

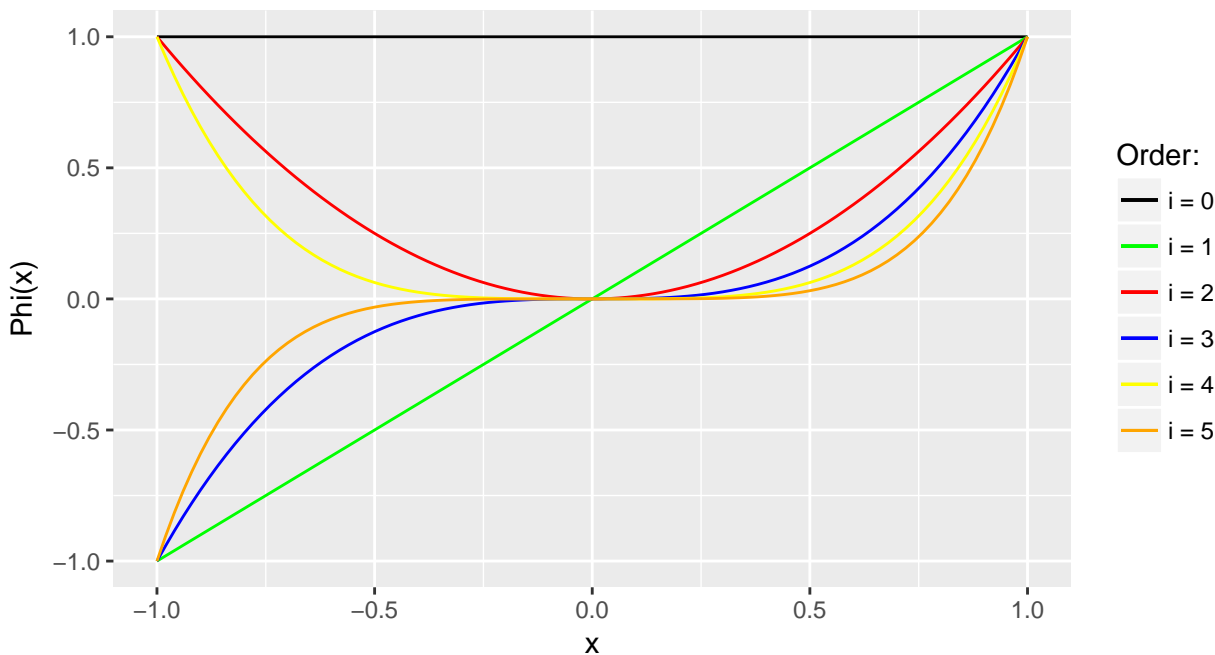
# Problem Solutions

## Chapter 4

*Pierre Paquay*

### Problem 4.1

Below we plot the monomials of order  $i$ ,  $\phi_i(x) = x^i$ .



It is easy to see that as the order  $i$  increases, so does the complexity of the curve (in the sense that it is able to fit more complex target functions).

### Problem 4.2

We may write

$$\begin{aligned} h(x) &= \begin{pmatrix} 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} L_0(x) \\ L_1(x) \\ L_2(x) \end{pmatrix} \\ &= L_0(x) - L_1(x) + L_2(x) \\ &= \frac{3}{2}x^2 - x + \frac{1}{2} \end{aligned}$$

So we get a degree 2 polynomial.

