

# Statistical Modelling and Inference: Week 2 Exercises

## Questions from Bayesian Regression

**1. For a Bayesian linear regression model with prior on  $p(w) = N(w|\mu, D^{-1})$  and  $q$  known, show that  $w_{Bayes}$  solves  $(D + q\phi^T\phi)w_{Bayes} = q\phi^T t + D\mu$**  To solve for bayes we take the known posterior probability:

$$-2\log p(w|t, X) = -2qt^T\phi w + qw^T\phi^T\phi w + (w - \mu)^T D(w - \mu) + const$$

To get the best estimate of  $w$ , we take the derivative of the posterior probability and set it equal to zero:

$$\frac{d(-2\log p(w|t, X))}{dw} = 0$$

The derivative of the **first term**:

$$\frac{d(-2qt^T\phi w)}{dw} = -2qt^T\phi$$

The derivative of the **second term** uses the property  $\frac{d(x^T Ax)}{dx} = 2Ax$

$$\frac{d(qw^T\phi^T\phi w)}{dw} = 2q\phi^T\phi w$$

Expanding the third term  $(w - \mu)^T D(w - \mu)$  gives:

$$= (w^T D - \mu^T D)(w - \mu)$$

$$= w^T D w - \mu^T D w - w^T D \mu + \mu^T D \mu$$

$$\partial(w^T D w - \mu^T D w - w^T D \mu + \mu^T D \mu) / \partial w = 2w^T D - 2\mu^T D$$

So we have:

$$0 = -2qt^T\phi + 2qw^T\phi^T\phi + 2w^T D - 2\mu^T D$$

Move negative terms to the LHS and divide both sides by 2:

$$qt^T\phi + \mu^T D = qw^T\phi^T\phi + w^T D$$

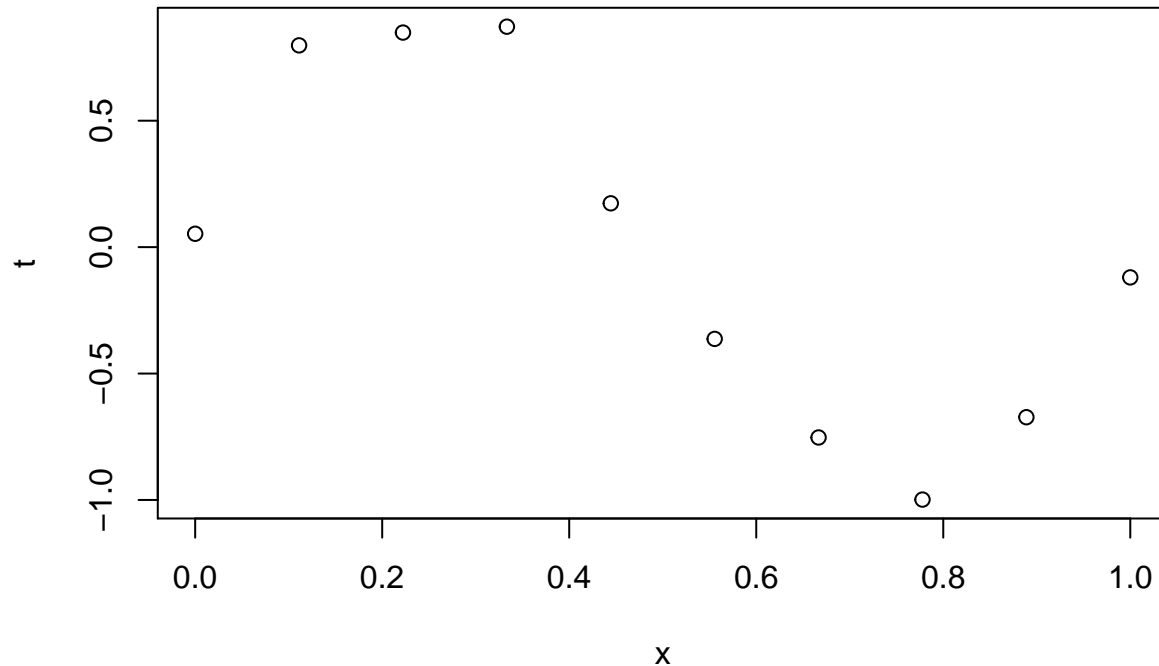
$$qw^T\phi^T\phi + w^T D = qt^T\phi + \mu^T D$$

