UNIVERSIDADE DE SÃO PAULO
Programa de Pos-Graduação em Estatística

Alex Monito Nhancololo

Questão da Lista 3:

Seja $(Y_1, X_1), \ldots, (Y_n, X_n)$ uma amostra aleatória de (Y, X) tal que $Y|X = x \sim N(\mu_\theta(x), \sigma_\theta(x)^2)$ e a distribuição de X não tenha informação sobre os parâmetros. Apresente uma rede neural que modele o quantil 75% (3º quartil) de $Y|X = x$. Mostre a aplicação do método para os dados gerados no R:
set.seed(32)
n = 1000
x = sort(runif(n, -4, 4))
$y = 3/(3 + 2 \times abs(x)^3) + \exp(-x^2) + \cos(x) \times \sin(x) + rnorm(n) \times 0.3$
O código deve ser generalizável para qualquer quantil.
Sugestão: reescreva a esperança e variância condicionais em termos do quantil e construa a função de verossimilhança.

Resolução:

Seja $Y_i|X = x_i \sim N(\mu_\theta(x_i), \sigma^2_\theta(x_i))$ para $i = 1, \ldots, n$. Queremos modelar o quantil de ordem τ (onde $\tau = 0.75$), denotado por $q_\tau(x_i; \theta)$. Sabemos que da noção de quantís que:

$$
\begin{aligned}
P(Y_i \leq q_\tau(x_i;\theta)) &= P\left(\underbrace{\frac{Y_i - \mu_\theta(x_i)}{\sigma_\theta(x_i)}}_{z_\tau(x_i,\theta)\sim \mathrm{N}(0,1)} \leq \frac{q_\tau(x_i;\theta) - \mu_\theta(x_i)}{\sigma_\theta(x_i)} \right) = \tau, \quad \text{Nota: aqui há abuso de notação, } z_\tau(x_i,\theta) = Z \\
&\implies \Phi\left(\frac{q_\tau(x_i;\theta) - \mu_\theta(x_i)}{\sigma_\theta(x_i)} \right) = \tau \\
&\implies \frac{q_\tau(x_i;\theta) - \mu_\theta(x_i)}{\sigma_\theta(x_i)} = \Phi^{-1}(\tau) = z_\tau(x_i,\theta) \\
&\implies \mu_\theta(x_i) = q_\tau(x_i;\theta) - z_\tau(x_i,\theta)\sigma_\theta(x_i)
\end{aligned}
$$

Assim, a rede neural estimará diretamente $q_\tau(x_i;\theta)$ e $\sigma_\theta(x_i)$, e $\mu_\theta(x_i)$ será uma função derivada destas saídas. Como referido acima, estou usando um abuso de notação, Z é um quantil e sua distribuição não depende de θ.

Função de Log-Verossimilhança
A função de densidade de probabilidade para uma observação y_i dado x_i é:

$$
\begin{aligned}
f(y_i|x_i;\theta) &= \frac{1}{\sqrt{2\pi}\sigma_\theta(x_i)} \exp\left(-\frac{1}{2} \left[\frac{y_i - \mu_\theta(x_i)}{\sigma_\theta(x_i)} \right]^2 \right) \\
&= \frac{1}{\sqrt{2\pi}\sigma_\theta(x_i)} \exp\left(-\frac{1}{2} \left[\frac{y_i - [q_\tau(x_i;\theta) - z_\tau(x_i,\theta)\sigma_\theta(x_i)]}{\sigma_\theta(x_i)} \right]^2 \right) \\
&= \frac{1}{\sqrt{2\pi}\sigma_\theta(x_i)} \exp\left(-\frac{1}{2} \left[\frac{y_i - q_\tau(x_i;\theta) + z_\tau(x_i,\theta)\sigma_\theta(x_i)}{\sigma_\theta(x_i)} \right]^2 \right) \\
&= \frac{1}{\sqrt{2\pi}\sigma_\theta(x_i)} \exp\left(-\frac{1}{2} \left[\frac{y_i - q_\tau(x_i;\theta) + z_\tau(x_i,\theta)\sigma_\theta(x_i)}{\sigma_\theta(x_i)} \right]^2 \right) \\
&= \frac{1}{\sqrt{2\pi}\sigma_\theta(x_i)} \exp\left(-\frac{1}{2} \left[\frac{y_i - q_\tau(x_i;\theta)}{\sigma_\theta(x_i)} + z_\tau(x_i,\theta) \right]^2 \right) \\
L(\theta) &\overset{ind}{=} \prod_{i=1}^{n} f(y_i|x_i;\theta) \\
&= \prod_{i=1}^{n} \left\{ \frac{1}{\sqrt{2\pi}\sigma_\theta(x_i)} \exp\left[-\frac{1}{2} \left(\frac{y_i - q_\tau(x_i;\theta)}{\sigma_\theta(x_i)} + z_\tau(x_i,\theta) \right)^2 \right] \right\}
\end{aligned}
$$