### Problem 4.3

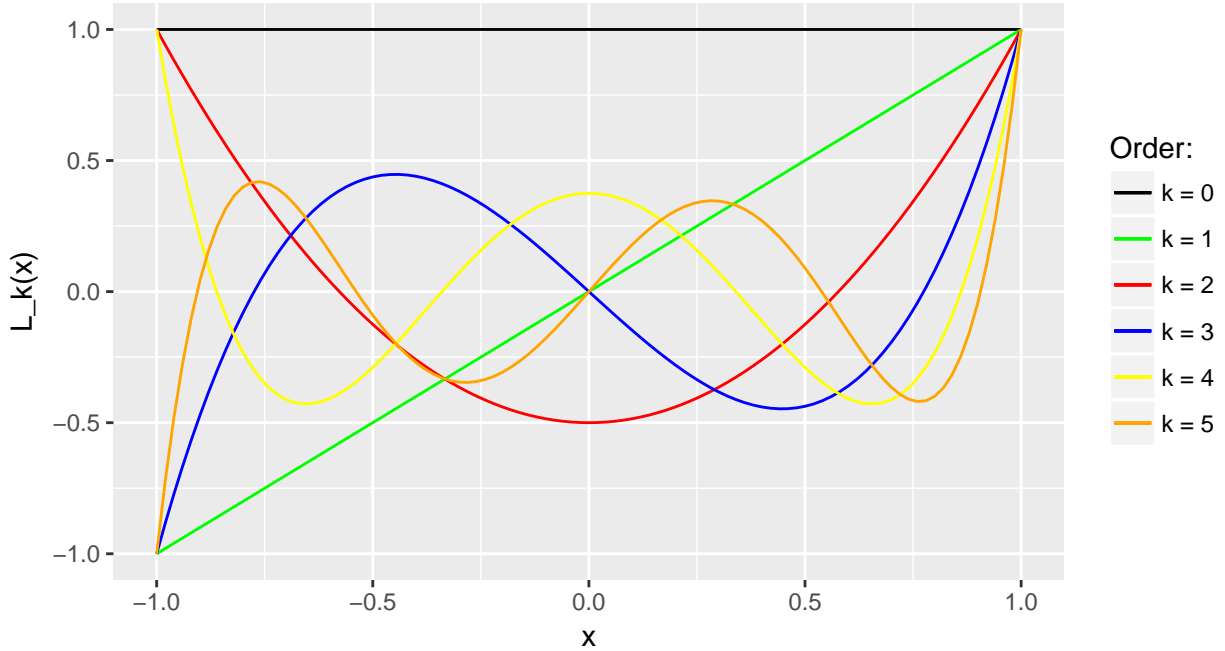
(a) We use the recursive definition of the Legendre polynomials to develop an algorithm to compute  $L_k(x)$  given  $x$ .

```

Legendre <- function(x, k) {
  if (k == 0)
    return(1)
  if (k == 1)
    return(x)
  else
    return(((2 * k - 1) / k) * x * Legendre(x, k - 1) - ((k - 1) / k) * Legendre(x, k - 2))
}

```

Now we plot the first six Legendre polynomials below.



(b) We prove this fact by induction. For  $k = 0$ , we have  $L_0(x) = 1$  which is a monomial of order 0. For  $k = 1$ , we have  $L_1(x) = x$  which is a monomial of order 1. Now we assume that the result is true for all order less than  $k + 2$ , and we will prove it is still true for order  $k + 2$ . We will also assume that  $k$  is even (the case when it is odd is proved in the same way). We have

$$\begin{aligned}
L_{k+2}(x) &= \underbrace{\frac{2k+3}{k+2}}_{=c_1} x \cdot \underbrace{L_{k+1}(x)}_{=a_{k+1}x^{k+1}+a_{k-1}x^{k-1}+\dots+a_1x} - \underbrace{\frac{k+1}{k+2}}_{=c_0} \cdot \underbrace{L_k(x)}_{=b_kx^k+b_{k-2}x^{k-2}+\dots+b_0} \\
&= c_1 a_{k+1} x^{k+2} + (c_1 a_{k-1} - c_0 b_k) x^k + \dots + (c_1 a_1 - c_0 b_2) x^2 - c_0 b_0
\end{aligned}$$

which is actually a linear combination of monomials all of even order with highest order  $k + 2$ . In this case we obviously have

$$L_k(-x) = (-1)^k L_k(x).$$

(c) Once again we proceed by induction on  $k$ . For  $k = 1$ , we have

$$\frac{x^2 - 1}{1} \underbrace{\frac{dL_1(x)}{dx}}_{=1} = x^2 - 1 = xL_1(x) - L_0(x).$$

Now we assume that the result is true for all order less than  $k$ , and we prove it is still true for  $k$ . We have that