Take the transpose of both sides (note:  $D^T = D$ ):

$$q\phi^T\phi w + Dw = q\phi^T t + D\mu$$

$$(D + q\phi^T\phi)w_{Bayes} = q\phi^T t + D\mu$$

**2. Curve fitting (pt1)** The aim is to learn a smooth function from the cloud of points stored in `curve_data.txt` using Bayesian linear regression models. In all that follows the prior  $p(w) = N(w|0, \delta^{-1}I)$  is used and  $q, \delta$  are constants specified by the user. **2.1** Plot the data



**2.2** Write a function in R, called `phix` that takes as input a scalar  $x$  (the input in curve fitting), with values in  $[0, 1]$ ,  $M$  the number of bases functions, and a categorical variable that specifies the type of basis used, and returns the vector of basis functions evaluated at  $x$ . Hence a call of the function `phix(0.3,4,"poly")` should return `c(1.0000, 0.3000, 0.0900, 0.0270, 0.0081)`. Code it up so that the option "poly" gives the polynomial bases and "Gauss" the Gaussian kernels with means  $\mu_i$  equally spaced in  $[0, 1]$ , with  $\mu = 0$  and  $\mu_M = 1$ .

```
phix <- function(x, M, option) {
  phi <- rep(0, M)
  if (option == "poly") {
    for (i in 1:(M)) {
      phi[i] <- x**i
    }
  }
  if (option == "Gauss") {
    for (i in 1:(M)) {
      phi[i] <- exp(-((x-i/(M))**2)/0.1)
    }
  }
  phi
}
```

**2.3** Write a function in R, called `post.params`, that takes as input the training data,  $M$ , the type of basis, the function `phix`,  $\delta$  and  $q$  and returns the parameters of the posterior distribution,  $w_{Bayes}$  and  $Q$ .

```
post.params <- function(data, M, option, delta, q) {
  phi = phix(data$x[1],M, option)
  for (i in 2:length(data$x)) {
    phi_ <- phix(data$x[i], M, option)
    phi = rbind(phi, phi_)
  }
}
```

```

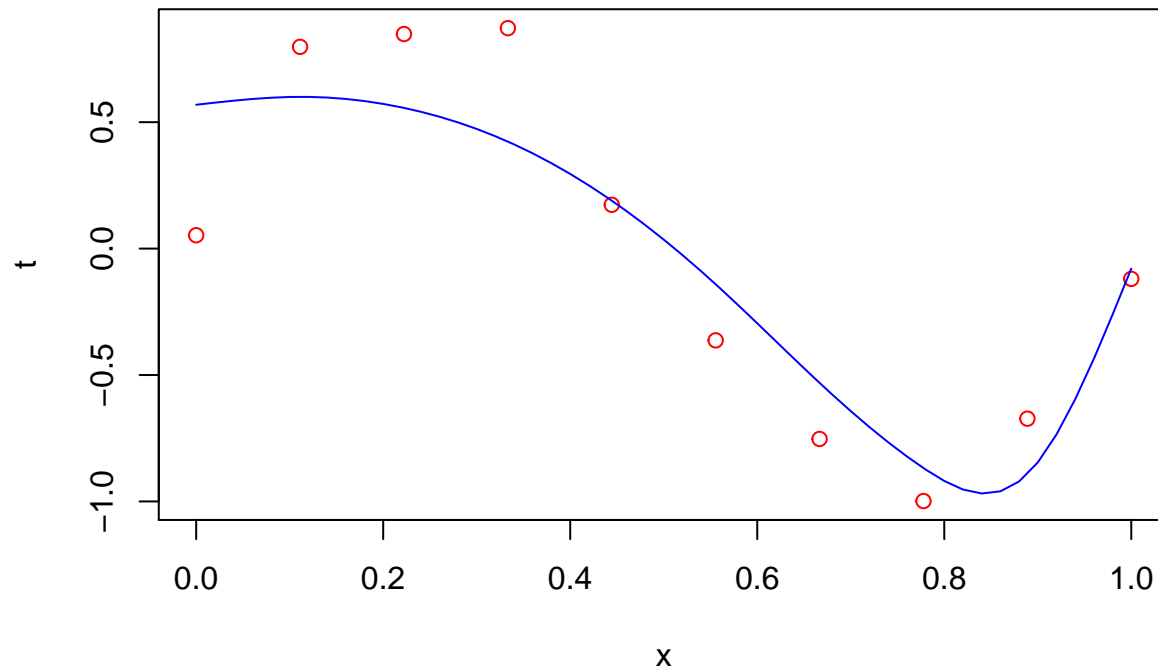
}
phi <- cbind(rep(1,M+1), phi)

Q = delta * diag(ncol(phi)) + q * (t(phi) %*% phi)
w = solve(Q)%*%(q*t(phi)%*%data$t)
t <- phi%*%w

return(list(Q, w, t))
}

