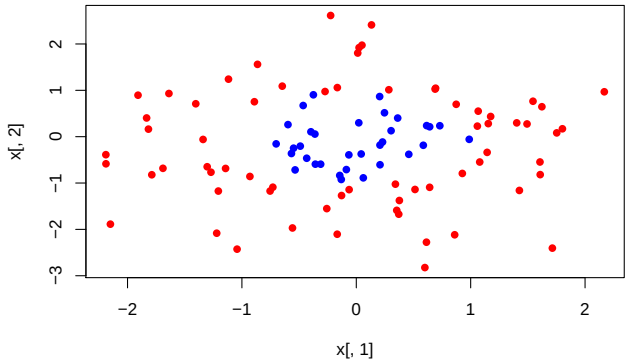
1 Kernalized ridge regression

Code the kernalized ridge regression for the 2d data that you already used for the simple classifier and the perceptron. Show your code. Show 3 examples of results obtain with simulated data in 2 dimensions, using 3 different kernels.

1.1 2d data

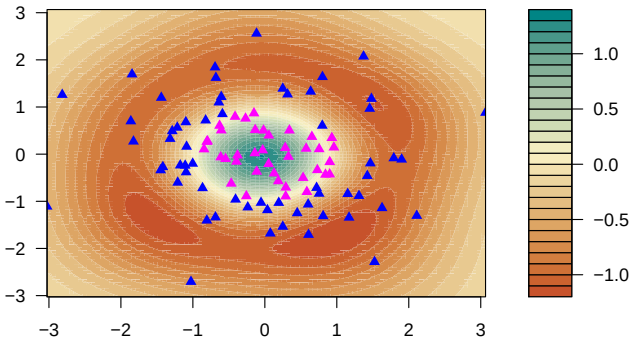
```
# Define Kernel
rm(list=ls())
k1 = function(x,y) return(sum(x*y))
k12 = function(x,y)  return(sum(x*y)+1)
k2 = function(x,y)  return(sum(x*y)^2)
k3 = function(x,y)  return((1+sum(x*y))^2)
d=4
k4 = function(x,y)  return((1+sum(x*y))^d)
sigma=1
k5 = function(x,y)  return(exp(-sum((x-y)^2)/(2*sigma^2)))
k51 = function(x,y)  return(exp(-sum(x^2)-sum(y^2)+2*sum((x*y))))
kappa=1
theta=1
k6 = function(x,y)  return(tanh(kappa*sum(x*y)+theta))
kernel = function(x,y)  return(k5(x,y))
```

```
# Generating training data and test data
n=100
library(mvtnorm)
x <- rmvnorm(n=n,mean=c(-2,-2)+c(2,2),sigma=diag(rep(1,2)))
y<-ifelse(x[,1]^2+x[,2]^2<1,1,-1)
plot(x[,1],x[,2],col=y+rep(3,n),pch=16)
```




```
test.x <- rmvnorm(n=n,mean=c(-2,-2)+c(2,2),sigma=diag(rep(1,2)))
test.y <- ifelse(test.x[,1]^2+test.x[,2]^2<1,1,-1)
## compute the classifier
lambda=0.01
KK <- matrix(rep(0,n^2),n,n)
KK=outer(1:n,1:n,Vectorize(function(i,j) kernel(x[i,],x[j,])))
alpha = solve(KK + lambda*n*diag(rep(1,n)))%*%y
f=t(KK)%*%alpha
# test
k.t <- matrix(rep(0,n^2),n,n)
k.t=outer(1:n,1:n,Vectorize(function(i,j) kernel(x[i,],test.x[j,])))
hat.y=sign(t(k.t)%*%alpha)
c11=sum(hat.y[test.y==1,]==1)
c12=sum(hat.y[test.y==1,]==-1)
c21=sum(hat.y[test.y==-1,]==1)
c22=sum(hat.y[test.y==-1,]==-1)
print(matrix(c(c11,c21,c12,c22),nrow=2))
```

