Algorithmic Trading

Arthur Li

February 1, 2025

Contents

1 Preface and Prerequisites

1.1 Brief Overview

Lorem Ipsum
To be completed once most of the book is done

1.2 Reading Roadmap

Lorem Ipsum. To be completed once most of the book is done

This content builds upon the foundational works of Rishi K. Narang (2013), Raja Velu (2020), and Marcos Lopez Prado (2018), among others, whose insights form the backbone of our discussion.

1.3 Overview of Systematic Investments

The figure below illustrates a live, production-level trading strategy. (Note that the diagram does not include ancillary components—such as research tools—that are also essential for building a complete strategy.)

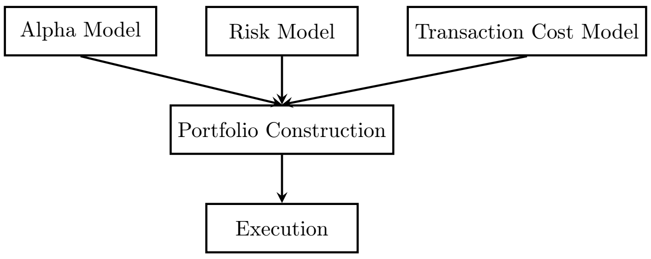


Figure 1: Live Production Trading Strategy Overview

At its core, the trading system is organized into three primary modules:

i. **Alpha Model:** Predicts the future behavior of the instruments under consideration, thereby generating directional alpha.

ii. **Risk Model:** Limits exposure by controlling risk factors that might not generate returns but could lead to losses (i.e., by setting directional exposure limits on asset classes).

iii. **Transaction Cost Model:** Assesses whether the trading costs required to transition from the current portfolio to a new one are acceptable within the portfolio construction framework.

These models feed into a portfolio construction model that balances profitability and risk to determine the optimal portfolio configuration. This model identifies the necessary trade adjustments, which are then executed by the system.

The execution model receives the required trades and, based on inputs such as trade urgency and current market liquidity, carries out the transactions in an efficient and cost-effective manner.

Method 1.3.1. *Chains of Production for Alpha Signals*
The production process for generating alpha signals consists of the following stages:

i. **Data Curation:** Involves collecting, cleaning, indexing, storing, adjusting, and delivering data to the production pipeline. This stage requires expertise in market microstructure and data protocols (e.g., FIX).

ii. **Feature Analysis:** Converts raw data into meaningful signals through techniques in information theory, signal extraction, visualization, labeling, weighting, classification, and feature importance assessment. Feature analysts compile and catalogue these insights.

iii. **Strategy Development:** Transforms informative features into actionable investment algorithms. Strategists mine feature libraries for ideas, blending deep financial market knowledge with data science to develop transparent (white-box) strategies—even if some features originate from black-box methods.

iv. **Back-Testing:** Evaluates the investment strategy under various scenarios. This process demands rigorous empirical analysis and includes metadata on how the strategy was formulated.

v. **Deployment:** Integrates the strategy code into the production system. This step is managed by algorithm specialists and mathematical programmers to ensure fidelity to the prototype while minimizing latency.

vi. **Portfolio Oversight:** Monitors the strategy post-deployment throughout its lifecycle:

   1. **Embargo:** Initially, the strategy is run on post-backtest data. If performance is consistent with backtesting, the strategy advances to the next phase.
   2. **Paper Trading:** The strategy is executed on a live feed, simulating real trading conditions. This phase accounts for data parsing, calculation delays, execution lags, and other operational latencies.
   3. **Graduation:** The strategy is allocated real capital, either individually or as part of an ensemble, with detailed evaluations of risk, returns, and costs.
   4. **Re-allocation:** Allocations are dynamically adjusted based on live performance. Initial allocations are small and may be increased as the strategy proves itself, with a subsequent reduction if performance declines.
   5. **Decommission:** If the strategy underperforms over an extended period, it is phased out.

1.3.1 Alpha Models Overview

Alpha models are designed to test and exploit theories about market behavior. These fall into two categories:

- **Theory-Driven Models**, which derive signals from economic or behavioral principles, and
- **Data-Driven Models**, which rely on complex data mining and statistical techniques.

In practice, practitioners often blend several models together to capture multiple facets of market dynamics.

Definition 1.3.2. *Theory-Driven Models*
Theory-driven models use fundamental principles to explain and predict market behavior. These include:

i. **Trend Following:** Assumes that once a market trend is established, it will persist long enough to be identified and exploited. As supporting data accumulate for a bullish or bearish outlook, more participants join the trend, shifting the asset price to a new equilibrium. Note that strategies such as moving average crossovers often yield less than a one-to-one return relative to the downside risk because market behavior can be unstable and episodic.

ii. **Mean Reversion:** Based on the idea that asset prices will revert to their historical average after deviating. Short-term liquidity imbalances can force prices to move abruptly, but these are typically followed by corrective movements. Statistical arbitrage, which bets on the convergence of prices between similar stocks that have diverged, is a common application of this strategy.

iii. **Value/Yield:** Evaluates securities by comparing fundamental ratios to their market price (often using an inverted ratio for consistency). A higher yield indicates a relatively cheaper instrument. This approach underpins *carry trades*, where investors buy undervalued assets and sell overvalued ones. In *Quant Long Short (QLS)* strategies, stocks are ranked on factors like value, and long positions are taken in the most attractive stocks while shorting the least attractive.

iv. **Growth:** Focuses on the future or historical growth potential of an asset. Forward-looking growth expectations are central, with the belief that companies with strong growth will increasingly dominate their sectors. Macro-level growth factors can drive foreign exchange decisions, while micro-level factors are key for individual companies.

v. **Quality:** Emphasizes capital preservation by favoring high-quality assets—those with robust earnings, sound balance sheets, and low leverage—over lower quality ones.

While theory-driven models incorporate clear economic rationale, **data-driven models** use advanced statistical and machine learning methods to extract patterns directly from data. These techniques, though more mathematically complex and often applied in high-frequency environments, can identify subtle market signals without relying on traditional economic theory.

Method 1.3.3. *Strategy Parameters*
Implementing an alpha strategy requires careful specification of several parameters:

i. **Forecast Target:** Define what the model predicts – whether it is the direction, magnitude, duration of a move, or even the probability of a particular outcome. A stronger signal (in terms of higher expected return or likelihood) typically warrants a larger position.

ii. **Time Horizon:** Forecast horizons can range from microseconds to years. Short-term strategies usually involve a high frequency of trades, whereas long-term strategies trade less frequently.

iii. **Bet Structure:** Models may generate absolute forecasts or relative forecasts (comparing one instrument to another). For relative forecasts, grouping assets into pairs or clusters (e.g., sectors) is common. While pair trading allows for precise comparisons, larger groupings can isolate idiosyncratic movements from overall market trends. Grouping can be achieved through statistical methods or heuristic industry classifications, each with its own tradeoffs.

iv. **Investment Universe:** Selection is based on factors such as geography, asset class, and instrument type. High liquidity is essential for reliable transaction cost estimation and consistent behavior. Consequently, common stocks, futures (on bonds and equity indices), and forex are frequently modeled, while more volatile instruments (e.g., certain biotech stocks) are typically excluded.

v. **Model Specification:** This defines the mathematical structure of the strategy, including any machine learning or data mining techniques used. Regular model refitting is necessary to adapt to changing market conditions, although it may increase the risk of overfitting.

vi. **Run Frequency:** This parameter sets how often the model is executed, ranging from monthly to real time. A higher run frequency increases transaction costs and sensitivity to noise, whereas a lower frequency can lead to larger trades that might themselves influence market prices.

Method 1.3.4. *Blending of Models*
Integrating multiple alpha signals can be accomplished using different approaches:

i. **Linear Models:** Assume that factors are independent and additive. Multiple regression techniques are typically employed to determine the weight of each factor.
ii. **Nonlinear Models:** Used when factor relationships are interdependent or evolve over time. Conditional models adjust the weight of one factor based on the performance of another, while rotational models dynamically shift weights according to recent performance metrics.
iii. **Machine Learning Models:** Although developing machine learning strategies requires substantial effort in data curation, infrastructure, feature engineering, and backtesting, these models can capture micro-level signals that traditional econometric methods may miss. Despite the complexity, ML approaches are becoming more prevalent as computational resources improve.

1.3.2 Risk Models

Risk models are indispensable tools in algorithmic trading, providing a quantitative framework to assess, monitor, and manage the inherent risks of financial markets. They not only serve to measure exposure but also guide portfolio construction, hedging strategies, and overall risk control.

Method 1.3.5. *Factor-Based Models*
Factor-based models decompose asset returns into contributions from systematic factors and idiosyncratic components. The most common factors include:

i. **Market Factor:** Captures the overall movement of the market.
ii. **Size Factor:** Reflects the differential risk associated with companies of varying market capitalizations.
iii. **Value Factor:** Accounts for risk due to discrepancies between market prices and fundamental valuations.
iv. **Momentum Factor:** Measures the tendency of asset prices to continue in their current trajectory.

This allows traders to understand which elements drive portfolio risk and adjust exposures accordingly.

Method 1.3.6. *Statistical Models*
Statistical risk models leverage historical data and probabilistic techniques to quantify risk parameters.

i. **Historical Simulation:** Directly computing risk metrics from past return distributions.
ii. **Monte Carlo Simulation:** Generating a large number of potential future return scenarios to estimate risk under diverse conditions.
iii. **Parametric Methods:** Employing analytical formulas based on assumed return distributions to calculate key risk measures.

These are useful for dynamically updating risk assessments as new market data become available.

Method 1.3.7. *Limiting Size of Risk*
Quantitative risk models are designed to limit the size of exposures to enhance return consistency.

i. **Constraint Mechanisms:**
    a. **Hard Constraints:** Absolute limits imposed on position sizes, which may be set arbitrarily.
    b. **Penalty Functions:** Flexible constraints where positions can exceed the limit if the alpha model forecasts significantly higher returns, with penalties applied for surpassing the prescribed levels.
ii. **Risk Measurement Approaches:**
    a. **Longitudinal Analysis:** Evaluates risk by assessing the volatility of an instrument over time.
    b. **Dispersion Analysis:** Measures risk by analyzing the correlation or covariance between assets.

These methods can be applied to single positions, groups of positions (such as sectors or asset classes), different types of risk exposures, and overall portfolio leverage.

Method 1.3.8. *Limiting the Types of Risk*
To eliminate unintentional exposure as there is no expectation of being compensated sufficiently for accepting them. This can be achieved the following measures:

i. **Market Exposure Restriction:** Focus on specific market segments to avoid undue exposure to volatile or unpredictable markets.

ii. **Leverage Management:** Limit the use of leverage by enforcing conservative leverage ratios to mitigate amplified losses.

iii. **Stop-Loss Policies:** Implement strict stop-loss rules to automatically exit positions that breach predetermined loss thresholds.

iv. **Position Size Controls:** Cap individual position sizes relative to overall portfolio risk, ensuring no single trade unduly influences portfolio performance.

v. **Regular Risk Assessments:** Continuously monitor risk metrics and adjust risk parameters to reflect evolving market conditions.

1.3.3 Transaction Cost Models

A trade is executed only if it improves the probability or magnitude of a return (as predicted by alpha model) or reduces likelihood or extent of a loss (as determined by risk model). However, this improvement must be greater than cost of trading. Note that transaction cost model is intended not to minimize trading costs directly, but rather to inform the portfolio construction engine of the costs associated with executing a given trade.

Remark 1.3.9. *Transaction Cost Components*

i. **Commissions and Fees:** These are payments made to brokerages (for accessing other market participants), exchanges (for enhanced transaction security), and regulators (for maintaining operational infrastructure). In the context of quantitative trading, where bank infrastructure is utilized, brokerage commissions tend to be minimal on a per-trade basis.

   Brokers also charge clearing and settlement fees. *Clearing* involves activities such as regulatory reporting and monitoring, tax handling, and failure management, all of which occur before settlement. *Settlement* is the final delivery of securities in exchange for full payment.

ii. **Slippage:** This refers to the change in price between the moment the trading system decides to execute a transaction and the time the order is actually sent to the exchange. Trend-following strategies tend to experience more slippage because the assets are already moving in the desired direction, whereas mean-reverting strategies are less affected. Reduced latency to market minimizes slippage, while higher asset volatility increases it.

iii. **Market Impact:** This measures the extent to which an order influences the market through its demand for liquidity. The market impact remains uncertain until after the trade is executed. Additionally, there can be an interaction between slippage and market impact (i.e., selling during an upward trending market).

Definition 1.3.10. *Types of Transaction Cost Models*

i. **Flat Model:** Assumes that the cost of trading remains constant regardless of the order size. This model is appropriate when the traded size is nearly uniform and liquidity remains stable.

ii. **Linear Model:** In this model, the trading cost increases at a constant rate relative to the order size. It provides a better estimate than the flat model.

iii. **Piece-Wise Linear Model:** This approach employs piece-wise linear functions to model costs. It strikes a balance between simplicity and accuracy, offering improved precision compared to flat or linear models.

iv. **Quadratic Model:** Although the most computationally intensive, the quadratic model delivers the highest accuracy in cost estimation.

1.3.4 Portfolio Construction Models

Portfolio construction models generally come in two major forms: rule-based models and optimisers. Rule-based models are built on heuristics—they can be very simple or quite complex—and are often derived from human experience and trial-and-error. In contrast, optimiser-based models rely on an objective function and use algorithmic methods to achieve the desired portfolio outcome.

Definition 1.3.11. *Rule-Based Models*

i. **Equal Position Weighting:** Applied when portfolio manager believes that once a position is deemed good enough to own, no further information is needed to decide its size. Strength of signal does not influence the weighting. Model assumes there is enough statistical power to predict not only direction but also magnitude relative to other forecasts within the portfolio. As a result, portfolio may place a few large bets on the strongest forecast and many smaller bets on less dramatic signals; however, this can lead to taking excessive risk in an idiosyncratic event on an apparently attractive position, which may cause adverse selection bias.

ii. **Equal Risk Weighting:** Strategy adjusts position sizes inversely to their volatility or another measure of risk. More volatile positions has smaller allocations, whereas less volatile positions has more allocation. Because the unit of risk is typically a backward-looking measure, such as historical volatility, if volatility shifts over time, the model might be misled.

iii. **Alpha-Driven Weighting:** Position size is primarily determined by alpha model. Alpha signal guides size of position, usually subject to predetermined size limits. Additional constraints often include limits on the total bet size within a group. A function may also be used to relate the forecast's magnitude to the position size. In futures trend following, it might suffer from sharp drawdowns, as it heavily relies on the accuracy of the alpha signals.

iv. **Decision-Tree Weighting:** A decision-making process is used to determine the allocation for each instrument based on both the type of alpha model and the type of instrument. Constraints, such as percentage limits for allocation, may also be imposed. However, as more alpha models or instrument types are introduced, the decision tree can grow significantly in complexity.

Remark 1.3.12. *Optimisers Models Parameters*
The pioneering model in optimiser-based portfolio construction is Harry Markowitz's mean variance optimisation (MVO), which is founded on the principles of modern portfolio theory (MPT). The main inputs to these models include the expected return (mean), asset variance, and the expected correlation matrix. Other inputs typically involve the portfolio's size in currency terms, the desired risk level (such as volatility or expected drawdown), and additional constraints like liquidity and universe limits. Model uses an objective function paired with an algorithm that seeks to maximise the portfolio's return relative to its volatility.

i. **Expected Return:** Derived from alpha models, this input captures not only the direction but also the magnitude of the expected returns.

ii. **Expected Volatility:** Typically estimated using stochastic volatility forecasting methods (i.e., GARCH), this input accounts for periods of high and low volatility along with occasional jumps.

iii. **Expected Correlation:** Given that instrument correlations can fluctuate over time, it is often more effective to group similar assets together before calculating the correlations within each group.

Method 1.3.13. *Optimisation Techniques*

i. **Unconstrained Optimisation:** Most basic form of optimisation with no constraints applied. Might result in a portfolio that invests all available capital in a single instrument—specifically, the one with the highest risk-adjusted return.

ii. **Constrained Optimisation:** Constraints such as position limits or limits on groups of instruments are applied. These constraints can sometimes have a greater influence on the portfolio construction than the optimiser itself.

iii. **Black-Litterman Optimisation:** This combines investor forecasts with a measure of confidence in those forecasts, blending them with historical data. It adjusts the historically observed correlation levels by incorporating the investor's return expectations for various instruments.

iv. **Grinold and Kahn's Approach:** Instead of directly sizing positions, this constructs a portfolio of signals. It creates factor portfolios, each typically based on a single type of alpha forecast. After back-testing these factor portfolios, their return series are then treated as instruments for the optimiser. Since the number of factor portfolios is usually limited (typically no more than 20), the optimisation problem becomes more manageable. This method also allows for the inclusion of a risk model, transaction cost model, overall portfolio size, and risk targets as additional inputs.

v. **Resampled Efficiency:** Seeks to improve the input parameters for optimisation by reducing the over-sensitivity to estimation errors. It does so by employing Monte Carlo simulation to resample data, thereby mitigating the estimation error in the inputs.

vi. **Data-Mining Approaches:** Leverage machine learning techniques—such as supervised learning or genetic algorithms—to explore a wide range of potential portfolios and identify the optimal one.

1.3.5 Execution Model

There are two primary methods to execute a trade: electronically or via a human intermediary. In electronic execution, direct market access (DMA) is employed, allowing traders to leverage the brokerage firm's infrastructure and exchange connectivity to trade directly on electronic markets.
Execution algorithms can be developed in-house, sourced from brokers, or obtained from third-party vendors. Brokerages also provide portfolio bidding services. In these arrangements, the "blind" portfolio for the transaction is characterized by features such as valuation ratios of long and short positions, sector allocation, market

capitalization, and similar metrics. The broker then quotes a fee—expressed in basis points relative to the gross market value of the portfolio traded—which offers the trader a measure of certainty. Once an agreement is reached, the broker collects the fee and assumes the risk of executing the portfolio at future market prices, which may turn out to be more or less favorable than the initially guaranteed prices..

Remark 1.3.14. *Order Execution Algorithm Parameters*

i. **Aggressive vs Passive:** Algorithm decides whether to use an aggressive or passive order based on how immediately the trade must be executed. Market orders are inherently aggressive. Limit order placed at current best quote is relatively aggressive; limit order positioned below current bid is considered passive. Many exchanges reward liquidity providers for placing passive orders, while charging traders for consuming liquidity. When an order crosses the spread, it effectively uses liquidity from another trader's passive order, thereby reducing available liquidity. In return, if the passive order is executed, the trader can benefit from a better transaction price along with a commission rebate from the exchange.
   Momentum-based strategies typically favour aggressive orders, whereas mean reversion strategies lean towards passive orders. A strong, reliable signal justifies a more aggressive execution, while a weaker or less certain signal may call for a more passive approach. An intermediate strategy might involve placing limit orders between the best current bid and offer.

ii. **Large vs Small Order:** Large order may be divided into several smaller orders to reduce market impact, though this carries risk of adverse price movements. The optimal size of each order segment is determined by estimates from a transaction cost model and an analysis of the appropriate level of aggressiveness.

iii. **Hidden vs Visible Order:** Visible order discloses certain trading intentions, whereas a hidden order conceals this information, helping to prevent market imbalances; however, hidden orders typically suffer from lower execution priority. In algorithmic trading, the practice of using hidden orders—where a large order is segmented into many smaller parts, with most being placed as hidden—is known as "iceberging."

iv. **Order Routing:** When multiple liquidity pools exist for same instrument, smart order routing systems are used to determine most suitable pool for executing a given order. These systems evaluate factors such as depth of liquidity on various electronic communication networks (ECNs) and connectivity speeds.

v. **Cancelling and Replacing Orders:** Traders may submit a large number of orders without the expectation of execution, only to rapidly cancel and replace them. This practice helps gauge the market's response to changes in order book depth, offering insights into potential profitable patterns. For example, if a trader intends to buy a significant number of shares, they might place numerous small orders at prices further from the market and then cancel them, thereby improving market perception.

