# 1 Classifier

## 1.1 proof

$g(x) =< C_+ - C_-, X - C >=< C_+, X > - < C_-, X > - < C_+, C > + < C_-, C >;$

$< C_+, X >=< \frac{1}{n_+} \sum_{i \in I_+}^{n} x_i, x >;$

$< C_-, X >=< \frac{1}{n_-} \sum_{i \in I_-}^{n} x_i, x >;$

$< C_+, C >=< C_+, \frac{1}{2}C_+ > + < C_+, \frac{1}{2}C_- >= \frac{1}{2n_+^2} \sum_{(i,j) \in I_+} < x_i, x_j > + \frac{1}{2} < C_+, C_- >$

$< C_-, C >=< C_-, \frac{1}{2}C_+ > + < C_-, \frac{1}{2}C_- >= \frac{1}{2} < C_+, C_- > + \frac{1}{2n_-^2} \sum_{(i,j) \in I_-} < x_i, x_j >$

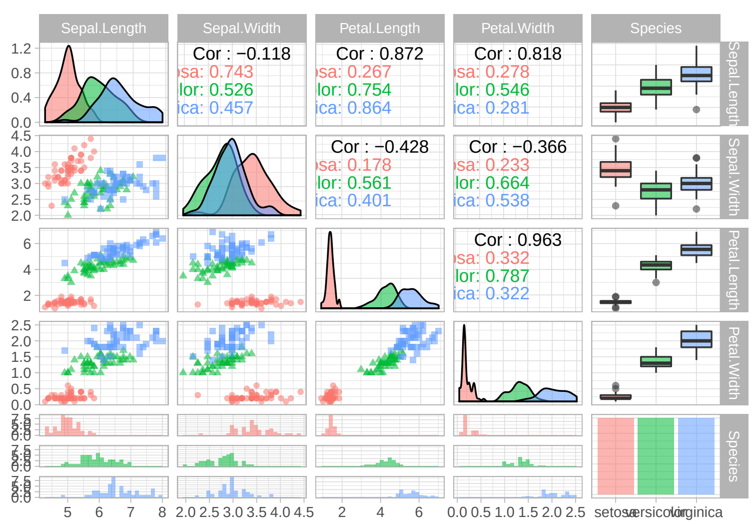
$$g(x) =< \frac{1}{n_+} \sum_{i \in I_+}^{n} x_i, x > - < \frac{1}{n_-} \sum_{i \in I_-}^{n} x_i, x > - \frac{1}{2n_+^2} \sum_{(i,j) \in I_+} < x_i, x_j > - \frac{1}{2} < C_+, C_- > + \frac{1}{2} < C_+, C_- > + \frac{1}{2n_-^2} \sum_{(i,j) \in I_-} < x_i, x_j >$$

$$= \sum_{i=1}^{n} \alpha_i < x_i, x > + b$$

where $\alpha_i = \begin{cases} \frac{1}{n_+} & y_i = +1 \\ -\frac{1}{n_-} & y_i = -1 \end{cases}$; $b = \frac{1}{2n_-^2} \sum_{(i,j) \in I_-} < x_i, x_j > - \frac{1}{2n_+^2} \sum_{(i,j) \in I_+} < x_i, x_j >$

## 1.2 iris data clasification

- Import the iris data

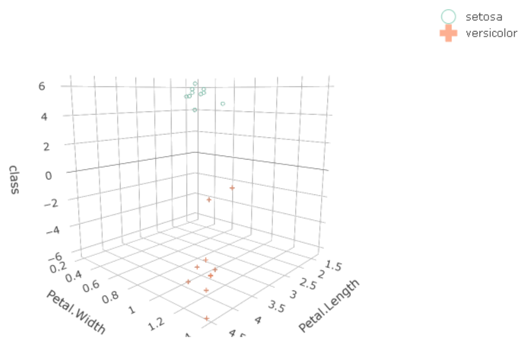


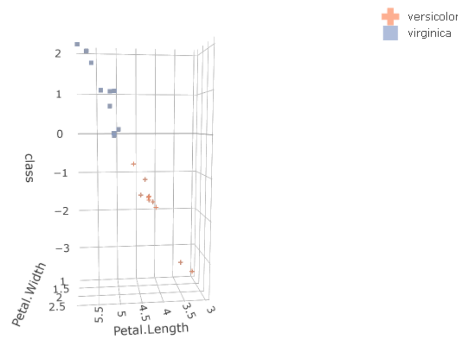

```
# Define the train and test sets
iris$class <- NA
iris_setosa <- iris[iris$Species=="setosa",]
iris_versicolor <- iris[iris$Species=="versicolor",]
iris_virginica <- iris[iris$Species=="virginica",]
iris_train_se<- iris_setosa[1:40,]
iris_train_ve<- iris_versicolor[1:40,]
iris_train_vi<- iris_virginica[1:40,]
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))
# setosa v.s.versicolor
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)
# virginica v.s.versicolor
k.pp=outer(1:40,1:40,Vectorize(function(i,j) k(iris_train_vi[i,1:4],iris_train_vi[j,1:4])))
```

```r
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")
error_rate1 <- length(which(iris_test_se.ve$evaluate=="Wrong"))/20
iris_test_ve.vi$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")
error_rate2 <- length(which(iris_test_ve.vi$evaluate=="Wrong"))/20
```