$$\begin{aligned}
& \frac{x^2-1}{k} \frac{dL_k(x)}{dx} \\
&= \frac{x^2-1}{k} \left( \frac{2k-1}{k} L_{k-1}(x) + \frac{(2k-1)x}{k} \frac{dL_{k-1}(x)}{dx} - \frac{k-1}{k} \frac{dL_{k-2}(x)}{dx} \right) \\
&= \frac{(x^2-1)(2k-1)}{k^2} L_{k-1}(x) + \frac{(2k-1)(k-1)x}{k^2} \underbrace{\frac{x^2-1}{k-1} \frac{dL_{k-1}(x)}{dx}}_{=xL_{k-1}(x)-L_{k-2}(x)} - \frac{(k-1)(k-2)}{k^2} \underbrace{\frac{x^2-1}{k-2} \frac{dL_{k-2}(x)}{dx}}_{=xL_{k-2}(x)-L_{k-3}(x)} \\
&= \frac{(2k-1)(kx^2-1)}{k^2} L_{k-1}(x) - \frac{(k-1)(3kx-3x)}{k^2} L_{k-2}(x) + \frac{(k-1)(k-2)}{k^2} L_{k-3}(x) \\
&= x \underbrace{\left( \frac{2k-1}{k} xL_{k-1}(x) - \frac{k-1}{k} L_{k-2}(x) \right)}_{=L_k(x)} - \frac{2k-1}{k^2} L_{k-1}(x) - \frac{(k-1)^2}{k^2} \underbrace{\left( \frac{2k-3}{k-1} xL_{k-2}(x) - \frac{k-2}{k-1} L_{k-3}(x) \right)}_{=L_{k-1}(x)} \\
&= xL_k(x) - \frac{(2k-1) + (k-1)^2}{k^2} L_{k-1}(x) \\
&= xL_k(x) - L_{k-1}(x).
\end{aligned}$$

(d) We may write that

$$\begin{aligned}
\frac{d}{dx} \left( (x^2-1) \frac{dL_k(x)}{dx} \right) &= \frac{d}{dx} \left( xkL_k(x) - kL_{k-1}(x) \right) \\
&= kL_k(x) + xk \frac{dL_k(x)}{dx} - k \frac{dL_{k-1}(x)}{dx} \\
&= kL_k(x) + \frac{k^2 x^2}{x^2-1} L_k(x) - \frac{k^2 x}{x^2-1} L_{k-1}(x) - \frac{k(k-1)}{x^2-1} xL_{k-1}(x) + \frac{k(k-1)}{x^2-1} L_{k-2}(x) \\
&= \frac{kx^2-k+k^2 x^2}{x^2-1} L_k(x) - \frac{k}{x^2-1} [(2k-1)xL_{k-1}(x) - (k-1)L_{k-2}(x)] \\
&= \frac{kx^2-k+k^2 x^2}{x^2-1} L_k(x) - \frac{k^2}{x^2-1} L_k(x) \\
&= \frac{k}{x^2-1} [(x^2-1) + kx^2 - k] L_k(x) \\
&= k(k+1)L_k(x).
\end{aligned}$$

(e) We will first consider the case where  $l \neq k$ . We have that

$$\frac{d}{dx} \left( (1-x^2) \frac{dL_k(x)}{dx} \right) + k(k+1)L_k(x) = 0$$

and

$$\frac{d}{dx} \left( (1-x^2) \frac{dL_l(x)}{dx} \right) + l(l+1)L_l(x) = 0,$$

now we multiply the first identity by  $L_l(x)$  and the second by  $L_k(x)$ , if we subtract and integrate the two identities obtained, we get

$$\int_{-1}^1 L_l(x) \frac{d}{dx} \left( (1-x^2) \frac{dL_k(x)}{dx} \right) - L_k(x) \frac{d}{dx} \left( (1-x^2) \frac{dL_l(x)}{dx} \right) dx + [k(k+1) - l(l+1)] \int_{-1}^1 L_k(x) L_l(x) dx = 0.$$

Using integration by parts for the first integral, we get

$$\underbrace{\left( L_l(x)(1-x^2) \frac{dL_k(x)}{dx} \right) \Big|_{-1}^1}_{=0} - \underbrace{L_k(x)(1-x^2) \frac{dL_l(x)}{dx} \Big|_{-1}^1}_{=0} - \underbrace{\int_{-1}^1 \frac{dL_l(x)}{dx} (1-x^2) \frac{dL_k(x)}{dx} - \frac{dL_k(x)}{dx} (1-x^2) \frac{dL_l(x)}{dx} dx}_{=0} = 0.$$

Finally, we obtain

$$\int_{-1}^1 L_k(x) L_l(x) dx = 0.$$

Now, we consider the case where  $l = k$ . We have that

$$\begin{aligned} A_k = \int_{-1}^1 L_k^2(x) dx &= \frac{2k-1}{k} \int_{-1}^1 x L_k(x) L_{k-1}(x) dx - \frac{k-1}{k} \underbrace{\int_{-1}^1 L_k(x) L_{k-2}(x) dx}_{=0} \\ &= \frac{(2k-1)(k+1)}{k(2k+1)} \underbrace{\int_{-1}^1 L_{k+1}(x) L_{k-1}(x) dx}_{=0} + \frac{(2k-1)k}{k(2k+1)} \int_{-1}^1 L_{k-1}^2(x) dx \\ &= \frac{2k-1}{2k+1} \int_{-1}^1 L_{k-1}^2(x) dx. \end{aligned}$$