Aplicando o logaritmo natural para obter a log-verossimilhança $\ell(\theta)$:

$$\begin{aligned}\ell(\theta) &= \sum_{i=1}^{n} \log\left[\frac{1}{\sqrt{2\pi}\sigma_\theta(x_i)} \exp\left(-\frac{1}{2}\left(\frac{y_i - q_\tau(x_i;\theta)}{\sigma_\theta(x_i)} + z_\tau(x_i,\theta)\right)^2\right)\right] \\ &= \sum_{i=1}^{n}\left[-\log(\sqrt{2\pi}) - \log(\sigma_\theta(x_i)) - \frac{1}{2}\left(\frac{y_i - q_\tau(x_i;\theta)}{\sigma_\theta(x_i)} + z_\tau(x_i,\theta)\right)^2\right]\end{aligned}$$

Como $\log(\sqrt{2\pi})$ é constante em relação a θ, podemos ignorar-a e definir a função perda como o negativo da log-verossimilhança:

$$\begin{aligned}J(\theta) &= -\ell(\theta) \\ J(\theta) &\propto \sum_{i=1}^{n}\left[\log(\sigma_\theta(x_i)) + \frac{1}{2}\left(\frac{y_i - q_\tau(x_i;\theta)}{\sigma_\theta(x_i)} + z_\tau(x_i,\theta)\right)^2\right]\end{aligned}$$

Função Escore

Sabe-se que, para utilizar métodos de otimização baseados em gradiente (como SGD ou Adam), calculamos as derivadas parciais da função de perda $J_i(\theta)$ (contribuição da i-ésima observação) em relação às saídas da rede neural, a fim de obter os estimadores dos parâmetros.

Seja, $r_i = \frac{y_i - q_\tau(x_i;\theta)}{\sigma_\theta(x_i)} + z_\tau(x_i,\theta)$ (Note que r_i define um resíduo padronizado deslocado $z_\tau(x_i,\theta)$ unidades), a função de perda e suas respectivas derivadas parciais tornam-se:

$$J_i(\theta) \propto \log(\sigma_\theta(x_i)) + \frac{1}{2}\left(\frac{y_i - q_\tau(x_i;\theta)}{\sigma_\theta(x_i)} + z_\tau(x_i,\theta)\right)^2 = \log(\sigma_\theta(x_i) + \frac{1}{2}r_i^2$$

$$\begin{aligned}\frac{\partial J_i}{\partial q_\tau(x_i;\theta)} &= \frac{\partial}{\partial q_\tau(x_i;\theta)}\left[\frac{1}{2}\left(\frac{y_i - q_\tau(x_i;\theta)}{\sigma_\theta(x_i)} + z_\tau(x_i,\theta)\right)^2\right] = \left(\frac{y_i - q_\tau(x_i;\theta)}{\sigma_\theta(x_i)} + z_\tau(x_i,\theta)\right)\cdot\frac{\partial}{\partial q_\tau(x_i;\theta)}\left(\frac{y_i - q_\tau(x_i;\theta)}{\sigma_\theta(x_i)}\right) \\ &= -\frac{1}{\sigma_\theta(x_i)}\left(\frac{y_i - q_\tau(x_i;\theta)}{\sigma_\theta(x_i)} + z_\tau(x_i,\theta)\right) \\ &= r_i \cdot \left(-\frac{1}{\sigma_\theta(x_i)}\right)\end{aligned}$$

$$\begin{aligned}\frac{\partial J_i}{\partial \sigma_\theta(x_i)} &= \frac{\partial}{\partial \sigma_\theta(x_i)}[\log(\sigma_\theta(x_i))] + \frac{\partial}{\partial \sigma_\theta(x_i)}\left[\frac{1}{2}r_i^2\right] \\ &= \frac{1}{\sigma_\theta(x_i)} + r_i \cdot \frac{\partial r_i}{\partial \sigma_\theta(x_i)}\end{aligned}$$

$$\begin{aligned}\frac{\partial r_i}{\partial \sigma_\theta(x_i)} &= \frac{\partial}{\partial \sigma_\theta(x_i)}\left((y_i - q_\tau(x_i;\theta))\sigma_\theta(x_i)^{-1} + z_\tau(x_i,\theta)\right) \\ &= -(y_i - q_\tau(x_i;\theta))\sigma_\theta(x_i)^{-2} \\ &= -\frac{y_i - q_\tau(x_i;\theta)}{\sigma_\theta(x_i)^2}\end{aligned}$$