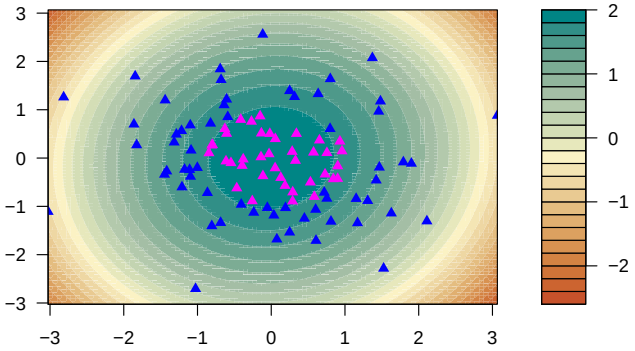
```
##      [,1] [,2]
## [1,]   34    5
## [2,]    2   59
```



1.1.1 Using 3 Kernels

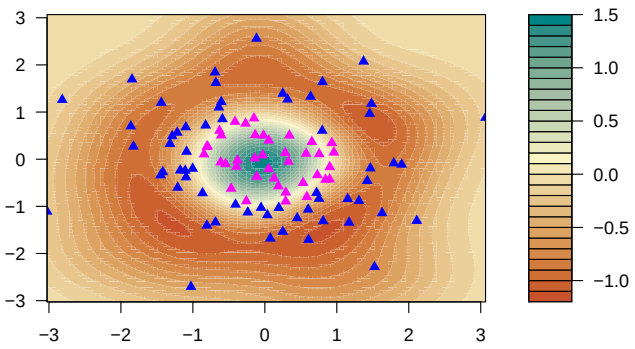
```
## use  kernel (x*y)^2
kernel = function(x,y)
  return(k2(x,y))
KK <- matrix(rep(0,n^2),n,n)
KK=outer(1:n,1:n,Vectorize(function(i,j) kernel(x[i,],x[j,])))
alpha = solve(KK + lambda*n*diag(rep(1,n)))%*%y
f=t(KK)%*%alpha
# test
k.t <- matrix(rep(0,n^2),n,n)
k.t=outer(1:n,1:n,Vectorize(function(i,j) kernel(x[i,],test.x[j,])))
hat.y=sign(t(k.t)%*%alpha+0.25)
c11=sum(hat.y[test.y==1,]==1)
c12=sum(hat.y[test.y==1,]==-1)
c21=sum(hat.y[test.y==-1,]==1)
c22=sum(hat.y[test.y==-1,]==-1)
print(matrix(c(c11,c21,c12,c22),nrow=2))
```

```
##      [,1] [,2]
## [1,]   36    3
## [2,]    7   54
```



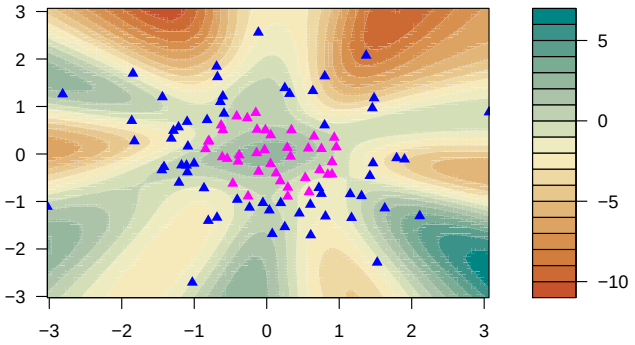

```
## use kernel Gaussian-1
kernel = function(x,y)
  return(k51(x,y))
KK <- matrix(rep(0,n^2),n,n)
KK=outer(1:n,1:n,Vectorize(function(i,j) kernel(x[i,],x[j,])))
alpha = solve(KK + lambda*n*diag(rep(1,n)))%*%y
f=t(KK)%*%alpha
# test
k.t <- matrix(rep(0,n^2),n,n)
k.t=outer(1:n,1:n,Vectorize(function(i,j) kernel(x[i,],test.x[j,])))
hat.y=sign(t(k.t)%*%alpha)
c11=sum(hat.y[test.y==1,]==1)
c12=sum(hat.y[test.y==1,]==-1)
c21=sum(hat.y[test.y==-1,]==1)
c22=sum(hat.y[test.y==-1,]==-1)
print(matrix(c(c11,c21,c12,c22),nrow=2))
```

```
##      [,1] [,2]
## [1,]   34    5
## [2,]    1   60
```




```
## use kernel 6
kernel = function(x,y)
  return(k6(x,y))
KK <- matrix(rep(0,n^2),n,n)
KK=outer(1:n,1:n,Vectorize(function(i,j) kernel(x[i,],x[j,])))
alpha = solve(KK + lambda*n*diag(rep(1,n)))%*%y
f=t(KK)%*%alpha
# test
k.t <- matrix(rep(0,n^2),n,n)
k.t=outer(1:n,1:n,Vectorize(function(i,j) kernel(x[i,],test.x[j,])))
hat.y=sign(t(k.t)%*%alpha)
c11=sum(hat.y[test.y==1,]==1)
c12=sum(hat.y[test.y==1,]==-1)
c21=sum(hat.y[test.y==-1,]==1)
c22=sum(hat.y[test.y==-1,]==-1)
print(matrix(c(c11,c21,c12,c22),nrow=2))
```

```
##      [,1] [,2]
## [1,]   25   14
## [2,]   13   48
```