Finally, we are able to obtain that

$$\begin{aligned} A_k &= \frac{2k-1}{2k+1} A_{k-1} \\ &= \frac{2k-1}{2k+1} \cdot \frac{2k-3}{2k-1} A_{k-2} \\ &= \frac{2k-1}{2k+1} \cdot \frac{2k-3}{2k-1} \cdots \frac{3}{5} \frac{1}{3} \underbrace{A_0}_{=2} \\ &= \frac{2}{2k+1}. \end{aligned}$$

## Problem 4.4

The following code is an implementation of the experimental framework used to study various aspects of overfitting.

```
Legendre2 <- function(x, q) {
  vec <- rep(NA, q + 1)
  for (k in 0:q) {
    vec[k + 1] <- (choose(q, k))^2 * (x - 1)^(q - k) * (x + 1)^k / 2^q
  }

  return(sum(vec))
}

f <- function(x, Qf, aq) {
  Lq <- rep(0, Qf + 1)
  for (k in 0:Qf) {
```

```

    Lq[k + 1] <- Legendre2(x, k)
  }

  return(sum(aq * Lq))
}
f <- Vectorize(f, vectorize.args = "x")

experiment <- function(Qf, N, sigma, Ntest) {
  aq <- rnorm(Qf + 1)
  norm <- rep(0, Qf + 1)
  for (q in 0:Qf)
    norm[q + 1] <- 1 / (2 * q + 1)
  norm_fac <- 1 / sqrt(sum(norm))
  aq <- norm_fac * aq

  xn <- runif(N, min = -1, max = 1)
  eps <- rnorm(N)
  yn <- f(xn, Qf, aq) + sigma * eps
  D <- data.frame(x = xn, y = yn)

  y <- D$y
  D2 <- data.frame(x = D$x, x_sq = D$x^2)
  Z2 <- as.matrix(cbind(1, D2))
  Z2_cross <- solve(t(Z2) %*% Z2) %*% t(Z2)
  w2 <- as.vector(Z2_cross %*% y)
  D10 <- data.frame(x = D$x, x_sq = D$x^2, x_cub = D$x^3, x_quad = D$x^4,
    x_quint = D$x^5, x_six = D$x^6, x_seven = D$x^7,
    x_eight = D$x^8, x_nine = D$x^9, x_ten = D$x^10)
  Z10 <- as.matrix(cbind(1, D10))
  Z10_cross <- solve(t(Z10) %*% Z10) %*% t(Z10)
  w10 <- as.vector(Z10_cross %*% y)

  x <- runif(Ntest, min = -1, max = 1)
  eps <- rnorm(Ntest)
  y <- f(x, Qf, aq) + sigma * eps
  Dtest <- data.frame(x = x, y = y)
  Eout2 <- mean((as.matrix(cbind(1, Dtest$x, Dtest$x^2)) %*% w2 - Dtest$y)^2)
  Eout10 <- mean((as.matrix(cbind(1, Dtest$x, Dtest$x^2, Dtest$x^3, Dtest$x^4,
    Dtest$x^5, Dtest$x^6, Dtest$x^7, Dtest$x^8,
    Dtest$x^9, Dtest$x^10)) %*% w10 - Dtest$y)^2)

  return(c(Eout2, Eout10))
}

```

(a) To normalize  $f$ , we compute  $\mathbb{E}_{a,x}[f^2]$  as follows,

$$\begin{aligned}
\mathbb{E}_{a,x}[f^2] &= \mathbb{E}_x[\mathbb{E}_{a|x}[f^2|x]] \\
&= \mathbb{E}_x[\underbrace{\text{Var}_{a|x}[f]}_{=1} + (\underbrace{\mathbb{E}_{a|x}[f]}_{=0})^2] \\
&= \sum_q L_q^2(x) \underbrace{\text{Var}_{a|x}[a_q]}_{=1} = \sum_q L_q(x) \underbrace{\mathbb{E}_{a|x}[a_q]}_{=0} \\
&= \sum_{q=0}^{Q_f} \mathbb{E}_x[L_q^2(x)].
\end{aligned}$$

Moreover, we may write that

$$\mathbb{E}_x[L_q^2(x)] = \frac{1}{2} \int_{-1}^1 L_q^2(x) dx = \frac{1}{2q+1},$$

with which we can conclude that

$$\mathbb{E}_{a,x}[f^2] = \sum_{q=0}^{Q_f} \frac{1}{2q+1}.$$

This means that, to normalize  $f$ , we have to multiply each coefficient  $a_q$  by the constant factor  $1/\sqrt{\sum_q \frac{1}{2q+1}}$ . Obviously, if the signal  $f$  is normalized to  $\mathbb{E}[f^2] = 1$ , this implies that the noise level  $\sigma^2$  is automatically calibrated to the signal level.