Definition 1.3.15. *High Frequency Trading*
High Frequency Trading (HFT) involves alpha-driven strategies that focus on extremely short-term bets—often executed in seconds or less—referred to as *microstructure alphas*. These strategies analyze liquidity patterns in the order book. Larger quantitative groups may also incorporate these insights into their execution models to reduce the costs associated with entering trades. Even marginal improvements per trade can accumulate significantly over time. However, pursuing microstructure alpha as a standalone high frequency strategy necessitates substantial investments in both infrastructure and research.
Machine learning techniques may be applied to detect patterns in how other market participants execute orders. When competitors' execution models are less refined, their patterns become more apparent, allowing an ML strategy to exploit these patterns in the future. Short-term patterns tend to exhibit a degree of stability.

Definition 1.3.16. *HFT Shark Strategy*
This is designed to detect large orders that have been fragmented (or "iceberged") by observing the rapid filling of a series of very small trades. Quick execution of these small orders can indicate the presence of a large hidden order. The strategy then involves front-running this iceberg order by placing visible trades ahead of it. Consequently, the iceberg order is forced to move market prices upward in order to execute its trades. Once the iceberg order is fully executed, the resulting favorable price movement enables the shark to exit its position quickly, thereby securing a relatively risk-free profit.

Remark 1.3.17. *HFT Trading Infrastructure*
By using a broker that acts as a trading agent, traders can offload infrastructure requirements and avoid dealing directly with regulatory and other operational constraints. High frequency strategies may also employ colocation or sponsored access. In a colocation setup, traders place their trading servers in close physical proximity to the exchange to minimize latency.
The Financial Information eXchange (FIX) protocol is the standard for real-time electronic communication among market participants. Although open source FIX engines are available, high frequency traders often develop their own proprietary FIX engines to ensure optimal execution speeds.

1.3.6 Research

Definition 1.3.18. *Scientific Method*

1. Researcher observe a phenomenon in the market and construct a theory.
2. Researcher seeks out information to test the theory.
3. Researcher tests the theory, and with enough confidence, risk some capital on the validity of the theory.

Remark 1.3.19. *Sources of Alpha Idea Generation*

1. Observing the market, using the scientific method to test the theory
2. Academic literature, requiring significant time to read academic journals, working papers, and conference presentations for ideas. Literature from other fields such as astronomy, physics, or psychology, may provide ideas relevant to quant finance problems.
3. Migration of a researcher or portfolio manager from one quant shop to another.
4. Lessons from activities of discretionary traders

Remark 1.3.20. *Model Quality Assessment*

i. **Cumulative Profit Graph:** A smooth profit curve is ideal; if the graph shows long periods of inactivity or exhibits sharp, erratic losses and gains, it may signal issues with the model.

ii. **Average Annual Rate of Return:** Indicates the historical performance level of the strategy.

iii. **Variability of Returns:** Lower variability in returns is preferable, as it suggests consistency. Examining the "lumpiness"—the share of total returns derived from periods significantly above average—can further measure return consistency.

iv. **Peak-to-Valley Drawdowns:** Measures maximum decline from any cumulative peak in the profit curve. Lower drawdowns, along with shorter recovery periods after drawdowns, reflect a more robust strategy.

v. **Predictive Power:** The R-squared statistic can be employed to assess how much of the variability in the predicted asset is explained by the model. For example, an exceptionally high R^2 (around 0.05 out of sample) may warrant further scrutiny. Additionally, bucketing instrument returns by deciles can help verify whether the model categorizes them accurately.

vi. **Percentage of Winning Trades and Winning Time Periods:** Determines whether the strategy relies on a small number of highly profitable trades or on a larger volume of moderately successful trades.

vii. **Risk-Adjusted Return Ratios:** Evaluate statistics such as the Sharpe ratio, information ratio, Sterling ratio, Calmar ratio, and Omega ratio to assess the balance between returns and risk.

viii. **Relationship with Other Strategies:** Consider the incremental value provided by a new strategy by comparing its performance with existing strategies, both independently and in combination.

ix. **Time Decay:** Examine how the returns of the strategy are affected if trades are initiated on a lagged basis after receiving a trading signal. This helps to determine the sensitivity of the strategy to the timeliness of information and the degree of market saturation.

x. **Sensitivity to Specific Parameters:** A high-quality strategy should show only minor variations in outcomes with small changes in its parameters; large fluctuations may indicate overfitting.

xi. **Overfitting:** By plotting parameter values against the corresponding outcomes, one should observe a relatively flat curve with no abrupt jumps. Models that are parsimonious—that is, those that rely on fewer parameters—tend to be less prone to overfitting.

Remark 1.3.21. *Other Considerations in Model Testing*
It is crucial to note that overestimating trading costs can lead to holding positions longer than optimal, whereas underestimating these costs might result in excessive portfolio turnover and a detrimental bleed from trading expenses. Moreover, assumptions regarding the availability of short positions must be carefully considered, especially with respect to hard-to-borrow lists.

1.3.7 Risk Assessment

Definition 1.3.22. *Model Risks*
Quantitative models inherently carry model risk—the risk that a model fails to accurately describe, match, or predict real-world phenomena. Each element of a quant model is subject to its own potential for error.

i. **Inapplicability of Modelling:** This risk arises when a quant model is applied to a problem for which it was not designed, or when a particular technique is misapplied to a given scenario.

ii. **Model Misspecification:** This occurs when the model does not properly reflect the real world. Although a model may perform adequately under normal conditions, it can fail under extreme circumstances.

iii. **Implementation Errors:** These include mistakes in coding or system design. Additionally, errors can occur if the model components are executed in an incorrect sequence.

Definition 1.3.23. *Regime Change Risk*
Quant models are built on historical data, relying on relationships that have prevailed over time. When a regime change occurs, these historical relationships and behaviors can shift, leading the model to lose its effectiveness.

Definition 1.3.24. *Exogenous Shock Risk*
This type of risk is driven by external events that are not inherent to the market itself, such as terrorist attacks, the outbreak of wars, bank bailouts, or regulatory changes (e.g., modifications to shorting rules). In such cases, discretionary overrides may be necessary.

Definition 1.3.25. *Contagion Risk*
Contagion risk occurs when multiple investors follow similar strategies. There are two components to this risk: first, the extent to which a quant strategy is crowded; and second, the additional positions held by other investors that might force them to exit the strategy in a panic (often referred to as the ATM effect).
Quant liquidation crisis may be triggered by factors such as the sheer size and popularity of quantitative strategies, suboptimal returns by operators leading up to a crisis, the cross-collateralisation of many strategies within funds, and risk targeting—where risk managers aim to maintain a specific level of volatility across their funds or strategies.

Method 1.3.26. *Risk Monitoring Methods*

i. **Exposure Monitoring Tools:** These tools aggregate current positions by various exposures (e.g., valuation, momentum, volatility) to monitor both gross and net exposures across sectors, industries, market capitalisation buckets, and style factors.

ii. **Profit and Loss Monitors:** By comparing the current portfolio against the previous day's closing prices, these monitors utilize intraday performance charts and also examine the source of profits and the hit rate (i.e., the percentage of positions where the strategy is profitable).

iii. **Execution Monitors:** These displays track the status of orders—indicating which are being processed and which have been completed—along with details such as transaction sizes and prices. They also measure fill rates for limit orders in passive strategies, and monitor slippage and market impact.

iv. **System Performance Monitors:** These monitors are responsible for detecting software or infrastructure errors, assessing CPU performance, measuring the speed of various stages of automated processes, and tracking communication latency.

1.4 Exploratory Data Analysis

1.4.1 Data Taxonomy

A brief overview of the types of data used in systematic trading.

Four essential types of financial data

Fundamental Data	Market Data	Analytics	Alternative Data
Assets	Price/Yield/IV	Analyst Recommendation	Satellite/CCTV
Liabilities	Volume	Credit Ratings	Google Searches
Sales	Dividend/Coupons	Earnings Expectations	Twitter/Chats
Costs/Earnings	Open Interest	News Sentiment	Metadata
Macro Variables	Quotes/Cancellations	...	...
...	Aggressor Side		
	...		

Remark 1.4.1. *Fundamental Data Characteristics*

i. Data is published with an index corresponding to last date in the report, which precedes the release date.
ii. Data is frequently backfilled or corrected, with the data vendor overwriting initial values as needed.
iii. The data is highly regularized and available at low frequency.

Remark 1.4.2. *Market Data Characteristics*

i. The raw feed consists of unstructured information, such as FIX messages (which allow full reconstruction of the trading book) or complete collections of BWIC (bids wanted in competition) responses.
ii. Processing FIX data is non-trivial, with approximately 10TB generated daily.

Remark 1.4.3. *Analytics Data Characteristics*

i. This is derivative data processed from the original source, with the relevant signal already extracted.
ii. It is costly to produce, and the methodology used in production may be biased or opaque.

Remark 1.4.4. *Alternative Data Characteristics*

i. This data is generated by individuals, business processes, and sensors.
ii. It provides primary information that is not available from traditional sources.
iii. Cost and privacy concerns; it may be particularly valuable if it challenges existing data infrastructure.

Definition 1.4.5. *Reference Data*

i. **Trading Universe:** Evolves daily to incorporate new listings and de-listings. Knowing when a stock ceases trading is crucial to avoid survivor bias.
ii. **Symbology Mapping:** Involves identifiers such as ISIN, SEDOL, RIC, and Bloomberg Tickers. Since symbols may change over time, mapping must persist as point-in-time data to support historical 'as-of-date' analyses, often requiring a bi-temporal data structure.
iii. **Ticker Changes:** Maintain a historical table of ticker changes (as described in symbology mapping) to ensure seamless continuity in time series data.
iv. **Corporate Actions Calendars:** Include events such as stock and cash dividends (with announcement and execution dates), stock splits, reverse splits, rights offers, mergers and acquisitions, spin-offs, adjustments in free float or shares outstanding, and quotation suspensions.
For dividends, announcements may coincide with increased volatility and price jumps, enabling strategies to capitalize on the added volatility.For splits and rights offers, historical data must be adjusted backward (both volume and price) to reflect these actions. For M&A and spin-offs, adjustments are needed to account for valuation changes, which are important in merger arbitrage strategies. Suspensions can create data gaps that impact backtesting.
v. **Static Data:** Comprises attributes like country, sector, primary exchange, currency, and quote factor. This data is used to group instruments based on fundamental similarities (e.g., for pairs trading), and maintaining a table of quotation currencies per instrument is essential for portfolio aggregation.
vi. **Exchange Specific Data:** Each exchange has unique features that must be considered.
First Group: Hours and Dates of Operations

1. **Holiday Calendar:** Different markets have their own holiday schedules. For strategies that trade across multiple markets, discrepancies in holiday closures can affect correlation.
2. **Exchange Session Hours:** Different sessions (Pre-Market, Continuous Core, After-Hours, etc.), auction times, cutoff times for order submission, lunch breaks, and pre-/post-lunch auctions. This also includes DST adjustments and variations in trading hours by venue.
3. **Disrupted Days:** Records of exchange outages or trading disruptions, which are important to filter out when building or testing strategies.

Second Group: Trading Mechanics

1. **Tick Size:** The minimum eligible price increment, which may vary by instrument and price level.
2. **Trade and Quote Lots:** The minimum size increments for trades or quotes.
3. **Limit-Up and Limit-Down Constraints:** Maximum daily price fluctuations and the rules for trading pauses or restrictions at threshold levels.
4. **Short Sell Restrictions:** Rules that may prevent short sales from trading at prices worse than the last trade, or from generating new quotes below the lowest prevailing price. These restrictions impact liquidity sourcing.

vii. **Market Data Condition Codes:** Vary by exchange and asset class. Each market event may have multiple codes (e.g., auction trade, lit/dark trade, cancelled/corrected trade, regular trade, off-exchange reporting, block trade, or multi-leg order such as an option spread). It is essential to build a mapping table for these codes so that trades published solely for reporting purposes can be excluded from liquidity updates in aggregated daily data.

viii. **Special Day Calendars:** Identify days with distinct liquidity characteristics that impact both execution strategies and alpha generation. Examples (non-exhaustive) include:

1. Half trading days before Christmas and after Thanksgiving (US).
2. Ramadan effects in Turkey.
3. Taiwan market opening on a weekend to compensate for holiday closures.
4. Adjusted trading hours in Korea on nationwide university entrance exam days.
5. Late openings in the Brazilian market following Carnival.
6. Last trading days of months and quarters, when portfolio rebalancing occurs.
7. Index rebalancing dates, where intraday volume skews toward the end of day.
8. Options and futures expiry dates (e.g., quarterly/monthly expiry, Triple Witching in the US, Special Quotations in Japan) that cause excess trading volume and altered intraday patterns due to hedging and portfolio adjustments.

Model normal days first; special days either modelled independently or by adjusting normal day baseline.

ix. **Futures-Specific Reference Data:** Essential for determining which contract was live at any point via an expiry calendar and identifying the most liquid contract. For example, equity index futures are generally most liquid for the front month, while energy futures (such as oil) might be more liquid in the second contract. Note that there is no standardised expiry frequency across markets. When computing rolling-window metrics, potential roll dates must be accounted for; volume data may be blended before and after a roll. Additionally, futures markets exhibit different intra-day phases with distinct liquidity characteristics, so market data metrics (volume profiles, average spreads, bid-ask sizes) should be computed for each session based on a schedule.

x. **Options-Specific Reference Data (Options Chain):** Consists of expiry date and strike price combinations. Mapping equity tickers to their corresponding option tickers, with strike and expiry information, facilitates more complex investment and hedging strategies (e.g., assessing distance to strike or changes in open interest between puts and calls).

xi. **Market-Moving News Releases:** Includes macroeconomic announcements (central bank statements such as FED/FOMC, ECB, BOE, BOJ, SNB; Non-Farm Payrolls; PMIs; Manufacturing Indices; Crude Oil Inventories) and stock-specific events like earnings releases. Maintaining a calendar of these events helps assess their impact on strategies.

xii. **Related Tickers:** Tickers that represent the same underlying asset, allowing for efficient opportunity identification. This includes mappings for primary tickers to composite tickers (in fragmented liquidity markets), dual-listed or fungible securities (e.g., in the US and Canada), ADRs/GDRs, or differences between local and foreign boards.

xiii. **Composite Assets:** Such as ETFs, indexes, and mutual funds, which are used to achieve desired exposures or as hedging instruments. They can also present arbitrage opportunities when deviating from their NAV. It is important to maintain data on their constituent time series, any cash component, the divisor for converting NAV to quoted price, and the constituent weights.

xiv. **Latency Tables:** For high-frequency trading, these tables record the distribution of latencies between different data centers for efficient order routing, as well as reordering data collected from different locations.

Definition 1.4.6. *Market Data Feed*

i. **Level I Data (Trade and BBO Quotes):** Contains trade executions and top-of-book quotes, sufficient to reconstruct the Best Bid and Offer (BBO). This data also includes trade status (e.g., cancelled, reported late) and qualifiers (e.g., odd lot, normal trade, auction trade, intermarket sweep, average price reporting, exchange details), which are useful for analyzing event sequences and deciding whether a print should update the last price and total traded volume.

ii. **Level II Data (Market Depth):** Provides full depth of the limit order book, including all updates (price changes, additions, or removals of shares) across all venues in fragmented markets.

iii. **Level III Data (Full Order View):** Provides detailed message data in which each incoming order is assigned a unique ID for tracking. Records precise details when an order is executed, cancelled, or amended, making it possible to reconstruct complete order book (with national depth) at any moment.

   1. **Timestamp:** Milliseconds elapsed since midnight.
   2. **Ticker:** Equity symbol (up to 8 characters).
   3. **Order:** Unique order identifier.
   4. **T (Message Type):** 'B' indicates an add buy order; 'S' an add sell order; 'E' a partial execution; 'C' a partial cancellation; 'F' a full execution; 'D' a full deletion; 'X' a bulk volume for a cross event; and 'T' an execution of a non-displayed order.
   5. **Shares:** Order quantity for messages 'B', 'S', 'E', 'X', 'C', and 'T'. Zero for 'F' and 'D'.
   6. **Price:** Order price for 'B', 'S', 'X', and 'T' messages; zero for cancellations and executions. The last 4 digits denote the decimal part (padded with zeroes), and the value should be divided by 1000 to convert to the currency unit.
   7. **MPID:** A 4-character Market Participant ID associated with the transaction.
   8. **MCID:** A 1-character Market Centre Code for the originating exchange.

iv. Special order types of note:

   1. **Order Subject to Price Sliding:** Execution price may be one cent less favourable than display price (e.g., at NASDAQ). Such orders are ranked at locking price as hidden orders but displayed at one minimum price variation inferior; a new order ID is issued if order is replaced as a display order.
   2. **Pegged Order:** Based on the NBBO, these orders are non-routable and receive a new timestamp upon repricing; display rules vary by exchange.
   3. **Mid-point Peg Order:** A non-displayed order that may result in half-penny executions.
   4. **Reserve Order:** The displayed size is treated as a displayed limit order, while the reserve size is subordinate to non-displayed and pegged orders. The minimum display quantity is 100 shares; when it falls below this threshold, the reserve is replenished, a new timestamp is generated, and the displayed size is re-ranked.
   5. **Discretionary Order:** Displayed at one price while passively trading at a more aggressive discretionary price. It only becomes active when shares are available within the discretionary price range and is ranked last in priority. The execution price may be less favourable than the display price.
   6. **Intermarket Sweep Order:** Can be executed without the need to verify the prevailing NBBO.

   Using this comprehensive Level III data, one can model the inter-arrival times of various events, as well as the arrival and cancellation rates as functions of distance from the best bid/offer and other variables (e.g., order book imbalance, queue length). Subsequently, analysis may include assessing the impact of market orders on the limit order book, estimating the likelihood of a limit order advancing in the queue, determining the probability of capturing the spread, and forecasting short-term price movements.

