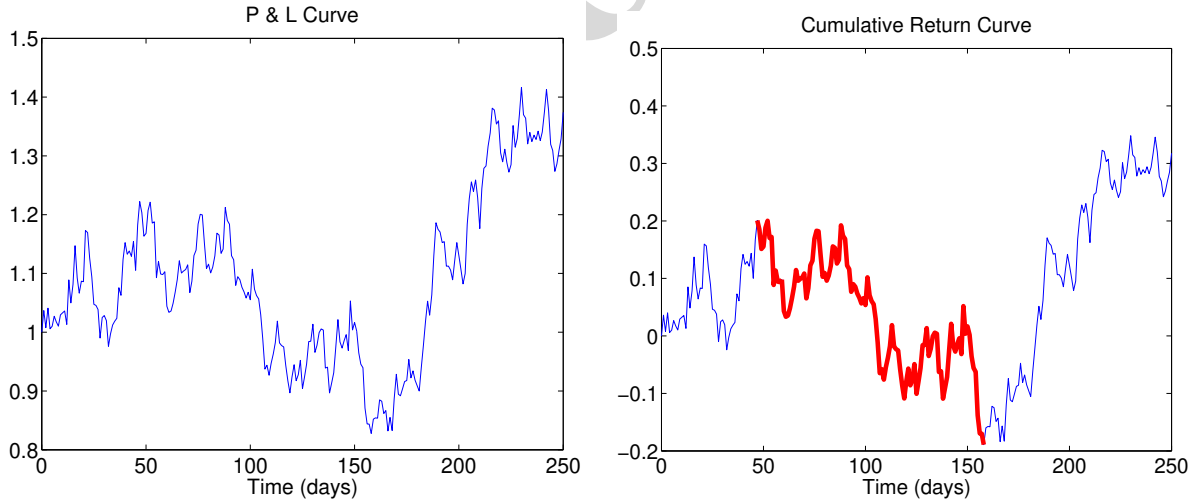
Chapter 6

Optimal Trading

6.1 Measures of Performance

Imagine investing \$1 in a trading strategy, and monitoring the status of your investment at regularly spaced times $t_1, t_2, \ldots t_n$. For convenience we set $t_0 = 0$ and assume that the spacing between the times is τ, so that $t_i = i\tau$. For daily trading strategies, $\tau = 1\ day = \frac{1}{250}\ years$ where there are approximately 250 trading days in a year. Let V_i be the value of your investment at time t_i ($V_0 = 1$). The sequence of V_i is known as the profit and loss curve (P&L curve) of the trading strategy.

The return sequence is given by $r_i = \log(V_i/V_{i-1}) \approx (V_i - V_{i-1})/V_{i-1}$ which is the percentage return. The cumulative return curve is given by $R_t = \sum_{i=1}^{t} r_i$. An example P&L curve and its corresponding cumulative return curve are shown below.



The mean τ-period return, μ_τ is the average of the returns, $\mu_\tau = \frac{1}{n}\sum_{i=1}^{n} r_i$. It is conventional to compute the mean annualized return which scales up this average return to 1 year, so that we define the average annualized return as

$$\mu = \frac{\mu_\tau}{\tau} = \frac{1}{n\tau}\sum_{i=1}^{n} r_i = \frac{1}{T}\sum_{i=1}^{n} r_i,$$

where $T = n\tau$ is the entire period of observation. Similarly, we can compute the variance of the returns, $\sigma_\tau^2 = \frac{1}{n}\sum_{i=1}^{n}(r_i - \mu_\tau)^2$. Assuming that each time period is independent, we can compute the annualized variance by scaling up the τ-period variance to 1 year,

$$\sigma^2 = \frac{\sigma_\tau^2}{\tau} = \frac{1}{n\tau}\sum_{i=1}^{n}(r_i - \mu_\tau)^2 = \frac{1}{T}\sum_{i=1}^{n}(r_i - \mu_\tau)^2.$$

The annualized volatility is σ. The Sharpe ratio is a risk adjusted measure of performance defined by the ratio of the annualized return and the annualized volatility,

$$Sharpe = \frac{\mu}{\sigma} = \frac{\mu_\tau}{\sigma_\tau\sqrt{\tau}}.$$

It is often the case that these measures will be defined not with respect to the raw returns, but the excess returns, where the excess is with respect to the risk free rate, rf_i. Thus we define the excess returns by $\bar{r}_i = r_i - rf_i$.

Exercise 6.1
Give linear time algorithms for computing the annualized average return, the anualized volatility and the Sharpe ratio.

The Sharpe ratio is an example of a risk adjusted measure of return because it scales the return by a normalizing factor which is how variable the return is, which is a measure of the risk in the trading strategy. For two trading strategies with the same return, the one with lower volatility (or risk) will have a higher Sharpe ratio. Generally, 3 years is an acceptable track record for a trading strategy, and a Sharpe ratio of 2 or higher over a period of more than 3 years is considered very good in the industry.

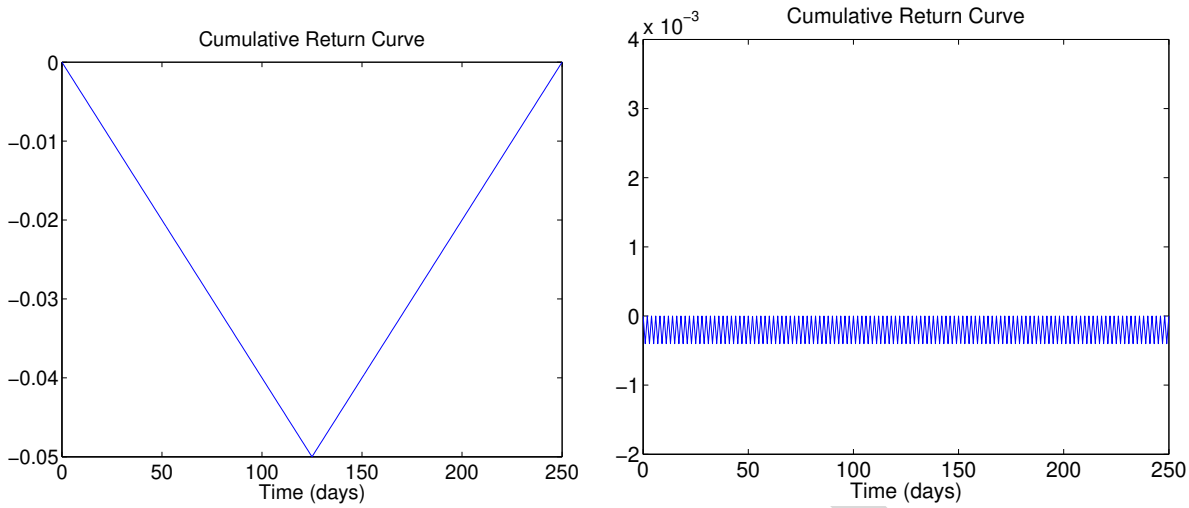
There are two properties of the Sharpe ratio that make it a little undesirable. The first is that it penalizes the downside and upside risk equally. So a trading strategy which makes only positive but variable returns can have a low Sharpe ratio.

Exercise 6.2
Given any $\epsilon > 0$, construct a return sequence (for appropriately defined τ) consisting only of positive returns for which the Sharpe ratio is less than ϵ.

One way to alleviate this problem is to only consider the negative returns in defining the risk. Thus we compute the the root mean square of the negative returns, sometimes called the downside deviation. The ratio or the average return to the downside deviation, usually denoted the downside deviation ratio, is another risk adjusted measure of performance which now does not penalize variability in positive returns. This is not often used in practice because there is yet another flaw in the Sharpe ratio which also affects the downside deviation.

Exercise 6.3
Show that any permutation in the return sequence results in the same Sharpe ratio.