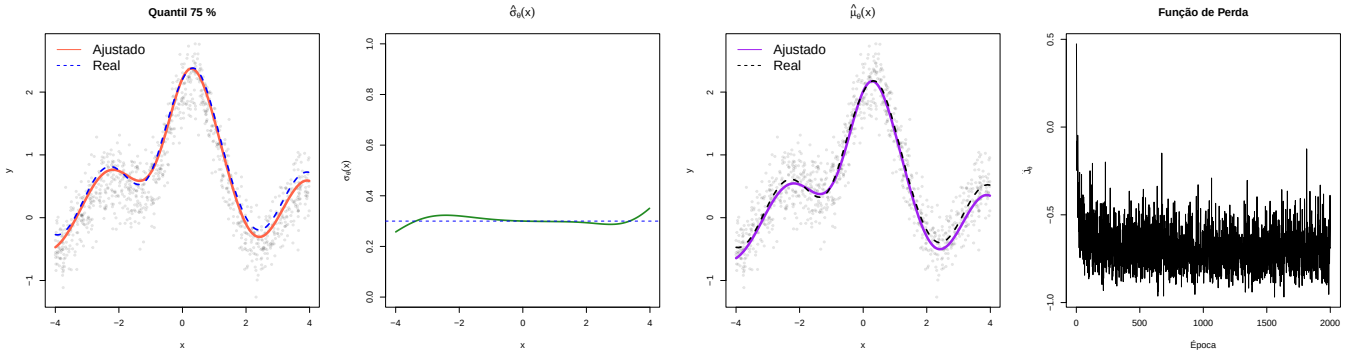
$$\begin{aligned}\frac{\partial J_i}{\partial \sigma_\theta(x_i)} &= \frac{1}{\sigma_\theta(x_i)} + r_i\left(-\frac{y_i - q_\tau(x_i;\theta)}{\sigma_\theta(x_i)^2}\right) \\ &= \frac{1}{\sigma_\theta(x_i)}\left[1 - \left(\frac{y_i - q_\tau(x_i;\theta)}{\sigma_\theta(x_i)} + z_\tau(x_i,\theta)\right)\left(\frac{y_i - q_\tau(x_i;\theta)}{\sigma_\theta(x_i)}\right)\right]\end{aligned}$$

Pela regra da cadeia, a derivada (gradiente) total em relação a um parâmetro θ_j e para o vetor θ da rede são, respectivamente, $\frac{\partial J(\theta)}{\partial \theta_j}$ e $\nabla_\theta J(\theta)$:

$$\begin{aligned}\frac{\partial J(\theta)}{\partial \theta_j} &= \sum_{i=1}^{n}\left[\frac{\partial J_i}{\partial q_\tau(x_i;\theta)}\frac{\partial q_\tau(x_i;\theta)}{\partial \theta_j} + \frac{\partial J_i}{\partial \sigma_\theta^2(x_i)}\frac{\partial \sigma_\theta^2(x_i)}{\partial \theta_j}\right] \\ \nabla_\theta J(\theta) &= \sum_{i=1}^{n}\left(\frac{\partial J_i}{\partial q_\tau(x_i;\theta)}\nabla_\theta q_\tau(x_i;\theta) + \frac{\partial J_i}{\partial \sigma_\theta(x_i)}\nabla_\theta \sigma_\theta(x_i)\right)\end{aligned}$$

Interpretação: A rede neural (Figura 1 capturou bem a não-linearidade do quantil 75% (painel 1) e identificou corretamente a homoscedasticidade dos dados, com a estimativa de $\hat{\sigma}_\theta(x)$ estabilizando próxima ao valor real de 0,3 (painel 2). Essa consistência permitiu a recuperação exata da esperança condicional (painel $\sigma_\theta(x) = 3$), derivada analiticamente das saídas da rede. Por fim, o comportamento da função de perda ajustada $J(\theta)$ (painel 4) demonstra uma convergência rápida e estável, indicando um treinamento bem-sucedido.

