.739	Rigt
versicolor	-5.097	Rigt	virginica	1.566	Rigt
versicolor	-6.206	Rigt	virginica	0.9795	Rigt
versicolor	-4.246	Rigt	virginica	-0.02223	Wrong
versicolor	-1.446	Rigt	virginica	1.968	Rigt
versicolor	-4.667	Rigt	virginica	1.795	Rigt
versicolor	-4.451	Rigt	virginica	0.968	Rigt
versicolor	-4.63	Rigt	virginica	0.119	Rigt

Species	class	evaluate	Species1	class1	evaluate1
versicolor	-5.402	Rigt	virginica	0.6535	Rigt
versicolor	-0.6631	Rigt	virginica	0.9918	Rigt
versicolor	-4.411	Rigt	virginica	0.02902	Rigt

3 Perceptron

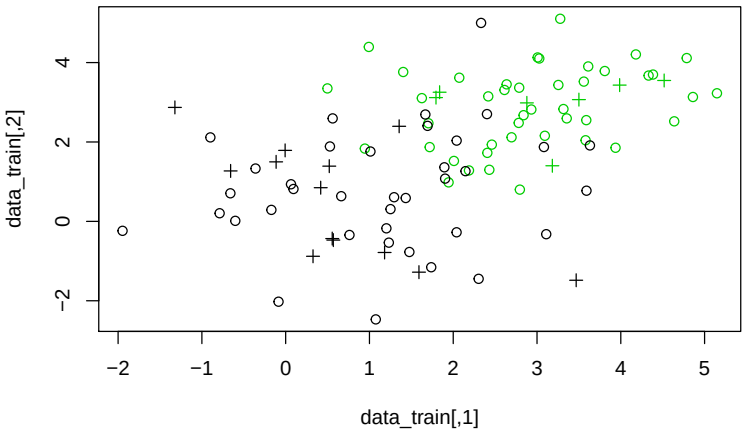
3.1 pseduo code

```
k_percetron <- function(data,T,n){
  alpha=rep(0,n)
  for (i in 1:n){
    for (j in 1:n){
      K[i,j]=alpha %*% k(dataX[i,],dataX[j,])
    }
  }
  for (t in 1:T){
    for (i in 1:n){
      ifelse (dataX$y[i]%*%alpha[t]%*%data$x[i,]<0, alpha[t]+data$y[i]%*%data$x[i,], alpha[t])
    }
  }
  return(sum(K[i,j]))
}
```

```
data <- data.frame(matrix(runif(200,min = -10, max=10),nrow = 100, ncol = 2))
data$y <- ifelse(data$X1<=data$X2,1,-1)

k_percetron <- function(data,T){
  alpha=rep(0,100)
  for (i in 1:100) {
    for (j in 1:100){
      Kij=alpha[i] %*% k(as.matrix(data[i,-3]),as.matrix(data[,-3]))
    }
  }
  for (t in 1:T){
    for (i in 1:100){
      alpha[t] =ifelse (as.matrix(data$y[i])%*%alpha[t]%*%as.matrix(data[i,-3])<0, alpha[t]+as.matrix(data$y[i])%*%as.matrix(data[i,-3]), alpha
    }
  }
}
# use kernel 1
k <- function(a,b){return(a %*% t(b))}
k_percetron(data,10)
# use kernel 2
k <- function(a,b){return((a %*% t(b))^2)}
k_percetron(data,10)
# use kernel 3
k <- function(a,b){return((a %*% t(b) +1)^2)}
k_percetron(data,10)
```

```
rm(list=ls())
## generate 2d data
n.p=50
n.m=50
n=n.p+n.m
library(mvtnorm)
data.p=rmvnorm(n=n.p,mean=c(1,1)+c(2,2),sigma=diag(rep(1,2)))
data.m=rmvnorm(n=n.m,mean=c(-1,-1)+c(2,2),sigma=diag(rep(2,2)))
class = c(rep(1,n.p),rep(-1,n.m))
data=rbind(data.p,data.m)
id_train=sort(sample(1:100,size=80))
id_test=sort(c(1:100)[-id_train])
data_train=data[id_train,]
class_train=class[id_train]
n_train=dim(data_train)[1]
data_train.m=data_train[class_train==-1,]
data_train.p=data_train[class_train==1,]
n_train.m=dim(data_train.m)[1]
n_train.p=dim(data_train.p)[1]
data_test=data[id_test,]
class_test=class[id_test]
n_test=dim(data_test)[1]
plot(data_train,col=class_train+rep(2,n_train),pch=1);points(data_test,col=class_test+rep(2,n_test),pch=3)
```




```
## compute the classifier by k1
k = function(x,y)  return(sum(x*y))
k_percetron=function(u){
  k.x=outer(1:n_train,1,Vectorize(function(i,j) k(data_train[i,],u)))
  return(sum(alpha*k.x))
}
alpha=rep(0,n_train)
alpha[1]=class_train[1]
for (i in (2:n_train)){
  if ((k_percetron(data_train[i,])*class_train[i])<0) alpha[i]=class_train[i]
}
alpha
z.hat=matrix(NA,nrow=20,ncol=1)
k.x=outer(1:80,1:20,Vectorize(function(i,j) k(data_train[i,],data_train[j,])))
for (i in (1:20)){
z.hat[i]=sum(t(k.x[,i])%*%alpha)
}
z.hat
```

```
## compute the classifier by k1
k = function(x,y)  return(sum(x*y))

k_percetron=function(n.test){

alpha=rep(0,n_train)
alpha[1]=class_train[1]

for (j in (2:n_train)){
  k.x=outer(1:n_train,1,Vectorize(function(i,j) k(data_train[i,],data_train[j,])))

  if ((sum(alpha*k.x)*class_train[j])<0) alpha[j]=class_train[j]
}

z.hat=matrix(NA,nrow=n.test,ncol=1)
k.x=outer(1:80,1:20,Vectorize(function(i,j) k(data_train[i,],data_train[j,])))
for (t in (1:n.test)){
z.hat[t]=sum(t(k.x[,t])%*%alpha)
}
return(z.hat)
}
```