To illustrate the problem, consider the following two cumulative return curves,



In the first one, all the negative returns occur first. In the second one the negative and positive returns alternate. These are two clearly different looking cumulative return curves, yet they will have the same Sharpe ratio.

The maximum drawdown risk measure takes these considerations into account. The maximum drawdown (MDD) of a cumulative return curve is the largest possible loss, assuming you entered and exited the trading strategy at the worst possible time. For the first curve in the above example, the MDD is 5%, but for the second curve, it is approximately 0. For the P&L curve shown at the begining of this chapter, the part of the cumulative return curve realizing the MDD is highlighted in red. Notice that the MDD as a measure of risk does not penalize positive returns, no matter how variable they are, and further, the MDD is sensitive to permutations of the return sequence.

Formally, the maximum drawdown can be defined by viewing the return sequence as a string of numbers. Then, the MDD is the minimum possible substring sum,

$$MDD = \min_{i \le j} \left\{ \sum_{k=i}^{j} r_k \right\}.$$

Thus, a straightforward algorithm to compute the MDD is to consider all possible substrings, and compute the substring sum, taking the minimum.

Exercise 6.4
Show that this algorithm is cubic.

If the minimum of the cumulative return curve occurs after the maximum, then the MDD is the maximum minus the minimum. Otherwise this is not the case, and in general, there is no relationship between the maximum, the minimum and the MDD. A cubic algorithm is not acceptable in practice, and in this case it turns out that one can compute the MDD in linear time.

Exercise 6.5
Let R_i denote the cumulative return curve of a trading strategy. Define the drawdown at time i by

$$DD_i = \max_{1 \le k \le i} R_k - R_i,$$

which is the previous maximum minus the current value.

(a) Show that $MDD = \max_i DD_i$.
(b) Give a linear time algorithm to compute the MDD given as input the return sequence $r_1, \ldots, r_n$. The algorithm should have a run time which is linear in n.

The MDD is itself a very useful measure of risk. In fact, most hedge funds would like to have small MDD, because large drawdowns lead to fund redemptions. In addition, the Sterling ratio is a very common risk adjusted return measure obtained by dividing the return by the MDD,

$$Sterling = \frac{\mu}{MDD}.$$

Typically, the MDD is calculated over a period of 3 years, and the return is also scaled to 3 years. The choice of 3 years is a conventiional practice that has arisen because until recently, the scaling law for the MDD has not been known, and so the notion of an annualized MDD was not possible. Recently, **?** the behavior of the MDD with time has been computed, and so the notion of an annualized MDD does make sense, and can now be used to obtain standardized risk adjusted measures for funds which have been around for more than 3 years or less than 3 years.

6.2 Optimal Trading Strategies

A trader has in mind the task of developing a trading system that optimizes some profit criterion, the simplest being the total return. A more conservative approach is to optimize a risk adjusted return. In an enviroment where markets exhibit frequent crashes and portfolios encounter sustained periods of losses, it should be no surprise that the Sterling ratio and the Sharpe ratio have emerged as the leading performance measures used in the industry.

Given a set of instruments, a trading strategy is a switching function that transfers the wealth from one instrument to another. We consider the problem of finding optimal trading strategies, i.e., trading strategies that maximize a given performance metric, on *historical data*. We focus on the total return as the measure of performance, but one can also construct optimal strategies efficiently for variants of the Sharpe and Sterling ratio **?**. Finding the optimal trading strategy for non-zero transactions cost is a path dependent optimization problem even when the price time series is known. A brute force approach to solving this problem would search through the space of all possible trading strategies, keeping only the one satisfying the optimality criterion. Since the number of possible trading strategies grows exponentially with time, the brute force approach leads to an exponential time algorithm[1], which for all practical purposes is infeasible – even given the pace at which computing power grows.

(i) Knowing what the optimal trades are, one can take an inductive approach to real trading: on historical data, one can construct the optimal trades; one can then correlate various market

[1]The asymptotic running time of an algorithm is measured in terms of the input size n. If the input is a time sequence of n price data points, then polynomial time algorithms have run time that is bounded by some polynomial in n. Exponential time algorithms have running time greater than some exponentially growing function in n **?**.

and/or technical indicators with the optimal trades. These indicators can then be used to identify future trading opportunities. In a sense, one can try to *learn* to predict good trading opportunities based on indicators by emulating the optimal trading strategy. A host of such activity within the inductive framework, goes under the name of *financial engineering*.

(ii) The optimal trading performance under certain trading constraints can be used as a benchmark for real trading systems. For example, how good is a trading system that makes ten trades with a Sterling ratio of 4 over a given time period? One natural comparison is to benchmark this trading strategy against a Sterling-optimal trading strategy that makes at most ten trades over the same time period.

(iii) Optimal trading strategies (with or without constraints) can be used to quantitatively rank various markets (and time scales) with respect to their profitability according to a given criterion. So for example, one could determine the optimal time scale on which to trade a particular market, or given a set of markets, which is the most profit-friendly.

(iv) Given a stochastic model for the behavior of a pair of instruments, one can use the efficient algorithms presented here to construct *ex-ante* optimal strategies using simulation. To be more specific, note that the optimal strategy constructed by our algorithms requires full knowledge of the future price paths. The stochastic model can be used to generate sample paths for the instruments. These sample paths can be used to compute the optimal trading strategy given the current history and information set. One then has a sample set of future paths and corresponding optimal trading strategies on which to base the current action. Note that such a stochastic model for future prices would have to take into account correlations (including auto-correlations) among the instruments.

It is beyond the scope of the current discussion to develop these applications. Our main goal here is to present the algorithms for obtaining optimal trading strategies, *given* a price time series.

6.2.1 Trading Model

We now make the preceeding discussion more precise. We consider optimal trading strategies on two instruments, for concreteness, a stock S and a bond B with price histories $\{S_0, \dots, S_n\}$ and $\{B_0, \dots, B_n\}$ over n consecutive time periods, $\{[t_0, t_1], [t_1, t_2], \dots, [t_{n-1}, t_n]\}$. The prices B_i, S_i correspond to the times t_i, $i \in \{0, \dots, n\}$. We can assume that $t_0 = 0$. Thus, for example, over time period $[t_{i-1}, t_i]$, the price of stock moved from S_{i-1} to S_i.

We denote the return sequence for the two instruments by $\{s_1, \dots, s_n\}$ and $\{b_1, \dots, b_n\}$ respectively: $s_i = \log \frac{S_i}{S_{i-1}}$, and correspondingly, $b_i = \log \frac{B_i}{B_{i-1}}$. We assume that one of the instruments is the benchmark instrument, and that all the equity is held in the benchmark instrument at the begining and end of trading. The bond is usually considered the benchmark instrument, and for illustration, we will follow this convention. The trivial trading strategy is to simply hold onto bond for the entire duration of the trading period. It is useful to define the excess return sequence for the stock, $\hat{s}_i = s_i - b_i$. When the benchmark instrument is the bond, the excess return as we defined it is the conventionally used one. However, one may want to measure performances of a trading strategy with respect to the S&P 500 as benchmark instrument, in which case the excess return would be determined relative to the S&P 500 return sequence. The excess return sequence for the bond is just the sequence of zeros, $\hat{b}_i = 0$. Conventionally, the performance of a strategy is measured relative to some trivial strategy, so the excess return sequence will be the basis of most of our performance measures. We make the following assumptions regarding the trading:

A1 *[All or Nothing]*: The position at all times is either entirely bond or entirely stock.

A2 *[No Market Impact]*: Trades can be placed without affecting the quoted price.

A3 *[Fractional Market]*: Arbitrary fractions of stock or bond can be bought or sold.

A4 *[Long Strategies]*: One can only hold long positions in stock or bond.