Definition 1.4.7. *Binned Data*

i. **Open, High, Low, Close (OHLC) and Previous Close Price:** These values indicate trading activity and intraday volatility. The range between the low and high prices reflects market sentiment, and the previous close must be adjusted for corporate actions and dividends.

ii. **Last Trade before Close (Price/Size/Time):** Captures any jump in the close price during the final trading moments, serving as an indicator of the stability of the close as a reference for the next day.

iii. **Volume:** Acts as an indicator of trading activity, particularly when it deviates sharply from long-term averages. It is useful to collect volume breakdowns between lit and dark venues for execution strategies.

iv. **Auctions Volume and Price:** Reflects price discovery events marked by significant volume prints.

v. **VWAP:** The Volume Weighted Average Price offers an indication of daily trading activity and is particularly useful for algorithmically executing large orders.

vi. **Short Interest/Days-to-Cover/Utilisation:** These metrics serve as proxies for investor positioning. High short interest suggests a bearish view from institutional investors, while the utilisation of borrowable securities indicates the potential for additional shorting. Days-to-Cover helps assess the potential severity of a short squeeze; higher values imply a greater chance of sudden price surges in heavily shorted securities.

vii. **Futures Data:** Provides insights into market activity or positions of large investors through open interest data. Arbitrage opportunities may arise if the basis is mispriced relative to dividend estimates.

viii. **Index-Level Data:** Supplies relative measures for instrument-specific features (such as index OHLC and volatility). Normalised features can help identify instruments that deviate from their benchmarks.

ix. **Options Data:** Offers information on trader positioning via open interest and the Greeks.

x. **Asset Class Specific Data:** Includes yield or benchmark rates (such as repo rates, 2-year, 10-year, and 30-year yields), CDS spreads, and the US Dollar Index.

Definition 1.4.8. *Granular Intraday Microstructure Activity*

i. **Number and Frequency of Trades:** Serves as a proxy for market activity and continuity; a low number may indicate challenging execution and higher volatility.

ii. **Number and Frequency of Quote Updates:** Provides a similar measure of market activity.

iii. **Top of Book Size:** Liquidity; larger sizes allow for larger orders to be executed almost immediately.

iv. **Depth of Book (Price and Size):** Also reflects the liquidity available in the market.

v. **Spread Size (Average, Median, Time-Weighted Average):** Proxy for trading costs. Parameterised distribution of spreads helps in identifying when trading opportunities are relatively inexpensive or costly.

vi. **Trade Size (Average, Median):** Useful for identifying intraday liquidity opportunities by examining the volume available in the order book.

vii. **Ticking Time (Average, Median):** Represents how frequently the top level of the order book is updated. This measure is critical for execution algorithms that must adapt their update frequency (e.g., for adding or cancelling child orders) to the characteristics of the traded instrument.

Daily distributions of these metrics can be used as starting estimates at the beginning of the day and updated intraday via online Bayesian methods.
Derived daily data include:

i. X-day Average Daily Volume (ADV) / Average Auction Volume

ii. X-day Volatility (e.g., close-to-close, open-to-close)

iii. Beta with respect to an index or sector (using standard beta or asymmetric up-/down-day beta)

iv. Correlation Matrix

When binning data, intervals may range from a few seconds to 30 minutes. Minute-bar data is typically used for volume and spread profiles to reduce noise from market friction.

Definition 1.4.9. *Fundamental Data and Other Data*

i. **Key Ratios:** Such as Earnings Per Share (EPS), Price-to-Earnings (P/E), Price-to-Book (P/B), etc.

ii. **Analyst Recommendations:** Aggregated consensus valuations from analysts.

iii. **Earnings Data:** Quarterly earnings estimates provided by research analysts serve as indicators of a stock's performance prior to the actual published figures.

iv. **Holders:** Significant changes in ownership can indicate shifts in sentiment among sophisticated investors.

v. **Insider Purchase/Sale:** Reflects potential future price movements, as insiders typically possess the best available information about the company.

vi. **Credit Ratings:** Downgrades, which lead to higher funding costs, can adversely impact equity prices.

1.4.2 Financial Data Structures

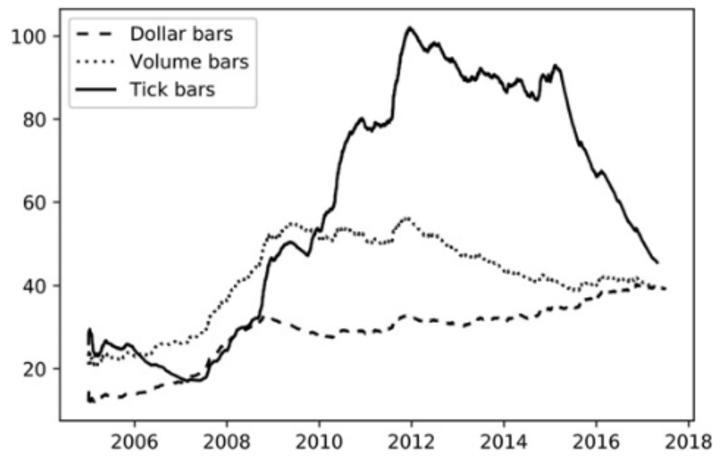


Figure 2: Average daily frequency of tick, volume, and dollar bars

Remark 1.4.10. *Standard BARS*
Method to transform a series of observations arriving at irregular frequency into a homogeneous series derived from regular sampling.

i. Time Bars: obtained by sampling information at fixed intervals. Information collected includes timestamp, volume-weighted average price (VWAP), open price, close price, high price, low price, volume etc.
To be avoided as markets do not process information at constant time interval. Time bars oversample information in low-activity periods and under-sample information in high-activity periods. Time bars exhibit poor statistical properties, i.e., serial correlation, heteroscedasticity, non-normality of returns.

ii. Tick Bars: sample variables extracted each time a pre-defined number of transactions take place. Allows synchronisation of sampling with a proxy of information arrival.
Sampling as a function of trading activity creates returns closer to IID Normal (Thierry and Helyette (2000)). When constructing tick bars, to be aware of outliers, as many exchanges carry out auction at open and at close; order book accumulates bids and offers without matching. Order fragmentation introduces some arbitrariness in number of ticks. Matching engine protocols may split one fill into multiple artificial partial fills as a matter of operational convenience.

iii. Volume Bars: samples every time a pre-defined amount of security's units that have been exchanged. Achieves better statistical properties than sampling tick bars.
Convenient artefact for studying market microstructure theories.

iv. Dollar Bars: samples an observation every time a pre-defined market value is exchanged. Used when the analysis involves significant price fluctuations. Robust against corporate actions such as splits, reverse splits, issuance of new shares, buying back existing shares.
Bar size could be dynamically adjusted as a function of free-floating market cap of a company or outstanding amount of issued debt.

Remark 1.4.11. *Information-Driven Bars*
Method to sample more frequently when new (micro-structural) information arrives to the market.

i. Tick Imbalance Bars: sample bars whenever tick imbalance exceeds expectations. To determine tick index T such that accumulation of signed ticks exceeds a given threshold.
Let $\{(p_t, v_t)\}_{t=1,\dots,T}$ be sequence of ticks where p_t and v_t is the price and volume associated with tick t. Let tick rule define a sequence $\{b_t\}_{t=1,\dots,T}$ where

$$b_t = \begin{cases} b_{t-1} & \text{if } \Delta p_t = 0 \\ \frac{|\Delta p_t|}{\Delta p_t} & \text{if } \Delta p_t \neq 0 \end{cases}$$

The tick imbalance at time T is defined as

$$\theta_T = \sum_{t=1}^{T} b_t$$

Compute expected value of θ_T at beginning of the bar,

$$E_0[\theta_T] = E_0[T](P[b_t = 1] - P[b_t = -1]) = E_0[T](2P[b_t = 1] - 1)$$

where $E_0[T]$ is expected size of tick bar, $P[b_t = 1]$ and $P[b_t = -1]$ is unconditional probability that a tick is classified as a buy and sell. In practice, $E_0[T]$ and $(2P[b_t = 1]-1)$ may be estimated as an exponentially weighted moving average of T and b_t values from prior bars.
Define the tick imbalance bar (TIB) as a T^* contiguous subset of ticks such that

$$T^* = \arg\min_T\{|\theta_T| \geq E_0[T] \mid 2P[b_t = 1] - 1\}$$

where the size of expected imbalance is implied by $|2P[b_t = 1] - 1|$.
When θ_T is more imbalanced than expected, a low T will satisfy the conditions.
TIBs are produced more frequently under presence of informed trading (asymmetric information that triggers one-side trading). TIBs are buckets of trades containing equal amounts of information.

ii. Volume/Dollar Imbalance Bars: sample bars when volume or dollar imbalances diverge from expectations. First, define imbalance at time T as

$$\theta_T = \sum_{t=1}^{T} b_t v_t$$

where v_t may represent ether number of securities traded (VIB) or dollar amount traded (DIB).
The expected value of θ_T at the beginning of the bar is then computed as

$$\begin{aligned}E_0[\theta_T] &= E_0\left[\sum_{t|b_t=1}^{T} v_t\right] - E_0\left[\sum_{t|b_t=-1}^{T} v_t\right]\\ &= E_0[T](P[b_t = 1]E_0[v_t|b_t = 1] - P[b_t = -1]E_0[v_t|b_t = -1])\\ &= E_0[T](v^+ - v^-)\end{aligned}$$

where the initial expectation of v_t is decomposed into component contributed by buys and sells. Then

$$E_0[\theta_T] = E_0[T](2v^+ - E_0[v_t])$$

In practice, $E_0[T]$ and $(2v^+ - E_0[v_t])$ may be estimated as exponentially weighted moving average of T and $b_t v_t$ values from prior bars. Next, define VIB or DIB as a T^*-contiguous subset of ticks such that

$$T^* = \arg\min_T\{|\theta_T| \geq E_0[T] \mid 2v^+ - E_0[v_t]\}$$

where the size of expected imbalance is implied by $|2v^+ - E_0[v_t]|$.
When θ_T is more imbalanced then expected, a low T will satisfy the conditions.
VIB and DIB addresses concerns on tick fragmentation and outliers, and also addresses the issues of corporate actions, as the bar size is adjusted dynamically.

iii. Tick Runs Bars: sample bars when the sequence of buys in overall volume diverges from expectations. For the case when large traders sweep order book, use iceberg orders, or slice parent orders into multiple children, all leaving a trace of runs in the $\{b_t\}_{t=1,\ldots,T}$ sequence. Define length of current run as

$$\theta_T = \max\left\{\sum_{t|b_t=1}^{T} b_t - \sum_{t|b_t=-1}^{T} b_t\right\}$$

The expected value of θ_T at beginning of bar is computed as

$$E_0[\theta_T] = E_0[T]\max\{P[b_t = 1], 1 - P[b_t = 1]\}$$

In practice, $E_0[T]$ and $P[b_t = 1]$ may be estimated as exponentially weighted moving average of T and proportion of buy ticks from prior bars. Next, define TRB as T^*-contiguous subset of ticks such that

$$T^* = \arg\min_T\{\theta_T \geq E_0[T]\max\{P[b_t = 1], 1 - P[b_t = 1]\}\}$$

where the expected count of ticks from runs is implied by $\max\{P[b_t = 1], 1 - P[b_t = 1]\}$.
When θ_T exhibits more runs than expected, a low T will satisfy these conditions.

Instead of measuring length of longest sequence, count number of ticks of each side without offsetting.

iv. Volume/Dollar Runs Bars: sample bars when volume or dollars traded by one side exceed expectation for a bar. First, define volume or dollars associated with a run as

$$\theta_T = \max\left\{ \sum_{t|b_t=1}^{T} b_t v_t - \sum_{t|b_t=-1}^{T} b_t v_t \right\}$$

where v_t may either represent volume (VRB) or dollar amount exchanged (DRB). The expected value of θ_T at beginning of the bar is then

$$E_0[\theta_T] = E_0[T] \max\{P[b_t = 1]E_0[v_t|b_t = 1], (1 - P[b_t = 1])E_0[v_t|b_t = -1]\}$$

In practice, $E_0[T], P[b_t = 1], E_0[v_t|b_t = 1], E_0[v_t|b_t = -1]$ may be estimated as exponentially weighted moving average of T, proportion of buy ticks, buy volumes, and sell volumes from prior bars. Next, define a volume runs bar (VR) as T^*-contiguous subset of ticks such that

$$T^* = \arg\min_T \{\theta_T \geq E_0[T] \max\{P[b_t = 1]E_0[v_t|b_t = 1], (1 - P[b_t = 1])E_0[v_t|b_t = -1]\}\}$$

expected volume from runs is implied by $\max\{P[b_t = 1]E_0[v_t|b_t = 1], (1 - P[b_t = 1])E_0[v_t|b_t = -1]\}\}$. When θ_T exhibits more runs than expected, volume from runs is greater than expected, a low T will satisfy these conditions.

Definition 1.4.12. *Multi-Product Series: ETF Trick*
To model a basket of securities as if it was a single cash product. To transform any complex multi-product dataset into a single dataset that resembles a total return ETF.

Method 1.4.13. *ETF Trick*
Produce a time series that reflects the value of \$1 invested. Changes in the series will reflect changes in PnL, series will be strictly positive, and implementation shortfall will be taken into account. The bars contain:

i. Raw open price of instrument $i = 1, \ldots, I$ at bar $t = 1, \ldots, T$: $o_{i,t}$
ii. Raw close price of instrument $i = 1, \ldots, I$ at bar $t = 1, \ldots, T$: $p_{i,t}$
iii. USD value of one point of instrument $i = 1, \ldots, I$ at bar $t = 1, \ldots, T$: $\varphi_{i,t}$. This includes forex rate.
iv. Volume of instrument $i = 1, \ldots, I$ at bar $t = 1, \ldots, T$: $v_{i,t}$
v. Carry, dividend, or coupon paid by instrument i at bar t: $d_{i,t}$. Variable can also be used to charge margin costs or costs of funding.

All instruments $i = 1, \ldots, I$ were tradable at bar $t = 1, \ldots, T$. Even if some instruments were not tradable over entirety of time interval $[t - 1, t]$, at least they were tradable at times $t - 1$ and t.
For basket of securities with allocation vector ω_t rebalanced (or rolled) on bars $B \subseteq \{1, \ldots, T\}$, the \$1 investment value $\{K_t\}$ is derived as

$$h_{i,t} = \begin{cases} \frac{\omega_{i,t} K_t}{o_{i,i+1}\varphi_{i,t} \sum_{i=1}^{I} |\omega_{i,t}|} & \text{if } t \in B \\ h_{i,t-1} & \text{otherwise} \end{cases}$$

$$\delta_{i,t} = \begin{cases} p_{i,t} - o_{i,t} & \text{if } (t-1) \in B \\ \Delta p_{i,t} & \text{otherwise} \end{cases}$$

$$K_t = K_{t-1} + \sum_{i=1}^{I} h_{i,t-1}\varphi_{i,t}(\delta_{i,t} + d_{i,t})$$

where $K_0 = 1$ is the initial AUM. Variable $h_{i,t}$ is the holdings of instrument i at time t, $\delta_{i,t}$ is change of market value between $t - 1$ and t for instrument i. Note profits or losses are being reinvested whenever $t \in B$, hence preventing negative prices. Dividends $d_{i,t}$ are already embedded in K_t.
The purpose of $\omega_{i,t} \left(\sum_{i=1}^{I} |\omega_{i,t}|\right)^{-1}$ is to de-lever the allocations.
Let τ_i be transaction cost associated with trading \$1 of the instrument. Three additional variables that the strategy needs to know for every observed bar t are:

i. Rebalance Costs: variable cost $\{c_t\}$ associated with allocation rebalance is

$$c_t = \sum_{i=1}^{I} (|h_{i,t-1}|\, p_{i,t} + |h_{i,t}|\, o_{i,t+1})\varphi_{i,t}\tau_i \quad \forall t \in B$$

Note c_t is not embedded in K_t, as shorting will generate fictitious proceeds when allocation is rebalanced. In code, $\{c_t\}$ is treated as a (negative) dividend.

ii. Bid-Ask Spread: the cost $\{\tilde{c}_t\}$ of buying or selling one unit of this ETF,

$$\tilde{c}_t = \sum_{i=1}^{I} |h_{i,t-1}|\, p_{i,t}\varphi_{i,t}\tau_i$$

When a unit is bought or sold, strategy must charge this cost $\tilde{c}_t$.

iii. Volume: volume traded $\{v_t\}$ is determined by least active member in the basket. Let $v_{i,t}$ be volume traded by instrument i over bar t. The number of tradable basket units is

$$v_t = \min_i \left\{ \frac{v_{i,t}}{|h_{i,t-1}|} \right\}$$

Transaction costs functions may not be linear, and can be simulated by the strategy.

Method 1.4.14. *ETF Trick: Computation of Allocation Vector with PCA*
Consider an IID multivariate Gaussian process with means vector μ of size $N \times 1$, and covariance matrix V of size $N \times N$. First, perform spectral decomposition $VW = W\Lambda$, where columns in W are reordered so that elements of Λ diagonal are sorted in descending order. Second, given allocations vector ω, portfolio risk is

$$\sigma^2 = \omega' V \omega = \omega' W \Lambda W' \omega = \beta' \Lambda \beta = (\Lambda^{1/2}\beta)'(\Lambda^{1/2}\beta)'$$

where β is projection of ω on orthogonal basis. Third, Λ is a diagonal matrix, thus

$$\sigma^2 = \sum_{n=1}^{N} \beta_n^2 \Lambda_{n,n}$$

The risk attributed to the nth component is

$$R_n = \beta_n^2 \Lambda_{n,n} \sigma^{-2} = [W'n]_n^2 \Lambda_{n,n} \sigma^{-2}$$

with $R' 1_N = 1$, and 1_N is a vector of N ones.
Note $\{R_n\}_{n=1,\dots,N}$ is distribution of risks across orthogonal components.
Next, compute vector ω which delivers user-defined risk distribution R. Note from earlier,

$$\beta = \left\{ \sigma \sqrt{\frac{R_n}{\Lambda_{n,n}}} \right\}_{n=1,\dots,N}$$

which represents allocation in new (orthogonal basis).
The allocation in old basis is $\omega = W\beta$. Rescaling ω re-scales σ, hence keeping risk distribution constant.

Method 1.4.15. *ETF Trick: Single Futures Roll*
To work with non-negative rolled series, derive price series of \$1 investment as follows:

i. Compute time series of rolled futures prices
ii. Compute return r as rolled price change divided by previous roll price
iii. Form a price series using these returns

These methods allow us to produce a continuous, homogeneous, and structured dataset from collection of unstructured financial data. Note however, that several ML algorithms do not scale well with sample size. ML algorithms achieve higher accuracy when they attempt to learn from relevant examples.

Method 1.4.16. *Sampling for Reduction*
To reduce the amount of data used to fit ML algorithm, downsampling could be used.

i. Sequential sampling at constant step size (linspace sampling)

ii. Sampling randomly using uniform distribution (uniform sampling)

Note both samples do not necessarily contain subset of most relevant observations.