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




The left figure of setosa v.s. versicolor shows that the two specises are separated by zero plane, while virginica v.s. versicolor shows that a few points are close to zero. The tables also show a misclassified point in virginica v.s. versicolor.

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 |

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 |
| versicolor | -5.402 | Rigt | virginica | 0.6535 | Rigt |
| versicolor | -0.6631 | Rigt | virginica | 0.9918 | Rigt |
| versicolor | -4.411 | Rigt | virginica | 0.02902 | Rigt |

# 2 Perceptron

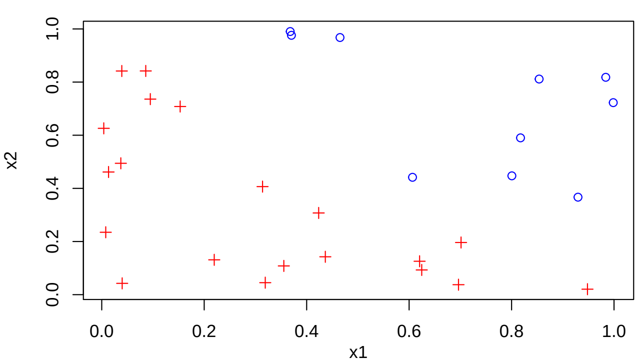
## 2.1 pseduo code

Init:
– define the classifier $f(x) = w^T x$
– define perceptron algorithm
– set the sample size =n
– set initial weight with $w \leftarrow y_1 x_1$
– set initial misclassified argument with TRUE
While misclassfied=TURE
– change argument with FALSE
– For $i = 2 \ldots n$ do
— If $y_i w^T x_i < 0$ then $w \leftarrow w + y_i x_i$
— change argument with TRUE
— EndIf
– EndFor
EndWhile

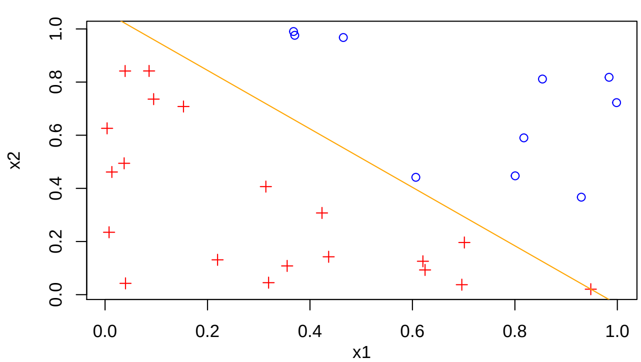
## 2.2 Using 3 Kernels

```
n <- 30; dim <- 2; threshold <-1           ## generate 2d data
x <- runif(n * dim)
x <- matrix(x, ncol = dim)
y <- ifelse(apply(x, 1, sum) < threshold, -1, 1)
data <- cbind(x, y, alpha = rep(1, n))
k<- function(x,w){return(sum(x*w))}        ## define the kernel function
k_perceptron <- function(data,dim) {       ## difine perceptron algorithm
n <- nrow(data)                            ## get the sample size
y <- data[ ,dim+1]                         ## select y
w <- c(data[1,1:dim]*data[1,dim+2],0)      ## initialize weight with y_1*x_1
misclassfied <- TRUE                       ## initialize the argument
  while (misclassfied) {                   ## start to loop
    misclassfied <- FALSE
    for (i in 2:n) {                       ## loop times = sample size-1
      if (y[i]*k(data[i,-dim-1],w) < 0) { ## if expectation does not match y
         w <- w + y[i] * data[i,-dim-1]    ## update the weights
         misclassfied <- TRUE              ## reset argument and check next one
  }}}
  return(w)
}
k <- function(x,y) {return(sum(x*y))} ## Use k1
(w1 <- k_perceptron(data,2))
##
##  2.068041  1.879500 -2.000000
k <- function(x,y) {return(sum(x*y+1))} ## Use k2
(w2 <- k_perceptron(data,2))
##
##  6.081670  6.213451 -9.000000
k <- function(x,y) {return(sign(x%*%y))} ## Use k3
(w3 <- k_perceptron(data,2))
##
##  2.068041  1.879500 -2.000000
```