Assumption **A1** is in fact not the case in many trading funds, for it does not allow legging into a trade, or holding positions in both instruments simultaneously. While this is technicaly a restriction, for many optimality criteria (for example return optimal strategies), one can show that there always exists an all-or-nothing optimal strategy. Thus, we maintain this simplifying assumption for our discussion. Further, such assumptions are typically made in the literature on optimal trading (see for example **?**). Assumptions **A2**–**A4** are rather mild and quite accurate in most liquid markets, for example foreign exchange. Assumption **A3** is needed for **A1**, since if all the money should be transfered to a stock position, this may necessitate the purchase of a fractional number of shares. Note that if $\mathcal{T}[i-1] \neq \mathcal{T}[i]$, then at the begining of time period $[t_{i-1}]$, the position was transfered from one instrument to another. Such a transfer will incur an instantaneous per unit transaction cost equal to the bid-ask spread of the instrument being transfered into. We assume that the bid-ask spread is some fraction (f_b for bond and f_s for stock) of the bid price.

With these constraints in mind, we define a trading strategy $\mathcal{T}$ as a boolean $n+1$-dimensional vector indicating where the money is at time t_i:

$$\mathcal{T}[i] = \begin{cases} 1 & \text{if money is in stock at time } t_i, \\ 0 & \text{if money is in bond at time } t_i. \end{cases}$$

Exercise 6.6
How many possible trading strategies are there?

We assume that $\mathcal{T}[0] = \mathcal{T}[n] = 0$, i.e., all the money begins and ends in bond. If $\mathcal{T}[i] = 0$ and $\mathcal{T}[i+1] = 1$ then infinitessimally after time t_i, the money is moved from bond to stock. We say that a trade is entered at time t_i. A trade is exited at time t_i if $\mathcal{T}[i] = 1$ and $\mathcal{T}[i+1] = 0$. The number of trades made by a trading strategy is equal to the number of trades that are entered. The return (or excess return) of the trading strategy over time period $[t_i, t_{i+1}]$ depends on the values of $\mathcal{T}[i]$ and $\mathcal{T}[i+1]$. We let $r_{\mathcal{T}}[i]$ for $i \in \{1, 2, \ldots, n\}$ be the vector which contains the returns of the trading strategy over the time period $[t_{i-1}, t_i]$. Then,

$$r_{\mathcal{T}}[i] = \begin{cases} b_i & \text{if } \mathcal{T}[i-1] = 0, \mathcal{T}[i] = 0; \\ b_i - f_b & \text{if } \mathcal{T}[i-1] = 1, \mathcal{T}[i] = 0; \\ s_i & \text{if } \mathcal{T}[i-1] = 1, \mathcal{T}[i] = 1; \\ s_i - f_s & \text{if } \mathcal{T}[i-1] = 0, \mathcal{T}[i] = 1. \end{cases} \tag{6.1}$$

where f_b is the transactions cost incurred in terms of return for switching positions from stock to bond, and f_s is the transactions cost incurred for switching positions from bond to stock. We assume that these transactions costs are constants. In words, the return over time period $[t_{i-1}, t_i]$ is the return for the instrument you end the period in minus a transactions cost if you started the period in the other instrument.

Exercise 6.7
The equity curve for a trading strategy $\mathcal{T}$ is the vector $\mathcal{E}_\mathcal{T}$, where $\mathcal{E}_\mathcal{T}[i]$ is the value at time t_i, with $\mathcal{E}_\mathcal{T}[0] = 1$. The return sequence $r_\mathcal{T}$ is then $r_\mathcal{T}[i] = \log \frac{\mathcal{E}_\mathcal{T}[i]}{\mathcal{E}_\mathcal{T}[i-1]}$, for $i \geq 1$. Suppose that the bid-ask spread for bond is a fraction $\hat{f}_b$ of the bid price, and for the stock is a fraction $\hat{f}_s$ of the bid price. Show that $f_s = \log(1 + \hat{f}_s)$ and $f_b = \log(1 + \hat{f}_b)$.

Note that when the bid ask spread is a constant, not a fraction of the bid price, then it is more convenient to work in the value (as opposed to the returns) space. The total return for a strategy is

$$\mu(\mathcal{T}) = \sum_{i=1}^{n} r_\mathcal{T}[i].$$

We will typically suppress the dependence on $\mathcal{T}$ when it is clear what trading strategy we are refering to. We will focus on maximizing the total return, and refer the reader to the literature for the more complex problems of maximizing the Sharpe and Sterling ratios **?**.

Exercise 6.8
Consider the 6 times $[0, 1, 2, 3, 4, 5]$, over which the return sequence for bond was $[1, 1, 1, 1, 1]$ and the return sequence for stock was $[1, -2, 3, 2, 1]$. Assume that $f_s = f_b = 1$ and compute the total return μ for the trading strategy $\mathcal{T} = [0, 0, 1, 0, 1, 1]$.

We now consider efficient algorithms for computing total return optimal trading strategies, with and without constraints on the number of trades [2]. In particular, it is possible to construct return optimal trading strategies in *linear time*:

(i) **Unconstrained Trading.** A trading strategy $\mathcal{T}^*$ can be computed in $O(n)$ such that for any other strategy $\mathcal{T}$, $\mu(\mathcal{T}^*) \geq \mu(\mathcal{T})$.

(ii) **Constrained Trading.** A trading strategy $\mathcal{T}_K^*$ making at most K trades can be computed in $O(K \cdot n)$ time such that for any other strategy $\mathcal{T}_K$ making at most K trades, $\mu(\mathcal{T}_K^*) \geq \mu(\mathcal{T}_K)$.

Exercise 6.9
For return optimal trading strategies, show that the all-or-nothing assumption can be made without loss of generality. In particular, show that there always exists a return optimal strategy which is all-or-nothing.
[Hint: You may want to use induction on the number of time steps n.]

In order to compute the return optimal strategies, we will use a dynamic programming approach to solve a more general problem. Specifically, we will construct the return optimal strategies for every prefix of the returns sequence. First we consider the case when there is no restriction on the number of trades, and then the case when the number of trades is constrained to be at most K.

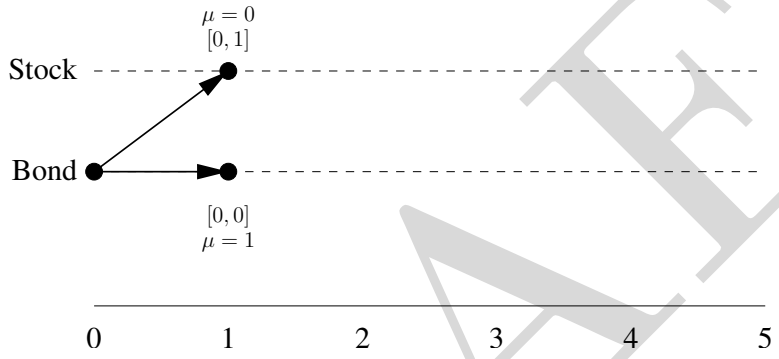
6.2.2 Overview of the Algorithm

The basic idea of the algorithm is to consider the optimal strategy to time t_i. This strategy must end in either stock or bond. Suppose that it ends in stock, then it must arrive at the final position

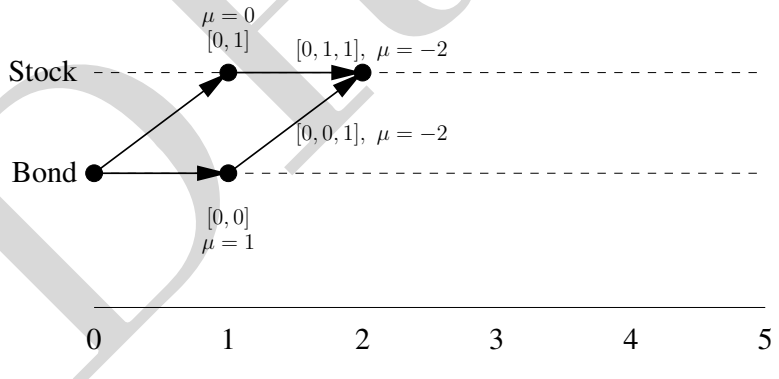
[2] We will use standard $O()$ notation in stating our results: let n be the length of the returns sequences; we say that the run time of an algorithm is $O(f(n))$ if, for some constant C, the runtime is $\leq Cf(n)$ for any possible return sequences. If $f(n)$ is linear (quadratic), we say that the runtime is linear (quadratic).

in stock at t_i by either passing through stock or bond at time t_{i-1}. Thus, the optimal strategy which ends in stock at time t_i must be either the optimal strategy which passes through stock at time t_{i-1} followed by holding the stock for one more time period, or the optimal strategy which passes through bond at time t_{i-1} and then makes a trade into the stock for the next time period. Whichever is better among these two options yields the optimal strategy to time period t_i that ends in stock. A similar argument applies to the optimal strategy to time t_i that ends in bond. Thus, having computed the optimal strategies which end in stock and bond to time t_{i-1}, we can compute the optimal strategies which end in stock and bond to time t_i. This induction can be propagated to obtain the final result.