Method 1.4.17. *Event-Based Sampling: CUMSUM Filter*
Bets are often placed after some event takes place, hence to let ML algorithm learn whether there is an accurate prediction function under these circumstances, CUSUM filter could be used.
This is a quality-control method, to detect shift in mean value of measured quantity away from a target value.
Let $\{y_t\}_{t=1,\dots,T}$ be IID observations arising from a locally stationary process. The cumulative sums are

$$S_t = \max\{0, S_{t-1} + y_t - E_{t-1}[y_t]\}, \quad S_0 = 0$$

An action will be recommended at the first t satisfying $S_t \geq h$ for some threshold h (filter size).
Note $S_t = 0$ whenever $y_t = E_{t-1}[y_t] - S_{t-1}$, The zero floor means some downward deviations will be skipped.
The filter is set up to identify a sequence of upside divergences from any reset level zero.
The threshold is activated when

$$S_t \geq h \Leftrightarrow \exists \tau \in [1,t] \mid \sum_{i=\tau}^{t}(y_i - E_{i-1}[y_t]) \geq h$$

This concept of run-ups can be extended to include run-downs, giving symmetric CUSUM filter.

$$\begin{aligned} S_t^+ &= \max\{0, S_{t-1}^+ + y_t - E_{t-1}[y_t]\}, \quad S_0^+ = 0 \\ S_t^- &= \min\{0, S_{t-1}^- + y_t - E_{t-1}[y_t]\}, \quad S_0^- = 0 \\ S_t &= \max\{S_t^+, -S_t^-\} \end{aligned}$$

1.4.3 Data Labelling Techniques

Method 1.4.18. *Labelling with Fixed-Time Horizon Method*
Given features matrix X with I rows, $\{X_i\}_{i=1,\dots,I}$ drawn from some bards with index $t = 1,\dots,T$, where $I \leq T$, let an observation X_i be assigned a label $y_i \in \{-1,0,1\}$,

$$\begin{aligned} y_i &= \begin{cases} -1 & \text{if } r_{t_{i,0},t_{i,0}+h} < -\tau \\ 0 & \text{if } \left|r_{t_{i,0},t_{i,0}+h}\right| \leq \tau \\ 1 & \text{if } r_{t_{i,0},t_{i,0}+h} > \tau \end{cases} \\ r_{t_{i,0},t_{i,0}+h} &= \frac{p_{t_{i,0}+h}}{p_{t_{i,0}}} - 1 \end{aligned}$$

where τ is a pre-defined constant threshold, $t_{i,0}$ is index of bar immediately after X_i takes place, $t_{i,0}+h$ is index of h-th bar after $t_{i,0}$, and $r_{t_{i,0},t_{i,0}+h}$ is price return over bar horizon h.

Remark 1.4.19. *Limitations of Fixed-Time Horizon Method*

i. Time bars do not exhibit good statistical properties (as seen earlier)
ii. The same threshold τ is applied regardless of observed volatility.
Compute daily volatility at intraday estimation points, applying span of n days t an exponentially weighted moving standard deviation.

Method 1.4.20. *Labelling with Triple-Barrier Method*
Labels an observation according to first barrier touched out of three barriers.

i. Set two horizontal barriers and one vertical barrier. Horizontal barriers are defined by profit-taking and stop-loss limits, which are a dynamic function of estimated volatility (realised or implied). Third barrier is the number of bars elapsed since the position was taken (expiration limit).
ii. If upper barrier is touched first, label observation as 1. If lower barrier is touched first, label observation as -1. If vertical barrier is touched first, either label by sign of the return or with 0.

Note that the method is path-dependent. To label an observation, need to account for entire path spanning $[t_{i,0}, t_{i,0}+h]$ where h defines the vertical barrier (expiration limit). Let $t_{i,1}$ be the time of first barrier touch with return as $r_{t_{i,0},t_{i,1}}$. The horizontal barriers may not be symmetric.

Remark 1.4.21. *Triple-Barrier Method Configurations*
Denote a barrier configuration by triplet $[pt, sl, t1]$ which are the upper barrier, lower barrier, physical barrier.
Set value as 0 if barrier is inactive, and 1 if barrier is active.
The three useful configurations are:

i. $[1, 1, 1]$: to realise profit, but have set a maximum tolerance for losses and a holding period.
ii. $[0, 1, 1]$: to exit after a number of bars, unless stopped-out.
iii. $[1, 1, 0]$: take profit as long as not stopped-out.

The three less realistic configurations are:

i. $[0, 0, 1]$: equivalent to fixed-time horizon method.
ii. $[1, 0, 1]$: position held until a profit is made or maximum holding period is exceeded, without regard for immediate unrealised losses
iii. $[1, 0, 0]$: position is held until a profit is made. Could lock in loose position for years.

The two illogical configurations are:

i. $[0, 1, 0]$: aimless. Hold position until stopped-out.
ii. $[0, 0, 0]$: no barriers. Position locked forever, no label generated.

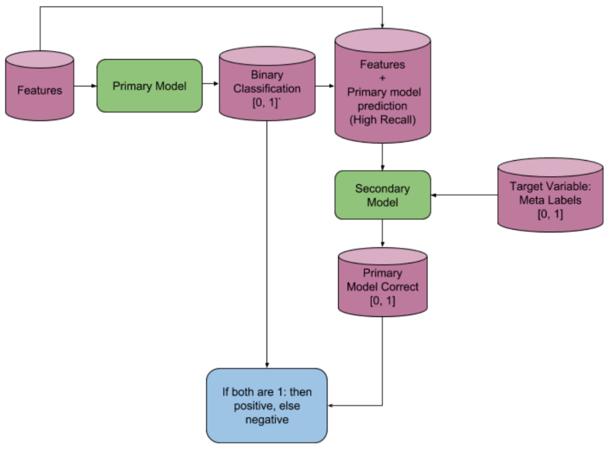

Features
Primary Model
Binary Classification [0, 1]`
Features + Primary model prediction (High Recall)
Secondary Model
Target Variable: Meta Labels [0, 1]
Primary Model Correct [0, 1]
If both are 1: then positive, else negative

Figure 3: Meta-Labelling Process

Method 1.4.22. *Meta-Labelling*
The technique is particularly helpful to achieve higher F1-scores.
First, build a model that achieves high recall, even if precision is not particularly high. Second, correct for low precision by applying meta-labelling to positives predicted by primary model.
Meta-labelling will filter out false positives, where majority of positives have been identified by primary model.
The second model's purpose is to determine if the positive from primary model is true or false.

i. Train a primary model (binary classification)
ii. A threshold level is determined at which the primary model has a high recall, ROC curves could be used to help determine a good level.
iii. Typical features of second model are as follows:
    i. Primary model features concatenated with predictions from first model.
    ii. Market state
    iii. Features indicative of false positives
    iv. Distribution related
    v. Recent model performance

    Meta Labels are used as target variable in second model. Fit the second model
iv. Prediction from the secondary model is combined with the prediction from the primary model and only where both are true, is your final prediction true.

Remark 1.4.23. *Limitations of Meta-Labelling*

i. If model has overfit the data, meta-labelling will not add much value
ii. If every trade is not treated as an independent observation, the meta-model is forced to determine day-to-day exposures, which is the wrong way to apply the technique
iii. Technique trades recall for precision. Require a large number of trades to train on, while being happy with reduction in trade frequency

1.4.4 Data Sample Weights

Note that most of ML literature is based on IID assumption, and ML applications usually fail in finance as these assumptions are unrealistic in the case of financial time series.

Remark 1.4.24. *Overlapping Outcomes*
Let label y_i be assigned to an observed feature X_i, where $y_i = f([t_{i,0}, t_{i,1}])$ is a function over the interval. When $t_{i,1} > t_{j,0}$ and $i < j$, then y_j will depend on common return $r_{t_{j,0},\min\{t_{i,1},t_{j,1}\}}$ (over interval $[t_{j,0}, \min\{t_{i,1}, t_{j,1}\}]$). The series of labels $\{y_i\}_{i-1,\ldots,J}$ are not IID whenever there is overlap between any two consecutive outcomes, i.e., $\exists i \mid t_{i,1} > t_{i+1,0}$. If this is resolved by restricting bet horizon to $t_{i,1} \leq t_{i+1,0}$, there is no overlap, but this will lead to coarse models where features sampling frequency is limited by horizon used to determine outcome. To investigate outcomes that lasted a different duration, samples have to be resampled with different frequency. In addition, if path-dependent labelling technique is to be applied, the sampling frequency will be subordinated to first barrier's touch. Hence, to use $t_{i,1} > t_{i+1,0}$, leading to overlapping outcomes.

Method 1.4.25. *Estimating Uniqueness of Label*
Let two labels y_i and y_j be concurrent at time t, both a function of at least one common return $r_{t-1,t} = \frac{p_t}{p_{t-1}} = 1$. To compute the number of labels that are a function of given return $r_{t-1,t}$:

i. For each $t = 1, \ldots, T$, form a binary array $\{1_{t,i}\}_{i=1,\ldots,I}$ where $1_{t,i} \in \{0, 1\}$.
   Variable $1_{t,i} = 1$ if and only if $[t_{i,0}, t_{i,1}]$ overlaps with $[t-1, t]$ and $1_{t,i} = 0$ otherwise.

ii. Compute the number of labels concurrent at t, $c_t = \sum\limits_{i=1}^{I} 1_{t,i}$

Method 1.4.26. *Average Uniqueness of Label*
To estimate label's uniqueness (non-overlap) across its lifespan.

i. Uniqueness of label i at time t is $u_{t,i} = 1_{t,i} c_t^{-1}$.

ii. Average uniqueness of label i is average $u_{t,i}$ over label's lifespan, $\overline{u}_i = (\sum_{t=1}^{T} u_{t,i})(\sum_{t=1}^{T} 1_{t,i})^{-1}$.

Note that $\{\overline{u}_i\}_{i=1,\ldots,I}$ are not used for forecasting the label, hence there is no information leakage.

Remark 1.4.27. *IID and Oversampling*
Probability of not selecting item i after I draws with replacement on set of I items is $(1 - I^{-1})^I$. As $I \to \infty$, note that $(1 - I^{-1})^I \to e^{-1}$. Number of unique observations drawn to be expected is $(1 - e^{-1}) \approx \frac{2}{3}$.
If maximum number of overlapping outcomes is $K \leq I$, probability of not selecting a particular item i after I draws with replacement on set of I items is $(1-K^{-1})^I$. As sample size increase, probability can be approximated as $(1 - I^{-1})^{I\frac{K}{I}} \approx e^{-\frac{K}{I}}$. Implication is that incorrectly assuming IID draws lead to oversampling.

Method 1.4.28. *Sampling with Bootstrap, Redundancy*
Sampling with bootstrapping on observations where $I^{-1} \sum_{i=1}^{I} \overline{u}_i \ll 1$, in-bag observations will increasingly be redundant to each other, and very similar to out-of-bag observations. Two solutions may be:

i. Drop overlapping outcomes before performing bootstrap.
   As overlaps are not perfect, dropping an observation due to overlap will lead to extreme loss in information.

ii. Utilise the average uniqueness $I^{-1} \sum_{i=1}^{I} \overline{u}_i$ to reduce undue influence of outcomes that contain redundant information. Ensure in-bag observations are not sampled at frequency much higher than uniqueness.

Method 1.4.29. *Sequential Bootstrap*
Draws made according to changing probability that controls for redundancy.

i. Observation X_i is drawn from uniform distribution, $i \sim U[1, I]$.
   Probability of drawing any value i is $\delta_i^{(1)} = I^{-1}$.

ii. Second draw, to reduce probability of drawing observation X_j with highly overlapping outcome.
   Let φ be sequence of draws (may include repetitions), where $\{\varphi^{(1)}\} = \{i\}$.
   Uniqueness of j at time t is $u_{t,j}^{(2)} = 1_{t,j}(1+\sum_{k\in\varphi^{(1)}} 1_{t,k})^{-1}$, which is the uniqueness from adding alternative j's to existing sequence of draws $\varphi^{(1)}$.
   Average uniqueness of j is average $u_{t,j}^{(2)}$ over j's lifespan, $\overline{u}_j^{(2)} = (\sum_{t=1}^{T} u_{t,j})(\sum_{t=1}^{T} 1_{t,j})^{-1}$.
   A second draw can be made based on updated probabilities $\{\delta_j^{(2)}\}_{j=1,\ldots,I}$:

$$\delta_j^{(2)} = \overline{u}_j^{(2)} \left(\sum_{k=1}^{I} \overline{u}_j^{(2)} \right)^{-1}$$

   where $\sum_{j=1}^{I} \delta_j^{(2)} = 1$. Do a second draw, update $\varphi^{(2)}$, and re-evaluate $\{\delta_j^{(3)}\}_{j=1,\ldots,I}$.

iii. Process is repeated until I draws have taken place.

Process draws samples much close to IID, verified by increase in $I^{-1}\sum_{i=1}^{I}\overline{u}_i$.

Method 1.4.30. *Weighting Observations by Uniqueness and Absolute Return*
Let labels be a function for return sign ($\{-1,1\}$ for standard label, $\{0,1\}$ for meta-label). The sample weights can be defined in terms of sum of attributed returns over event's life-span, $[t_{i,0}, t_{i,1}]$,

$$\tilde{w}_i = \left| \sum_{t=t_{i,0}}^{t_{i,1}} \frac{r_{t-1,t}}{c_t} \right|, \quad w_t = \tilde{w}_i \left(\sum_{j=1}^{I} \tilde{w}_j \right)^{-1}$$

where $\sum_{i=1}^{I} w_i = I$. The method weigh an observation as a function of absolute log returns that can be attributed uniquely to it. Lower returns should be assigned higher weights.

Method 1.4.31. *Time Decay Weighting*
To let sample weights decay as new observations arrive.
Let $d[x] \geq 0 \ \forall x \in [0, \sum_{i=0}^{I}\overline{u}_i]$ be time-decay factors multiplying sample weights from earlier.
The final weight has no decay, $d[\sum_{i=1}^{I}\overline{u}_i] = 1$, and all other weights will adjust relative to that.
Let $c \in (-1,1]$ be user-defined parameters that determines decay function as follows:

i. If $c \in [0,1]$, then $d[1] = c$ with linear decay
ii. If $c \in (-1,0)$, then $d[-c\sum_{i=1}^{I}\overline{u}_i] = 0$, with linear decay between $[-c\sum_{i=1}^{I}\overline{u}_i, \sum_{i=1}^{I}\overline{u}_i]$, and $d[x] \ \forall x \leq -c\sum_{i=1}^{I}\overline{u}_i$.

If given linear piecewise function $d = \max\{0, a+bx\}$, requirements are met by following boundary conditions:

i. $d = a + b\sum_{i=1}^{I}\overline{u}_i = 1 \Rightarrow a = 1 - b\sum_{i=1}^{I}\overline{u}_i$
ii. Contingent on c:
    1. $d = a + b0 = c \Rightarrow b = (1-c)(\sum_{i=1}^{I}\overline{u}_i)^{-1} \quad \forall c \in [0,1]$
    2. $d = a - bc\sum_{i=1}^{I}\overline{u}_i = 0 \Rightarrow b = [(c+1)\sum_{i=1}^{I}\overline{u}_i]^{-1} \ \forall c \in (-1,0)$

In the implementation, decay takes place according to cumulative uniqueness. Note that

i. $c = 1$ means there is no time decay
ii. $0 < c < 1$ means weights decay linearly over time, but every observation still receives strictly positive weight, regardless of age
iii. $c = 0$ means weights converge linearly to zero over time
iv. $c < 0$ means oldest portion cT of observations receive zero weight (erased from memory)

Method 1.4.32. *Class Weighting*
Weights for underrepresented labels. Critical in classification problems where the most important classes have rare occurrences. To assign higher weights to samples associated with those rare labels.

1.4.5 Fractionally Differentiated Features

Standard stationarity transformations (i.e. integer differentiation) reduce signal by removing memory. Although stationarity is necessary for inferential purposes, it is rarely the case that we want all memory to be erased. Fractionally differentiated processes exhibit long-term persistence and anti-persistence, hence enhancing the forecasting power compared to standard ARIMA approach.

Definition 1.4.33. *BackShift Operator*
Let B be the backshift operator applied to a matrix of real-valued features $\{X_t\}$, where $B^k X_t = X_{t-k}$ for any integer $k \geq 0$. By binomial expansion, we then have

$$\begin{aligned}(1-B)^d = \sum_{k=0}^{\infty}\binom{d}{k}(-B)^k &= \sum_{k=0}^{\infty}\prod_{i=0}^{k-1}(d-i)\frac{(-B)^k}{k!} \\ &= \sum_{k=0}^{\infty}(-B)^k\prod_{i=0}^{k-1}\frac{d-i}{k-i} \\ &= 1 - dB + \frac{d(d-1)}{2!}B^2 - \frac{d(d-1)(d-2)}{3!}B^3 + \cdots\end{aligned}$$

Remark 1.4.34. *Properties of Fractionally Differentiated Features*
Let d be a real (non-integer) positive number. The arithmetic series consists of dot product

$$\tilde{X}_t = \sum_{k=0}^{\infty} \omega_k X_{t-k}$$
$$\omega = \left\{1, -d, \frac{d(d-1)}{2!}, -\frac{d(d-1)(d-2)}{3!}, \dots, (-1)^k \prod_{i=0}^{k-1} \frac{d-1}{k!}, \dots\right\}$$
$$X = \{X_t, X_{t-1}, \dots, X_{t-k}, \dots\}$$

where ω are the weights, X are the values. Properties of these features are:

i. Long memory: if d is a positive integer number, then

$$\prod_{i=0}^{k-1} \frac{d-i}{k!} = 0 \quad \forall > d$$

and memory beyond that point is cancelled.

ii. Iterative weight generation: given sequence of weights ω, for $k = 0, \dots, \infty$, the weights are

$$\omega_k = -\omega_{k-1} \frac{d-k+1}{k}, \quad \omega_0 = 1$$

iii. Convergence: For $k > d$, if $\omega_{k-1} \neq 0$, then

$$\left| \frac{\omega_k}{\omega_{k-1}} \right| = \left| \frac{d-k+1}{k} \right| < 1$$

and $\omega_k = 0$ otherwise. Hence weights converge asymptotically to zero.
For positive d and $k < d+1$, then $\frac{d-k+1}{k} \geq 0$, which makes initial weights alternate in sign.
For non-integer d, once $k \geq d+1$, ω_k will be negative if int$[d]$ is even, and positive otherwise.
In summary, $\lim_{k\to\infty} = 0^-$ when int$[d]$ is even, and $\lim_{k\to\infty} = 0^+$ when int$[d]$ is odd.
In special case $d \in (0,1)$, that $-1 < \omega_k < 0 \; \forall k > 0$. Alternate weight signs makes $\{\tilde{X}_t\}_{t=1,\dots,T}$ stationary, as memory wanes or is offset over the long run.

