STAT 527- Assignment 2

Joseph David

May 6 2022

Problem 1.

Proof. (a) Let L be the weighted mean square error:

$$L = \sum_{i \le n} \left(y_i - (\beta_0 + \beta_1 x_i + \cdots + \beta_k x_i^k) \right)^2.$$

Note that

$$\beta^T z_i = \beta_0 + \beta_1 x_i + \cdots + \beta_k x_i^k,$$

and therefore

$$L = \sum_{i \le n} \left(y_i - \beta^T z_i \right)^2.$$

Taking the derivative with respect to β, we get

$$\begin{aligned} \frac{\partial L}{\partial \beta} &= \sum_{i=1}^{n} 2w_i (y_i - \beta^T z_i)(-z_i) \\ &= \sum_{i \le n} 2w_i(\beta^T z_i) z_i - \sum_{i \le n} 2 w_i y_i z_i. \end{aligned}$$

Setting this equal to zero, we must solve

$$\sum_{i \le n} w_i (\beta^T z_i) z_i = \sum_{i \le n} w_i y_i z_i.$$

A linear algebra calculation shows that $\sum_{i \le n} w_i y_i z_i = Z^T W y$. We also see that

$$\begin{aligned} \sum_{i \le n} w_i(\beta^T z_i) z_i &= w_1(\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_1^k) z_1 + \cdots + w_n(\beta_0 + \beta_1 x_n + \cdots + \beta_k x_n^k) z_n \\ &= \beta_0(w_1 z_1 + \cdots + w_n z_n) + \beta_1(w_1 x_1 z_1 + \cdots + w_n x_n z_n) + \cdots \beta_k(w_1 x_1^k z_1 + \cdots + w_n x_n^k z_k) \\ &= Z^T W Z \beta. \end{aligned}$$

Therefore, the above equation yields

$$Z^T W Z \beta = Z^T W y$$

so solving for β we get $\beta = (Z^T W Z)^{-1} Z^T W y$. $\square$

Proof. (b) Let $x_1, \ldots, x_n \sim N(\mu, \sigma^2)$ for some unknown μ, σ. By the Central Limit Theorem, we know that

$$\sqrt{n}(\widehat{\mu} - \mu) \to^D N(0, \sigma^2),$$

so $\widehat{\mu} - \mu = O_p(n^{-1/2})$. The Central Limit Theorem also implies that $\widehat{\sigma}^2 - \sigma^2 = O_P(n^{-1/2})$, and also that $\sqrt{n}(\mu - \mu)$ and $\sqrt{n}(\widehat{\sigma}^2 - \sigma^2)$ converges to a joint multivariate Gaussian distribution.

Let $x_0 \in \mathbb{R}$ be fixed. Note that $\phi_{\widehat{\mu},\widehat{\sigma}}(x_0)$ can be viewed as a function of the sample and thus a function of $\widehat{\mu}$ and $\widehat{\sigma}$. Call this function f, so it is defined by $f(\widehat{\mu}, \widehat{\sigma}) = \phi_{\widehat{\mu},\widehat{\sigma}}(x_0)$. Note also then with this definition we have that $f(\mu, \sigma) = \phi_{\mu,\sigma}(x_0)$ is the truth.

By the multivariate delta method,

$$\sqrt{n}\Big(\phi_{\widehat{\mu},\widehat{\sigma}}(x_0) - \phi_{\mu,\sigma}(x_0)\Big) \to^D N(0, \tau),$$

for some variance τ. Thus,

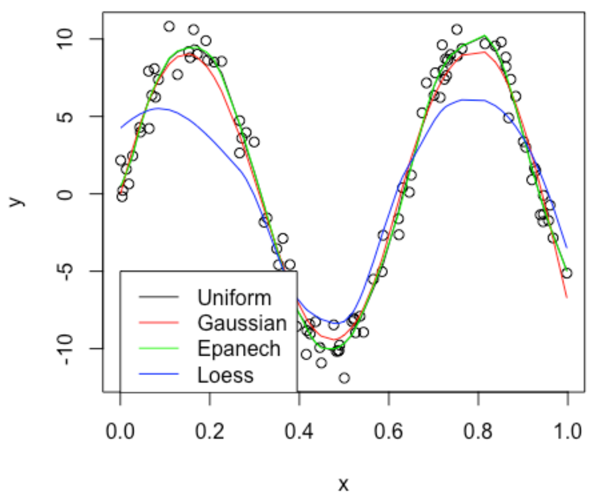
$$\Big(\phi_{\widehat{\mu},\widehat{\sigma}}(x_0) - \phi_{\mu,\sigma}(x_0)\Big)^2 = O_p(1/n).$$