1.2 MNIST data clasification

Code a version of the kernalized ridge regression algorithm that let you run the algorithm for various sets of images from the MNIST dataset such that you can perform controled experiments. The input of the algorithm could be a list of filenames, corresponding to images for training, testing, as well as the corresponding labels.

Run the kernelized ridge regression for one digit versus another one of your choice. You will see that the size of the full MNIST training set might be too large for the algorithm to run in a reasonable time. Instead, sample smaller training sets, say of size 100. Now, you should be able to run the algorithm. Experiment with smaller and larger training set size. Report the performance by specifying the total number of images

correctly classified, as well as the learning time and testing time. Show graphs, with on the horizontal axis the number of images used for training or for testing.

Import data: Omitted.

```
# First Run: 50 train and 50 test
digit1=1
digit2=9
id.train = sample(which(train$y==digit1 | train$y==digit2),size=50)
id.test= sample(which(test$y==digit1 | test$y==digit2),size=50)
x=train$x[id.train,]
y=train$y[id.train]
test.x=test$x[id.test,]
test.y=test$y[id.test]
# start to train by ridge regression
ptm <- proc.time()
n <- 50; m <- 50
lambda=0.001
kk <- tcrossprod(x)
dd <- diag(kk)
gau.kernel <- exp((-matrix(dd,n,n)-t(matrix(dd,n,n))+2*kk))
alpha = solve(gau.kernel + lambda*n*diag(rep(1,n)))%*%y
train_50<- proc.time() - ptm
# start to test by ridge regression
kt <- tcrossprod(test.x)
dd <- diag(kt)
gau.kernel <- exp((-matrix(dd,m,m)-t(matrix(dd,m,m))+2*kt))
hat.y=sign(gau.kernel%*%alpha-4.76)
test_50<-proc.time() - ptm
# show the result
c11=sum(hat.y[y==digit1,]<0)
c12=sum(hat.y[y==digit2,]<0)
c21=sum(hat.y[y==digit1,]>0)
c22=sum(hat.y[y==digit2,]>0)
print(matrix(c(c11,c21,c12,c22),nrow=2))
```

```
##      [,1] [,2]
## [1,]   31    0
## [2,]    0   19
```

```
# Second Run : 100 train and 100 test
digit1=1
digit2=9
id.train = sample(which(train$y==digit1 | train$y==digit2),size=100)
id.test= sample(which(test$y==digit1 | test$y==digit2),size=100)
x=train$x[id.train,]
y=train$y[id.train]
test.x=test$x[id.test,]
test.y=test$y[id.test]
# start to train by ridge regression
ptm <- proc.time()
n <- 100; m <- 100
lambda=0.001
kk <- tcrossprod(x)
dd <- diag(kk)
gau.kernel <- exp((-matrix(dd,n,n)-t(matrix(dd,n,n))+2*kk))
alpha = solve(gau.kernel + lambda*n*diag(rep(1,n)))%*%y
train_100 <-proc.time() - ptm
# start to test by ridge regression
kt <- tcrossprod(test.x)
dd <- diag(kt)
gau.kernel <- exp((-matrix(dd,m,m)-t(matrix(dd,m,m))+2*kt))
hat.y=sign(gau.kernel%*%alpha-4.55)
test_100 <-proc.time() - ptm
# show the result
c11=sum(hat.y[y==digit1,]<0)
c12=sum(hat.y[y==digit2,]<0)
c21=sum(hat.y[y==digit1,]>0)
c22=sum(hat.y[y==digit2,]>0)
print(matrix(c(c11,c21,c12,c22),nrow=2))
```

```
##      [,1] [,2]
## [1,]   55    0
## [2,]    0   45
```

```
# Third Run : 1000 train and 1000 test
digit1=1
digit2=9
id.train = sample(which(train$y==digit1 | train$y==digit2),size=1000)
id.test= sample(which(test$y==digit1 | test$y==digit2),size=1000)
x=train$x[id.train,]
y=train$y[id.train]
test.x=test$x[id.test,]
test.y=test$y[id.test]
# start to train by ridge regression
ptm <- proc.time()
n <- 1000; m <- 1000
lambda=0.001
kk <- tcrossprod(x)
dd <- diag(kk)
gau.kernel <- exp((-matrix(dd,n,n)-t(matrix(dd,n,n))+2*kk))
alpha = solve(gau.kernel + lambda*n*diag(rep(1,n)))%*%y
train_1000 <-proc.time() - ptm
lambda_0.001 <-proc.time() - ptm
# start to test by ridge regression
kt <- tcrossprod(test.x)
dd <- diag(kt)
gau.kernel <- exp((-matrix(dd,m,m)-t(matrix(dd,m,m))+2*kt))
hat.y=sign(gau.kernel%*%alpha-2.5)
test_1000 <-proc.time() - ptm
# show the result
c11=sum(hat.y[y==digit1,]<0)
c12=sum(hat.y[y==digit2,]<0)
c21=sum(hat.y[y==digit1,]>0)
c22=sum(hat.y[y==digit2,]>0)
print(matrix(c(c11,c21,c12,c22),nrow=2))
```

```
##      [,1] [,2]
## [1,]  507    0
## [2,]    0  493
```