```

**2.4 Plot the estimated linear predictor, by plugging in  $w_{Bayes}$ , and superimpose the training data; use  $q = (1/0.1)^2$ ,  $\delta = 2.0$  and  $M = 9$**



### Questions from Bayesian Prediction

1. Continuation of your work on smooth function estimation with the data in `curve_data.txt`.

```

library(MASS)

M = 9
basis_type = 'Gauss'
delta = 1.0
q = (1/0.1)**2
data <- read.table("curve_data.txt")
test.x <- seq(0,1,1/1000)

precision.bayes <- function(data, x, M, basis_type, delta, q) {
  input_data_params <- post.params(data, M, basis_type, delta, q)
  Qbayes <- input_data_params[[1]]
  wbayes <- input_data_params[[2]]

```

```

# create phi(testx) by sending all the test x to phix
phi.test.x <- matrix(nrow = length(test.x), ncol = M+1)
for (n in 1:length(test.x)) {
  phi.test.x[n,] <- c(1, phix(test.x[n], M, basis_type))
}

test.y <- matrix(nrow = length(test.x), ncol = 2)
dimnames(test.y)[[2]] <- list("test.x", "test.y")
for (n in 1:nrow(phi.test.x)) {
  test.y[n,"test.x"] <- test.x[n]
  test.y[n,"test.y"] <- t(phi.test.x[n,]) %*% wbayes
}

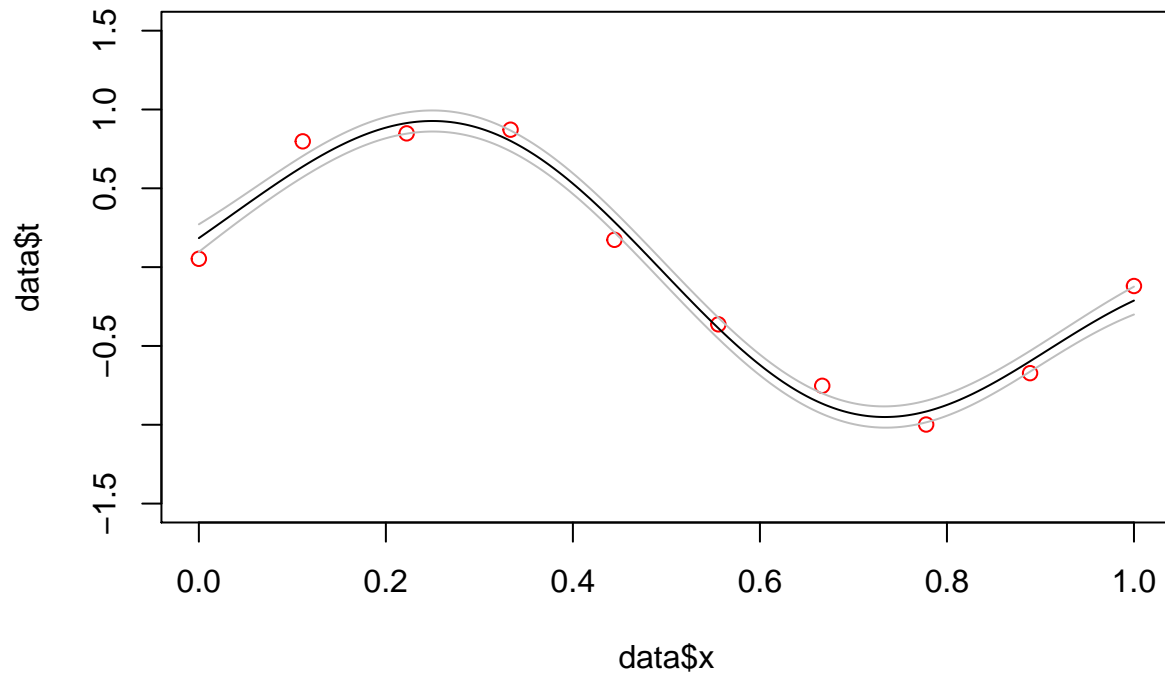
sds <- rep(0, nrow(phi.test.x))
Qbayesinv <- solve(Qbayes)
for (row_idx in 1:nrow(phi.test.x)) {
  sds[row_idx] <- sqrt(t(phi.test.x[row_idx,]) %*% Qbayesinv %*% phi.test.x[row_idx,])
}