3.2 Using 3 Kernels

```
## evaluate the classifier

    for (j in 1:100){
    for (i in 1:100){
      alpha[j] =ifelse (as.matrix(data$y[i])%*%alpha[j]%*%as.matrix(data[i,-3])<0, alpha[j]+as.matrix(data$y[i])%*%as.matrix(data[i,-3]), alpha
    }
  }
```

```
## compute the classifier by k1
k1_test <- k_percetron(20)
## compute the classifier by k2
k = function(x,y)
  return(sum(x*y)+1)
k2_test <- k_percetron(20)
```

```r
## compute the classifier by k3
k = function(x,y)
  return((1+sum(x*y))^2)
k3_test <- k_percetron(20)

pander(cbind(k1_test,k2_test,k3_test),caption="k1/ k2 /k3")
```

Table 3: k1/ k2 /k3

-9.144	-12.14	-143.2
0.5389	-2.461	-48.8
-8.718	-11.72	-110.3
-5.982	-8.982	-108.9
-7.44	-10.44	-140.5
-9.713	-12.71	-185.3
-5.013	-8.013	-58.68
4.359	1.359	-15.14
-3.22	-6.22	-86.1
-8.657	-11.66	-90.31
-11.85	-14.85	-247.6
-11.53	-14.53	-167
-11.28	-14.28	-190
-7.231	-10.23	-77.17
-9.244	-12.24	-137.7
-3.69	-6.69	-58.54
-8.871	-11.87	-106.2
-6.999	-9.999	-164.5
-6.375	-9.375	-198.4
-1.091	-4.091	-22.01

4 Kernels over $\mathcal{X} = \mathbb{R}^2$

4.1 Verify that $\phi(x)^T\phi(y) = (x^Ty)^2$

1. Find a function $\phi(x)$:$\mathbb{R}^2 \mapsto \mathbb{R}^6$ such that for any (x, y), $\phi(x)^T\phi(y) = (x^Ty + 1)^2$
2. Find a function $\phi(x)$:$\mathbb{R}^2 \mapsto \mathbb{R}^9$ such that for any (x, y), $\phi(x)^T\phi(y) = (x^Ty + 1)^2$
3. Verify that
$$K(x, y) = (1 + x^Ty)^d$$
for $d = 1, 2 \ldots$ is a positive definite kernel
4. find a function ϕ:$\mathbb{R}^2 \mapsto H$, where H is an inner product space such that for any (x, y), $< \phi(x), \phi(y) >_H = x^Ty - 1$