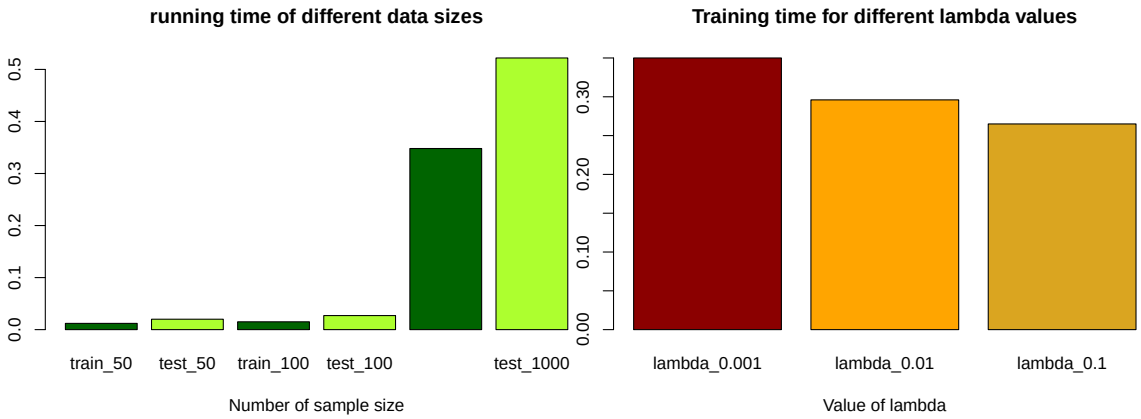
```
# Fouth Run : lambda=0.01
# start to train by ridge regression
ptm <- proc.time()
n <- 1000; m <- 1000
lambda=0.01
kk <- tcrossprod(x)
dd <- diag(kk)
gau.kernel <- exp((-matrix(dd,n,n)-t(matrix(dd,n,n))+2*kk))
alpha = solve(gau.kernel + lambda*n*diag(rep(1,n)))%*%y
lambda_0.01 <-proc.time() - ptm
# start to test by ridge regression
kt <- tcrossprod(test.x)
dd <- diag(kt)
gau.kernel <- exp((-matrix(dd,m,m)-t(matrix(dd,m,m))+2*kt))
hat.y=sign(gau.kernel%*%alpha-0.45)
# show the result
c11=sum(hat.y[y==digit1,]<0)
c12=sum(hat.y[y==digit2,]<0)
c21=sum(hat.y[y==digit1,]>0)
c22=sum(hat.y[y==digit2,]>0)
print(matrix(c(c11,c21,c12,c22),nrow=2))
```

```
##      [,1] [,2]
## [1,]  507    0
## [2,]    0  493
```

```
# Fifth Run : lambda=0.1
# start to train by ridge regression
ptm <- proc.time()
n <- 1000; m <- 1000
lambda=0.1
kk <- tcrossprod(x)
dd <- diag(kk)
gau.kernel <- exp((-matrix(dd,n,n)-t(matrix(dd,n,n))+2*kk))
alpha = solve(gau.kernel + lambda*n*diag(rep(1,n)))%*%y
lambda_0.1 <-proc.time() - ptm
# start to test by ridge regression
kt <- tcrossprod(test.x)
dd <- diag(kt)
gau.kernel <- exp((-matrix(dd,m,m)-t(matrix(dd,m,m))+2*kt))
```

```
hat.y=sign(gau.kernel%*%alpha-0.045)
# show the result
c11=sum(hat.y[y==digit1,]<0)
c12=sum(hat.y[y==digit2,]<0)
c21=sum(hat.y[y==digit1,]>0)
c22=sum(hat.y[y==digit2,]>0)
print(matrix(c(c11,c21,c12,c22),nrow=2))
```

```
##      [,1] [,2]
## [1,]  507    0
## [2,]    0  493
```