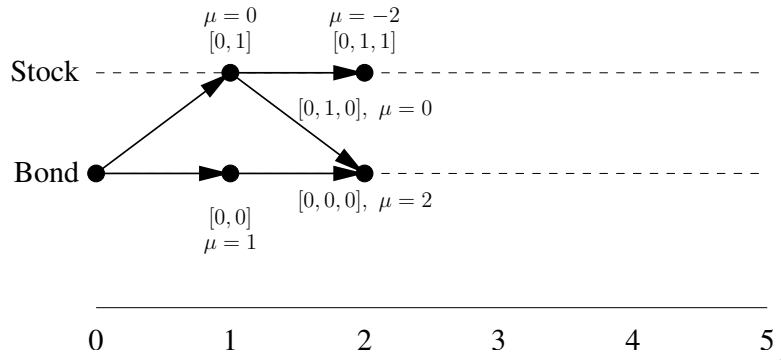
We will illustrate this idea with the example in Exercise 6.8. First we consider the optimal strategies up to time 1, ending in stock and bond. In fact there is only one such strategy which ends in bond (namely to hold bond) and one such strategy to end in stock (namely to switch to stock), hence these are the optimal ones. We can compute the returns of these two strategies, summarized in the following picture,



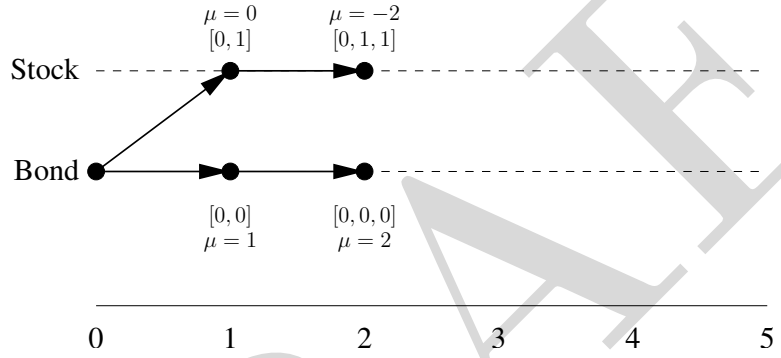
We now consider the optimal strategies to time 2, ending in stock. There are two options, either you came from bond or from stock, in either case, gettinig to the previous point optimally,



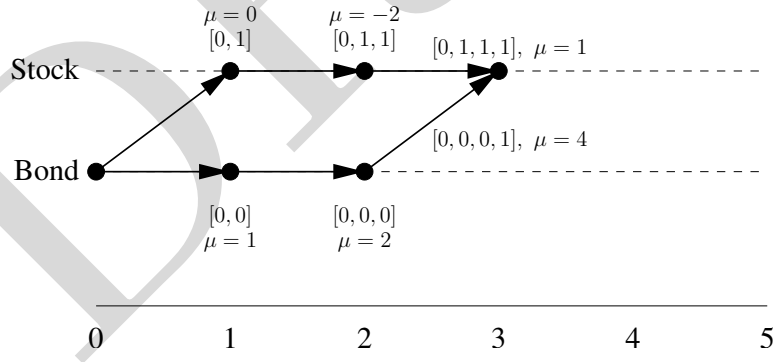
Since both of these options have the same return, we may pick either as the optimal strategy to time 2 ending in stock. Similarily, we can consider the two options for the optimal strategy to time 2 ending in bond,



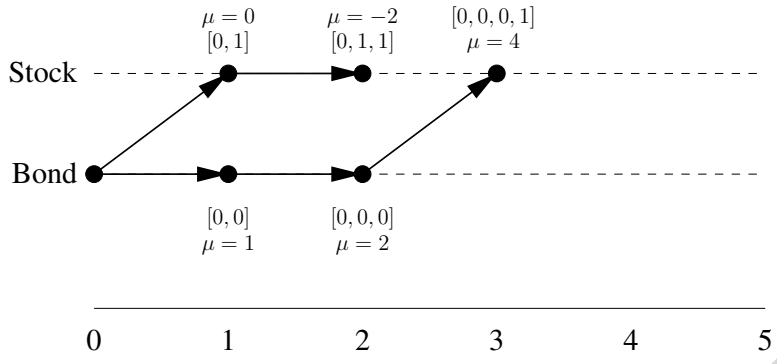
Since the option passing through bond at time 1 has higher return, this is the optimal strategy to time 2 ending in bond,



Continuing, we consider the two options for the optimal strategy to time 3, ending in stock,

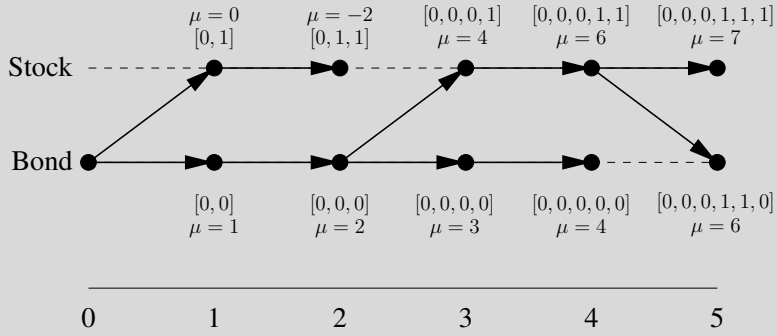


Since the option coming through the optimal strategy to time 2 ending in bond has higher return we have the optimal strategy ending in stock at time 3 is,



Exercise 6.10
Continue the analysis of the example to obtain the optimal strategies ending in stock and bond at time 5, pictorially representing the solutions as above. Give a return optimal strategy ending in bond, and what is its return?

[Answer:



]

When there are constraints on the number of trades, we only need to slightly modify the above argument. We would like to compute the optimal strategy which ends (say) in stock and makes at most K trades. Any such strategy has to be one of two possibilities: it makes at most K trades ending in stock at time t_{i-1}, or it makes at most $K-1$ trades, ending in bond at time t_{i-1}. If it ended in bond, it can only make at most $K-1$ trades because one additional trade will be required to convert from bond at t_{i-1} to stock at t_i. Thus the inductive construction will start with $K = 0$ which is to hold bond. Assuming we have computed all the optimal strategies for $K = k$ to all times $\{t_i\}$, we can then compute all the optimal strategies for $K = k + 1$ to all times.

