Universiteit van Amsterdam
Amsterdam School of Economics

Machine Learning in Finance

Assignment 2

Gauss Process Regression for Option Pricing

AJ* and SU†

Spring 2022

*Student number: ∞
†Student number: π

Summary

This report details the implementation of *Gauss Process Regression* to determine call option prices and illustrate how a calibrated machine learning model can be used to value an option portfolio. Having implemented the method, the results and limitations will be discussed.

Implementation

We first import the requisite libraries and define the function `BLSPrice`, following the conventions used in *Exercise 6*:

```
## libraries
library(MASS)
library(GauPro)
library(ggplot2)

## a) Create the dataset of call option prices
##    Plain Vanilla Call
BLSPrice <- function(S, M, cap.T, sigma, r){
  ## M = K/S
  d1 <- (log(1/M) + (r + 0.5*sigma^2)*cap.T) / (sigma*sqrt(cap.T))
  d2 <- d1 - sigma*sqrt(cap.T)
  C <- S*pnorm(d1) - S*M*exp(-r*cap.T)*pnorm(d2)
  return(C)
}
```

Now, set the simulation parameters and create the dataset:

```
## parameters for simulation
set.seed(1)
N <- 100
S <- runif(N, 80, 120)
M <- runif(N, .5, 1.5)
cap.T <- runif(N, 1/12, 1)
sigma <- runif(N, 0.1, 0.4)
r <- runif(N, 0, 0.05)

## dataset
vanilla.C <- BLSPrice(S, M, cap.T, sigma, r)

## target Y dim(n x 1)
Y <- vanilla.C
```

```
## feature matrix X dim(n x 5)
X <- cbind(S, M, cap.T, sigma, r)
```

Next, we fit the model:

```
## b) fit GPR using GauPro
GPR.call <- GauPro(X, Y)
```

Using the fitted model, we determine the price of call options for different strike prices K:

```
## c) the price of call options for different strike prices
Np <- 100
Sp <- 100 + rep(0, Np)
Mp <- seq(0.5, 1.5, length.out = Np)
Tp <- 1/12 + rep(0, Np)
sigp <- 0.3 + rep(0, Np)
rp <- 0.01 + rep(0, Np)

## create new design matrix
Xp <- cbind(Sp, Mp, Tp, sigp, rp)

## predictions
Cp <- BLSPrice(Sp, Mp, Tp, sigp, rp)
Yp <- GPR.call$pred(Xp)

## d) plot model predictions with confidence bounds
upper <- GPR.call$predict(Xp)+2*GPR.call$predict(Xp, se=T)$se
lower <- GPR.call$predict(Xp)-2*GPR.call$predict(Xp, se=T)$se

ggplot() +
  geom_line(aes(Mp, Cp, color = "C"), size = 1.5) +
  geom_line(aes(Mp, Yp, color = "C.hat"), size = 1.5) +
  geom_line(aes(Mp, upper, color = "Confidence bounds"), size = 1) +
  geom_line(aes(Mp, lower, color = "Confidence bounds"), size = 1) +
  scale_color_manual("",
                     breaks = c("C", "C.hat", "Confidence bounds"),
                     values = c("C" = "blue", "C.hat" = "red",
                     "Confidence bounds" = "orange")) +
  labs(x = "M", y = "C") +
  theme(text = element_text(size = 20))
```