2 Semi-parametric regression

Let $\mathcal{D} = \{(x_i, y_i), 1 \leq i \leq n\}$ be a training set where $x_i \in \mathbb{R}^d$ is a feature vector, and $y_i \in \mathbb{R}$ is the target, or independent variable. We are interested in the following semi-parametric model for predicting y, $f(x) = \theta^T x + g(x)$ where $\theta \in \mathbb{R}^d$ is a vector of parameters and $g : \mathbb{R}^d \mapsto \mathbb{R}$ belongs to a RKHS with kernel $k(.,.)$. This model is called semi-parametric because it is the sum of a parametric component, here the linear term $\theta^T x$ and a non-linear component, the function $g(.)$. Consider the functional $J(\theta, g) = \sum_{i=1}^n \left(y_i - \theta^T x_i - g(x_i)\right)^2 + \lambda ||g||_H^2$

2.1 Show that for any θ, a function $g \in H$ that minimizes $J(\theta, g)$ has the following form $g(.) = \sum_{i=1}^n \alpha_i k(x_i, .)$ where $\alpha \in \mathbb{R}^n$

Let $v = \text{span}[k(\cdot, x_i), .., k(\cdot, x_n)]$ $\mathcal{V}$ is closed linear subspace of $\mathcal{H}$. Then all minimizers of J belong to $\mathcal{V}$. Thus, there is an unique decomposition $g = g_v + g_\perp$ with $g_v \in \mathcal{V}$

$\forall g \in \mathcal{V}, \langle g_\perp, f \rangle = 0$

$$\|g\|_{\mathcal{H}}^2 = \|g_v + g_\perp\|_{\mathcal{H}}^2 = \langle g_v + g_\perp, g_v + g_\perp \rangle = \langle g_v, g_v \rangle + \langle g_\perp, g_\perp \rangle + \underbrace{2\langle g_v, g_\perp \rangle}_{0} = \|g_v\|_{\mathcal{H}}^2 + \|g_\perp\|_{\mathcal{H}}^2$$

$$g(x_i) = \langle g, k(\cdot, x_i) \rangle = \langle g_v + g_\perp, k(\cdot, x_i) \rangle = \langle g_v, k(\cdot, x_i) \rangle + \underbrace{\langle g_\perp, k(\cdot, x_i) \rangle}_{0} = g_v(x_i)$$