6.2.3 Unconstrained Return-Optimal Trading Strategies

First we give the main definitions that we will need in the dynamic programming algorithm to compute the optimal strategy. Consider a return-optimal strategy for the first $m+1$ times $\{t_0, \ldots, t_m\}$. Define $\mathcal{S}(m, 0)$ (resp. $\mathcal{S}(m, 1)$) to be a return-optimal strategy ending in bond (resp. stock) at time t_m. Up to time t_1, there is only one strategy ending in bond, and one strategy ending in stock, so $\mathcal{S}(1, 0) = [0, 0]$ and $\mathcal{S}(1, 1) = [0, 1]$ For $\ell \in \{0, 1\}$, let $\mu(m, \ell)$ denote the return of $\mathcal{S}(m, \ell)$, i.e., $\mu(m, \ell) = \mu(\mathcal{S}(m, \ell))$. Let $\textsc{Prev}(m, \ell)$ denote the penultimate position of the optimal strategy

$\mathcal{S}(m,\ell)$ at time t_{m-1}. Note that $\textsc{Prev}(1,1) = \textsc{Prev}(1,0) = 0$, since both optimal strategies to time t_1 started in bond.

We are after $\mathcal{S}(n,0)$, the optimal strategy to time t_n ending in bond. Denote this strategy by $\mathcal{T}^*$. If we know $\textsc{Prev}(m,\ell)$ for $m \geq 1$ and $\ell \in \{0,1\}$, then we can construct $\mathcal{S}(n,0)$ in linear time as follows. First, we have the obvious fact that $\mathcal{T}^*[n] = \mathcal{S}(n,0)[n] = 0$. The previous position is given by $\textsc{Prev}(n,0)$. Suppose $\textsc{Prev}(n,0) = \mathcal{T}^*[n] = 0$, i.e., the previous position was 0. Then the position previous to that is exactly the previous position for the strategy $\mathcal{S}(n-1,0)$ which is $\textsc{Prev}(n-1,0)$. If on the other hand, $\textsc{Prev}(n,0) = \mathcal{T}^*[n] = 0$, i.e., the previous position was 1. Then the position previous to that is exactly the previous position for the strategy $\mathcal{S}(n-1,1)$ which is $\textsc{Prev}(n-1,1)$. More generally, suppose the optimal strategy at time m is $\mathcal{T}^*[m]$. Then the previous position is exactly $\textsc{Prev}(m,\mathcal{T}^*[m])$. We thus have the following backward recursion for $\mathcal{T}^*$

$$\begin{aligned}\mathcal{T}^*[n] &= 0,\\ \mathcal{T}^*[m-1] &= \textsc{Prev}(m,\mathcal{T}^*[m]),\ \text{for } 1 \leq m \leq n.\end{aligned}$$

Thus, a single backward scan is all that is required to compute all the elements in $\mathcal{T}^*$. This backward scan is typically called the backtracking step in a dynamic programming algorithm which is typically the step that is used in constructing the solution in a dynamic programming approach. Note that storing the entire $\textsc{Prev}$ array requires memory that is linear in n. The remainder of the discussion focusses on the computation of the array $\textsc{Prev}(m,\ell)$ for $m \geq 1$ and $\ell \in \{0,1\}$.

The optimal strategy $\mathcal{S}(m,\ell)$ must pass through either bond or stock at time t_{m-1}. Thus, $\mathcal{S}(m,\ell)$ must be the extension of one of the optimal strategies $\{\mathcal{S}(m-1,0), \mathcal{S}(m-1,1)\}$ by adding the position ℓ at time period t_m. More specifically,

$$S(m,\ell) = \begin{cases}[S(m-1,0),\ell] & \text{or,}\\ [S(m-1,1),\ell].\end{cases}$$

In particular, $\mathcal{S}(m,\ell)$ will be the extension that yields the greatest total return. Thus,

$$\mu(m,\ell) = \max\{\mu([S(m-1,0),\ell]), \mu([S(m-1,1),\ell])\}.$$

Using (6.1), we have that

$$\begin{aligned}\mu([\mathcal{S}(m-1,0),\ell]) &= \begin{cases}\mu(m-1,0)+b_m & \ell=0,\\ \mu(m-1,0)+s_m-f_s & \ell=1;\end{cases}\\ \mu([\mathcal{S}(m-1,1),\ell]) &= \begin{cases}\mu(m-1,1)+b_m-f_b & \ell=0,\\ \mu(m-1,1)+s_m & \ell=1.\end{cases}\end{aligned}$$

Using these expressions, we can compute $\mu(m,\ell)$ for $m \geq 1$ and $\ell \in \{0,1\}$ using the following recursion,

$$\mu(m,\ell) = \begin{cases}\max\{\mu(m-1,0)+b_m, \mu(m-1,1)+b_m-f_b\} & \ell=0,\\ \max\{\mu(m-1,0)+s_m-f_s, \mu(m-1,1)+s_m\} & \ell=1.\end{cases}$$

Simultaneously, as we compute $\mu(m,\ell)$, we can also compute $\textsc{Prev}(m,\ell)$ as follows,

$$\begin{aligned}\textsc{Prev}(m,0) &= \begin{cases}0 & \text{if } \mu(m-1,0)+b_m \geq \mu(m-1,1)+b_m-f_b,\\ 1 & \text{otherwise;}\end{cases}\\ \textsc{Prev}(m,1) &= \begin{cases}0 & \text{if } \mu(m-1,0)+s_m-f_s \geq \mu(m-1,1)+s_m,\\ 1 & \text{otherwise.}\end{cases}\end{aligned}$$

It should be evident that if we already know $\mu(m-1,0)$ and $\mu(m-1,1)$, then we can compute $\mu(m,\ell)$ and $\textsc{Prev}(m,\ell)$ for $\ell \in \{0,1\}$ in constant time. Further, we have that $\mu(1,0) = b_1$ and $\mu(1,1) = s_1 - f_s$, and so, by a straight forward induction, we can compute $\mu(m,\ell)$ and $\textsc{Prev}(m,\ell)$ in linear time.

Exercise 6.11
Implement this dynamics programming algorithm as a function which takes as input two return series and the corresponding transactions costs, and outputs an optimal strategy, together with the return of the optimal strategy.

Is it possible to have more than one optimal strategy? If so, what can you say about the returns of the optimal strategies?

The generalization of this algorithm to $N > 2$ instruments is straightforward by suitably generalizing a trading strategy. $\mathcal{S}(m,\ell)$ retains its definition, except now $\ell \in \{0,\dots,N-1\}$. To compute $\mu(m,\ell)$ will need to take a maximum over N terms depending on $\mu(m-1,\ell')$, and so the algirithm will have runtime $O(Nn)$.

One of the assumptions we maintained was the all or nothing assumption. The next exercise shows that we did not lose any generality in doing so.

Exercise 6.12
Show that there always exists an all or nothing trading strategy which is return optimal. in particular, show that for any trading strategy which makes K trades, there is a trading strategy which makes at most K trades with at least as much return. (This also shows that the all or nothing assumption is also not a serious restriction to constrained return optimal trading.)

[Hint: You may want to use induction on n.]

One concern with the unconstrained optimal strategy is that it may make too many trades. It is thus useful to compute the optimal strategy that makes at most a given number of trades. We discuss this next.

6.2.4 Constrained Return-Optimal Strategies

We now suppose that the number of trades is constrained to be at most K. The general approach is similar to the unconstrained case. It is more convenient to consider the number of position switches a strategy makes, which we define as the number of times the position switches. For a valid trading strategy, the number of trades entered equals the number of trades exited, so $k = 2K$. Analogous to $\mathcal{S}(m,\ell)$ in the previous section, we define $\mathcal{S}(m,k,\ell)$ to be the optimal trading strategy to time t_m that makes at most k position switches ending with position ℓ. Let $\mu(m,k,\ell)$ be the return of strategy $\mathcal{S}(m,k,\ell)$, and let $\textsc{Prev}(m,k,l)$ store the pair (k',ℓ'), where ℓ' is the penultimate position of $\mathcal{S}(m,k,\ell)$ at t_{m-1} that leads to the end position ℓ, and k' is the number of position switches made by the optimal strategy to time period t_{m-1} that was extended to $\mathcal{S}(m,k,\ell)$.