return(list(test.y, sds, Qbayes, wbayes))
}

result <- precision.bayes(data, x, M, basis_type, delta, q)
test.ys <- result[1][[1]]
sds <- result[2][[1]]
Qbayes <- result[3][[1]]
wbayes <- result[4][[1]]

plot(data$x, data$t, ylim = range(-1.5:1.5), col = 'red')
lines(x = test.ys[, "test.x"], y = test.ys[, "test.y"])
lines(x = test.ys[, "test.x"], y = (test.ys[, "test.y"] + sds), col = 'grey')
lines(x = test.ys[, "test.x"], y = (test.ys[, "test.y"] - sds), col = 'grey')

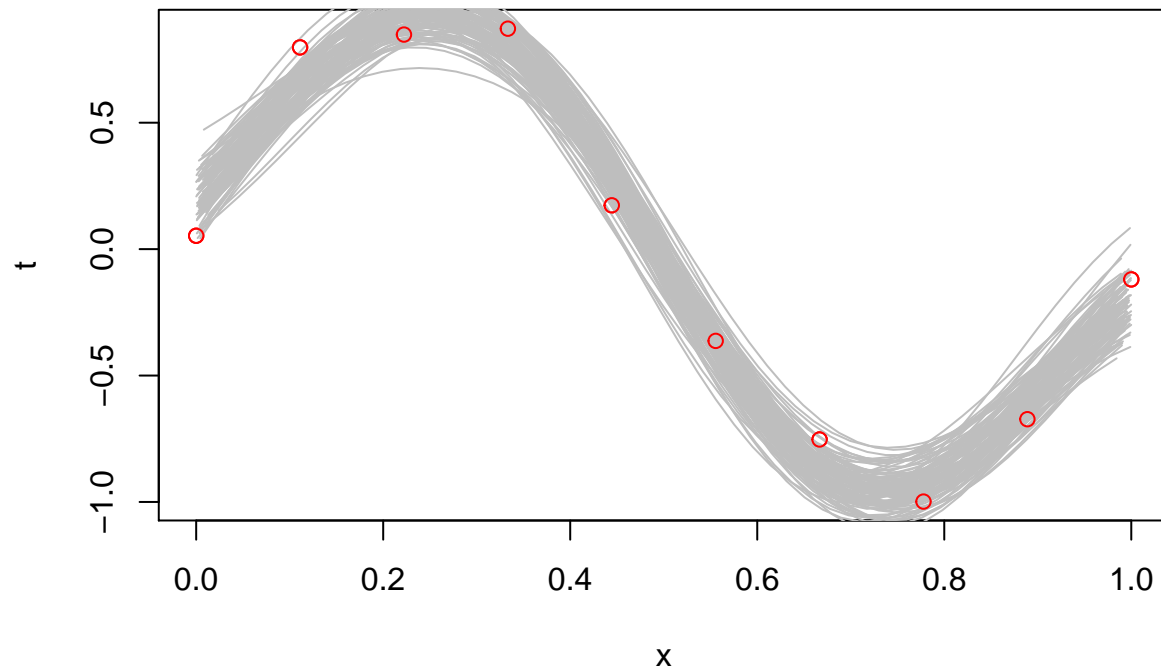
```



```

plot(data, col = 'red')
nsims <- 100
mins <- rep(0, nsims)
maxs <- rep(0, nsims)
for (sim in 1:nsims) {
  # generate random draws of  $N(w, cov)$  and plot random  $x$ s evaluated for each
  # Procedure:
  # 1) generate a random draw of  $N(w, cov)$  -> this becomes a new (random)  $w$  (from the same distribution)
  wbayesrand <- mvrnorm(n = 1, wbayes, Sigma = solve(Qbayes))
  # 2) generate 100 random  $x$ s to calculate  $\phi(x, \dots)$ 
  xs <- runif(100)
  # 3) generate their  $y$ 's as  $t(\phi(x)) \%*\% wbayesrand$ 
  ys <- rep(0, length(xs))
  for (n in 1:length(xs)) {
    testphi <- c(1, phi(xs[n], M, basis_type))
    ys[n] <- t(testphi) \%*\% wbayesrand
  }
  mins[sim] <- min(ys)
  maxs[sim] <- max(ys)
  lines(predict(splines::interpSpline(xs, ys)), col = 'grey')
}
points(data, col = 'red')