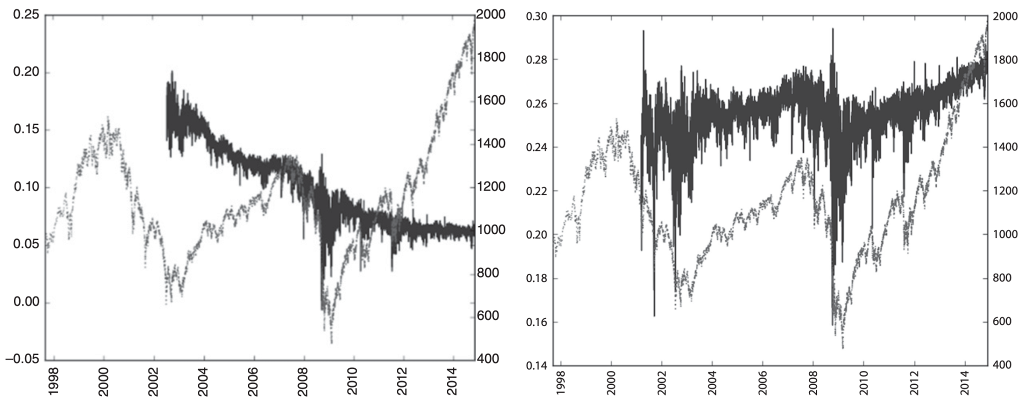


Figure 4: Fractional differentiation controlling for weight loss with expanding and fixed-width window

Method 1.4.35. *Expanding Window*
Given time series T with real observations $\{X_t\}_{t=1,\dots,T}$, for each l, the relative weight loss is defined as

$$\lambda_l = \sum_{j=T-l}^{T} |\omega_j| \Bigg/ \sum_{i=0}^{T-1} |\omega_i|$$

Given tolerance level $\tau \in [0,1]$, determine value l^* such that $\lambda_{l^*} \leq \tau$ and $\lambda_{l^*+1} > \tau$. This value l^* corresponds to the first results $\{\tilde{X}_t\}_{t=1,\dots,l^*}$, where weight-loss is beyond acceptable threshold $\lambda_t > \tau$.
From Remark 1.4.34, it is clear λ_{l^*} depends on convergence speed of $\{\omega_k\}$, which in turn depends on $d \in [0,1]$.
For $d = 1, \omega_k = 0 \; \forall k > 1$, and $\lambda_l = 0 \; \forall l > 1$, hence it suffices to drop $\tilde{X}_1$.

As $d \to 0^+$, l^* increases, and larger portion of initial $\{\tilde{X}_t\}_{t=1,\dots,l^*}$ needs to be dropped to keep the weight loss $\lambda_{l^*} < \tau$. Note that there will be negative drift caused by negative weights added to initial observations as window is expanded. By controlling for weight loss, negative drift is still substantial as $\{\tilde{X}_t\}_{t=l^*+1,\dots,T}$ are computed on an expanding window.

Method 1.4.36. *Fixed-Width Window*
Drop weights after their modulus $|\omega_k|$ decreases below a given threshold τ. This is equivalent to finding the first l^* such that $|\omega_{l^*}| \geq \tau$ and $|\omega_{l^*+1}| \leq \tau$, setting a new variable $\tilde{\omega}_k$:

$$\tilde{\omega_k} = \begin{cases} \omega_k & \text{if } k \leq l^* \\ 0 & \text{if } k > l^* \end{cases} \quad , \qquad \tilde{X}_t = \sum_{k=0}^{l^*} \tilde{\omega}_k X_{t-k} \;\; \text{for } t = T - l^* + 1, \dots, T$$

Note that the same vector of weights is used across all estimates of $\{\tilde{X}_t\}_{t=l^*,\dots,T}$, hence avoiding negative drift caused by expanding window's added weights.
Distribution has skewness and excess kurtosis from memory, but it is stationary.

2 Mathematical Primer

Quantitative trading requires mastery in the following fields of math:

i. Time Series Analysis
ii. Stochastic Processes
iii. Machine Learning

This section serves as a guide in providing the fundamental knowledge required.

2.1 Time Series Analysis

Based on the books by James Douglas Hamilton (1994), and ...

2.1.1 Stationary Time Series

Definition 2.1.1. A *linear first-order difference equation* is defined as

$$y_t = \phi y_{t-1} + w_t$$

where y_t is the target variable (with y_{t-1} the lag 1 variable), w_t is an input variable at time t

The difference equation may be solved by recursive substitution to arrive at

$$y_t = \phi^{t+1} y_{-1} + \phi^t w_0 + \phi^{t-1} w_1 + \phi^{t-2} w_2 + \cdots + \phi w_{t-1} + w_t$$

The *dynamic multiplier* calculates effect of w_t on y_{t+j}, and is given by

$$\frac{\partial y_{t+j}}{\partial w_t} = \phi^j$$

If $|\phi| < 1$, the system is stable. If $|\phi| > 1$, then the system is explosive.
We may generalise the process to *p-th order difference equation*, i.e.,

$$y_t = \phi_t y_{t-1} + \phi_2 t_{t-2} + \cdots + \phi_p y_{t-p} + w_t$$

We may rewrite this as a first-order difference equation in a $(p \times 1)$ vector $\boldsymbol{\xi}_t$:

$$\boldsymbol{\xi}_t = \begin{bmatrix} y_t \\ y_{t-1} \\ \vdots \\ y_{t-p+1} \end{bmatrix}$$

Define the $(p \times p)$ matrix $\boldsymbol{F}$ by

$$\boldsymbol{F} = \begin{bmatrix} \phi_1 & \phi_2 & \phi_3 & \cdots & \phi_{p-1} & \phi_p \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$$

Finally, define the $(p \times 1)$ vector $\boldsymbol{v}_t$ by

$$\boldsymbol{v}_t = \begin{bmatrix} w_t \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Then the system of p equations $\boldsymbol{\xi}_t = \boldsymbol{F}\boldsymbol{\xi}_{t-1} + \boldsymbol{v}_t$ is identical to the first order difference equation, and by recursive substitution, we have the following for the case of p-th order difference equation:

$$\boldsymbol{\xi}_{t+j} = \boldsymbol{F}^{j+1}\boldsymbol{\xi}_{t-1} + \boldsymbol{F}^j \boldsymbol{v}_t + \boldsymbol{F}^{j-1}\boldsymbol{v}_{t+1} + \cdots + \boldsymbol{F}\boldsymbol{v}_{t+j-1} + \boldsymbol{v}_{t+j}$$

Hence the dynamic multiplier is then $\frac{\partial y_{t+j}}{\partial w_t} f_{11}^{(j)}$, where $f_{11}^{(j)}$ is the $(1,1)$ element of $\boldsymbol{F}^j$.

2.1.2 Univariate Time Series Models

Trading activities through an exchange can be described by a sequence of time stamps ('ticks') $t_0 < t_1 < \cdots < t_n$, and 'marks' y_i at time t_i, where t_i denote market open, t_n denote market close. The marks y_i is a characteristic of the order book at time of ith activity. Events with marks associated with the ticks can be described mathematically as a marked point process.

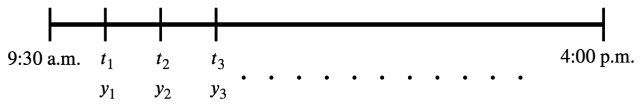


Figure 5: Ticks and Marks

A data aggregation method is where aggregation is conducted when there is a change in marker. An alternative aggregation method would be to divide the time span T for exchange hours into K intervals, so regularly spaced intervals are of size $\Delta t = T/K$. The time series method to be discussed will apply to all aggregated data.

Let $p_{it} = \ln P_{it}$ denote price of ith asset at time t; $p_t = (p_{1t}, p_{2t}, \ldots, p_{nt})$ denote price vector for n assets; y_{it} denote vector of characteristics of ith asset at time t. These quantities are aggregated from high frequency data. Consider r factors $f_t = (f_{1t}, f_{2t}, \ldots, f_{rt})$ that may include market and industry factors, and asset characteristics. Trading rules can be broadly grouped as follows:

i. *Statistical Arbitrage*: $E(p_{i,t+1} \mid p_{i,t}, p_{i,t-1}, \ldots, y_{i,t}, y_{i,t-1}, \ldots)$, that predicts the price of ith stock at $t+1$ based on past trading information (also known as time series momentum)

ii. *Momentum*: $E(p_{t+1} \mid p_t, p_{t-1}, \ldots, y_t, y_{t-1}, \ldots)$, that predicts the cross-sectional momentum of a subset of stocks based on past trading characteristics. For portfolio formation and rebalancing, pairs trading

iii. *Fair Value*: $E(p_{t+1} \mid p_t, p_{t-1}, \ldots, y_t, y_{t-1}, \ldots, f_t, f_{t-1}, \ldots)$, predicts price using all relevant quantities. Factors normally include market, Fama-French; at a more macro level than timescale considered for price prediction, but may still be useful.

Hence price and volatility prediction can be formulated as a time series prediction problem. Autocorrelations and partial autocorrelations can be used to build autoregressive and ARCH models with some predictive power.

Let $Y_1, Y_2, \ldots, Y_T$ be a sequence of random variables with a joint probability distribution. A sequence of observations of stochastic process $\{Y_t, t = 1, \ldots, T\}$ is a realisation of the process.
A time series $\{Y_t\}$ is *stationary* if for every integer m, the set of variables $Y_{t_1}, Y_{t_2}, \ldots, Y_{t_m}$ depends only on the distance between times $t_1, t_2, \ldots, t_m$. Thus $E(Y_t) = \mu$, $Var(Y_t) = \sigma^2$ are constant for all t.
The *auto-covariance* function is defined as

$$\gamma(s) = Cov(Y_t, Y_{t-s}) = E[(Y_t - \mu)(Y_{t-s} - \mu)] \;\; \forall s = 0, \pm 1, \ldots$$

The *auto-correlation* of process at lag s is defined as

$$\rho(s) = Corr(Y_t, Y_{t-s}) = \frac{Cov(Y_t, Y_{t-s})}{(Var(Y_t)\ Var(Y_{t-s}))^{1/2}} = \frac{\gamma(s)}{\gamma(0)}, \quad \forall s = 0, \pm 1, \ldots$$

Some examples of stationary stochastic processes are as follows:

Example 2.1.2. *(White Noise)* Sequence of discrete independent random variables (r.v.) $\{\epsilon_t\}$ with $E[\epsilon_t] = 0$ and $E[\epsilon_t^2] = \sigma^2$. Set $Y_t = \mu + \epsilon_t$, then $E[Y_t] = \mu$, $Cov(Y_t, Y_{t-s}) = Var(Y_t) = \sigma^2$ if $s = 0$, and $Cov(Y_t, Y_{t-s}) = 0$ if $s \neq 0$. Hence the process is stationary.

Example 2.1.3. *(Moving Average)* Let $\{\epsilon_t\}$ be independent r.v., with process $\{Y_t\}$ where $Y_t = \mu + \epsilon_t + \epsilon_{t_1}$ for $t = 0, 1, 2, \ldots$, and μ is constant. Then $E[Y_t] = \mu$ $\forall t$, and

$$Cov(Y_t, Y_{t-s}) = \gamma(s) = \begin{cases} 2\sigma^2 & \text{if } s = 0 \\ \sigma^2 & \text{if } s = 1 \\ 0 & \text{if } s > 1 \end{cases}$$

Hence $\{Y_t\}$ is stationary, with $\rho(s) = \gamma(s)/\gamma(0)$ such that $\rho(0) = 1$, $\rho(1) = 1/2$, $\rho(s) = 0$ for $|s| > 1$.

Some examples of non-stationary stochastic processes are as follows:

Example 2.1.4. *(Random Walk with Drift)* Let $\{\epsilon_t\}$ be sequence of independent r.v. with $E[\epsilon_t] = 0$, $E[\epsilon_t^2] = \sigma^2$, and define process $\{Y_t\}$ by $Y_t = Y_{t-1} + \delta + \epsilon_t$ with $Y_0 = 0$. Then the process can be summarised as

$$Y_t = \delta t + \sum_{j=1}^{t} \epsilon_j$$

Note that $E[Y_t] = \delta t$ and $Var[Y_t] = t\sigma^2$, hence process $\{Y_t\}$ is not stationary.

Given T observations from stationary process $\{Y_t\}$, the *sample mean* is $\overline{Y} = \frac{1}{T}\sum\limits_{t=1}^{T} Y_t$. The *sample auto-covariance* function is defined by $\hat{\gamma}(s) = \frac{1}{T}\sum\limits_{t=1}^{T-s}(Y_t - \overline{Y})(Y_{T-s} - \overline{Y})$ for $s = 0, 1, \ldots$. The *sample auto-correlation* function (ACF) is defined as $\hat{\rho}(s) = \frac{\hat{\gamma}(s)}{\hat{\gamma}(0)} = r(s)$. The *sample variance* is $Var(\overline{Y}) = \frac{\gamma(0)}{T}\left[1 + 2\sum\limits_{u=1}^{Y-1}\left(\frac{T-u}{T}\right)\rho(u)\right]$; note that it has to account for auto-correlations.

Definition 2.1.5. A stochastic process $\{Y_t\}$ is a *linear process* if it can be represented as

$$Y_t = \mu + \sum_{j=0}^{\infty} \Psi_j \epsilon_{t-j}$$

where ϵ_t are independent with mean 0, variance σ_ϵ^2, and $\sum\limits_{j=0}^{\infty} < \infty$.

2.2 Classical Machine Learning

2.2.1 Ensemble Methods

2.3 Deep Learning

2.3.1 Deep Feedforward Networks

Remark 2.3.1. *Deep Feedforward Networks (DFNs)*
Defines a mapping $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$ and learns value of parameters $\boldsymbol{\theta}$ that results in best function approximation.

Remark 2.3.2. *Linear Models*
Linear models may fit data efficiently and reliably, either in closed form or with convex optimisation.
Model capacity limited to linear functions, does not model interaction between any two input variables

Remark 2.3.3. *Nonlinear Models*
To represent nonlinear functions of $\boldsymbol{x}$, apply linear model to transformed input $\phi(\boldsymbol{x})$ where ϕ is a nonlinear transformation. Kernel trick may be applied to obtain nonlinear learning algorithm. To choose mapping ϕ:

i. Choose generic ϕ, such as that used by kernel machines based on RBF kernel.
   If $\phi(\boldsymbol{x})$ is of high enough dimension, can find enough capacity to fit training set, but generalisations to test set remains poor. Mappings are based on principle of local smoothness and do not encode enough prior information to solve advanced problems.
ii. Manually engineer ϕ. Requires decades of human effort for each task, and practitioners specialising in different domains (i.e, speech recognition, computer vision) with little transfer between domains.
iii. Learn ϕ through deep learning. Model is $y = f(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{w}) = \phi(\boldsymbol{x}; \boldsymbol{\theta})^T \boldsymbol{w}$, where parameters $\boldsymbol{\theta}$ can be used to learn ϕ from broad class of functions, and parameters $\boldsymbol{w}$ that map from $\phi(\boldsymbol{x})$ to desired output.
   Do not require training problem to be convex, and only require finding the right general function.

Definition 2.3.4. *Cost Functions*

i. Learning Conditional Distributions with Maximum Likelihood: cost function is negative log-likelihood, which is the cross-entropy between training data and model distribution:

$$J(\boldsymbol{\theta}) = -\mathbb{E}_{\boldsymbol{x},\boldsymbol{y} \sim \hat{p}_{\text{data}}} \log p_{\text{model}}(\boldsymbol{y} \mid \boldsymbol{x})$$

   Specific form of cost function changes from model to model, depending on form of $\log p_{\text{model}}$.
   Method removes the burden of designing cost functions for each model, as specifying a model $p(\boldsymbol{y} \mid \boldsymbol{x})$ automatically determines a cost function $\log p(\boldsymbol{y} \mid \boldsymbol{x})$.
ii. Learning Conditional Statistics: to learn just one conditional statistic of $\boldsymbol{y}$ given $\boldsymbol{x}$.
   With sufficiently powerful neural network, this can represent any function f from a wide class of function, limited by features of continuity and boundedness. Hence the cost function is a functional. Learning is choosing a functional rather than a set of parameters.
   By calculus of variable, solving the optimisation problem yields the below function,

$$f^* = \arg\min_f \mathbb{E}_{\boldsymbol{x},\boldsymbol{y} \sim p_{\text{data}}} \|\boldsymbol{y} - f(\boldsymbol{x})\|^2$$
$$f^* = \mathbb{E}_{\boldsymbol{y} \sim p_{\text{data}}(\boldsymbol{y} \mid \boldsymbol{x})}[\boldsymbol{y}]$$

   If infinitely many samples from the true data-generating distribution could be trained, then minimising the mean squared error cost function gives a function that predicts mean of $\boldsymbol{y}$ for each value of $\boldsymbol{x}$.
   A second result from calculus of variations is:

$$f^* = \arg\min_f \mathbb{E}_{\boldsymbol{x},\boldsymbol{y} \sim p_{\text{data}}} \|\boldsymbol{y} - f(\boldsymbol{x})\|_1$$

   which is a function that predicts the median value of $\boldsymbol{y}$ for each $\boldsymbol{x}$. This is the mean absolute error.
   Note that mean squared error and mean absolute error often lead to poor results when used with gradient-based optimisation. Output units that saturate may produce very small gradients when combined with these cost functions. Hence the reason that cross-entropy cost function is more popular.

Definition 2.3.5. *Output Units*

i. Linear Units: base on affine transformation with no nonlinearity.
   Given features $\boldsymbol{h}$, a layer of output units produces vector $\hat{\boldsymbol{y}} = \boldsymbol{W}^T \boldsymbol{h} + \boldsymbol{b}$. Linear output layers are used to produce mean of conditional Gaussian distribution:

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{y}; \hat{\boldsymbol{y}}, \boldsymbol{I})$$

   Maximising log-likelihood is equivalent to minimising mean squared error.
   Linear units do not saturate, hence may be used for wide variety of optimisation algorithms.

ii. Sigmoid Units: define Bernoulli distribution y conditioned on $\boldsymbol{x}$. Neural net to predict $P(y=1 \mid \boldsymbol{x})$, which lies in interval $[0,1]$. The sigmoid unit is defined by

$$\hat{y} = \sigma(\boldsymbol{w}^T\boldsymbol{h} + b)$$

where σ is the logistic sigmoid function.
Note that the cost function used with maximum likelihood is $-\log P(y \mid \boldsymbol{x})$, preventing saturation.
The loss function for MLE of Bernoulli parametrised by sigmoid is

$$J(\boldsymbol{\theta}) = -\log P(y \mid \boldsymbol{x}) = -\log \sigma((2y-1)z) = \zeta((1-2y)z)$$

Function saturates only when $(1-2y)z$ is very negative, i.e., when model has the right answer.

iii. Softmax Units: used to represent probability distribution over n different classes.
A linear layer predicts unnormalised log probabilities:

$$\boldsymbol{z} = \boldsymbol{W}^T\boldsymbol{h} + \boldsymbol{b}$$

where $z_i = \log \tilde{P}(y = i \mid \boldsymbol{x})$. Softmax function is then

$$\text{softmax}(\boldsymbol{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

The function is to maximise $\log P(y=i;\boldsymbol{z}) = \log \text{softmax}(\boldsymbol{z})_i = z_i - \log \sum_j \exp(z_j)$.
Note that the input z_i always has direct contribution to cost function, and the term cannot saturate.
Un-regularised maximum likelihood will drive the model to learn parameters that drive the softmax to predict fraction of counts for each outcome observed in the training set:

$$\text{softmax}(z(\boldsymbol{x};\boldsymbol{\theta}))_i \approx \frac{\sum\limits_{j=1}^{m} \mathbf{1}_{y^{(j)}=i, \boldsymbol{x}^{(j)}=x}}{\sum\limits_{j=1}^{m} \mathbf{1}_{\boldsymbol{x}^{(j)}=x}}$$

Note that objective functions other than log-likelihood does not work well with softmax function. Squared error is poor loss function for softmax units, and can fail to train the model to change its output.
Softmax function can saturate, and many functions based on softmax also saturate, unless they are able to invert the saturating activating function.

iv. Other Output Types: generally, given a conditional distribution $p(\boldsymbol{y} \mid \boldsymbol{x};\ \boldsymbol{\theta})$, principle of maximum likelihood suggests using $-\log p(\boldsymbol{y} \mid \boldsymbol{x};\ \boldsymbol{\theta})$ as the cost function.
Neural networks represent a function $f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{\omega}$. The outputs of the function are not direct predictions of value $\boldsymbol{y}$, but the parameters for distribution over $\boldsymbol{y}$. The loss function is then $-\log p(\boldsymbol{y};\boldsymbol{\omega}(\boldsymbol{x}))$.