Figura 1: Resultados da aplicação da Rede Neural para a estimação do quantil condicional $\tau = 0,75$ no modelo $Y|X = x \sim N(\mu_\theta(x), \sigma^2_\theta(x))$. ($1^o$ Painel) Curva do quantil estimado $\hat{q}_{0,75}(x;\hat{\theta})$ (linha sólida) e o quantil teórico verdadeiro (linha tracejada), sobrepostos à dispersão dos dados observados. (2^o Painel) Função de desvio padrão estimada $\hat{\sigma}_{\hat{\theta}}(x)$ (linha sólida) vs desvio padrão real constante $\sigma = 0,3$ (linha tracejada). (3^o Painel) Estimativa da média condicional $\hat{\mu}_{\hat{\theta}}(x) = \hat{q}_{0,75} - z_{0,75}\hat{\sigma}$ (linha sólida) vs função média verdadeira $f(x)$ (linha tracejada). (4^o Painel) Convergência do modelo: trajetória da função de perda $\hat{J}(\theta)$ (negativo da log-verossimilhança média) ao longo das 2000 épocas de treinamento via otimização estocástica (Adam).



Pseudocódigo do Algoritmo de Treinamento

O Algoritmo 1 descreve o procedimento de estimação dos parâmetros θ. A notação $\hat{q}_j$ e $\hat{\sigma}_j$ representa, respectivamente, as estimativas $q_\tau(x_j;\theta)$ e $\sigma_\theta(x_j)$ para a j-ésima amostra.

Algorithm 1: Otimização via Máxima Verossimilhança Reparametrizada

Data: Conjunto de treinamento $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, Quantil alvo $\tau \in (0,1)$, taxa de aprendizado η, épocas E, tamanho do batch B.
Result: Vetor de parâmetros otimizados $\hat{\theta}$.

// 1. Pré-cálculo da constante (independe dos dados)
$z_\tau \leftarrow \Phi^{-1}(\tau)$ // Para $\tau = 0.75$, $z_\tau \approx 0.674$
// 2. Inicialização
$\theta \leftarrow$ InicializarPesos($\mathcal{N}(0, 0.01)$)
Definir otimizador Adam com taxa η
// 3. Loop de Otimização Estocástica
for *época* $e = 1$ **to** E **do**
  Permutar aleatoriamente $\mathcal{D}$
  for *cada minibatch* $\mathcal{B} \subset \mathcal{D}$ **do**
    // Passo Forward
    for *cada amostra* $j \in \mathcal{B}$ **do**
      $(\hat{q}_j, h_j) \leftarrow$ QuantileNet($x_j;\theta$) // $\hat{q}_j$: quantil estimado; h_j: log-sigma bruto
      $\hat{\sigma}_j \leftarrow \exp(h_j)$ // Função Exponencial para garantir $\sigma > 0$
    end
    // Cálculo da Perda Média no Batch ($J(\theta)$)
    $J_\mathcal{B}(\theta) \leftarrow \frac{1}{|\mathcal{B}|} \sum_{j \in \mathcal{B}} \left[\ln(\hat{\sigma}_j) + \frac{1}{2} \left(\frac{y_j - \hat{q}_j}{\hat{\sigma}_j} + z_\tau \right)^2 \right]$
    // Passo Backward e Atualização
    $g \leftarrow \nabla_\theta J_\mathcal{B}(\theta)$ // Gradiente via diferenciação automática
    $\theta \leftarrow \text{Adam}(\theta, g, \eta)$
  end
end
return $\hat{\theta}$