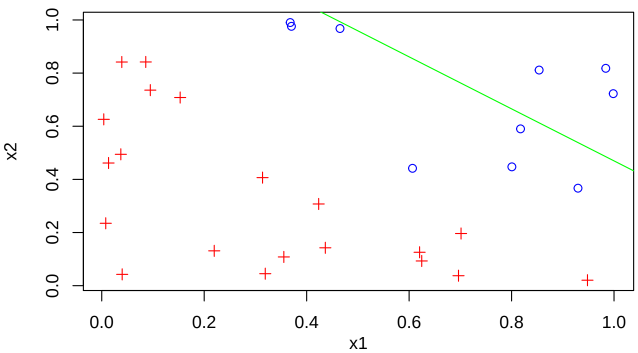
The output shows that the perceptrons with $k = x^T y$ and $k = \text{sign}(x^T y)$ can classify the data. The $k = x^T y + 1$ is biased.



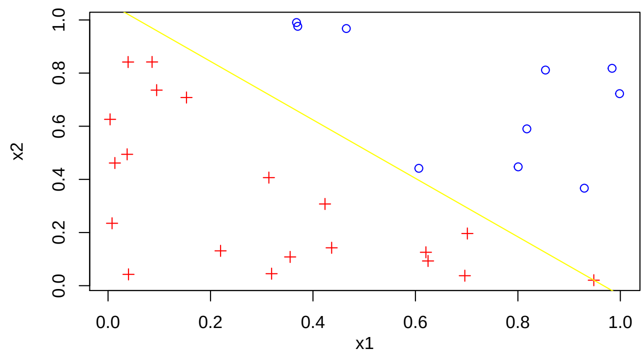
Before perceptron



Use k1 perceptron



Use k2 perceptron



Use k3 perceptron

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

$x = (x_1, x_2) \in \mathbb{R}^2$ and $y = (y_1, y_2) \in \mathbb{R}^2$,

## 3.1 Let $\phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$,verify that $\phi(x)^T\phi(y) = (x^Ty)^2$

$$(x^Ty)^2 = (\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{2\times 1} \begin{bmatrix} y_1 & y_2 \end{bmatrix}_{1\times 2})^2 = (x_1y_1 + x_2y_2)^2 = x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2$$

$$= \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}_{3\times 1} \begin{bmatrix} y_1^2 & \sqrt{2}y_1y_2 & y_2^2 \end{bmatrix}_{1\times 3} = \phi(x)^T\phi(y) \quad \blacksquare$$

## 3.2 Find a function $\phi(x)$:$\mathbb{R}^2 \mapsto \mathbb{R}^6$ such that for any $(x, y)$, $\phi(x)^T\phi(y) = (x^Ty + 1)^2$

$$(x^Ty + 1)^2 = (x_1y_1 + x_2y_2 + 1)^2 = x_1^2y_1^2 + 2x_1y_1 + 2x_1x_2y_1y_2 + 2x_2y_2 + x_2^2y_2^2 + 1$$

$$= \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_2 \\ x_2^2 \\ 1 \end{bmatrix}_{6\times 1} \begin{bmatrix} y_1^2 & \sqrt{2}y_1 & \sqrt{2}y_1y_2 & \sqrt{2}y_2 & y_2^2 & 1 \end{bmatrix}_{1\times 6}$$

Thus,

$$\phi(x) = (x_1^2, \sqrt{2}x_1, \sqrt{2}x_1x_2, \sqrt{2}x_2, x_2^2, 1) \quad \blacksquare$$

## 3.3 Find a function $\phi(x)$:$\mathbb{R}^2 \mapsto \mathbb{R}^9$ such that for any $(x, y)$, $\phi(x)^T\phi(y) = (x^Ty + 1)^2$

$$(x^Ty+1)^2 = x_1^2y_1^2 + 2x_1y_1 + \frac{1}{2}x_1x_2y_1y_2 + \frac{1}{2}x_2x_1y_2y_1 + \frac{1}{2}x_1^{0.5}x_2x_1^{0.5}y_1^{0.5}y_2y_1^{0.5} + \frac{1}{2}x_2^{0.5}x_1x_2^{0.5}y_2^{0.5}y_1y_2^{0.5} + 2x_2y_2 + x_2^2y_2^2 + 1$$

$$= \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1 \\ \frac{1}{\sqrt{2}}x_1x_2 \\ \frac{1}{\sqrt{2}}x_2x_1 \\ \frac{1}{\sqrt{2}}x_1^{0.5}x_2x_1^{0.5} \\ \frac{1}{\sqrt{2}}x_2^{0.5}x_1x_2^{0.5} \\ \sqrt{2}x_2 \\ x_2^2 \\ 1 \end{bmatrix}_{9\times 1} \begin{bmatrix} y_1^2 & \sqrt{2}y_1 & \frac{1}{\sqrt{2}}y_1y_2 & \frac{1}{\sqrt{2}}y_2y_1 & \frac{1}{\sqrt{2}}y_1^{0.5}y_2y_1^{0.5} & \frac{1}{\sqrt{2}}y_2^{0.5}y_1y_2^{0.5} & \sqrt{2}y_2 & y_2^2 & 1 \end{bmatrix}_{1\times 9}$$