Remark 2.3.6. *Learning Distribution Parameters*

i. Heteroscedastic Model: to predict different variance in $\boldsymbol{y}$ for different values of $\boldsymbol{x}$, formulate the Gaussian distribution using precision rather than variance. In multi-variate case, the diagonal precision matrix is used, diag$(\boldsymbol{\beta})$. The log-likelihood of Gaussian distribution parametrised by $\boldsymbol{\beta}$ involves only multiplication by β_i and addition of $\log \boldsymbol{\beta}_i$. The gradient of these operations are well-behaved.
Let $\boldsymbol{\alpha}$ be raw activation of the model used to determine diagonal precision. The softplus function may be used to obtain a positive precision vector $\boldsymbol{\beta} = \zeta(\boldsymbol{\alpha})$. Same strategy applies equally if using variance variance or standard deviation rather than precision.
If covariance is full and conditional, then parametrisation must be chosen that guarantees positive-definiteness of predicted covariance matrix.

$$\boldsymbol{\Sigma}(\boldsymbol{x}) = \boldsymbol{B}(\boldsymbol{x})\boldsymbol{B}^T(\boldsymbol{x})$$

where $\boldsymbol{B}$ is unconstrained square matrix. Note that if matrix is full rank, then computing likelihood requires $O(d^3)$ a $d \times d$ matrix for the determinant and inverse of $\boldsymbol{\Sigma}(\boldsymbol{x})$.

ii. Mixture Density Networks: to perform multimodal regression (predict real values that come from conditional distribution $p(\boldsymbol{y} \mid \boldsymbol{x})$ that can have several different peaks in $\boldsymbol{y}$ for the same $\boldsymbol{x}$.

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \sum_{i=1}^{n} p(c = i \mid \boldsymbol{x}) \mathcal{N}(\boldsymbol{y}; \boldsymbol{\mu}^{(i)}(\boldsymbol{x}), \boldsymbol{\Sigma}^{(i)}(\boldsymbol{x}))$$

The neural network will have three outputs:

1. Mixture components $p(c = i \mid \boldsymbol{x})$, forming a multinoulli distribution over n different components with latent variable c, obtained by softmax over n-dimensional vector.
2. Means $\boldsymbol{\mu}^{(i)}(\boldsymbol{x})$
3. Covariances $\boldsymbol{\Sigma}^{(i)}(\boldsymbol{x})$

Gradient-based optimisation of conditional Gaussian mixtures can be unreliable as the divisions can be numerically unstable. May be solved by clipping gradients, or scaling gradients heuristically.

Definition 2.3.7. *Hidden Units*
Even if hidden units are not differentiable at all input points, gradient descent still performs well enough as the training algorithms do not usually arrive at local minimum of cost function, but reduce its value significantly. Most hidden units has input vector $\boldsymbol{x}$, computes an affine transformation $\boldsymbol{z} = \boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b}$, then applying element-wise non-linear function $g(\boldsymbol{z})$.

i. Rectified Linear Units (ReLUs): uses activation function $g(z) = \max\{0, z\}$.
Typically used on top of an affine transformation $\boldsymbol{h} = g(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b})$. In initialisation, set all elements of $\boldsymbol{b}$ to small positive values, so that ReLUs will be initially active for most inputs in training set.
Generalisations of ReLUs have non-zero slope α_i for $z_i < 0$: $h_i = \max(0, z_i) + \alpha_i \min(0, z_i)$.

   1. Absolute Value Rectification: sets $\alpha_i = -1$ to obtain $g(z) = |z|$. Used for object recognition from images, to seek features that are invariant under polarity reversal of input illumination.
   2. Leaky ReLU: fixes α_i to small positive value
   3. Parametric ReU: treats α_i as a learnable parameter

ii. Maxout Units: divide $\boldsymbol{z}$ into groups of k values. Each maxout unit then outputs maximum element of one of these groups: $g(\boldsymbol{z})_i = \max_{j \in \mathbb{G}^{(i)}} z_j$, where $\mathbb{G}^{(i)}$ is indices of inputs for group i, which is $\{(i-1)k+1, \dots, ik\}$.
This allows learning of piecewise linear function that responds to multiple directions in input $\boldsymbol{x}$ space .
Maxout units learn the activation function itself. With large k, maxout unit can learn to approximate any convex function. Each maxout unit is parametrised by k weight vectors, hence need more regularisation than ReLUs. Benefits include:

   1. Can work well without regularisation if training set is large and number of pieces per unit is low.
   2. Can gain statistical and computation advantages by requiring fewer parameters.
   3. Have redundancy that resists catastrophic forgetting, where neural networks forgot how to perform tasks that were trained on in the past.

iii. Logistic Sigmoid and Hyperbolic Tangent: the logistic sigmoid activation function is $g(z) = \sigma(z)$, and the hyperbolic tangent activation function is $g(z) = \tanh(z)$. Note that $\tanh(z) = 2\sigma(2z) - 1$.
Sigmoidal units saturate across most of the domain, which makes gradient-based learning very difficult. Hence the use in hidden units in feedforward networks is now discouraged.
If sigmoidal activation function must be used, hyperbolic tangent activation function performs better, as it resembles identity function more closely. Training $\hat{y} = \boldsymbol{w}^T \tanh(\boldsymbol{U}^T \tanh(\boldsymbol{V}^T\boldsymbol{x}))$ resembles training a linear model $\hat{y} = \boldsymbol{w}^T\boldsymbol{U}^T\boldsymbol{V}^T\boldsymbol{x}$ as long as the activations of the network can be kept small.
Sigmoidal functions are more common in recurrent networks, probabilities models, auto-encoders

iv. Linear Unit: consider neural network layer with n inputs, p outputs, $\boldsymbol{h} = g(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b})$. Replace with two layers, one using weight matrix $\boldsymbol{U}$ and the other using weight matrix $\boldsymbol{V}$. If first layer has no activation function, then the factored approach is to compute $\boldsymbol{h} = g(\boldsymbol{V}^T\boldsymbol{U}^T\boldsymbol{x} + \boldsymbol{b})$. If $\boldsymbol{U}$ produces q outputs, then $\boldsymbol{U}$ and $\boldsymbol{V}$ together only contains $(n + p)q$ parameters, while $\boldsymbol{W}$ contains np parameters. For small q, this is considerable saving in parameters, while the cost is constraining linear transformation to be low rank. This is an efficient way of reducing number of parameters in the model.

v. Softmax: naturally represent probability distribution over discrete variable with k possible values. Used only in more advanced architectures that explicitly learn to manipulate memory.

vi. Radial Basis Function (RBF): function becomes more active as $\boldsymbol{x}$ approaches a template $\boldsymbol{W}_{:,i}$. As it saturates to 0 for most $\boldsymbol{x}$, it can be difficult to optimise.

$$h_t = \exp\left(-\frac{1}{\sigma_i^2}\|\boldsymbol{W}_{:,i} - \boldsymbol{x}\|^2\right)$$

vii. Softplus: smooth version of rectifier for functional approximation and for conditional distributions of undirected probabilistic models. Usage is generally discouraged.

$$g(a) = \zeta(a) = \log(1 + e^a)$$

viii. Hard tanh: shaped similarly to tanh but bounded.

$$g(a) = \max(-1, \min(1, a))$$

Theorem 2.3.8. *Universal Approximation Theorem*
A feedforward network with linear output layer and at least one hidden layer with any 'squashing' activation function can approximate any Borel measurable function from one finite-dimensional space to another with any desired non-zero amount of error, provided the network is given enough hidden units.

Method 2.3.9. *Architecture Design*
Neural network layers are arranged in a chain structure, with each layer being a function of preceding layer.

$$\begin{aligned}\boldsymbol{h}^{(1)} &= g^{(1)}(\boldsymbol{W}^{(1)T}\boldsymbol{x} + \boldsymbol{b}^{(1)}) \\ \boldsymbol{h}^{(k)} &= g^{(k)}(\boldsymbol{W}^{(k)T}\boldsymbol{h}^{(k-1)} + \boldsymbol{b}^{(k)}), \quad k \geq 2\end{aligned}$$

The main considerations are the depth of network and width of each layer. Deeper networks use far fewer units per layers and far fewer parameters, and often generalise to the test set, but are harder to optimise.

Remark 2.3.10. *Depth of Network and Universal Approximation Theorem*
A feedforward network with single layer is sufficient to represent any function, but the layer may be infeasibly large and fail to learn and generalise correctly. Using deeper models can reduce number of units required to represent the desired function and can reduce generalisation error.
Shallow networks with broad family of non-polynomial activation functions have universal approximation properties. Piecewise linear networks can represent functions with number of regions that is exponential to depth. The number of linear regions carved out by deep rectifier network with d inputs, depth l, and n units per hidden layer is $O(\binom{n}{d}^{d(d-1)} n^d)$. For maxout networks with k filters per unit, this is $O(k^{(k-1)+d})$.

Remark 2.3.11. *Backpropagation*
To calculate the gradient of loss function with respect to each of individual parameters of the neural network. Model training begins with random initialisation of weights and biases.

i. Forward pass: input is sampled from training data. Nodes receive input vector and passes their value (multiplied by random initial weight) to nodes of first hidden layer. The hidden units take weighted sum of these output values as an input to the activation function, whose output is used for next hidden layer.
ii. Error computation: the final output of the network is compared to the ground truth, difference is calculated for the error value.
iii. Backwards pass: the error value computed earlier is used to compute the gradient of loss function. The gradient is then propagated back through the network, and the weights are updated according to their contribution to the error. The learning rate determines the size of weight updates.
iv. Weights update: the weights are updated in opposite direction of the gradient

Algorithm 1: Backpropagation Learning Algorithm

```
Require:
  A set of training examples $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$
  A multilayer network with $L$ layers, weights $w_{ij}^l$, and activation function $f$
  Loss function $J(y, o)$
  Learning rate $0 < \alpha < 1$
  Number of epochs *epochs*.

  *Initialize all weights:*
  for $w_{ij}^l$ in the network do
     $w_{ij}^l \leftarrow$ small random number
  end for

  for $i = 1$ to *epochs* do
     for all training examples $(x, y) \in \mathcal{D}$ do

        *Propagate the inputs forward to compute the outputs*
        for all neurons $i$ in the input layer do
           $a_i^0 \leftarrow x_i$
        end for
        for $l = 2$ to $L$ do
            for all neurons $i$ in layer $l$ do
                $z_i^l \leftarrow \sum_j w_{ij}^l \, a_j^{l-1}$
                $a_i^l \leftarrow f\left(z_i^l\right)$
            end for
        end for

        *Propagate deltas backward from output layer to input layer*
        for all neurone $i$ in the output layer do
            *Let $o_i \equiv a_i^L$ be output of neurone $i$*
            $\delta_i^L \leftarrow \frac{\partial J(y_i, o_i)}{\partial o_i} f'\left(z_i^L\right)$
        end for

        for $l = L - 1$ to $1$ do
            for all neurons $i$ in layer $l$ do
                $\delta_i^l \leftarrow f'\left(z_i^l\right) \sum_j \left(w_{ji}^{l+1} \, \delta_j^{l+1}\right)$
            end for
        end for

        *Update weights using the deltas*
        for all $w_{ij}^l$ in the network do
            $w_{ij}^l \leftarrow w_{ij}^l - \alpha \, \delta_i^l \, a_j^{l-1}$
        end for
    end for
end for
```

2.3.2 Regularisation for Deep Learning

Definition 2.3.12. *Regularisation* refers to adding a parameter norm penalty $\Omega(\boldsymbol{\theta})$ to the objective function J. The regularised objective function is then

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha\Omega(\boldsymbol{\theta})$$

where $\alpha in[0, \infty)$ is a hyper-parameter that weights the contribution of norm penalty term.
For neural networks, the parameter norm penalty Ω is chosen such that it penalises only the weights of the affine transformation at each layer and leaves the biases un-regularised.

Definition 2.3.13. *L^2/Ridge Regularisation*
The regularisation term added to objective function is

$$\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$$

Remark 2.3.14. *Behaviour of Weight Decay (L^2) Regularisation*
Assuming no bias parameter, a model have the following objective function and parameter gradient:

$$\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w} + J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$$
$$\nabla_{\boldsymbol{w}}\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \alpha\boldsymbol{w} + \nabla_{\boldsymbol{w}}J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$$

On a single gradient step, the update is as follows:

$$\boldsymbol{w} \leftarrow (1 - \epsilon\alpha)\boldsymbol{w} - \epsilon\nabla_{\boldsymbol{w}}J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$$

The addition of weight decay has modified learning rule to multiplicatively shrink weight vector by constant factor on each step before gradient update.
Using quadratic approximation to objective function at minimal unregularised training cost, approximation is

$$\hat{J}(\overline{\theta}) = J(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)\boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*), \qquad \boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} J(\boldsymbol{w})$$

where $\boldsymbol{H}$ is Hessian matrix of J with respect to $\boldsymbol{w}$ evaluated at $\boldsymbol{w}^*$. Minimum of $\hat{J}$ occurs where gradient is

$$\nabla_{\boldsymbol{w}}\hat{J}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w}-\boldsymbol{w}^*) = \boldsymbol{0}$$

Adding weight decay gradient, solve for minimum of regularised version of $\hat{J}$. Let $\tilde{\boldsymbol{w}}$ be minimum, then

$$\begin{aligned}\alpha\tilde{\boldsymbol{w}} + \boldsymbol{H}(\tilde{\boldsymbol{w}}-\boldsymbol{w}^*) &= 0\\ (\boldsymbol{H}+\alpha\boldsymbol{I})\tilde{\boldsymbol{w}} &= \boldsymbol{H}\boldsymbol{w}^*\\ \tilde{\boldsymbol{w}} &= (\boldsymbol{H}+\alpha\boldsymbol{I})^{-1}\boldsymbol{H}\boldsymbol{w}^*\end{aligned}$$

As $\alpha \to 0$, regularised solution $\tilde{\boldsymbol{w}} \to \boldsymbol{w}^*$. Note $\boldsymbol{H}$ is real and symmetric, hence decompose into diagonal matrix $\boldsymbol{A}$ and orthonormal basis of eigenvectors $\boldsymbol{Q}$ such that $\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$, to get

$$\tilde{\boldsymbol{w}} = (\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T+\alpha\boldsymbol{I})^{-1}\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T\boldsymbol{w}^* = \boldsymbol{Q}(\boldsymbol{\Lambda}+\alpha\boldsymbol{I})^{-1}\boldsymbol{\Lambda}\boldsymbol{Q}^T\boldsymbol{w}^*$$

The effect of weight decay rescale $\boldsymbol{w}^*$ along axes defined by eigenvectors of $\boldsymbol{H}$. The component of $\boldsymbol{w}^*$ aligned with i-th eigenvector of $\boldsymbol{H}$ is rescaled by factor $\frac{\lambda_i}{\lambda_i+\alpha}$. For components where $\lambda_i \gg \alpha$, the effects of regularisation is relatively small. For components where $\lambda_i \ll \alpha$, components will be shrunk to nearly zero magnitude.

Definition 2.3.15. *L^1 Regularisation*
The L^1 regularisation is the sum of absolute values of individual parameters.

$$\Omega(\boldsymbol{\theta}) = \|\boldsymbol{w}\|_1 = \sum_i |w_i|$$

Definition 2.3.16. *Behaviour of L^1 Regularisation*
Assuming no bias parameter, a model has the following objective function and parameter gradient:

$$\begin{aligned}\tilde{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y}) &= \alpha\|\boldsymbol{w}\|_1 + J(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})\\ \nabla_{\boldsymbol{w}}\tilde{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y}) &= \alpha\ \text{sign}(\boldsymbol{w}) + \nabla_{\boldsymbol{w}}J(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})\end{aligned}$$

Note that the regularisation contribution to gradient is a constant factor with sign equal to sign(w_i). Minimum of $\tilde{J}$ occurs at where

$$\nabla_{\boldsymbol{w}}\tilde{J}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w}-\boldsymbol{w}^*)$$

Assuming the Hessian is diagonal, $\boldsymbol{H} = \text{diag}([H_{1,1},\ldots,H_{n,n}])$, where each $H_{i,i} > 0$. The quadratic approximation of regularised objective function is then

$$\begin{aligned}\tilde{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y}) &= J(\boldsymbol{w}^*;\boldsymbol{X},\boldsymbol{y}) + \sum_i\left[\frac{1}{2}H_{i,i}(\boldsymbol{w}_i-\boldsymbol{w}_i^*)^2 + \alpha\,|w_i|\right]\\ w_i &= \text{sign}(w_i^*)\max\left\{|w_i^*| - \frac{\alpha}{H_{i,i}}, 0\right\}\end{aligned}$$

For the case where $w_i^* > 0$ for all i, then

i. if $w_i^* \le \frac{\alpha}{H_{i,i}}$, the optimal value is $w_i = 0$ as contribution of $J(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})$ to regularised objective $\tilde{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})$ is overwhelmed in direction i by L^1 regularisation which pushes w_i to zero.
ii. if $w_i^* > \frac{\alpha}{H_{i,i}}$, regularisation shifts optimal value of w_i in the direction by $\frac{\alpha}{H_{i,i}}$.

For the case where $w_i^* < 0$, this happens similarly, but with L^1 penalty decreasing w_i by $\frac{\alpha}{H_{i,i}}$, with min value 0. Note that L^1 produces a more sparse solution, which is a feature selection mechanism.

Remark 2.3.17. *Norm Penalties as Constrained Optimisation*
Let cost function regularised by parameter norm penalty be

$$\tilde{J}(\boldsymbol{\theta};\boldsymbol{X},\boldsymbol{y}) = J(\boldsymbol{\theta};\boldsymbol{X},\boldsymbol{y}) + \alpha\Omega(\boldsymbol{\theta})$$

Construct a generalised generalised Lagrange function,

$$\mathcal{L}(\boldsymbol{\theta};\boldsymbol{X},\boldsymbol{y}) = J(\boldsymbol{\theta};\boldsymbol{X},\boldsymbol{y}) + \alpha(\Omega(\boldsymbol{\theta}) - k)$$

The solution to the constrained problem is then

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}}\max_{\alpha,\alpha\ge 0}\mathcal{L}(\boldsymbol{\theta},\alpha)$$

Note that optimal value α^* will shrink $\Omega(\boldsymbol{\theta})$, but not such that it is less than k.
Fixing α^*, the problem is then a function of $\boldsymbol{\theta}$,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \alpha^*) = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha^* \Omega(\boldsymbol{\theta})$$

This is the regularised training problem of minimising $\tilde{J}$. The parameter norm penalty is imposing a constraint on the weights. If explicit constraints are to be used rather than penalties, use stochastic gradient descent on $J(\boldsymbol{\theta})$, then take the projected $\boldsymbol{\theta}$ to the nearest point that satisfies $\Omega(\boldsymbol{\theta}) < k$. This method is used if k is predefined by user, and time is not to be spent on searching for α value that corresponds to this k.
Explicit constraints and re-projection work better in circumstances where non-convex optimisation is involved, and this may avoid getting stuck in local minima. Explicit constraints also impose stability, and together with high learning rate allows rapid exploration of parameter space.