$\Longrightarrow$ Código exercício R

```
> #%%%%%%%%%%%
> if (!require(pacman)) install.packages("pacman")
Loading required package: pacman
> pacman::p_load(torch, cli, ggplot2, patchwork, dplyr)
> #
```

```
> set.seed(32)
> n <- 1000
> x <- sort(runif(n, -4, 4))
> f <- function(x) 3/(3+2*abs(x)^3) + exp(-x^2) + cos(x)*sin(x)
> y <- f(x) + rnorm(n) * 0.3
> # Quantil e Z-score
> TAU <- 0.75
> Z_TAU <- qnorm(TAU)
> #Definição da Rede Neural
> QuantileNet <- torch::nn_module(
+   "QuantileNet",
+   initialize = function(in_dim, out_dim, N) {
+
+     # Rede para estimar o Quantil q_tau(x)
+     self$seq_q <- torch::nn_sequential(
+       torch::nn_linear(in_dim, round(N)),
+       torch::nn_gelu(),
+       torch::nn_linear(round(N), out_dim)
+     )
+
+     # Rede para estimar o log(sigma(x))
+     self$seq_log_sigma <- torch::nn_sequential(
+       torch::nn_linear(in_dim, round(N)),
+       torch::nn_gelu(),
+       torch::nn_linear(round(N), out_dim)
+     )
+   },
+
+   forward = function(x) {
+     q_hat <- self$seq_q(x)
+     # Aplicamos exp para garantir que sigma > 0
+     sigma_hat <- torch_exp(self$seq_log_sigma(x))
+
+     return(list(q_hat, sigma_hat))
+   }
+ )
> #Função de Estimação
> estimacao_batch <- function(x, y, model, epochs = 1000, lr = 0.1,
+                             f_true = NULL, print = TRUE, batch = 32) {
+
+   n <- length(y)
+   yy <- torch_tensor(y)$unsqueeze(-1)
+   xx <- torch_tensor(x)$unsqueeze(-1)
+
+   opt <- optim_adam(model$parameters, lr = lr)
+
+   i <- 0
+   k <- 0
+   #
+   loss_store <- numeric(epochs)
+
+   while(i < epochs) {
+     sam <- sample(1:n)
+     i <- i + 1
+
+     # Loop de Batches
+     for(j in 1:round(n/batch)) {
+       k <- k + 1
+       idx_start <- 1 + batch * (j - 1)
+       idx_end   <- batch * j
+       if (idx_end > n) idx_end <- n
+       aux <- sam[idx_start:idx_end]
+
+       # Forward
```

```
+       fit <- model(xx[aux])
+       q_hat <- fit[[1]]
+       sigma_hat <- fit[[2]]
+
+       # J(theta) = log(sigma) + 0.5 * ((y - q)/sigma + z_tau)^2
+       res_padronizado <- (yy[aux] - q_hat) / sigma_hat
+       L <- torch_mean(torch_log(sigma_hat) + 0.5 * (res_padronizado + Z_TAU)^2)
+
+       # Backward
+       opt$zero_grad()
+       L$backward()
+       opt$step()
+     }
+
+     #
+     loss_store[i] <- L$item()
+
+     #
+     if(i %% round(epochs/10) == 0 && print) {
+
+       #
+       model$eval()
+       fit_full <- model(xx)
+       q_full <- as.numeric(fit_full[[1]])
+       sigma_full <- as.numeric(fit_full[[2]])
+
+       #mu = q - z * sigma
+       mu_full <- q_full - Z_TAU * sigma_full
+
+       model$train()
+
+       cli::cli_progress_message(paste("Época:", i, "Perda:", round(L$item(), 4)))
+
+       #
+       par(mfrow = c(1, 4))
+
+       #Quantil
+       plot(x, y, pch = 19, col = rgb(0,0,0,0.1), main = paste("Quantil", TAU*100, "%"),
+            cex=0.5, xlab="x", ylab="y")
+       lines(x, q_full, col = "tomato", lwd = 3)
+       # Quantil
+       q_true <- f(x) + Z_TAU * 0.3
+       lines(x, q_true, col = "blue", lwd = 2, lty = 2)
+       legend("topleft", legend=c("Ajustado", "Real"), col=c("tomato", "blue"),
+lty=1:2, bty="n", cex=1.5)
+
+       #sigma_theta (x)
+       plot(x, sigma_full, type="l", col="forestgreen", lwd=2,
+            ylim=c(0, 1), main = expression(hat(sigma)[theta](x)), xlab="x",

+ylab=expression(sigma[theta](x)))
+       abline(h=0.3, col="blue", lty=2)
+
+       #mu_theta (x)
+       plot(x, y, pch = 19, col = rgb(0,0,0,0.1), main = expression(hat(mu)[theta](x)),
+            cex=0.5, xlab="x", ylab="y")
+       lines(x, mu_full, col = "purple", lwd = 3)
+       curve(f, add=TRUE, col="black", lwd=2, lty=2)
+       legend("topleft", legend=c("Ajustado", "Real"), col=c("purple", "black"),

+lty=1:2, bty="n", cex=1.5)
+       #J(theta)
+       plot(1:i, loss_store[1:i], type="l", main="Função de Perda",
+            xlab="Época", ylab=expression(hat(J)[theta]), col="black")
```

```
+     }
+   }
+ }
> # (N = 50 neurônios)
> model <- QuantileNet(in_dim = 1, out_dim = 1, N = 50)
> #
> estimacao_batch(x, y, model, epochs = 2000, lr = 0.01, batch = 64)
```