# COMP6212 Computational Finance 2017/18, Assignment Part I

## (a)

1. Both of the assets have the same expected return (e.g. mean) and variance. This means that any holdings combination will result in the same E and V pair and consequently, they will map into the same unique Markowitz efficient frontier.
2. The following figures, which were plotted using MATLAB’s financial toolbox, show the Markowitz efficient frontier for different combinations of securities. The function used to produce them was portstats, using the Expected Return vector and the Covariance matrix of the securities as the core arguments.

|  |  |
| --- | --- |
|  |  |
| Figure 1: Efficient Portfolio for the 3 Securities | Figure 2: Efficient Portfolio of Securities 1 and 2 |
|  |  |
| Figure 3: Efficient Portfolio of Securities 1 and 3 | Figure 4: Efficient Portfolio of Securities 2 and 3 |

Figure 1 shows the Markowitz efficient frontier of the three securities with a scatter of 100 random portfolios. The random portfolios cover a wide area of the plot, showing examples of good (e.g. close to the frontier) and bad (e.g. far from the frontier) portfolios. Distinctively, Figure 2 to Figure 4 show that all of the randomly generated portfolios are positioned on the efficient frontier. This implies that holding any pair combination of these 3 securities leads to an efficient portfolio.

1. Matlab’s CVX tools were used to replace the built-in linprog and quadprog functions within the **NaiveMV** function.

|  |  |  |
| --- | --- | --- |
|  | cvx\_begin      variable weights(3)      maximize( ERet \* weights )      subject to          V1 \* weights == 1; % equality constraint, where V1 = ones(3,1)          weights >= 0; % inequality constraint (Lower Bound)  cvx\_end |  |
|  | Code 1: linprog in CVX |  |

Code 1 shows the procedure in which CVX can be used to find the weights (e.g. holdings) of the securities that maximize the expected return, subject to the following conditions:

|  |  |  |  |
| --- | --- | --- | --- |
|  | (1) |  | (2) |

Code 2 shows the CVX implementation for the quadprog function. This time, the task involves finding the weights that minimize the variance of the portfolio, , subject to the same constraints (1) and (2).

|  |  |  |
| --- | --- | --- |
|  | cvx\_begin      variable weights(3)      minimize( weights' \* ECov \* weights )      subject to          V1 \* weights == 1; % equality constraint, where V1 = ones(3,1)          weights >= 0; % inequality constraint (Lower Bound)  cvx\_end |  |
|  | Code 2: quadprog in CVX |  |

Lastly, using the information collected from Code 1 and Code 2, that is, the maximum achievable return and the minimum achievable variance, Code 3 is used to iteratively solve the optimization problem by discretely incrementing the desired expected return to trace the efficient frontier.

|  |  |  |
| --- | --- | --- |
|  | cvx\_begin      variable weights(3)      minimize( weights' \* ECov \* weights )      subject to          A \* weights == B;%equality constraint using several target returns          weights >= 0; % inequality constraint (Lower Bound)  cvx\_end |  |
|  | Code 3: quadprog in CVX for several target returns |  |

This new implementation of **NaiveMV** using the CVX tools was tested by plotting the Markowitz efficient frontier for the same securities in Question 2, against its previous version (e.g. using linprog and quadprog), as shown in Figure 5. The plot shows that both approaches lead to the same result.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | Figure 5: Comparison of Efficient Frontier using NaiveMV with CVX and linprog/quadprog |  |

1. The three randomly selected stocks from the constituent companies of the FTSE 100 were: Ashtead Group (ASH), Croda International (CRDA) and Sage Group (SGE). The daily closing prices of the stocks for the past three years were first imported from Yahoo Finance into a CSV file. The data from the first year and half (e.g. half of the data), was used to compute the expected returns and covariance of the securities. The daily returns were calculated by using the following equation:

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

The resulting expected return vector and covariance matrix for the securities ASH, CRDA and SGE, in that order, were:

|  |  |
| --- | --- |
|  |  |

Using this information, the efficient frontier, shown in Figure 6, was then plotted using MATLAB’s financial tools.

|  |  |
| --- | --- |
|  |  |
| Figure 6: Efficient Frontier for ASH, CRDA and SGE | Figure 7: Four Efficient Portfolio Samples |

To compare the performance of performance of several efficient portfolios obtained by using the information in the first half of the data, four portfolios, represented as red dots in Figure 7, were randomly selected to be tested using the remaining half of the data against the naïve portfolio. The obtained results are shown in the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Weights => | AHT | CRDA | SGE | Return (%) |
| Naive | 0.3333 | 0.3333 | 0.3333 | 32.08 |
| Portfolio 1 | 0.0000 | 0.0494 | 0.9506 | -3.07 |
| Portfolio 2 | 0.0000 | 0.1970 | 0.8025 | 3.04 |
| Portfolio 3 | 0.0003 | 0.3453 | 0.6544 | 9.20 |
| Portfolio 4 | 0.0682 | 0.5572 | 0.3746 | 22.73 |

Table 1: Returns of sample portfolios from the efficient frontier against the naïve portfolio

The results show that the returns generated from the naïve portfolio outperform the rest. The best performing random portfolio is Portfolio 4, which is the portfolio with the highest mean of return (e.g. forth red dot, from left to right).