Exercise 6.13
How many possible trading strategies are there with k position switches?

The algorithm once again follows from the observation that the the optimal strategy $\mathcal{S}(m,k,\ell)$ must pass through either bond or stock at t_{m-1}. A complication is that if the penultimate position

is bond and $\ell = 0$, then at most k position switches can be used to get to thhe penultimate position, however, if $\ell = 1$, then only at most $k-1$ position switches may be used. Similar reasoning applies if the penultimate position is stock. We thus get the following recursion,

$$\begin{aligned}\mu(m,k,0) &= \max\left\{\mu(m-1,k,0),\ \mu(m-1,k-1,1) - f_b\right\},\\ \mu(m,k,1) &= \max\left\{\mu(m-1,k-1,0) + s_m - f_s,\ \mu(m-1,k,1) + s_m\right\}.\end{aligned}$$

This recursion is initialized with $\mu(m,0,0) = \sum_{i=1}^{m} b_i$ and $\mu(m,0,1) = -\infty$ for $1 \le m \le n$. Once $\mu(m,k,\ell)$ is computed for all m,ℓ, then the above recursion allows us to compute $\mu(m,k+1,\ell)$ for all m,ℓ. Thus, the computation of $\mu(m,k,\ell)$ for $1 \le m \le n$, $0 \le k \le 2K$ and $\ell \in \{0,1\}$ can be accomplished in $O(nK)$ time. As in the unconstrained case, the strategy that was extended gives $\textsc{Prev}(m,k,\ell)$,

$$\begin{aligned}\textsc{Prev}(m,k,0) &= \begin{cases}(k,0) & \text{if } \mu(m-1,k,0) > \mu(m-1,k-1,1) - f_b,\\ (k-1,1) & \text{otherwise.}\end{cases}\\ \textsc{Prev}(m,k,1) &= \begin{cases}(k-1,0) & \text{if } \mu(m-1,k-1,0) + s_m - f_s > \mu(m-1,k,1) + s_m,\\ (k,1) & \text{otherwise.}\end{cases}\end{aligned}$$

Thus, $\textsc{Prev}(m,k,\ell)$ can be computed as we compute $\mu(m,k,\ell)$ in $O(nK)$ time.

The optimal trading strategy T_K^* making at most K trades is then given by $\mathcal{S}(n,2K,0)$, and the full strategy can be reconstructed in a single backward scan using the following backward recursion (we introduce an auxilliary vector κ),

$$\begin{aligned}\mathcal{T}_K^*[n] &= 0,\\ \kappa[n] &= 2K\\ (\kappa[m-1],\mathcal{T}_K^*[m-1]) &= \textsc{Prev}(m,\kappa[m],\mathcal{T}_K^*[m]),\ \text{for } 1 \le m \le n.\end{aligned}$$

Since the algorithm needs to store $\textsc{Prev}(m,k,\ell)$ for all m,k, the memory requirement is $O(nK)$. Once again, it is not hard to generalize this algorithm to work with N instruments, and the resulting run time will be $O(nNK)$.

Exercise 6.14
Implement the dynamic programming algorithm as a function which takes as input the return sequences, the transactions costs and the maximum number of trades and returns the optimal trading strategy together with its return.

6.2.5 Other Work on Optimal Trading

The body of literature on optimal trading is so enormous that we only highlight here some representative papers. The reasearch on optimal trading falls into two broad categories. The first group is on the more theoretical side where researchers assume that instrument prices satisfy some particular model, for example the prices are driven by a stochastic process of known form; the goal is to derive closed-form solutions for the optimal trading strategy, or a set of equations that the optimal strategy must follow. The main drawbacks of such theoretical approaches is that their prescriptions can only be useful to the extent that the assumed models are correct. Our work

does not make any assumptions about the price dynamics to construct ex-post optimal trading strategies.

The second group of research which is more on the practical side is focused on exploring data driven / learning methods for the prediction of future stock prices moves and trading opportunities. Intelligent agents are designed by training on past data and their performance is compared with some benchmark strategies. Our results furnish (i) optimal strategies on which to train intelligent agents and (ii) benchmarks with which to compare their performance.

Theoretical Approaches Boyd et al. in **?** consider the problem of single-period portfolio optimization. They consider the maximization of the expected return subject to different types of constraints on the portfolio (margin, diversification, budget constraints and limits on variance or shortfall risk). Under certain assumptions on the returns distribution, they reduce the problem to numerical convex optimization. Similarily, Thompson in **?** considered the problem of maximizing the (expected) total cumulative return of a trading strategy under the assumption that the asset price satisfies a stochastic differential equation of the form $dS_t = dB_t + h(X_t)dt$, where B_t is a Brownian motion, h is a known function and X_i is a Markov Chain independent of the Brownian motion. In this work, he assumes fixed transaction costs and imposes assumptions **A1, A2, A4** on the trading. He also imposes a stricter version of our assumption **A3**: at any time, the trader can have only 0 or 1 unit of stock. He proves that the optimal trading strategy is the solution of a free-boundary problem, gives explicit solutions for several functions h and provides bounds on the transaction cost above which it is optimal never to buy the asset at all.

Pliska et al. in **?** and Bielecki et al. in **?** considered the problems of optimal investment with stochastic interest rates in simple economies of bonds and a single stock. They characterize the optimal trading strategy in terms of a nonlinear quasi-variational inequality and develop a numerical approaches to solving these equations.

Some work has been done within risk-return frameworks. Berkelaar and Kouwenberg in **?** considered asset allocation in a return versus downside-risk framework, with closed-form solutions for asset prices following geometric Brownian motions and constant interest rates. Liu in **?** consider the optimal investment policy of a constant absolute risk aversion (CARA) investor who faces fixed and proportional transaction costs when trading multiple uncorrelated risky assets.

Zakamouline in **?** studies the optimal portfolio selection problem using Markov chain approximation for a constant relative risk averse investor who faces fixed and proportional transaction costs and maximizes expected utility of the investor's end-of-period wealth. He identifies three disjoint regions (Buy, Sell and No-Transaction) to describe the optimal strategy.

Choi and Liu in **?** considered trading tasks faced by an autonomous trading agent. An autonomous trading agent works as follows. First, it observes the state of the environment. According to the environment state, the agent responds with an action, which in turn influences the current environment state. In the next time step, the agent receives a feedback (reward or penalty) from the environment and then perceives the next environment state. The optimal trading strategy for the agent was constructed in terms of the agent's expected utility (expected accumulated reward).

Cuoco et al. in **?** considered Value at Risk as a tool to measure and control the risk of the trading portfolio. The problem of a dynamically consistent optimal porfolio choice subject to the Value at Risk limits was formulated and they proved that the risk exposure of a trader subject to a Value at Risk limit is always lower than that of an unconstrained trader and that the probability of extreme losses is also decreased.

Mihatsch and Neuneier in **?** considered problem of optimization of a risk-sensitive expected return of a Markov Decision Problem. Based on an extended set of optimality equations, risk-

sensitive versions of various well-known reinforcement learning algorithms were formulated and they showed that these algorithms converge with probability one under reasonable conditions.

Data Driven Approaches Moody and Saffell in **?** presented methods for optimizing portfolios, asset allocations and trading systems based on a direct reinforcement approach, which views optimal trading as a stochastic control problem. They developed reccurent reinforcement learning to optimize risk-adjusted investment returns like the Sterling Ratio or Sharpe Ratio, while accounting for the effects of transaction costs.

Liu et al. in **?** proposed a learning-based trading strategy for portfolio management, which aims at maximizing the Sharpe Ratio by actively reallocating wealth among assets. The trading decision is formulated as a non-linear function of the latest realized asset returns, and the function cam be approximated by a neural network. In order to train the neural network, one requires a Sharpe-Optimal trading strategy to provide the supervised learning method with target values. In this work they used heuristic methods to obtain a locally Sharp-optimal trading strategy. The transaction cost was not taken into consideration. Our methods can be considerably useful in the determination of target trading strategies for such approaches.