```



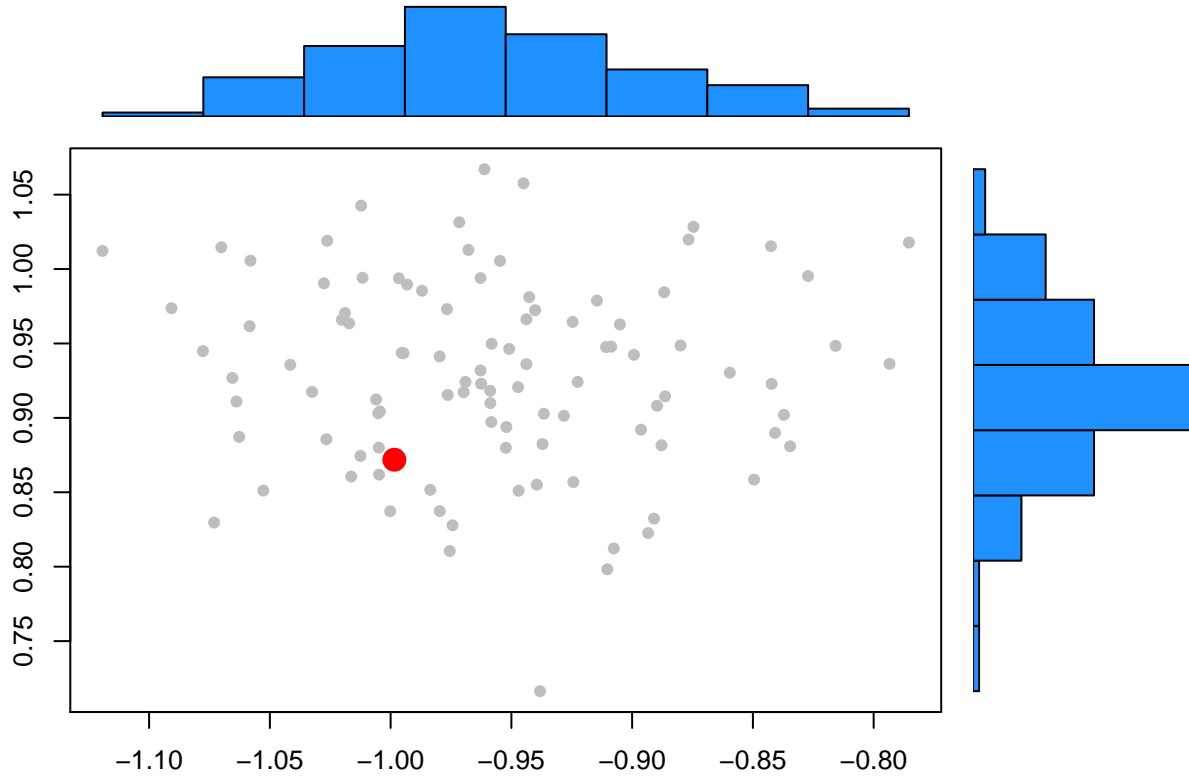
```

observed_min <- min(data$t)
observed_max <- max(data$t)

scatterhist <- function(x, y, x1, y1, xlab="", ylab=""){
  zones=matrix(c(2,0,1,3), ncol=2, byrow=TRUE)
  layout(zones, widths=c(4/5,1/5), heights=c(1/5,4/5))
  xhist = hist(x, plot=FALSE)
  yhist = hist(y, plot=FALSE)
  top = max(c(xhist$counts, yhist$counts))
  par(mar=c(3,3,1,1))
  plot(x,y, col = 'gray', pch = 16)
  points(x1, y1, col = 'red', pch = 16, cex = 2)
  par(mar=c(0,3,1,1))
  barplot(xhist$counts, axes=FALSE, ylim=c(0, top), space=0, col = 'dodgerblue')
  par(mar=c(3,0,1,1))
  barplot(yhist$counts, axes=FALSE, xlim=c(0, top), space=0, horiz=TRUE, col = 'dodgerblue')
  par(oma=c(3,3,0,0))
  mtext(xlab, side=1, line=1, outer=TRUE, adj=0,
        at=.8 * (mean(x) - min(x))/(max(x)-min(x)))
  mtext(ylab, side=2, line=1, outer=TRUE, adj=0,
        at=(.8 * (mean(y) - min(y))/(max(y) - min(y))))
}

scatterhist(mins, maxs, observed_min, observed_max)