1. The MacKinlay and Pastor Enhancement is used to use the information on the covariance matrix of the residuals to make up for the missing factors that affect the mispricing of returns. This is achieved by generating a new covariance matrix of the form , where the parameters , and are obtained from solving the likelihood function shown below:

The computed value for the parameters using the three securities used in exercise 4 (e.g. ASH, CRDA and SGE), were = [, , ], and . This resulted in a new covariance matrix:

This new covariance matrix was then used to plot the Markowitz efficient frontier, against the previous result. It can be observed that it offers a broader set of standard deviation options for efficient portfolios.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | Figure 8: Comparison of Efficient Frontier with and without using MacKinlay and Pastor enhancement |  |

1. a) A greedy forward selection algorithm was used to select the top 6 (e.g. a fifth) of the 30 constituent stocks from the FTSE 100, that best track the index. This was done by using Matlab’s sequentialfs function as shown below.

|  |  |  |
| --- | --- | --- |
|  | train = @(x\_train, y\_train, x\_test, y\_test) sum((y\_test - x\_test\*(x\_train \ y\_train)).^2);  counter = zeros(1,30);    for c = 1:50      inmodel = sequentialfs(train,X,y,'cv',10);      counter = counter + inmodel;  end    [B,I] = maxk(counter, 6); |  |
|  | Code 4: Greedy Forward Selection algorithm to select 6 stocks |  |

The function train is the cost function used to provide a quantitative measure of how close a subset of selected features in matrix X is to the objective vector Y. This is computed using a simple linear regression cost function. The matrix X, of size 760x30, contains 760 sequential daily returns for each of the 30 available stocks and the vector Y, of size 760x1, contains the daily returns for the FTSE 100. The argument ‘cv’, represents the n-fold cross validation parameter, which in this case is 10. The algorithm was run for 50 times, each time keeping track of the chosen stocks, to plot a histogram depicting how many times each one was picked by the algorithm. This is shown below in Figure 9.

|  |  |
| --- | --- |
|  |  |
| Figure 9: Stocks selected by Forward Selection algorithm | Figure 10: Stocks selected by Lasso Regression |

The resulting 6 stocks were Stocks 7, 18, 20, 24, 25 and 29. To measure the correlation and similarity between this 6-stock portfolio and the FTSE 100, the Pearson’s correlation coefficient and the expected return were computed. The correlation coefficient was calculated using MATLAB’s corrcoef function, which returned a result of 0.1245. The expected return for this portfolio was , whereas the expected return for the FTSE 100 for the same time period was .

b) In this approach, a sparse index tracking portfolio was constructed by using regularization. Lasso regression was applied to the same X matrix and Y vector using the lasso function in MATLAB with a 10-fold cross validation. This was executed 50 times, and in each time, using the regularization parameter yielding the minimum MSE, the stocks for which the weights were assigned with a non-zero value were counted. The results are shown in Figure 10. For a visual presentation, Figure 11 illustrates the variation of the mean squared error (e.g. similarity to the FTSE 100) for different values of the regularization parameter. This was generated using MATLAB’s lassoplot function.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | Figure 11: MSE against different values of the regularization parameter Lambda |  |

Taking all this information into account, the regularization parameter which was chosen was 0.0005 which resulted in weights of 0.1416 for stocks 18 and 20, and 0.3584 for stocks 25 and 29, with the rest of the stocks set to 0. The Pearson correlation and the expected return obtained for this 4-stock portfolio were 0.155 and , respectively. This shows that the Lasso regression resulted in a more accurate selection of stocks than the forward selection algorithm.

1. In the paper of Lobo *et al.*, transaction costs are introduced as functions that take the transactions , as inputs, and output the costs incurred from them. These are then included in the portfolio optimization portfolio problem as an additional constraint, leading to the following optimization problem:

|  |  |  |
| --- | --- | --- |
| maximise | subject to , | (4) |

where is the expected returns of the assets, are the current holdings and the transactions, in which when buying, and for selling. The condition is valid for self-financing portfolios, as the condition only allows to buy securities by selling others to fill the cost. The second constraint simply encapsulates all the possible portfolios that can be built. The paper also suggest that optimization problem could be written in the following form:

|  |  |
| --- | --- |
| minimize | subject to , |

where the objective is to minimize the transaction costs subject to a minimum expected return .

