Tradeoff Viés e Variância

Paulo Cirino Ribeiro Neto

03/04/2017

Viés e Variância

Em *Machine Learning* o problema de viés e variância é conceitualmente uma problema de minização de 2 fontes de erro.

O viés de um modelo está relacionado com a falta de capacidade que ele têm em aprender as relações entre entrada e saída, de forma geral é definido como a diferença entre valor esperado médio do modelo e a saída real.

A variância por sua vez, está relacionada com o erro devido a variabilidade entre modelos treinados com dados diferentes da mesma função geradora.

A relação entre os dois tipos de erro é quase sempre inversa, onde um modelo que possui robustez suficiente para não variar o resultado dado amostragens diferentes de uma função geradora, quase nunca é capaz de apredender as relações entre entrada e saída.

Definição Matemática

Definindo que existe uma variável resposta y que é gerada por $y = f(x) + \epsilon$. Criamos uma função $\hat{f}(x)$ de forma que $(y - \hat{f}(x))^2$ o mais proximo de 0.

Podemos definir o problema de viés e variância da seguinte forma:

$$\mathrm{E}\left[\left(y - \hat{f}(x)\right)^2\right] = \mathrm{Vies}\left[\hat{f}(x)\right]^2 + \mathrm{Var}\left[\hat{f}(x)\right] + \sigma^2$$

Onde definimos a variância como:

$$\mathrm{Var}[X] = \mathrm{E}[X^2] - \mathrm{E}[X]^2$$

E por dedução matemática é possível obter :

$$\mathrm{Bias}\left[\hat{f}(x)\right] = \mathrm{E}\left[\hat{f}(x) - f(x)\right]$$

Material Base

O material base desse trabalho é baseado na plataforma https://paulocirino.shinyapps.io/BiasAndVariance/ (https://paulocirino.shinyapps.io/BiasAndVariance/) que possibilita a visualização do problema viés e variância para diversas funções e modelos de forma interativa, o código fonto da plataforma é aberto no meu (github) [https://github.com/PauloCirino/BiasAndVariance (https://github.com/PauloCirino/BiasAndVariance)] pessoal.

Os cálculos de viés e variância foram feitos com base no exemplo fornecido em python da (sckit-learn) [http://scikit-learn.org/stable/auto_examples/ensemble/plot_bias_variance.html (http://scikit-learn.org/stable/auto_examples/ensemble/plot_bias_variance.html)].

Solução prática do Trabalho

Modelos

Foram pedidos 5 modelos com flexibilidades diferentes para esse trabalho, de forma a simplificar e grantir resultados nos extremos foram escolhidos modelos **KNN** com $K = \[1, 5, 10, 20, 40\]$. Onde, por definição, $K = 1$ interpola os dados de treinamento sendo o modelo com maior flexibilidade e $K = 40$ é exatamente a média amostral(porque utilizaremos 40 amostras de treinamento).

```
K <- c(40, 20, 10, 5, 1)
require('FNN')
ModelFunc <- function(Xtrain, Ytrain, K, Xtest) {
 fittedTrain <- FNN::knn.reg(train = as.matrix(Xtrain),
                             test = as.matrix(Xtest),
                             y = Ytrain,
                             k = K,
                             algorithm = "kd_tree")
 fittedTrain$pred
}
```

Dados

Foram pedidas 2 funções:

$$f_1(x) = Y(x) = \frac{(X-2)(2X+1)}{1+X^2} + \epsilon$$

$$f_2(x) = Y(x) = \sin(X) + \epsilon$$

```
f1 <- function(nObs = 40, minX = -8, maxX = 12, step = 0.1){
    Xtrain <- runif(n = nObs, min = minX, max = maxX)
    Ytrain <- ((Xtrain-2) * (2*Xtrain + 1)) / (1 + Xtrain**2)
    noise <- ( rnorm(length(Ytrain), sd = 0.25))
    Ytrain <- Ytrain + noise

    Xtest <- seq(from = minX, to = maxX, by = step)
    Ytest <- ((Xtest-2) * (2*Xtest + 1)) / (1 + Xtest**2)

    list(train = data.frame(Xtrain = Xtrain, Ytrain = Ytrain),
         test = data.frame(Xtest = Xtest, Ytest = Ytest),
         noiseVar = var(noise)
    )
}

f2 <- function(nObs = 40, minX = 0, maxX = 6.28, step = 0.1){
    Xtrain <- runif(n = nObs, min = minX, max = maxX)
    Ytrain <- sin(Xtrain)
    noise <- ( rnorm(length(Ytrain), sd = 0.1))
    Ytrain <- Ytrain + noise

    Xtest <- seq(from = minX, to = maxX, by = step)
    Ytest <- sin(Xtest)

    list(train = data.frame(Xtrain = Xtrain, Ytrain = Ytrain),
         test = data.frame(Xtest = Xtest, Ytest = Ytest),
         noiseVar = var(noise)
    )
}

Funcs <- c(f1, f2)
```

Nota-se que o ruído é uma normal de $média = 0$ e $sd = 0.1$.