6.3 Problems

Chapter 7

Trade Entry

We now develop another application of dynamic programming, to optimal trade execution. This is sometimes called *legging* into a trade. To be concrete, we focus on selling shares in a stock, but the same general approach applies equally well to buying.

The general formulation of the problem is that you have a (usually large) number of shares, K which you would like to sell and the entire trade must be executed over the next n time steps $t = 1, 2, \ldots, n$. Your decision to sell is typically predicated on a market view that the price will be dropping, and so you would like to sell as fast as possible. On the other hand, selling a large amount of stock, will have a *market impact*, which means that as you sell the stock, the price of the stock will change, and can also affect the future market view. When you sell a larger quantities, you have a larger impact, causing the average price at which you sell to be lower. This adverse market impact encourages spreading out your trade. Hence, your market view and your market impact are competing against each other, the former encouraging you to sell as quickly as possible and the latter encouraging you to sell as slowly as possible. Optimally balancing these two competing forces of market view market impact can provide significant profit gain. We formalize the problem in a general way before considering simplifications which we can efficiently solve using dynamic programming approaches.

The number of shares to be sold is K over the times $t = 1, 2, \ldots n$. Suppose that we sell k_i shares at time i, so the exit strategy can be represented by the n-dimensional vector $\mathbf{k} = [k_1, k_2, \cdots, k_n]$, where $k_i \geq 0$ and $\sum_i k_i = K$.

Exercise 7.1
Given K and n, how many possible exit strategies are there?

Before any shares are sold, you have some view as to how the market will behave. Specifically, the no-market impact price p_i at time i is known, which can be summarized in the no-market impact price vector $\mathbf{p} = [p_1, p_2, \cdots, p_n]$. If the trade is to sell K shares, then typically the p_i's are decreasing (one sells if one believes that the market is going down). If you sell according to the strategy $\mathbf{k}$, the prices will change, in particular drop, both as you sell and in the future. We will make some simplifying assumptions as to how this happens.

At time i, suppose that the price is p_i. If you sell k shares at time i, assume that you will execute your k shares at an average price of $p_i - g(k)$, where $g(k)$ is the *execution impact* of selling the k shares. There will also be a *future price impact* due to this sale. In particular, all your future realizations of the price will drop by an amount $f(k)$. Since the price is typically dropping during the execution, the average price for the execution will be higher than the final price after the execution, thus in a practical setting, one usually has that $f(k) \geq g(k)$.

At time i for exit strategy $\mathbf{k}$, let q_i be the amount by which the price has already dropped,

$$q_i = \sum_{j=1}^{i-1} f(k_j).$$

Let c_i be the proceeds from selling k_i shares at time i at an average price of $p_i - q_i - g(k_i)$. Then,

$$c_i = k_i(p_i - q_i - g(k_i)).$$

We can thus compute the proceeds from the entire sale,

$$\begin{aligned} C(\mathbf{k}) &= \sum_{i=1}^{n} c_i, \\ &= \sum_{i=1}^{n} k_i(p_i - q_i - g(k_i)), \\ &= \sum_{i=1}^{n} k_i(p_i - g(k_i)) - \sum_{i=1}^{n}\sum_{j=1}^{i-1} k_i f(k_j). \end{aligned}$$

The functions g, f are specified as vectors: $\mathbf{g} = [g_0, g_1, \ldots, g_K]$ and $\mathbf{f} = [f_0, f_1, \ldots, f_K]$. Note that $g_0 = f_0 = 0$. Given $\mathbf{p}, \mathbf{g}, \mathbf{f}$ and K, the task is to maximize C over all strategies $\mathbf{k} \geq 0$ such that $\sum_i k_i = K$. We can assume that $\mathbf{k}$ is a non-negative integer vector, because one can only execute an integral number of shares at a time.

Exercise 7.2

Suppose that the price vector is a constant vector equal to 100. Assume that $K = 10$ and that $f(k) = g(k) = 0$ if $0 \leq k \leq 1$ and 1 otherwise.

(a) Compute C, q_i for the strategy $\mathbf{k} = [4, 3, 2, 1, 0, 0, 0, 0, 0, 0]$.

(b) What is the optimal exit strategy and corresponding to it, what is C.

We now develop a dynamic programming solution for obtaining the optimal exit strategy. Suppose that you are at time i and the price has already dropped by an amount q and you have k shares remaining to sell. Let $C^*(k, q, i)$ denote the maximum possible proceeds from optimally executing the remainder of the trade (k shares) starting at time i.

We would like to know for starters, what is $C^*(K, 0, 1)$, the maximum possible proceeds from the sale of the K shares, in addition to the exit strategy to obtain that maximum. We begin by observing that at time n, there is nothing to be done but sell all the remaining k shares no matter what the price drop has been, so,

$$C^*(k, q, n) = k(p_n - q - g(k)).$$

Now consider $C^*(k, q, i)$ for a time $i < n$. Of the k shares remaining to be sold, there are only $k+1$ possibilities, corresponding to selling $0, 1, \ldots, k$ shares at time i. After selling $0 \leq \ell \leq k$ shares at time i, the maximum amount of money which can be made is

$$\ell(p_i - q - g(\ell)) + C^*(k - \ell, q + f(\ell), i + 1).$$

To obtain $C^*(k,q,i)$, which is the *maximum* amount of money that can be made at time i, we should take the maximum over all possible choices of ℓ to obtain,

$$C^*(k,q,i) = \max_{0\le\ell\le k} \left\{\ell(p_i - q - g(\ell)) + C^*(k-\ell, q+f(\ell), i+1)\right\}. \tag{7.1}$$

The value of ℓ attaining the maximum is needed for reconstructing the optimal strategy through the usual process of backtracking in a dynamic program. Let $\ell^*(k,q,i)$ be this value of ℓ,

$$\ell^*(k,q,i) = \operatorname*{argmax}_{0\le\ell\le k} \left\{\ell(p_i - q - g(\ell)) + C^*(k-\ell, q+f(\ell), i+1)\right\}. \tag{7.2}$$

This backward induction allows us to compute C^*, ℓ^* at time i for all k, q if we have already computed C^*, ℓ^* at time $i+1$ for all k, q. Since we know C^* at time n for all k, q, we can initialize the process at time n and continue all the way back to time 1, where we need $C^*(K,0,1)$. Note that $\ell^*(k,q,n) = k$ since there is nothing more to do than sell off all the remaining shares.

To get the optimal strategy, we use a forward induction on ℓ^*. Clearly $k_1 = \ell^*(K,0,1)$. Let κ_i be the number of shares remaining to execute in the optimal strategy at time i, $\kappa_1 = K$. Note that the price drop at time 1 is 0, $q_1 = 0$. In general, if there are κ_i shares to execute at time i and the price has dropped by q_i, then the optimal strategy sells $k_i = \ell^*(\kappa_i, q_i, i)$ shares, and we update $\kappa_{i+1} = \kappa_i - k_i$ and $q_{i+1} = q_i + f(k_i)$. Summarizing, we have that $\kappa_1 = K, q_1 = 0$ and for $i \ge 1$,

$$\begin{aligned} k_i &= \ell^*(\kappa_i, q_i, i), \\ \kappa_{i+1} &= \kappa_i - k_i, \\ q_{i+1} &= q_i + f(k_i). \end{aligned}$$

This forward induction allows us to compute k_i for $i \ge 1$.

Exercise 7.3
We will explore the maximum possible market impact that one can have when executing the trade. The total market impact is $q = \sum_i f(k_i)$, the total amount by which the market was moved. We would like to compute the maximum possible value of q under the restriction that $\sum_i k_i = K$. Thus define

$$q^*(K) = \max_{\sum_i k_i = K} \sum_i f(k_i).$$

Give a dynamic programming algorithm to compute $q^*(K)$.