2.3.3 Optimisation for Deep Learning

Remark 2.3.18. *Optimisation*
A cost function $J(\boldsymbol{\theta})$ is reduced to improve some performance measure P.
Optimisation algorithms include some specialisation on specific structure of machine learning objective functions.
Cost function is average over training set,

$$J(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\hat{p}_{\text{data}}} L(f(\boldsymbol{x};\boldsymbol{\theta}), y)$$

where L is per-example loss function, $f(\boldsymbol{x};\boldsymbol{\theta})$ is predicted output given input $\boldsymbol{x}$, $\hat{p}_{\text{data}}$ is empirical distribution.
For supervised learning, y is target output.
The goal is to usually to minimise objective function where expectation is taken on data generating distribution p_{data} rather than over the finite training set:

$$J^*(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},y)\sim p_{\text{data}}} L(f(\boldsymbol{x};\boldsymbol{\theta}), y)$$

Note that machine learning algorithm usually minimises a surrogate loss function, and halts when convergence criterion based on early stopping is satisfied.

Definition 2.3.19. *Empirical Risk Minimisation*
To minimise expected loss on training set, where $\hat{p}(\boldsymbol{x}, y)$ is the empirical distribution. The empirical risk is

$$\mathbb{E}_{\boldsymbol{x},y\sim\hat{p}_{\text{data}}(\boldsymbol{x},y)}[L(f(\boldsymbol{x},\boldsymbol{\theta}), y)] = \frac{1}{m}\sum_{i=1}^{m} L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}), y^{(i)})$$

where m is number of training samples. Note that empirical risk minimisation is prone to overfitting, as models with high capacity may memorise the training set.

Remark 2.3.20. *Batch and Minibatch Algorithms*
Objective algorithms for machine learning computes each update to the parameters based on expected value of cost function using only a subset of terms of full cost function.
For maximum likelihood estimation problems, note that

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p_{\text{model}}(\boldsymbol{x}^{(i)}, y^{(i)}; \boldsymbol{\theta})$$
$$J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x},y\sim\hat{p}_{\text{data}}} \log p_{\text{model}}(\boldsymbol{x}, y; \boldsymbol{\theta})$$

The gradient is then

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x},y\sim\hat{p}_{\text{data}}} \nabla_{\boldsymbol{\theta}} \log\log p_{\text{model}}(\boldsymbol{x}, y; \boldsymbol{\theta})$$

If the optimisation algorithm uses the entire training set, then it is a batch gradient method.
If a single sample is used at a time, then this is a stochastic/online gradient method.
For algorithms that uses more than one but less than all samples, then this is a minibatch gradient method.

Remark 2.3.21. *Challenges: Ill-Conditioning of Hessian Matrix H*
May cause stochastic gradient descent to be stuck where very small step increases the cost function.
Note that a gradient descent step of $-\epsilon\boldsymbol{g}$ will add

$$\frac{1}{2}\epsilon^2 \boldsymbol{g}^T \boldsymbol{H} \boldsymbol{g} - \epsilon \boldsymbol{g}^T \boldsymbol{g}$$

to the cost. This will be a problem when $\frac{1}{2}\epsilon^2 \boldsymbol{g}^T \boldsymbol{H}\boldsymbol{g} > \epsilon \boldsymbol{g}^T \boldsymbol{g}$.
Monitor squared gradient norm $\boldsymbol{g}^T\boldsymbol{g}$ and $\boldsymbol{g}^T\boldsymbol{H}\boldsymbol{g}$ term. When ill-conditioning happens, gradient norm does not shrink significantly but $\boldsymbol{g}^T\boldsymbol{H}\boldsymbol{g}$ term grows more than order of magnitude, and learning rate shrinks.

Remark 2.3.22. *Challenges: Local Minimums*
Model is identifiable if a sufficiently large training set can rule out all but one setting of model parameters.
There may be an extremely large or uncountably infinite amount of local minima in a cost function. If the local minima have a high cost compared to global minimum, then this is problematic.

Remark 2.3.23. *Challenges: Plateaus, Saddle Points, Flat Regions*
For a function $f : \mathbb{R}^n \to \mathbb{R}$, expected ratio of saddle points to local minima grows exponentially with n. Hessian matrix at a saddle point has a mixture of positive and negative eigenvalues.
Degenerate regions with flat gradient may correspond to high value of objective function.

Remark 2.3.24. *Challenges: Cliffs and Exploding Gradients*
On extremely steep cliff structure, gradient update step may move parameters extremely far, losing most of the optimisation work done previously. The consequences may be mitigated using gradient clipping heuristics, which reduces the step size to be small enough.

Remark 2.3.25. *Challenges: Long-Term Dependencies*
Occurs when computational graph becomes very deep. Let a computational graph contain a path that repeatedly multiply a matrix $\boldsymbol{W}$. After t steps, this is $\boldsymbol{W}^t$. Let the eigen-decomposition of $\boldsymbol{W}$ be $\boldsymbol{W} = \boldsymbol{V}\text{diag}(\boldsymbol{\lambda})\boldsymbol{V}^{-1}$. Then we can see that $\boldsymbol{W}^t = (\boldsymbol{V}\text{diag}(\boldsymbol{\lambda})\boldsymbol{V}^{-1})^t = \boldsymbol{V}\text{diag}(\lambda)^t\boldsymbol{V}^{-1}$.
There will be a vanishing and exploding gradient problem if gradients on the graph are scaled according to $\text{diag}(\boldsymbol{\lambda})^t$. Note that recurrent networks uses same $\boldsymbol{W}$ at each time step, hence may face the problem.

```
Require: Learning rate $\epsilon_k$.
Require: Initial parameter $\boldsymbol{\theta}$
  while stopping criterion not met do
    Sample a minibatch of $m$ examples from the training set $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ with
    corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow +\frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon\hat{\boldsymbol{g}}$
  end while
```

Figure 6: Stochastic Gradient Descent (SGD) update at training iteration k

Remark 2.3.26. *Stochastic Gradient Descent (SGD)*
Crucial parameter is the learning rate, which may be changed over time; at iteration k this is ϵ_k.
The SGD gradient estimation introduces a source of noise that does not vanish even at minimum. A sufficient condition to guarantee converge of SGD is

$$\sum_{k=1}^{\infty} \epsilon_k = \infty, \quad \sum_{k=1}^{\infty} \epsilon_k^2 < \infty$$

In practice, it is common to decay learning rate linearly until iteration τ:

$$\epsilon_k = (1-\alpha)\epsilon_0 + \alpha\epsilon_\tau$$

where $\alpha = \frac{k}{\tau}$. After iteration τ, leave ϵ as a constant.
Note that SGD computation time per update does not growth with number of training examples.

```
Require: Learning rate $\epsilon$, momentum parameter $\alpha$.
Require: Initial parameter $\boldsymbol{\theta}$, initial velocity $\boldsymbol{v}$.
  while stopping criterion not met do
    Sample a minibatch of $m$ examples from the training set $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ with
    corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient estimate: $\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Compute velocity update: $\boldsymbol{v} \leftarrow \alpha\boldsymbol{v} - \epsilon\boldsymbol{g}$
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$
  end while
```

Figure 7: Stochastic Gradient Descent (SGD) with momentum

Remark 2.3.27. *Momentum Algorithm*
Algorithm introduces variable $\boldsymbol{v}$ that is set to an exponentially decaying average of negative gradient. Hyperparameter $\alpha \in [0, 1)$ determines how quickly the contributions of previous gradients exponentially decay. The update rule is given by

$$\begin{aligned}
&\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \nabla_{\boldsymbol{\theta}} \left(\frac{1}{m} \sum_{i=1}^{m} L(\boldsymbol{f}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}) \right) \\
&\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}
\end{aligned}$$

The larger α is relative to ϵ, the more previous gradients affect current direction.

```
Require: Learning rate $\epsilon$, momentum parameter $\alpha$.
Require: Initial parameter $\boldsymbol{\theta}$, initial velocity $\boldsymbol{v}$.
  while stopping criterion not met do
    Sample a minibatch of $m$ examples from the training set $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ with corresponding labels $\boldsymbol{y}^{(i)}$.
    Apply interim update: $\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta} + \alpha \boldsymbol{v}$
    Compute gradient (at interim point): $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\tilde{\boldsymbol{\theta}}} \sum_i L(f(\boldsymbol{x}^{(i)}; \tilde{\boldsymbol{\theta}}), \boldsymbol{y}^{(i)})$
    Compute velocity update: $\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \boldsymbol{g}$
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$
  end while
```

Figure 8: Stochastic Gradient Descent (SGD) with Nesterov momentum

Remark 2.3.28. *Nesterov Momentum*
Inspired by Nesterov's accelerated gradient method, the update rules are:

$$\begin{aligned}
&\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \nabla_{\boldsymbol{\theta}} \left(\frac{1}{m} \sum_{i=1}^{m} L(\boldsymbol{f}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta} + \alpha \boldsymbol{v}), \boldsymbol{y}^{(i)}) \right) \\
&\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}
\end{aligned}$$

Note, gradient is evaluated after current velocity is applied.

```
Require: Global learning rate $\epsilon$
Require: Initial parameter $\boldsymbol{\theta}$
Require: Small constant $\delta$, perhaps $10^{-7}$, for numerical stability
  Initialize gradient accumulation variable $\boldsymbol{r} = \boldsymbol{0}$
  while stopping criterion not met do
    Sample a minibatch of $m$ examples from the training set $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Accumulate squared gradient: $\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g}$
    Compute update: $\Delta\boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\delta + \sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$.   (Division and square root applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
  end while
```

Figure 9: AdaGrad Algorithm

Remark 2.3.29. *AdaGrad Algorithm*
Algorithm individually adapts learning rates of all model parameters by scaling them inversely proportional to square root of sum of all of their historical squared values. Has greater progress in more gently sloped directions of parameter space. On deep neural networks, accumulation of squared gradients from beginning of training can result in premature and excessive decrease in effective learning rate.

```
Require: Global learning rate $\epsilon$, decay rate $\rho$.
Require: Initial parameter $\boldsymbol{\theta}$
Require: Small constant $\delta$, usually $10^{-6}$, used to stabilize division by small
  numbers.
  Initialize accumulation variables $\boldsymbol{r} = 0$
  while stopping criterion not met do
    Sample a minibatch of $m$ examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with
    corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Accumulate squared gradient: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1-\rho)\boldsymbol{g} \odot \boldsymbol{g}$
    Compute parameter update: $\Delta\boldsymbol{\theta} = -\frac{\epsilon}{\sqrt{\delta + \boldsymbol{r}}} \odot \boldsymbol{g}$.   ($\frac{1}{\sqrt{\delta+\boldsymbol{r}}}$ applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
  end while
```

Figure 10: RMSPorp Algorithm

```
Require: Global learning rate $\epsilon$, decay rate $\rho$, momentum coefficient $\alpha$.
Require: Initial parameter $\boldsymbol{\theta}$, initial velocity $\boldsymbol{v}$.
  Initialize accumulation variable $\boldsymbol{r} = \boldsymbol{0}$
  while stopping criterion not met do
    Sample a minibatch of $m$ examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with
    corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute interim update: $\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta} + \alpha \boldsymbol{v}$
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\tilde{\boldsymbol{\theta}}} \sum_i L(f(\boldsymbol{x}^{(i)}; \tilde{\boldsymbol{\theta}}), \boldsymbol{y}^{(i)})$
    Accumulate gradient: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1-\rho)\boldsymbol{g} \odot \boldsymbol{g}$
    Compute velocity update: $\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \frac{\epsilon}{\sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$.   ($\frac{1}{\sqrt{\boldsymbol{r}}}$ applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$
  end while
```

Figure 11: RMSPorp Algorithm with Nesterov momentum

Remark 2.3.30. *RMSProp Algorithm*
Modifies AdaGrad to perform better in non-convex setting by changing gradient accumulation into exponentially weighted moving average. RMSProp is effective and practical for deep neural networks.

```
Require: Step size $\epsilon$ (Suggested default: 0.001)
Require: Exponential decay rates for moment estimates, $\rho_1$ and $\rho_2$ in $[0, 1)$.
  (Suggested defaults: 0.9 and 0.999 respectively)
Require: Small constant $\delta$ used for numerical stabilization. (Suggested default:
  $10^{-8}$)
Require: Initial parameters $\boldsymbol{\theta}$
  Initialize 1st and 2nd moment variables $\boldsymbol{s} = \boldsymbol{0}$, $\boldsymbol{r} = \boldsymbol{0}$
  Initialize time step $t = 0$
  while stopping criterion not met do
    Sample a minibatch of $m$ examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with
    corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    $t \leftarrow t + 1$
    Update biased first moment estimate: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1-\rho_1)\boldsymbol{g}$
    Update biased second moment estimate: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1-\rho_2)\boldsymbol{g} \odot \boldsymbol{g}$
    Correct bias in first moment: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1-\rho_1^t}$
    Correct bias in second moment: $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1-\rho_2^t}$
    Compute update: $\Delta\boldsymbol{\theta} = -\epsilon \frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}} + \delta}$   (operations applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
  end while
```

Figure 12: Adam Algorithm

Remark 2.3.31. *Adam Algorithm*
Variant on combination of RMSProp and momentum. Note, momentum is incorporated directly as an estimate of first order moment of gradient; bias corrections are made to estimates of both first-order moments and

second-order moments to account for initialisation at origin.
Algorithm is fairly robust to choice of hyper-parameters.

```
Require: Initial parameter $\boldsymbol{\theta}_0$
Require: Training set of $m$ examples
  while stopping criterion not met do
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}}\sum_i L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Compute Hessian: $\boldsymbol{H} \leftarrow \frac{1}{m}\nabla^2_{\boldsymbol{\theta}}\sum_i L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Compute Hessian inverse: $\boldsymbol{H}^{-1}$
    Compute update: $\Delta\boldsymbol{\theta} = -\boldsymbol{H}^{-1}\boldsymbol{g}$
    Apply update: $\boldsymbol{\theta} = \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
  end while
```

Figure 13: Newton's Method with Objective $J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^m L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}), y^{(i)})$

Remark 2.3.32. *Newton's Method*
Based on second-order Taylor's series approximating $J(\boldsymbol{\theta})$ near the point $\boldsymbol{\theta}_0$, ignoring higher derivatives

$$J(\boldsymbol{\theta}) \approx J(\boldsymbol{\theta}_0) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^T\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta}_0) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)^T\boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)$$

where $\boldsymbol{H}$ is the Hessian of J with respect to $\boldsymbol{\theta}$ evaluated at $\boldsymbol{\theta}_0$.
Solving for critical point will get the Newton parameter update rule

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}_0 - \boldsymbol{H}^{-1}\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta}_0)$$

Note that for deep learning, the surface of objective function is non-convex. Regularising the Hessian will solve many of the issues such as saddle points etc.

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}_0 - [H(f(\boldsymbol{\theta}_0)) + \alpha\boldsymbol{I}]^{-1}\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta}_0)$$

Due to requirement for inversion of the $k \times k$ matrix with time complexity $O(k^3)$ at every training iteration, only networks with very small number of parameters may be trained with this method.

```
Require: Initial parameters $\boldsymbol{\theta}_0$
Require: Training set of $m$ examples
  Initialize $\boldsymbol{\rho}_0 = \boldsymbol{0}$
  Initialize $g_0 = 0$
  Initialize $t = 1$
  while stopping criterion not met do
    Initialize the gradient $\boldsymbol{g}_t = \boldsymbol{0}$
    Compute gradient: $\boldsymbol{g}_t \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}}\sum_i L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Compute $\beta_t = \frac{(\boldsymbol{g}_t - \boldsymbol{g}_{t-1})^\top\boldsymbol{g}_t}{\boldsymbol{g}_{t-1}^\top\boldsymbol{g}_{t-1}}$ (Polak-Ribière)
    (Nonlinear conjugate gradient: optionally reset $\beta_t$ to zero, for example if $t$ is
    a multiple of some constant $k$, such as $k = 5$)
    Compute search direction: $\boldsymbol{\rho}_t = -\boldsymbol{g}_t + \beta_t\boldsymbol{\rho}_{t-1}$
    Perform line search to find: $\epsilon^* = \text{argmin}_\epsilon \frac{1}{m}\sum_{i=1}^m L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}_t + \epsilon\boldsymbol{\rho}_t), \boldsymbol{y}^{(i)})$
    (On a truly quadratic cost function, analytically solve for $\epsilon^*$ rather than
    explicitly searching for it)
    Apply update: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \epsilon^*\boldsymbol{\rho}_t$
    $t \leftarrow t + 1$
  end while
```

Figure 14: Conjugate Gradient Method

Remark 2.3.33. *Conjugate Gradient Method*
Method effectively avoids computation of inversion Hessian by descending conjugate directions, which is a search direction conjugate to the previous line search direction (will not undo progress made in that direction).
Let current and previous search directions be $\boldsymbol{d}_t, \boldsymbol{d}_{t-1}$. At training iteration t, the next search direction $\boldsymbol{d}_t$ is

$$\boldsymbol{d}_t = \nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta}) + \beta_t\boldsymbol{d}_{t-1}$$

note that β_t may be computed via Fletcher Reeves:

$$\beta_t = \frac{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)^T \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)}{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_{t-1})^T \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_{t-1})}$$

or may be computed with Polak-Ribiere:

$$\beta_t = \frac{(\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t) - \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_{t-1}))^T \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)}{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_{t-1})^T \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_{t-1})}$$

The non-linear conjugate gradient method is a variation which uses occasional resets where the method is restarted with line search along with the unaltered gradient.

Remark 2.3.34. *Broyden-Fletcher-Goldfarb-Shanno (BGFS) Algorithm*
Similar to conjugate gradient, but uses more direct approach to approximation of newton's update.
Method approximate the inverse of hessian $\boldsymbol{H}$ with matrix $\boldsymbol{M}_t$, iteratively refined by low rank updates to better approximate $\boldsymbol{H}^{-1}$. Direction of gradient descent is then determined by $\boldsymbol{\rho}_t = \boldsymbol{M}_t \boldsymbol{g}_t$. Line search is performed in this direction to determine step size ϵ^*. Final update of parameter is then $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \epsilon^* \boldsymbol{\rho}_t$.
Note, BGFS store matrix $\boldsymbol{M}$ which requires $O(n^2)$ memory, making it impractical for most modern models.

3 Market Microstructure

3.1 Market Fundamentals

The basic function of a market is to bring buyers and sellers together.