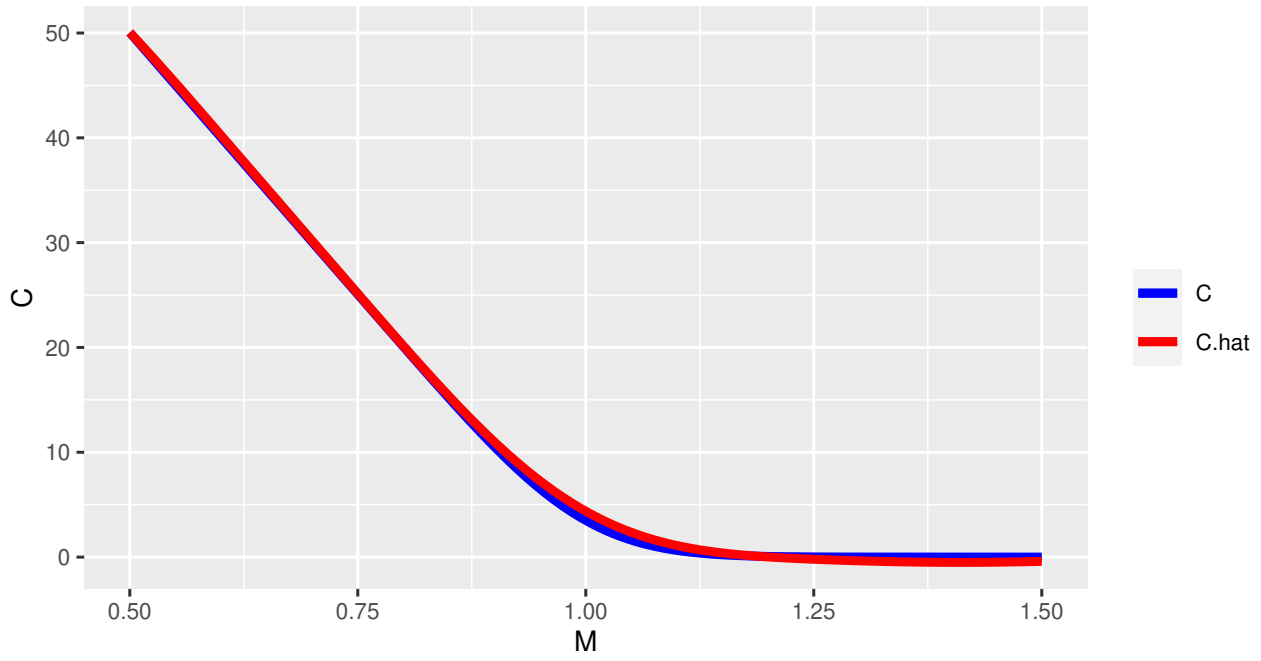

Figure 1: Price of the call option: Black-Scholes formula (*blue*) vs. fitted value (*red*).