(b) To obtain  $g_2$  and  $g_{10}$ , we first transform the original data  $x \in \mathcal{X}$  with a second (resp. tenth) order transformation  $z = \Phi_2(x) \in \mathcal{Z}_2$  (resp.  $z = \Phi_{10}(x) \in \mathcal{Z}_{10}$ ). Then, we find the best linear fit for the data in  $\mathcal{Z}_2$ -space (resp.  $\mathcal{Z}_{10}$ -space) to find  $\tilde{g}_2 = \tilde{w}^T z$  (resp.  $\tilde{g}_{10} = \tilde{w}^T z$ ). And finally, we get the best fit in  $\mathcal{X}$ -space

$$g_2(x) = \tilde{g}_2(\Phi_2(x)) = \tilde{w}^T \Phi_2(x) \text{ (resp. } g_{10}(x) = \tilde{g}_{10}(\Phi_{10}(x)) = \tilde{w}^T \Phi_{10}(x)).$$

(c) To compute analytically  $E_{out}$  for a given  $g_{10}$  we have to compute

$$E_{out}(g_{10}) = \mathbb{E}_{x,y}[(g_{10}(x) - y(x))^2] = \mathbb{E}_{x,y}[(g_{10}(x) - f(x) - \sigma\epsilon)^2] = \mathbb{E}_x[\mathbb{E}_{y|x}[(g_{10}(x) - f(x) - \sigma\epsilon)^2|x]].$$

(d) Below we plot the extent of overfitting depending on certain parameters of the learning problem. In the first plot, we fix  $Q_f = 20$  to study the stochastic noise.

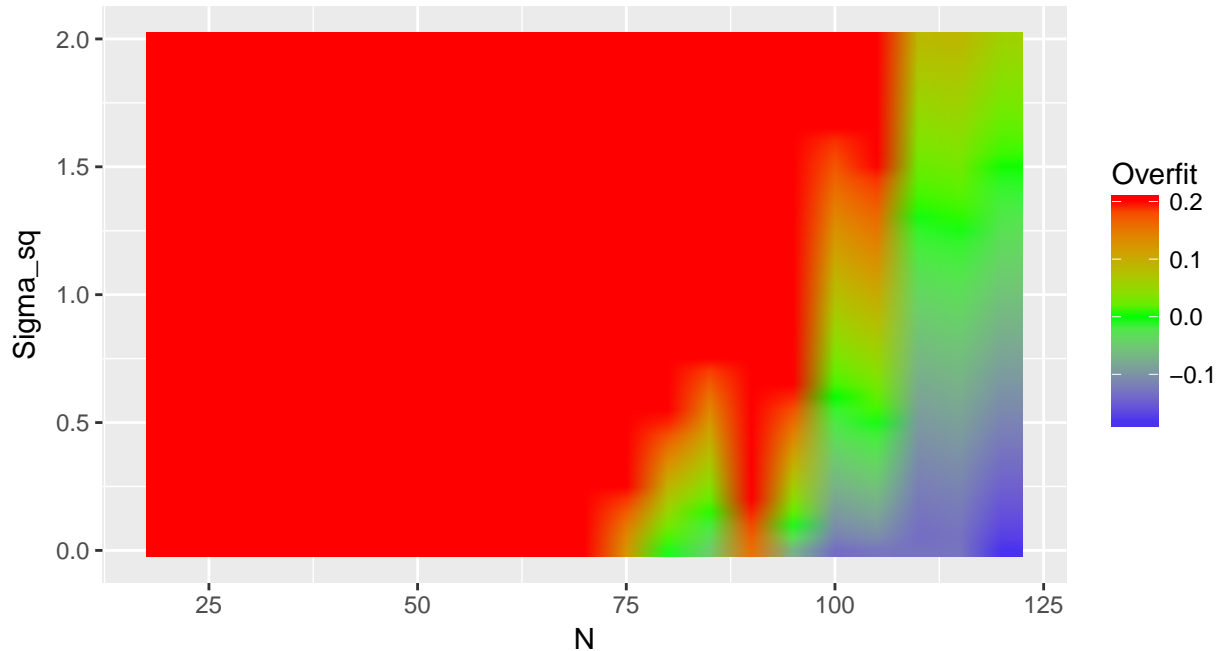
```

# Grid search with Qf = 20
Nexp <- 1000
grid <- expand.grid(N = seq(20, 120, by = 5), sigma_sq = seq(0, 2, by = 0.05))
E_out_Overfit <- foreach(i = 1:nrow(grid), .combine = "rbind") %dopar% {
  set.seed(1975)
  Eout_H2 <- numeric(Nexp)
  Eout_H10 <- numeric(Nexp)
  for (n in 1:Nexp) {
    tmp <- experiment(Qf = 20, grid$N[i], sqrt(grid$sigma[i]), Ntest = 100)
    Eout_H2[n] <- tmp[1]
    Eout_H10[n] <- tmp[2]
  }
  c(mean(Eout_H2), mean(Eout_H10))
}
Eout <- cbind(grid, E_out_Overfit)
colnames(Eout) <- c("N", "sigma_sq", "Eout_H2", "Eout_H10")
Eout["Overfit"] <- Eout$Eout_H10 - Eout$Eout_H2
Eout$Overfit <- ifelse(Eout$Overfit > 0.2, 0.2, Eout$Overfit)