[Hint: Show that $q^*(K) = \max_{1\le k\le K}\{f(k) + q^*(K-k)$, with $q^*(0) = 0$.]

7.0.1 Computational Considerations

For the particular model we are considering, we can get a more efficient algorithm for computing the optimal strategy by looking a little more closely at C^*.

Exercise 7.4
Show that $C^*(k,q,i) = C^*(k,0,i) - kq$. [Hint: Induction.]

Exercise 7.5
Use the previous exercise to show that $\ell^*(k,q,i)$ is independent of q, i.e. $\ell^*(k,q,i) = \ell^*(k,0,i)$.

The previous two exercises show that we can rewrite (7.1) and (7.2) because $C^*(k,q,i) = \mathcal{C}^*(k,i) - kq$, where $\mathcal{C}^*(k,i) = C^*(k,0,i)$:

$$\begin{aligned} C^*(k,q,i) &= \mathcal{C}^*(k,i) - kq, \\ \mathcal{C}^*(k,i) &= \max_{0\le\ell\le k} \left\{\ell p_i - \ell g(\ell) - (k-\ell)f(\ell) + \mathcal{C}^*(k-\ell, i+1)\right\}, \\ \ell^*(k,i) &= \operatorname*{argmax}_{0\le\ell\le k} \left\{\ell p_i - \ell g(\ell) - (k-\ell)f(\ell) + \mathcal{C}^*(k-\ell, i+1)\right\}. \end{aligned}$$

The boundary condition for $\mathcal{C}^*$ is $\mathcal{C}^*(k,n) = k(p_n - g(k))$. We reconstruct the optimal strategy from ℓ^*. In particular, $\kappa_1 = K$ and for $i \ge 1$,

$$\begin{aligned} k_i &= \ell^*(\kappa_i, i), \\ \kappa_{i+1} &= \kappa_i - k_i, \end{aligned}$$

The algorithm needs $O(nK)$ memory to store $\mathcal{C}^*(k,i)$ and $\ell^*(k,i)$ for $1 \le k \le K$ and $1 \le i \le n$. At each time step i, the K numbers $\mathcal{C}^*(k,i)$ for $1 \le k \le K$ need to be computed. Computing each number $\mathcal{C}^*(k,i)$ involves taking a maximum over k quantities. So the total computation at time step i is $\sum_{k=0}^{K} O(k) \in O(K^2)$. Therefore, the total computation is in $O(nK^2)$.

Exercise 7.6

Consider the following exit scenario. You wish to sell 10 shares (K=10) of a stock by time $T = 10$. Assume the stock price is decreasing linearly, $P_t = 100 - \alpha t$, where α is a parameter we will play with. Assume that the impact function is linear, $f(x) = \beta x$ where β is also a parameter to we will play with. You can only sell an integral number of shares at a time.

(a) Using a brute force search for the optimal exit strategy, how many exit strategies must be tested?

(b) Implement efficiently the dynamic programming algorithm to compute the optimal exit and determine the optimal exit strategy together with the maximum proceeds from the sale when $\alpha = \{0, 1, 2\}$, with $\beta = 1$. Repeat with $\beta = 2$.

(c) Explain intuitively what is going on.

7.0.2 Market Impact, Exectution Impact and the Order Book

Our discussion accommodates arbitrary price and execution impact functions. These impact functions can be computed from the order book and its dynamics. In particular, since we have been focussing on the sale of K shares, we should look at the bid side order book.

We postulate a very simple model for the order book dynamics. In particular, suppose that the zero impact market view $\mathbf{p}$ gives the top level of the bid side order book, i.e., the highest price at which someone is willing to buy.

Assume the order book has an equilibrium state which it can restore over the course of one time step. The order book state describes the orders which have been placed on the bid stack using a function F that specifies the number of orders at a particular price at or below the bid price. In particular, let p be the bid price (top level of the order book). Let δ be the tick size, the minimum possible difference between prices (for example $\delta = 1$ cent). The function $F(i)$ (for $\delta \ge 0$) which specifies the bid stack state is the number of bid orders with price at $p - i\delta$. Thus,

$$F(i) = \text{number of bid orders with price } p - i\delta.$$

So, for example, $F(0)$ is the number of orders placed at the bid. Typically (in equilibrium) the number of orders gets smaller (the order book gets thinner) as you move deeper into the bid stack.

The market impact for an order of size k is obtained by removing the k orders with highest prices

and the top level of the bid stack after these k orders are removed is the price after market impact. Thus, for example, if $k = F(0)$, then $f(k) = \delta$. Define the cumulative sequence $G(i) = \sum_{j=0}^{i} F(j)$. Then we obtain the market impact function as

$$f(k) = \begin{cases} 0 & 0 \le k < G(0), \\ \delta & G(0) \le k < G(1), \\ 2\delta & G(1) \le k < G(2), \\ \vdots & \\ i\delta & G(i-1) \le k < G(i). \end{cases}$$

Note that if $k > \sum_{i=0}^{\infty} F(i)$, then $f(k) = \infty$. We can now compute the average execution price for an order of size k, and hence the execution impact function $g(k)$ using the following logic. The first $F(0)$ shares will be sold at price p. The next $F(1)$ shares will be sold at price $p - \delta$. The next $F(2)$ shares will be sold at price $p - 2\delta$, and so on.

Exercise 7.7

Show that the execution impact function is given by

$$g(k) = \begin{cases} 0 & 0 \le k \le G(0), \\ \frac{\delta}{k}\left[\sum_{i=0}^{i-1} iF(i) + i(k - \sum_{i=0}^{i-1} F(i))\right] & G(i-1) < k \le G(i). \end{cases}$$

Typically, $F(i)$ is a non-increasing function. Some useful examples are the uniform order book, where $F(i) = \beta$; the linear order book, $F(i) = \lceil \max(1, \beta - \gamma i) \rceil$; polynomial decay, $F(i) = \lceil \max(1, \beta/(1+i)^{\rho}) \rceil$; exponential decay, $F(i) = \lceil \max(1, \beta e^{-\rho i}) \rceil$;

Exercise 7.8

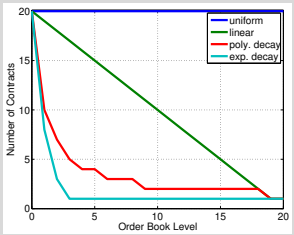
For the four types of order book state,

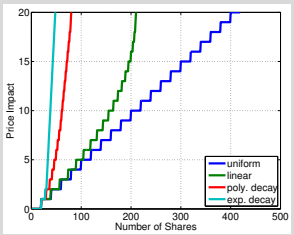
$$F(i) = \beta, \quad F(i) = \lceil \max(1, \beta - \gamma i) \rceil, \quad F(i) = \left\lceil \max(1, \tfrac{\beta}{(1+i)^{\gamma}}) \right\rceil, \quad F(i) = \lceil \max(1, \beta e^{-\gamma i}) \rceil,$$

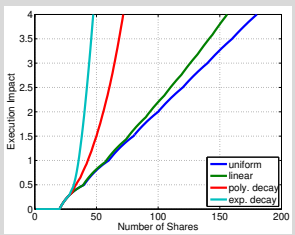
compute $f(k)$ and $g(k)$, giving plots. In all cases, $F(0) = \beta$.

[Answer: For the uniform order book (the other cases are more complicated), $F(i) = \beta$ and:

$$\begin{aligned} f(k) &= \delta \left\lfloor \frac{k}{\beta} \right\rfloor, \\ g(k) &= \delta \left[\left\lceil \frac{k}{\beta} \right\rceil - 1 - \frac{\beta}{2k} \left\lfloor \frac{k}{\beta} \right\rfloor \left(\left\lceil \frac{k}{\beta} \right\rceil - 1 \right) \right]. \end{aligned}$$







]