□

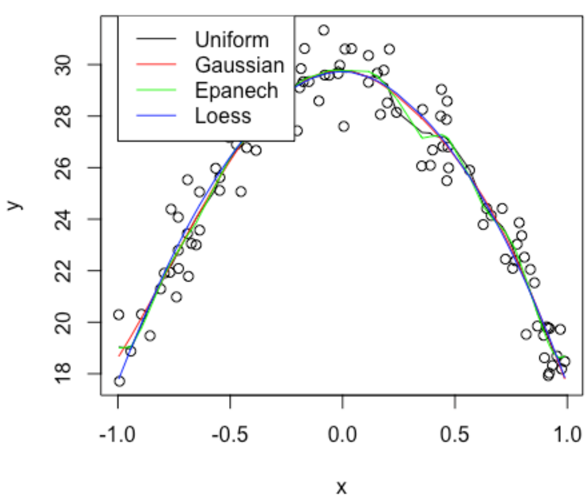
Problem 2. *3.*

The code attached in the appendix shows our implementation. We used the result of problem 1 in order to compute β^{x_0} for each x_0 that we are estimating. We implemented only for kernels Uniform, Gaussian, and Epanechikov.

We generated $n = 100$ data points uniformly on $[0, 1]$ and then generated data by $y_i = 10\sin(10x) + \epsilon_i$, where $\epsilon_i \sim N(0, 1)$. We then used our function `localPoly` to find a fitted model using degree 2 polynomials, bandwidth of 0.1, and using all three kernels. We also fitted a model using `loess` for comparison. Below are the results.



We then tested `localPoly` for the function $y_i = 75G(x_i) + \epsilon_i$ using bandwidth 0.2, where $x_i \sim Unif[-1, 1]$, $\epsilon_i \sim N(0, 1)$ and $G(x)$ denotes the pdf of $N(0, 1)$. Below are the results.



We see that with these choices of bandwidth, the results are good and the models are quite smooth. For the first function, `localPoly` performed significantly better than `loess`, but in the second case they all performed very similarly.

3 R Code

```
#Stat527 HW2

#Problem 3
local_poly <- function(x, y, deg, bandwidth, kernel = 1, new_x = x){
  n = length(new_x)
  f_hat = numeric(n)
  for(k in 1:n){
    x0 = x[k]
    beta_hat = get_beta(x0, x, y, deg, bandwidth, kernel)
    zi = numeric(deg)
    for(j in 1:(deg+1)){zi[j] = x0^(j-1)}
    f_hat[k] = zi %*% beta_hat
  }
  return(f_hat)
}

uniform <- function(x, bandwidth){
  if(abs(x) > bandwidth){
    return(0)
  }
  else{return(1)}
}

get_beta = function(x0,x,y,deg, bandwidth, kernel = 1){
  n = length(x)
  Z = matrix(0,n,deg+1)
 W = matrix(0,n,n)
  for(i in 1:n){
    for(j in 1:(deg+1)){
      Z[i,j] = x[i]^(j-1)
    }
  }
  if(kernel == 1){
  for(i in 1:n){
   W[i,i] = uniform(x[i] - x0, bandwidth)
  }
  }
  if(kernel == 2){
    for(i in 1:n){
      W[i,i] = dnorm(x[i] - x0, mean = 0, sd=bandwidth)
    }
  }
  if(kernel == 3){
    for(i in 1:n){
      if(abs(x0-x[i]) > bandwidth){W[i,i]=0}
```

```
    else{W[i,i] = (3/4)*(1-(abs(x[i]-x0)/bandwidth)^2)}
  }
}
return(solve(t(Z) %*% W %*% Z) %*% t(Z) %*% W %*% y)
```