```

```
Eout$Overfit <- ifelse(Eout$Overfit < -0.2, -0.2, Eout$Overfit)
```

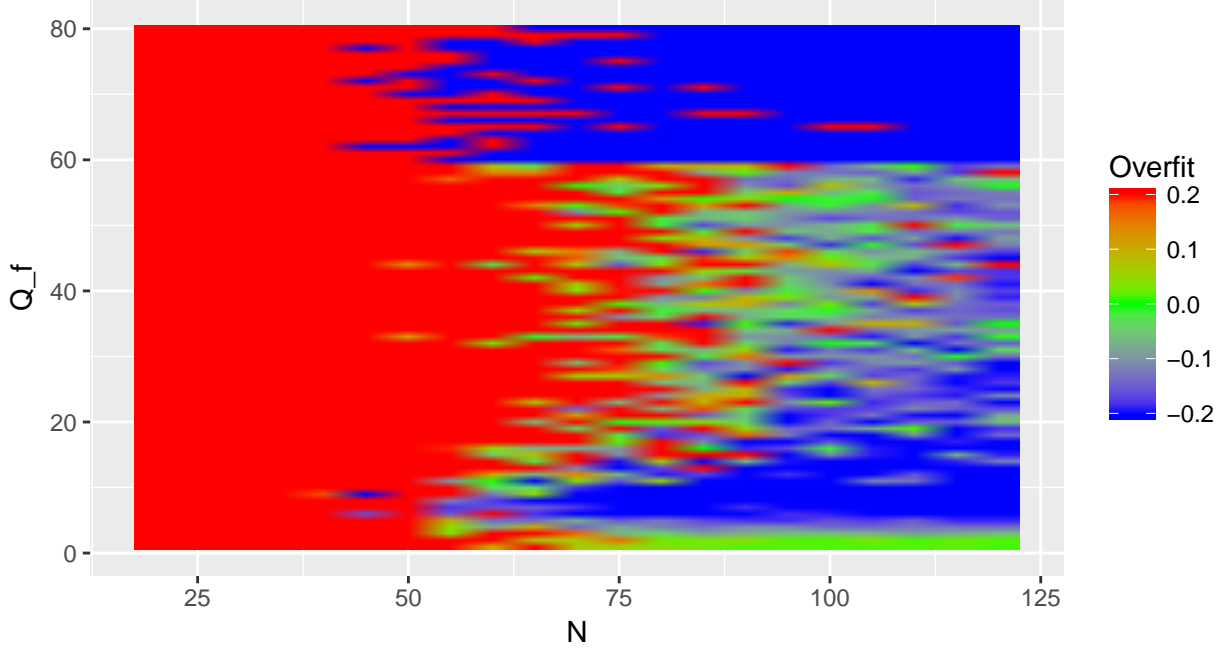
```
ggplot(Eout, aes(N, sigma_sq, fill = Overfit)) + geom_raster(interpolate = TRUE) +
  xlab("N") + ylab("Sigma_sq") +
  scale_fill_gradient2(low = "blue", mid = "green", high = "red")
```