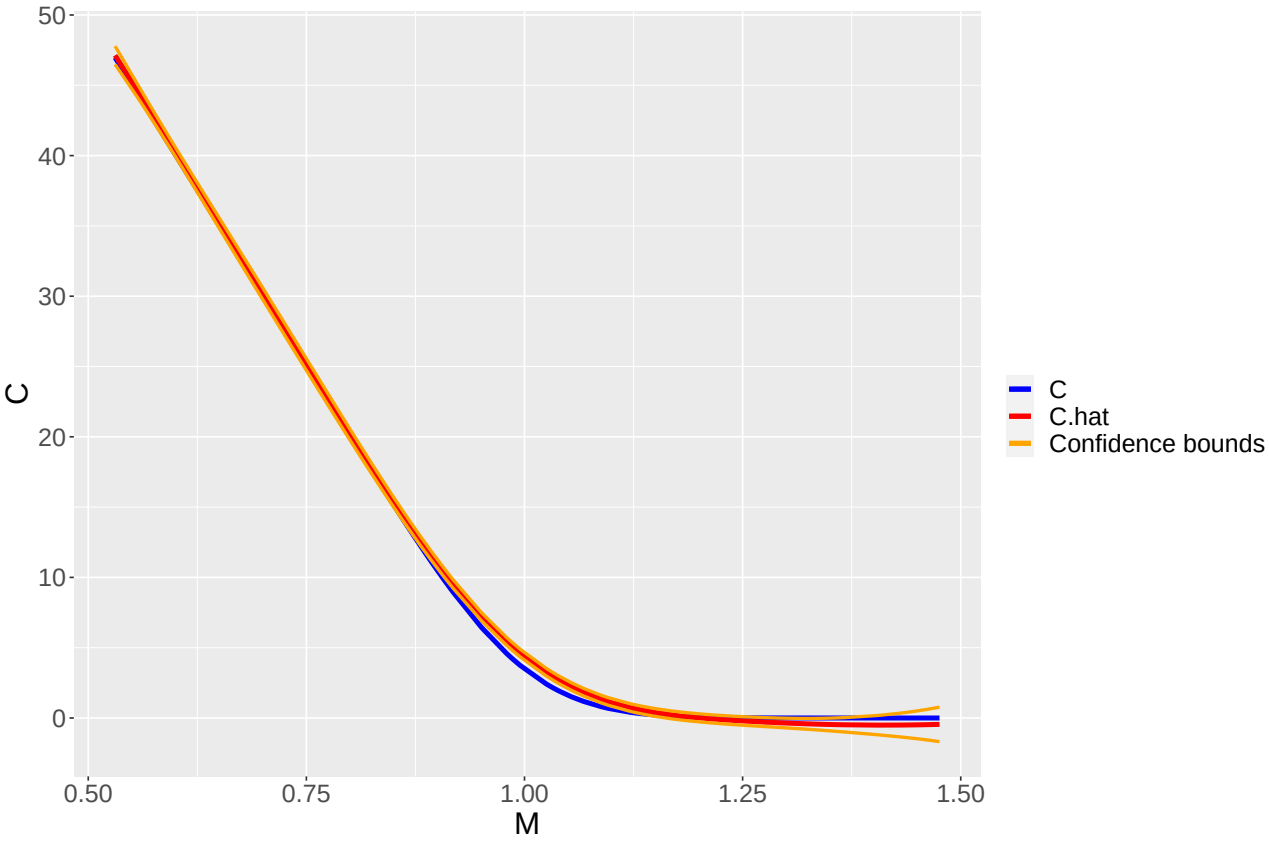

Figure 2: Confidence bounds (*orange*) for fitted value (*red*) [two standard deviations].

```
## e) construct new portfolio of options
set.seed(100)
N <- 100
S <- runif(N, 80, 120)
M <- runif(N, .5, 1.5)
cap.T <- runif(N, 1/12, 1)
sigma <- runif(N, 0.1, 0.4)
r <- runif(N, 0, 0.05)

## dataset
vanilla.C <- BLSPrice(S, M, cap.T, sigma, r)

## target Y dim(n x 1)
Y_port <- vanilla.C

## create feature matrix
X_port <- cbind(S, M, cap.T, sigma, r)

## set start time to determine calculation time
start_time <- Sys.time()

## calculate portfolio value with prediction from previous question
port_value <- sum(GPR.call$pred(X_port))

## set end time
end_time <- Sys.time()

## Print duration + computed portfolio values
print(end_time - start_time)
print(sum(Y_port))
print(port_value)

## Plot result
plot(Y_port, xlab='Option index', ylab='Option price', main='Fitted values
    vs. Black-scholes values')
points(GPR.call$pred(X_port), col='red')
legend(70,55,c('Black-scholes
    values', 'Fitted values'),cex=0.8,pch=c('O','O'),col=c('black','red'))
```

Discussion

Interestingly, we initially attempted to create the feature matrix $\mathbf{X}$ having normalised the features, carried out by the function `DataNorm`. This is primarily to control for the stock price `S` $\in [80, 120]$, whereas `M, T, sigma` and `r` are all ≤ 2. The features were normalised to the interval $[-1, 1]$. The function was defined as:

```
## Data normalization [-1,+1]
DataNorm = function(X){
  X[,1] = 2*(X[,1] - 100)/(120 - 80)     # S
  X[,2] = 2*(X[,2] - 1)/(1.5 - 0.5)      # M
  X[,3] = 2*(X[,3] - (11/24))/(1 - 1/12) # T
  X[,4] = 2*(X[,4] - 0.25)/(0.4 - 0.1)   # sigma
  X[,5] = 2*(X[,5] - 0.025)/(0.05 - 0)   # r
  return(X)
}


X_norm = DataNorm(X)
```

In the computation used by `GauPro`, the matrix $\mathbf{K}$ needs to be invertible, and hence positive definite. The transformation performed above made the Cholesky decomposition numerically unstable, and the matrix was no longer invertible. Given that the parameters of `sigma` and `r` are near zero, this is likely the source of the instability.

The target vector $\mathbf{Y}$ of option prices, as defined by the Black-Scholes formula, is calculated using vector arguments and is an efficient method of generating large training sets for any subsequent machine learning method. In this instance, Gauss Process Regression was fitted by calling the function `GauPro` with feature matrix $\mathbf{X}$ associated with the target $\mathbf{Y}$.

To assess the fit and uncertainty of the predictions, a dataset with new parameters and hence a new design matrix was created, as described in part **(c)**. Making use of the `pred` method of the `GauPro` function, predictions for the options prices could be made. They were stored in the vector `Yp` and compared to the vector with Black-Scholes option prices `Cp`.