```



**2.1** Show that the mean of the predictive distribution at an input location  $\mathbf{x}$ ,  $\phi(\mathbf{x})^T w_{Bayes}$  can be represented as

$$\sum_{n=1}^N q\phi(\mathbf{x})^T Q^{-1} \phi(x_n) t_n$$

therefore, as a linear combination of the training outputs  $t_n$ .

We know that the predictive distribution is given by (slide 3):

$$t_{N+1}|t, X, x_{N+1} \sim N(\phi(x_{N+1})^T w_{Bayes}, \phi(x_{N+1})^T Q^{-1} \phi(x_{N+1}) + q^{-1})$$

From this we know the mean of the predictive distribution at  $\mathbf{x}$  is:  $\phi(\mathbf{x})^T w_{Bayes}$

and the equation is given by  $w_{Bayes} = qQ^{-1}\phi^T t$

At  $\mathbf{x}$ ,  $\phi^T t$  is

$$\sum_{n=1}^N \phi(x_n) t_n$$

So we get the final result that the mean of the predictive distribution at  $\mathbf{x}$  is:

$$\sum_{n=1}^N q\phi(\mathbf{x})^T Q^{-1} \phi(x_n) t_n$$

**2.2** The weights above is a quantification of the similarity (in feature space) of the test input  $\mathbf{x}$  and the training inputs  $x_n$ . In particular, let  $k(x, y) := q\phi(x)^T Q^{-1} \phi(y)$ . Then, show that the weight of  $t_n$  is  $k(x, x_n)$

We know from 2.1 that the mean of the predictive distribution is given by:

$$\bar{x} = \sum_{n=1}^N q\phi(x)^T Q^{-1} \phi(x_n) t_n$$

We use this to solve for  $t_n$  and get

$$\frac{\bar{x}}{\sum_{n=1}^N q\phi(x)^T Q^{-1} \phi(x_n) t} = t_n$$

We find the summation in the denominator acts as a weight of the mean at  $\mathbf{x}$ .

**3. Let  $\mathbf{K}$  be the matrix with  $(n,k)$  element  $k(x_n, x_k)$ . Show that  $K = q\phi Q^{-1} \phi^T$**

**REVIEW**

We know that:

$$k(x, y) := q\phi(x)^T Q^{-1} \phi(y)$$

So we can exchange  $x$  and  $y$  for  $x_n$  and  $x_k$

$$k(x_n, x_k) := q\phi(x_n)^T Q^{-1} \phi(x_k)$$

The matrix produced for all  $n$  and all  $k$ , e.g. every row of  $\mathbf{x}$  ( $n$ ) and every column of  $\mathbf{x}$  ( $k$ ) is exactly the matrix  $\mathbf{K}$ :

$$q\phi(x_n)^T Q^{-1} \phi(x_k) = q\phi Q^{-1} \phi^T = K$$

**4. Notice that, by expanding  $\mathbf{Q}$ ,**

$$K = q\phi(\delta I + q\phi^T \phi)^{-1} \phi^T = \phi(\lambda I + \phi^T \phi)^{-1} \phi^T.$$

**Show that when  $\lambda = 0$ , and provided  $\phi^T \phi$  is invertible,  $\mathbf{K}$  is precisely the “hat” matrix of linear regression. Therefore, the matrix of kernel weights provides a Bayesian version of such matrix. We will revisit this later in the course.**

The “Hat” matrix from linear regression is given by:

$$H = \phi(\phi^T \phi)^{-1} \phi^T$$

When  $\lambda = 0$ , the first term in parenthesis:

$$K = \phi(\lambda I + \phi^T \phi)^{-1} \phi^T$$

is a matrix of zeroes and thus when multiplied by  $\phi$  is still a matrix of zeros so we remove it from the equation and the equation reduces to:

$$K = H = \phi(\phi^T \phi)^{-1} \phi^T$$