In the second plot, we fix  $\sigma^2 = 0.1$  to study the deterministic noise.

```
# grid search with sigma_sq = 0.1
Nexp <- 200
grid <- expand.grid(Qf = seq(1, 80, by = 1), N = seq(20, 120, by = 5))
E_out_Overfit <- foreach(i = 1:nrow(grid), .combine = "rbind") %dopar% {
  set.seed(1975)
  Eout_H2 <- numeric(Nexp)
  Eout_H10 <- numeric(Nexp)
  for (n in 1:Nexp) {
    tmp <- experiment(grid$Qf[i], grid$N[i], sqrt(0.1), Ntest = 10)
    Eout_H2[n] <- tmp[1]
    Eout_H10[n] <- tmp[2]
  }
  c(mean(Eout_H2), mean(Eout_H10))
}
Eout <- cbind(grid, E_out_Overfit)
colnames(Eout) <- c("Qf", "N", "Eout_H2", "Eout_H10")
Eout["Overfit"] <- Eout$Eout_H10 - Eout$Eout_H2
Eout$Overfit <- ifelse(Eout$Overfit > 0.2, 0.2, Eout$Overfit)
Eout$Overfit <- ifelse(Eout$Overfit < -0.2, -0.2, Eout$Overfit)

ggplot(Eout, aes(N, Qf, fill = Overfit)) + geom_raster(interpolate = TRUE) +
  xlab("N") + ylab("Q_f") +
  scale_fill_gradient2(low = "blue", mid = "green", high = "red")
```



(e) We take the average over many experiments because we want estimates of the expected out-of-sample error for a given learning scenario  $(Q_f, N, \sigma)$  using  $\mathcal{H}_2$  and  $\mathcal{H}_{10}$ .

### Problem 4.5

If we consider the following constrained optimization problem

$$\min_w E_{in}(w) \text{ subject to } w^T w \geq C,$$

the theory of Lagrange multipliers tells us that this problem is equivalent to the following unconstrained optimization problem

$$\min_w (E_{in}(w) - \lambda'_C w^T w) ; \lambda'_C \geq 0.$$

If we let  $\lambda_C = -\lambda'_C$ , we get that the original constrained optimization problem is equivalent to minimizing the augmented error

$$E_{aug}(w) = E_{in}(w) + \lambda_C w^T w ; \lambda_C \leq 0.$$

So, we may conclude that the soft order constraint corresponding to this problem is  $w^T w \geq C$ .

### Problem 4.6

(a) We begin by noting that

$$E_{in}(w_{reg}) = \frac{(w_{reg} - w_{lin})^T Z^T Z (w_{reg} - w_{lin}) + y^T (I - H)y}{N} \geq \frac{y^T (I - H)y}{N} = E_{in}(w_{lin}).$$

Now we suppose that  $\|w_{reg}\| > \|w_{lin}\|$ , in this case we may write that

$$E_{aug}(w_{reg}) = E_{in}(w_{reg}) + \lambda \|w_{reg}\|^2 > E_{in}(w_{lin}) + \lambda \|w_{lin}\|^2 = E_{aug}(w_{lin}),$$

which is not possible since  $w_{reg} = \text{argmin}_w E_{aug}(w)$ . So, we may conclude that  $\|w_{reg}\| \leq \|w_{lin}\|$ .

(b) First, we note that if  $v_i$  are eigenvectors with eigenvalues  $\lambda_i$  of a matrix  $A$ , then  $Av_i = \lambda_i v_i$ , and consequently

$$v_i = \lambda_i A^{-1} v_i \Leftrightarrow A^{-1} v_i = \frac{1}{\lambda_i} v_i \Rightarrow A^{-2} v_i = \frac{1}{\lambda_i^2} v_i,$$



which means that  $v_i$  are also eigenvectors of  $A^{-2}$  with eigenvalues  $1/\lambda_i^2$ .

Now, let  $v_i$  be the orthogonal eigenvectors of non-zero eigenvalues  $\lambda_i$  of  $Z^T Z$  (since  $Z^T Z$  is invertible and symmetric). We have that

$$\|w_{reg}\|^2 = y^T Z(Z^T Z + \lambda I)^{-2} Z^T y = u^T (Z^T Z + \lambda I)^{-2} u,$$

and

$$\|w_{lin}\|^2 = y^T Z(Z^T Z)^{-2} Z^T y = u^T (Z^T Z)^{-2} u$$

where  $u = Z^T y$ ; if we let  $V = (v_0, \dots, v_Q)$  be the orthogonal matrix of eigenvectors, we get

$$V^T Z^T Z V = \text{diag}(\lambda_i)$$

and

$$V^T (Z^T Z + \lambda I) V = V^T Z^T Z V + \lambda V^T V = \text{diag}(\lambda_i + \lambda).$$