Subsequently, two plots, one with the Black-Scholes option prices and the predicted option prices and the second one additionally with the 95% confidence bounds were created. To obtain the confidence bounds we again used the `pred` method but this time with the argument `se = TRUE` to obtain the standard errors. The confidence bounds are then created by taking the prediction and adding/subtracting twice the standard deviation. One can see that in general the predicted option prices are very close to the Black-Scholes option prices. Only in the area where the Option is at the money (M = 1) the prediction does not perform that well even though the confidence bounds signal that the prediction should be rather precise.

The running time of around 0.004 seconds for a portfolio consisting of 100 options and the computed portfolio values which just differ by around 2.50 with portfolio values of around 1605 reveal that a fitted Gauss Process Regression model provides an efficient method of determining option prices. Furthermore, the narrow nature of the confidence intervals generated by `GauPro` in this instance would indicate that we have a high degree of certainty in the predictions being generated. This finding would need to be scrutinised around $\texttt{M} = 1$, as the predictions appear to be mostly inaccurate in this region, yet our model indicates low prediction uncertainty. Last but not least we also include a plot to illustrate how well the Gauss Process Regression fitted our data in part **(e)** by plotting the different option values for 100 options with randomly chosen parameters, according to the Black-Scholes formula and the fitted model.

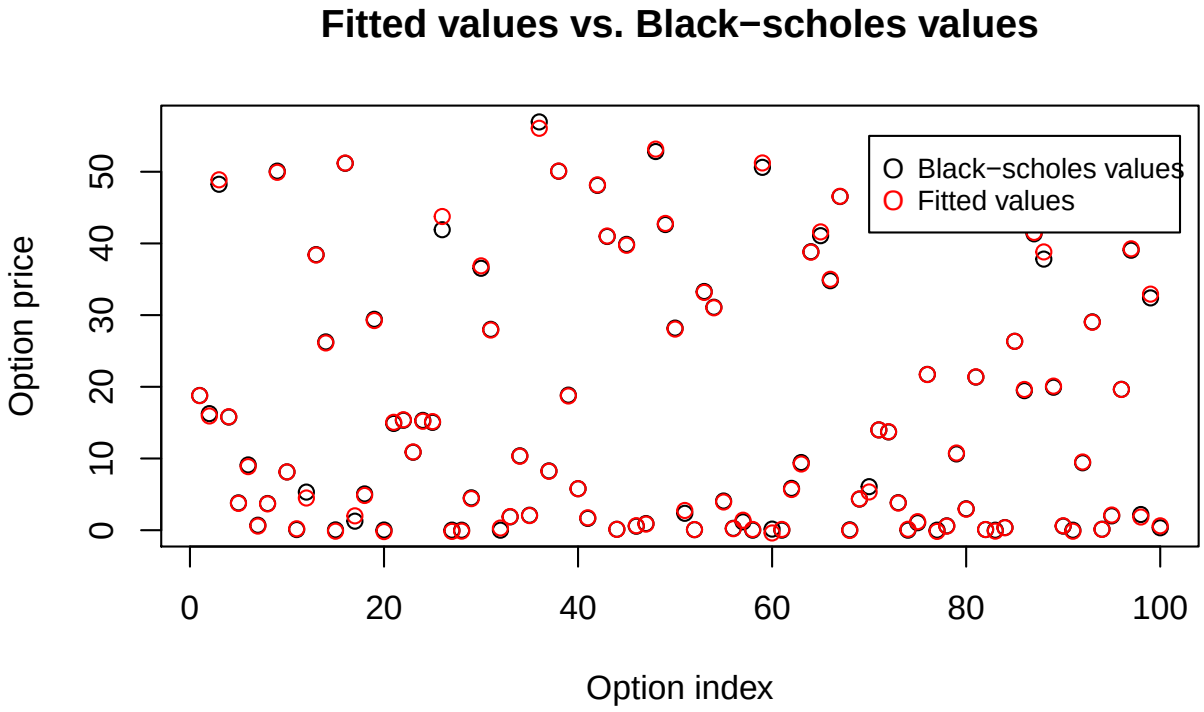

Figure 3: Price of the options in a new portfolio: Black-Scholes formula vs. fitted value.