For g is strictly increasing

$$\begin{aligned}
J(\theta, g) - J(\theta, g_v) &= \sum_{i=1}^{n} \left(y_i - \theta^T x_i - g(x_i)\right)^2 + \lambda \|g\|_{\mathcal{H}}^2 - \sum_{i=1}^{n} \left(y_i - \theta^T x_i - g_v(x_i)\right)^2 - \lambda \|g_v\|_{\mathcal{H}}^2 \\
&= \sum_{i=1}^{n} \left(y_i - \theta^T x_i - g(x_i)\right)^2 - \sum_{i=1}^{n} \left(y_i - \theta^T x_i - g_v(x_i)\right)^2 + \lambda \|g\|_{\mathcal{H}}^2 - \lambda \|g_v\|_{\mathcal{H}}^2 \\
&= \lambda \|g_\perp\|_{\mathcal{H}}^2 \geq 0 \quad \text{is free of} \theta
\end{aligned}$$

The representer theorem allows us to reduce the optimization problem to a finite dimensional optimization problem.

Let $\alpha = (\alpha_1, .., \alpha_n)^T \in \mathbb{R}$ is the solution of $\min J(\theta, g)$

For any θ, the function $g(.) = \sum_{i=1}^{n} \alpha_i k(x_i, .), g \in \mathcal{H}$ that minimizes $J(\theta, g)$

2.2 Show that $J(\sum_{i=1}^{n} \alpha_i k(x_i, .), \theta) = ||y - X\theta - K\alpha||^2 + \lambda \alpha^T K \alpha$ for some matrix K and X which you will specify together with their dimensions.

$$\|g\|_{\mathcal{H}}^2 = \langle g, g \rangle_{\mathcal{H}} = \langle \sum_{i=1}^{n} \alpha_i k(\cdot, x_i), \sum_{j=1}^{n} \alpha_j k(\cdot, x_j) \rangle_{\mathcal{H}} = \sum_{i,j=1}^{n} \alpha_i \alpha_j \underbrace{\langle k(\cdot, x_i), k(\cdot, x_j) \rangle_{\mathcal{H}}}_{k(x_i, x_j)} = \underset{(1,n)}{\alpha^T} \underset{(n,n)}{K} \underset{(n,1)}{\alpha}$$

$$g(x_i) = \sum_{j=1}^{n} \alpha_j k(x_i, x_j) = \sum_{j=1}^{n} \alpha_j [\underset{(n,n)}{K}]_{i,j} = [K\alpha]_i$$

$$J(\theta, \sum_{i=1}^{n} \alpha_i k(x_i, \cdot)) = \sum_{i=1}^{n} (y_i - \theta^T x_i - [K\alpha]_i)^2 + \lambda \|g\|_{\mathcal{H}}^2 = \| \underset{(n,1)}{Y} - \underset{(n,d)}{X} \underset{(d,1)}{\theta} - \underset{(n,n)}{K} \underset{(n,1)}{\alpha} \|^2 + \underset{(1,1)}{\lambda} \underset{(1,n)}{\alpha^T} \underset{(n,n)}{K} \underset{(n,1)}{\alpha}$$

2.3 Compute $\nabla_\alpha J$, the gradient of J with respect to α. Similarly, compute $\nabla_\theta J$, the gradient of J with respect to θ.

$\min_{g \in \mathcal{H}} J(\theta, g)$

$$\begin{aligned}
\nabla_\alpha J &= \frac{\partial}{\partial \alpha} [\|Y - X\theta - K\alpha\|^2 + \lambda \alpha^T K \alpha] \\
&= \frac{\partial}{\partial \alpha} \|K\alpha + X\theta - Y\|^2 + \lambda \frac{\partial}{\partial \alpha} \langle \alpha, K\alpha \rangle \\
&= 2(K\alpha + X\theta - Y) \frac{\partial}{\partial \alpha} (K\alpha + X\theta - Y) + \lambda (IK\alpha + K^T \alpha) \\
&= 2(K\alpha + X\theta - Y) K^T + 2\lambda K\alpha \\
&\underset{K = K^T}{=} 2K[(K + \lambda I)\alpha + X\theta - Y]
\end{aligned}$$

$$\begin{aligned}
\nabla_\theta J &= \frac{\partial}{\partial \theta} [\|Y - X\theta - K\alpha\|^2 + \lambda \alpha^T K \alpha] \\
&= 2(K\alpha + X\theta - Y) \frac{\partial}{\partial \theta} (K\alpha + X\theta - Y) \\
&= 2(K\alpha + X\theta - Y) X^T
\end{aligned}$$