If we expand  $u$  in the eigenbasis of  $Z^T Z$ , we get that  $u = \sum_i \alpha_i v_i$  and

$$\begin{aligned} \|w_{reg}\|^2 &= \sum_{i,j} \alpha_i \alpha_j v_i^T (Z^T Z + \lambda I)^{-2} v_j \\ &= \sum_{i,j} \alpha_i \alpha_j \frac{1}{(\lambda_i + \lambda)^2} v_i^T v_j \\ &= \sum_i \frac{\alpha_i^2}{(\lambda_i + \lambda)^2} \\ &\leq \sum_i \frac{\alpha_i^2}{\lambda_i^2} = \sum_{i,j} \alpha_i \alpha_j v_i^T (Z^T Z)^{-2} v_j = \|w_{lin}\|^2; \end{aligned}$$

for the above inequality to be true, we have to note that since  $Z^T Z$  is (at least) semi positive definite, its eigenvalues are non-negative.

## Problem 4.7

We begin by noting that

$$Z^T Z = V \Gamma U^T U \Gamma V^T = V \Gamma^2 V^T.$$

Let us first consider the vector  $Hy$ , we have

$$\begin{aligned} Hy &= Z(Z^T Z)^{-1} Z^T y \\ &= U \Gamma V^T (V^T)^{-1} \Gamma^{-2} V^{-1} V \Gamma U^T y \\ &= U U^T y; \end{aligned}$$

moreover, we also have for  $H(\lambda)y$  that

$$\begin{aligned}
H(\lambda)y &= Z(Z^T Z + \lambda I)^{-1} Z^T y \\
&= U \Gamma V^T (V \Gamma^2 V^T + \lambda I)^{-1} V \Gamma U^T y \\
&= U \Gamma V^T [V \underbrace{(\Gamma^2 + \lambda I)}_{=\text{diag}(\sigma_i^2 + \lambda)} V^T]^{-1} V \Gamma U^T y \\
&= U \Gamma V^T (V^T)^{-1} \text{diag}\left(\frac{1}{\sigma_i^2 + \lambda}\right) V^{-1} V \Gamma U^T y \\
&= U \text{diag}\left(\frac{\sigma_i^2}{\sigma_i^2 + \lambda}\right) U^T y.
\end{aligned}$$

Putting all of the above together, we get

$$(I - H(\lambda))y = (I - H)y + (H - H(\lambda))y = (I - H)y + U \text{diag}\left(1 - \frac{\sigma_i^2}{\sigma_i^2 + \lambda}\right) U^T y,$$

and consequently

$$\begin{aligned}
E_{in}(w_{reg}) &= \frac{1}{N} y^T (I - H(\lambda))^2 y \\
&= \frac{1}{N} y^T (I - H(\lambda))^T (I - H(\lambda)) y \\
&= \frac{1}{N} [y^T (I - H)y + 2y^T (I - H)U \text{diag}\left(1 - \frac{\sigma_i^2}{\sigma_i^2 + \lambda}\right) U^T y + y^T U \text{diag}\left(1 - \frac{\sigma_i^2}{\sigma_i^2 + \lambda}\right) U^T U \text{diag}\left(1 - \frac{\sigma_i^2}{\sigma_i^2 + \lambda}\right) U^T y] \\
&= \frac{1}{N} [y^T (I - H)y + y^T U \text{diag}\left(1 - \frac{\sigma_i^2}{\sigma_i^2 + \lambda}\right)^2 U^T y + 2y^T \underbrace{(I - H)U}_{=U - HU = U - U U^T U = 0} \text{diag}\left(1 - \frac{\sigma_i^2}{\sigma_i^2 + \lambda}\right) U^T y] \\
&= E_{in}(w_{lin}) + \frac{1}{N} \sum_i a_i^2 \left(1 - \frac{\sigma_i^2}{\sigma_i^2 + \lambda}\right)^2.
\end{aligned}$$

## Problem 4.8

First, we compute  $\nabla E_{aug}(w)$ , we immediately have

$$\nabla E_{aug}(w) = \nabla E_{in}(w) + 2\lambda w.$$

So the gradient descent update rule becomes

$$w(t+1) \leftarrow w(t) - \eta \nabla E_{aug}(w(t)) = (1 - 2\eta\lambda)w(t) - \eta \nabla E_{in}(w(t)).$$