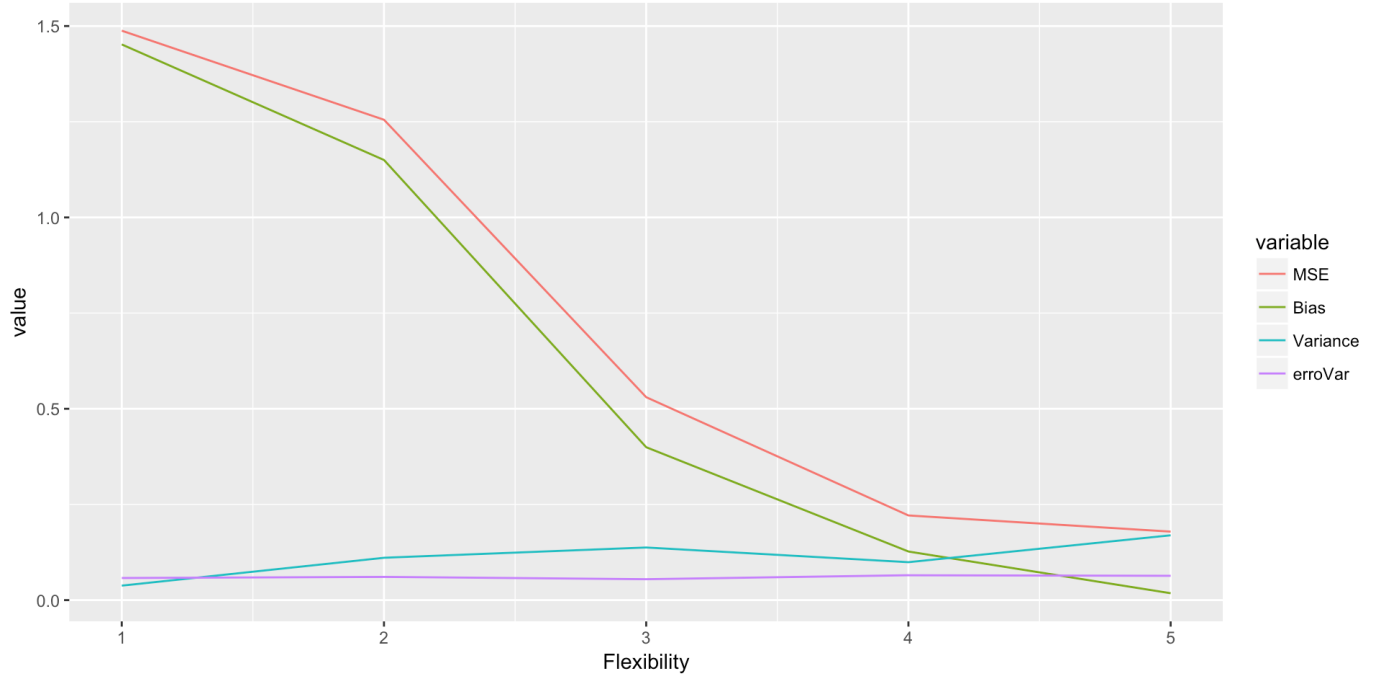
Gerando os Dados, Aplicando Modelos e Calculando Erros - (Questões 1 à 4)

Gerando 20 realizações de cada função temos:

```
nObs = 40
nModels = 20

resultDF <- data.frame()

for(fPos in 1:length(Funcs)){
    f <- eval(Funcs[[fPos]])

    for (kPos in 1:length(K)){
        k <- K[kPos]

        auxData <- f(nObs = nObs)

        YTestExpected <- auxData$test$Ytest
        YTestPredictMatrix <- matrix(ncol = nModels, nrow = length(YTestExpected))
        noiseVar <- numeric(length = nModels)

        for(nModel in 1:nModels){
            partialData <- f(nObs = nObs)
            noiseVar[nModel] <- partialData$noiseVar

            modelResult <- ModelFunc(Xtrain = partialData$train$Xtrain,
                                     Ytrain = partialData$train$Ytrain,
                                     K = k,
                                     Xtest = partialData$test$Xtest)

            YTestPredictMatrix[, nModel] <- modelResult
        }

        meanMSE <- mean( apply(YTestPredictMatrix, 2, function(x){
                mean( (x - YTestExpected) ** 2 )
        }))
        meanBias <- mean( (YTestExpected - apply(YTestPredictMatrix, 1, mean)) ** 2 )
        meanVariance <- mean( apply(YTestPredictMatrix, 1, var) )
        meanErrorVar <- mean(noiseVar)

        resultDF <- resultDF %>%
            dplyr::bind_rows(
                data.frame(MSE = meanMSE,
                           Bias = meanBias,
                           Variance = meanVariance,
                           erroVar = meanErrorVar,
                           Flexibility = kPos,
                           Function = fPos)
            )

    }
}
```

Visualizando Viés x Variância

```
resultDF %>%
    reshape2::melt(id = c("Flexibility", "Function")) %>%
    dplyr::filter(Function == 1) %>%
    dplyr::select(-Function) %>%
    ggplot2::ggplot(aes(x = Flexibility, y = value, color = variable)) +
    ggplot2::geom_line()
```




```
resultDF %>%
    reshape2::melt(id = c("Flexibility", "Function")) %>%
    dplyr::filter(Function == 2) %>%
    dplyr::select(-Function) %>%
    ggplot2::ggplot(aes(x = Flexibility, y = value, color = variable)) +
    ggplot2::geom_line()
```