2.4 Assume that the matrix X^TX is positive definite. Find one solution (α, θ) of the system $\nabla_\alpha J = 0, \nabla_\theta J = 0$.

p.d. K, X is symmetric, $K = K^T$, $X = X^T$; $K = P\Lambda P^T$; $I = PP^T$; Λ is diagonal matrix with $\gamma_1, .., \gamma_n$.

$\lambda > 0$,$\gamma_i > 0$, $K + \lambda I = P(\Lambda + \lambda I)P^T$ is inversible.

$$\nabla_\alpha J = (K + \lambda I)\alpha + X\theta - Y \overset{set}{=} 0 \implies (K + \lambda I)\alpha^\star = Y - X\theta \implies \alpha^\star = (K + \lambda I)^{-1}(Y - X\theta)$$

$$\nabla_\theta J = (K\alpha + X\theta - Y)X^T \overset{set}{=} 0 \implies X\theta^\star X^T = (Y - K\alpha)X^T \implies \theta^\star = (X^TX)^{-1}X^T(Y - K\alpha)$$

$\theta^\star = (X^TX)^{-1}X^T[Y - K(K + \lambda I)^{-1}(Y - X\theta^\star)] = (X^TX)^{-1}X^TY - (X^TX)^{-1}X^TK(K + \lambda I)^{-1}Y + (X^TX)^{-1}X^TK(K + \lambda I)^{-1}X\theta^\star$

$[I - (X^TX)^{-1}X^TK(K + \lambda I)^{-1}X]\theta^\star = (X^TX)^{-1}X^TY - (X^TX)^{-1}X^TK(K + \lambda I)^{-1}Y = (X^TX)^{-1}X^T[I - K(K + \lambda I)^{-1}]Y$

$\theta^\star = [I - (X^TX)^{-1}X^TK(K + \lambda I)^{-1}X]^{-1}(X^TX)^{-1}X^T[I - K(K + \lambda I)^{-1}]Y$

$\alpha^\star = (K + \lambda I)^{-1}Y - (K + \lambda I)^{-1}X[(X^TX)^{-1}X^T(Y - K\alpha^\star)] = (K + \lambda I)^{-1}[I - X[(X^TX)^{-1}X^T]Y + (K + \lambda I)^{-1}X[(X^TX)^{-1}X^TK\alpha^\star]$

$[I - (K + \lambda I)^{-1}HK]\alpha^\star = (K + \lambda I)^{-1}[I - H]Y$

$\alpha^\star = [I - (K + \lambda I)^{-1}HK]^{-1}(K + \lambda I)^{-1}[I - H]Y$

2.5 Write a code that demonstrate in one dimension the semi-parametric regression.

```
hmw3_data1 <- read.csv("hmw3-data1.csv")
n=10;n.plot=100
x = hmw3_data1$x
y = hmw3_data1$y
x.plot=seq(from=min(x),to=max(x),length.out=n.plot)
y.plot=seq(from=min(y),to=max(y),length.out=n.plot)
plot(x.plot,y.plot,type='n')
points(x,y,pch=16)
## compute the classifier
lambda=0.01
theta=1
I=diag(rep(1,n))
sigma=1
k = function(x,y)  return(exp(-sum((x-y)^2)/(2*sigma^2)))
kk=outer(1:n,1:n,Vectorize(function(i,j) k(x[i],x[j])))
alpha = solve(kk + lambda*diag(rep(1,n)))%*%(y-x*theta)
## evaluate the classifier
k.x=outer(1:n,1:n.plot,Vectorize(function(i,j) k(x[i],x.plot[j])))
hat.y=t(k.x)%*%alpha +x.plot*theta
lines(x.plot,hat.y,col=2)
legend("bottomleft",legend="estimated",col=2,lty=1)
```