Thus,

$$\phi(x) = (x_1^2, \sqrt{2}x_1, \frac{1}{\sqrt{2}}x_1x_2, \frac{1}{\sqrt{2}}x_2x_1, \frac{1}{\sqrt{2}}x_1^{0.5}x_2x_1^{0.5}, \frac{1}{\sqrt{2}}x_2^{0.5}x_1x_2^{0.5}, \sqrt{2}x_2, x_2^2, 1) \quad \blacksquare$$

### 3.4 Verify that $K(x, y) = (1 + x^T y)^d$ for $d = 1, 2 \ldots$ is a positive definite kernel

We know that $x^T y = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{2\times 1} \begin{bmatrix} y_1 & y_2 \end{bmatrix}_{1\times 2}) = x_1 y_1 + x_2 y_2 = y^T x$, then

$$k(x, y) = (1 + x^T y)^d = (1 + y^T x)^d = k(y, x)$$

Thus, $k(x, y)$ is symmetric.

$$\exists \phi(x) = \left( \sqrt{\binom{k}{0}} x_1^k x_2^0, \sqrt{\binom{k}{1}} x_1^{k-1} x_2^1, \cdots, \sqrt{\binom{k}{k-1}} x_1^1 x_2^{k-1}, \sqrt{\binom{k}{k}} x_1^0 x_2^k \right)$$

Make

$$k(\phi(x), \phi(y)) = \phi(x)^T \phi(y) = \sum_{l=0}^{k} \binom{k}{l} (x_1 y_1)^{k-l} (x_2 y_2)^l = (x_1 y_1 + x_2 y_2)^k = (x^T y)^k$$

$k(\phi(x), \phi(y))$ is a p.d. kernel and $\| \sum_{i=1}^{2} \alpha_i \phi_i(x) \|^2 \geq 0$. Therefore,

$$k(x, y) = (1 + x^T y)^d = \sum_{k=0}^{d} \binom{d}{k} (x^T y)^k 1^{d-k} = \sum_{k=0}^{d} \binom{d}{k} \phi(x)^T \phi(y)$$

For $d, k \in \mathbb{N}^+$,

$$\sum_{i=1}^{2} \sum_{j=1}^{2} \alpha_i \alpha_j < x_i, x_j >_{R^d} = \sum_{i=1}^{2} \sum_{j=1}^{2} \alpha_i \alpha_j \left[ \sum_{k=0}^{d} \binom{d}{k} \phi_i(x)^T \phi_j(x) \right] = \sum_{k=0}^{d} \binom{d}{k} \left[ \sum_{i=1}^{2} \sum_{j=1}^{2} \alpha_i \alpha_j < \phi_i(x)^T \phi_j(x) >_R \right]$$

$$= \sum_{k=0}^{d} \binom{d}{k} \| \sum_{i=1}^{2} \alpha_i \phi_i(x) \|^2 \geq 0$$

Therefore, for $x_1, x_2,\ x_i \in \mathbb{R};\ \alpha_1, ..\alpha_2,\ \alpha_i \in \mathbb{R}\ k(x, y) = (1 + x^T y)^d$ is a positive definite kernel. ■

### 3.5 find a function $\phi$:$\mathbb{R}^2 \mapsto H$, where $H$ is an inner product space such that for any $(x, y)$, $< \phi(x), \phi(y) >_H = x^T y - 1$

$$k(x, y) = < \phi(x), \phi(y) >_H = x^T y - 1 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{2\times 1} \begin{bmatrix} y_1 & y_2 \end{bmatrix}_{1\times 2} - 1 = x_1 y_1 + x_2 y_2 - 1$$

For all $|x| < \frac{\sqrt{2}}{2}$, when $\alpha_1 = \alpha_2 = 1$, $< \phi(x), \phi(x) >_H = 2x^2 - 1 < 0$.
By the property of Kernel, $k(x, y)$ should be positive.
Therefore, there isn't a function $\phi$:$\mathbb{R}^2 \mapsto H$ that can make $< \phi(x), \phi(y) >_H = x^T y - 1$. ■