Figure 2 (from Lobo *et al*.’s paper) illustrates a cumulative distribution plot of the returns of a group of assets,100 risky and 1 riskless, in a portfolio. It maps the probabilities for a portfolio to generate a return of less than a value , by assuming that they are normally distributed. The basis of the graph comes from the equation , in which we want to calculate the probability that the holdings of a portfolio , at the end of a particular timeframe, will be larger than . This assumes that is a Gaussian random variable, leading to the plot shown in Figure 2. Although there are some changes to the notation and representation of variables, the example given in Section 1.6 represents the same optimization problem as (4), with some tweaks and additions. The first addition is the short selling limit constraint given by , where the sold securities exceed the bough securities by more than . The other additions are the of two shortfall risks constraints, with the two main parameters and (e.g. , , , ) which assigns probabilities for the portfolio to produce a return of with , as depicted in Figure 2 (from Lobo *et al*.’s paper). These could be added to our portfolio optimization problem, by first imposing the transaction costs constraints and defining these shortfall risks to much our risk admissibility.

## (b)

1. The derivate of the cumulative normal distribution , is . To show that the expression (4) is true, we can work out and check if it matches its definition, as shown in (5).

|  |  |  |  |
| --- | --- | --- | --- |
|  | (4) |  | (5) |

First, we substitute and with their corresponding expressions to equation (4).

We then cancel out the constant and merge the exponentials together.

We then move *K* to the left-hand side of the equation and apply *log* to both sides of the equation.

We then rearrange the equation to move everything but to the right-hand side of the equation and substitute in.

We cancel out , and work out for

|  |  |  |
| --- | --- | --- |
| Ans. a) |  |  |

The derivative for can be calculated by first isolating the term that contains S, as shown below.

And then applying the derivative.

Because

|  |  |  |
| --- | --- | --- |
| Ans. b) |  |  |

To compute , we start by applying the derivative product rule to the second term of the equation.

We can then substitute equation (4) to replace the second term of the equation with .

Because we know that , we can compute , by applying the chain rule using a temporary variable and performing .

|  |  |  |
| --- | --- | --- |
|  |  |  |

Finally, we substitute this result into and we arrive to the final equation.

|  |  |  |
| --- | --- | --- |
| Ans. c) |  |  |

The partial derivative can be calculated by first applying the product rule to the first term.

We can then use the chain rule to write it in the following form.

By substituting the equations (4) and Ans. (b), the second and third terms are equivalent and therefore, they cancel each other out. Leading to.

|  |  |  |
| --- | --- | --- |
| Ans. d) |  |  |

To calculate the second derivative , we apply another derivative to Ans. (d), which leads to the previous chain rule We then simply substitute the equation from Ans. (b) to reach to the expression below.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Ans. e) |  |

Finally, we can prove that the expression for the call option price in indeed the solution to the Black-Scholes differential equation by substituting Ans. c), d) and e) in the equation below.

Where all the terms cancel out as expected.

1. MATLAB’s blsprice function was used to compute the Black-Scholes Model for each of the option prices, as shown in Appendix [n]. The arguments used to achieve this were the *strike price*, given in the file name; the *price* of the security at a given day; the *annual interest rate*, set to 6.0%; the *time*, given in a yearly scale; and the Hull *volatility*, which was computed using the equation , where is the standard deviation of the proportional price change and , the timescale computed in total number of trading days in a year, or 252. The Hull volatility was calculated each using the precedent 52 past prices of the security. Figures 12 and 13 illustrate a comparison between the actual option prices given in the dataset, against the option prices computed using the Black-Scholes Model.

|  |  |
| --- | --- |
|  |  |
| Figure 12: Actual Option Prices vs BS Option Prices for c2995 | Figure 13: Actual Option Prices vs BS Option Prices for p2995 |

It was consistently observed that for the call options generated by the Black-Scholes Model, the values were slightly higher to those of the actual prices, whereas for the put options, the values were lower. To quantify the differences between actual prices and the BS model, a histogram of the MSEs was plotted for all the different strike options, as shown in Figure 14. The bar plot shows that as the strike price of the call options increases, the Black-Scholes model generates values that are closer to the actual prices, whereas for the put options, it’s the opposite.

The implied volatility was computed for the call option, at a strike price of 2925, for a random 30-day period, resulting in the scatter plot shown in Figure 15. This shows that the implied volatilities from days 100 to 129, have a much higher variation than the Hull volatilities, previously computed.

|  |  |
| --- | --- |
|  |  |
| Figure 12: Actual Option Prices vs BS Option Prices for c2995 | Figure 13: Actual Option Prices vs BS Option Prices for p2995 |

Lastly, the implied volatilities for day 100, were calculated and plotted for each strike price as it can be seen below in Figure 14. The plot shows that there is an upward slope for the put option implied volatilities and the opposite for the call options. This is effect is known as the volatility smile.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | Figure 14: MSE against different values of the regularization parameter Lambda |  |

1. b) A simulation using MATLAB’s newrb function was performed to verify the claims from the paper by Hutchison *et al.*, where they suggest that the complex parametric relationship between the option prices and their variables can be approximated by ‘learning’ their respective Black-Scholes model. The first