Process 3.1.1. *(Four Components of a Trade)*

i. Acquisition of information and quotes
   1. Quality information and transparency are crucial to price discovery
   2. Transparent markets quickly disseminate high-quality information
   3. Opaque markets are those that lack transparency

ii. Routing of the trade order
   1. Selecting the brokers to handle the trades
   2. Deciding which markets will execute the trades and transmitting the trades to the markets

iii. Execution. Buys are matched and executed against sells according to the rules of the market

iv. Confirmation, clearance and settlement
   1. Clearance is the recording and comparison of the trade records
   2. Settlement involves the actual delivery of the security and its payment
   3. Might include trade allocation

Remark 3.1.2. *(Risks of Algorithmic Trading)*

i. Leaks might arise from competitor efforts to revere engineer them

ii. Many algorithms lack the capacity to handle or respond to exceptional or rare events.

Remark 3.1.3. *(Cornering-the-Market)* Trader takes huge long futures position and tries to exercise control over supply of underlying commodity. As maturity of futures contract is approaching, position is not closed, number of outstanding contracts exceed commodities available. Holders of short positions desperately try to close positions, leading to rise in both futures and spot prices.
Abuse is dealt with by increasing margin requirements or imposing stricter position limits or prohibiting trades that increase speculator's open position or requiring market participants to close their positions.

3.1.1 Liquidity Access in Equity Markets

(Exchanges) Account for 60% to 70% of all activity. The full order-book, arrivals/cancellations are all published, the liquidity information is transparent. Larger orders may impact the market. Liquidity at best price cannot be ignored, as the exchange will need to reroute to other exchanges where price is better while charging a fee (National Best Bid Offer, NBBO). All exchanges have almost exactly the same trading mechanism, and behave exactly the same way during the trading day except for opening and closing auctions.
Most exchanges use maker-taker fee model, but some exchanges (BYX, EDGA, NSX, BX) have an inverted fee model where rebate is provided for taking liquidity. IEX uses a speed bump to remove speed advantages of HFTs, providing a less 'toxic' liquidity pool.

(Dark Pools/ATS) Dark pools do not display any order information and use the NBBO as reference price. To avoid accessing protected venues, these pools trade only at the inside market (at or within the bid ask spread). Still possible to identify large blocks of liquidity by 'pinging' the pool at minimum lot size; counteracted by sending orders with minimum fill quantity tag, which allows block to be transparent from small pinging.
Most of ATS are run by major investment banks; some venues allow direct trading between investment firms. Note that many of the trading strategies used by firms tend to be highly correlated, hence liquidity is often on the same side; venues have to leverage sell side broker's liquidity to supplement their own.

(Single Dealer's Platform/Systematic Internalisers) Broker/Dealer and other institutional clients connect to Single Dealer Platform (SDP) directly. Not regulated ATS, hence can offer unique products. Brokers provide their own SDPs to expose their internal liquidity.

(Auctions) Exchanges begin and end day with a primary auction procedure that leverages special order types to accumulate supply and demand, then run algorithm that determines best price that would at best pair off the most volume. An opportunity for active and passive investors to exchange large amounts of liquidity.

3.1.2 Trading Mechanisms

Most common approach by modern electronic exchanges is the time/price priority, *continuous double auction* trading system. There are multiple buyers and multiple sellers participating at the same time.

(Limit Order Book, LOB) Stores all non-executed orders with associated instructions. Highly efficient, able to handle a high degree of concurrency to ensure the state is always correct. Has 2 copies of core data structure (for Buy, Sell orders). Supports 3 basic instruction types (insert, cancel, amend). There are 4 events on both Buy and Sell that may alter state of order book (limit order submission, limit order cancellation, limit order amendment, execution).

(Matching Algorithm) Responsible for interpreting various events to determine if any buy and sell orders can be matched. When multiple orders can be paired, price/time priority is used, where the order with most competitive prices are matched, and when prices are equal, the order that arrived prior is chosen.

Exchanges will publish order imbalance that exists among orders on opening and closing books during *Open Auctions*, with indicative price and volume. The following are published every second on market data feeds:

i. *Current Reference Price*: Price at which paired shares are maximised, imbalance is minimised, distance from bid-ask mid-point is minimised, in that order
ii. *Near Indicative Clearing Price*: The crossing price at which orders in opening, closing book and continuous book would clear against each other
iii. *Far Indicative Clearing Price*: The crossing price at which orders in opening, closing book would clear against each other
iv. *Number of Paired Shares*: The number of on-open or on-close shares that is able to pair off at the current reference price
v. *Imbalance Shares*: The number of opening or closing shares that would remain unexecuted at the current reference price
vi. *Imbalance Side*: The side of imbalance. B is buy-side imbalance; S is sell-side imbalance; N is no imbalance; O is no marketable on-open or on-close orders

In general, the following rules apply to match supply and demand:

i. Crossing price must maximise volume transacted
ii. If several prices result in similar volume transacted, the crossing price is the one the closest from last price
iii. Cross price is identical for all orders executed
iv. If two orders are submitted at same price, the order submitted first has priority
v. It is possible for an order to be partially executed if the other side quantity is not sufficient
vi. 'At Market' orders are executed against each other at the determined crossing price, up to available matching quantity on both sides, but generally do not participate in price formation process
vii. For Open Auction, unmatched 'At Market' orders are entered into continuous session of LOB as limit orders at the crossing price

The open auction is a major price discovery mechanism, as it occurs after a period of market inactivity when market participants were unable to transact even if they have information. Market participants with better information are more likely to participate, with more aggressive orders to extract liquidity, hence the mechanism is quite volatile and more suited for short-term alpha investors.
During execution, smaller orders may avoid participation in open auction and period thereafter; larger orders may participate to extract significant liquidity from the market.

Diversity of order types is key component of continuous double auction electronic markets.

Definition 3.1.4.

i. *Market Order*: trade carried out immediately at best price available in market.
ii. *Limit Order*: only executed at this price or at one more favourable to the trader.
iii. *Stop/Stop-Loss Order*: not visible in LOB. Only become active when certain price is reached or passed, then enter order book as ether limit or market order depending on user setup.

iv. *Stop-Limit Order*: combination of stop order and limit order. Order becomes limit order as soon as a bid or ask is made at the price equal to or less favourable than stop price. If stop price and limit price is the same, then the order is *stop-and-limit* order.

v. *Trailing Stop Order*: function like stop orders, but stop price is set dynamic rather than static.

vi. *Market-if-Touched (MIT)/Board Order*: executed at best available price after trade occurs at a specified price or more favourable. Ensure profit is taken if sufficiently favourable price movements occur.

vii. *Market-Not-Held/Discretionary Order*: traded as market order, execution may be delayed at broker's discretion for better price.

viii. *All-Or-None Order*: request full execution of order. Not executed until full quantity is available.

ix. *Peg Order*: specify a price level at which order should be continuously and automatically repriced. Used for mid-point executions in non-displayed markets.

x. *Iceberg Order*: limit order with specific display quantity, designed to prevent information leakage.

xi. *Hidden Order*: when available to trade, not directly available to other market participants in central LOB

xii. *On-Open*: request execution at open price. Can be limit-on-open or market-on-open.

xiii. *On-Close*: request execution at close price. Can be limit-on-close or market-on-close.

xiv. *Imbalance Only*: provide liquidity intended to offset on-open/on-close order imbalances during opening/-closing cross. These generally are limit orders.

xv. *D-Quote*: Special order on NYSE, mainly used during close auction period.

xvi. *Funari*: Special order on Tokyo Stock Exchange, allows limit order placed in book during continuous session to automatically enter closing auction as market orders.

Instruction validity may take the following forms:

i. *Day Order*: valid for full duration of trading session

ii. *Extended Day Orders*: allows for trading in extended hours

iii. *Good-Till-Cancel Order*: in effect until executed or until end of trading in particular contract

iv. *Immediate-or-Cancel Order*: will be immediately cancelled back to sender after reaching matching engine if it does not get immediate fill.

v. *Fill-or-Kill Order*: must be executed immediately on receipt or none at all.

3.1.3 Market Microstructure Primer

A market microstructure analysis framework follows three main categories:

i. Price Formation and Price Discover: how prices impound information over time, and how determinants of trading costs vary

ii. Market Design: impact of trading rules on price formation. Choice of tick size, circuit breakers that halt trading in event of large price swings, degree of anonymity, transparency of information to market participants. These create a diverse set of constraints and opportunities.

iii. Transparency: quantity, quality and speed of information effect on trading process. Classified into pre-trade (lit order book), post-trade (trade reporting to public).

Topics for research in microstructure includes the following:

i. Parent order sliced into several child orders sent to market for execution; difficult to discern the informed trader who use sophisticated dynamic algorithms. Retail trades usually cross the spread.

ii. Understanding of trading intensity in short intervals. Order imbalance is empirically shown to be unrelated to price levels.

iii. Informed traders usage of hidden orders in entering and exiting the market require further studies.

iv. Traders respond to changing market conditions by revising quoted prices. Quote volatility can provide valuable information about perceived uncertainty in the market.

4 Equities Trading

5 Fixed Income Trading

6 Derivatives Trading

Based on the classic by John C. Hull (2021)

6.1 Fundamentals of the Market

The derivatives market is much larger than the stock market in terms of underlying assets. Derivatives may be used for hedging, speculation, or arbitrage; and also transfer a wide range of risks from one entity to another.

Definition 6.1.1. A *derivative* involves two parties agreeing to a future transaction, with value depending on the values of other underlying variables.

A derivatives exchange is a market where individuals and companies trade standardised contracts as defined by the exchange. Once two traders have agreed to trade a product offered by the exchange, it is handled by the exchange clearing house, which takes care of the credit risk by requiring each trade to deposit margin.

If the trade is taken over-the-counter (OTC), participants may present it to a central counterparty (CCP) or clear the trade bilaterally. With the 2008 financial crisis, OTC market is forced to become more like the exchange-traded market, with changes as follows:

i. Standardised OTC derivatives between two financial institutions in US must, whenever possible, be traded on a swap execution facility (SEF), where market participants can post bid and ask quotes, and can trade by accepting the quotes of other market participants.
ii. Require that a CCP be used for most standardised derivatives transactions between financial institutions.
iii. All trades must be reported to a central repository.

6.1.1 Forward, Futures, and Options

Definition 6.1.2. A *spot contract* is an agreement to buy or sell an asset almost immediately. A *forward contract* is an agreement or buy or sell an asset at a certain future time for a certain price.

Let S_T be the spot price of asset at maturity, K is delivery price.
The payoff from a long position in a forward contract is $S_T - K$.
The payoff from a short position in a forward contract is $K - S_T$.

Futures contract are traded on an exchange, unlike forwards which are traded OTC. Majority of futures contract do not lead to delivery, as positions are closed prior to delivery period by entering an opposite trade to the original one.

Party in short position may file notice of intention to deliver with the exchange when they are ready to deliver. If the asset is a commodity, the grade of commodity are specified. The contract size specifies the amount of asset that has to be delivered. The place for delivery must also be specified, as commodities may involve significant transportation costs. The delivery month of the commodity may also be specified, and are chosen by the exchange to meet the needs of market participants. Trading typically ceases a few days before the last day on which delivery can be made. Daily price movement limits are also specified by exchange to prevent speculative excess causing large price movements; in this case trading ceases for the day.

As the delivery price for a futures contract is approached, the futures price converges to the spot price of the underlying asset.

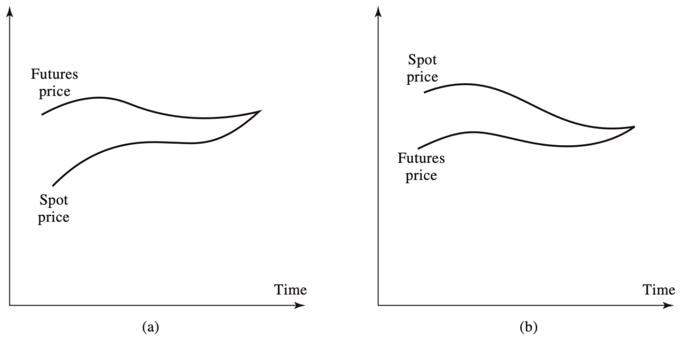


Figure 15: Convergence of futures price to spot price.

Suppose futures price is above spot price during delivery period. Traders have clear arbitrage opportunity: short futures, long asset, make delivery. The futures price will then fall.
If futures price is below spot price during delivery period, traders will long futures, wait for delivery, and the futures price will then rise.

Options are traded both on exchanges and OTC.

Definition 6.1.3. A *call option* (*put option*) gives the holder the right to sell (buy) the underlying asset by a certain date for a certain price.

Definition 6.1.4. An *American option* can be exercised at any time up to expiration date. An *European option* can only be exercised on expiration date itself.

6.1.2 Clearing House

Margin accounts are used by exchanges to organise trading so that contract defaults are avoided.
Trader has to keep funds in a margin account; the amount to be deposited at the time the contract is entered into is the *initial margin*. At each of trading day, margin account is adjusted to reflect trader's P&L (*daily settlement*, *marking to market*). Daily settlement leads to funds flowing daily between traders with long positions and traders with short positions; this daily flow of funds to reflect P&L is the *variation margin*. Trading via brokers requires a *maintenance margin*, which is lower than initial margin; if balance in margin account falls below maintenance margin, trader receives *margin call* and is required to top up to initial margin level within short period of time, or else broker closes out the position. Trader is also entitled to withdraw any balance in margin account that is in excess of the initial margin. Brokers pay interest on balance in margin account.
A forward contract is settled at end of life, while futures contract is settled daily. Minimum levels for initial and maintenance margins are set by exchange clearing house. Minimum margin levels are determined by variability of the price of underlying asset, revised when necessary.

The clearing house acts as intermediary in futures transactions, and keep track of all daily transactions for calculating net positions of each of its members. Members must provide initial margin reflecting the total number of contracts cleared. The maintenance margin is set equal to initial margin. In determining margin requirements, the number of contracts outstanding is calculated on a net basis rather than a gross basis. Members are required to contribute to a guaranty fund, which is used in the event that a member defaults, and the member's margin is insufficient to cover losses.

6.1.3 OTC Markets

OTC markets use central counterparties (CCPs), which perform the same role as exchange clearing houses. Members of CCP provide both initial margin and daily variation margin, and contribute to a guaranty fund. If an OTC derivative transaction has been agreed upon between parties A and B, and CCP accepts the transaction, they become the counterparty to both A and B. CCP hence takes on credit risk of both A and B.
Transactions are valued daily, and there are daily variation margin payments between members.

For bilaterally cleared OTC, two companies enter a master agreement covering all their trades (ISDA). The agreement includes a credit support annex (CSA), requiring both parties to provide collateral. Collateral agreements in CSAs usually require transactions to be valued daily. Since 2016, regulations require both initial margin and variation margin between financial institutions.

Definition 6.1.5. *Credit spread* is the difference between the interest rate and risk free rate.

(Treasury Rates) The rates on Treasury bills and Treasury bonds. The Treasury rates of developed countries are regarded as risk-free rates, as it is assumed that there is no chance the government will default.

(Overnight Rates) Borrowing and lending overnight by financial institutions to match asset and liabilities requirements for reserves; the overnight rate is the *federal funds rate*. The weighted average of rates in brokered transactions is the *effective federal funds rate*. The Federal Reserve may intervene with its own transactions to raise or lower the rates.

(Repo Rates) Secured borrowing rates; the difference between the price at which securities are sold and then repurchased. If structured carefully, involves very little credit risk. Most common type of repo is an *overnight repo*. In longer term arrangements, *term repos* are used.

The terminal value of an investment A invested at interest rate of R per annum, compounded m times per annum, is $A(1+\frac{R}{m})^{mn}$. If $m=1$, the rate is the *equivalent annual interest rate*. If continuous compounded is used, then the terminal value at the end of a year is Ae^R.

The *n-year spot-rate* is the interest rate earned on a zero-coupon bond. The *Bond Price* is the present value of all cash flows that will be received by owner of the bond, with different spot rate for each cash flow. The *Bond Yield* is the discount rate that, when applied to all cash flows, gives a bond price equal to its market price. The *Par Yield* for a certain bond maturity is the coupon rate that causes the bond price to equal its par value.

Definition 6.1.6. *Forward Rates* are rates implied by current spot rates for periods of time in the future.

Given R_1, R_2 the spot rates for maturities T_1, T_2 respectively, and R_F the forward rate between T_1 and T_2, then

$$R_F = \frac{R_2T_2 - R_1T_1}{T_2 - T_1} = R_2 + (R_2 - R_1)\frac{T_1}{T_2 - T_1}$$

Given the spot rate R for maturity T, the *instantaneous forward rate* for maturity of T is then

$$R_F = R + T\frac{\partial R}{\partial T}$$

If $P(0,T) = e^{-RT}$ is the price of zero-coupon bond maturity at time T, the equation is then

$$R_F = -\frac{\partial}{\partial T}\ln P(0,T)$$

Definition 6.1.7. *Forward Rate Agreement (FRA)* is an agreement to exchange a predetermined rate for a reference rate that will be observed in the market at a future time.

Let R_K be the fixed rate, R_F be the current forward rate for the reference rate, τ be the period of time to which the rates apply, L be the principal in the contract.
For the party that receives the fix rate, the FRA has a present value of

$$\tau(R_K - R_F)L$$

For the party that pays the fix rate, the FRA has a present value of

$$\tau(R_F - R_K)L$$

Definition 6.1.8. The *Duration* of a bond is a measure of how long the holder of the bond has to wait before receiving the present value of the cash payments.

Let c_i be cash flow at time t_i $(1 \le i \le n)$. Bond price B, yield y (continuously compounded) are related by

$$B = \sum_{i=1}^{n} c_i e^{-yt_i}$$

The Duration D of the bond is then

$$D = \sum_{i=1}^{n} t_i \left[\frac{c_i e^{-yt_i}}{B}\right]$$

where the term in square brackets is ratio of present value of cash flow at t_i to bond price. Duration is hence a time-weighted average of the times when payments are made.
The relationship between duration and yield is as follows:

$$\Delta B = \frac{dB}{dy}\Delta y$$
$$\frac{\Delta B}{B} = -D\Delta y$$

If y is expressed with compounding frequency of m times a year, then the relationship becomes

$$\Delta B = -\frac{BD\Delta y}{1 + y/m}$$

Hence, the *modified duration* is

$$D^* = \frac{D}{1 + y/m}$$

The duration relationship is then

$$\Delta B = -BD^* \Delta y$$

When duration is used for bond portfolios, it is assumed that the yields of all bonds will change by approximately the same amount, i.e., a parallel shift in the spot yield curve. The portfolio may still be exposed to shifts that are either large or non-parallel.
Convexity may be used to improve the relationship in the equation. Convexity is defined as

$$C = \frac{1}{B}\frac{d^2B}{dy^2}$$

By Taylor series, we then have the relationship

$$\frac{\Delta B}{B} = -D\Delta y + \frac{1}{2}C(\Delta y)^2$$

To determine the underlying reasons for the shape of the spot curve, theories on term structure of interest rates are referred to.

(Expectations Theory) A forward interest rate corresponding to a certain future period is equal to the expected future zero interest rate for that period.

(Market Segmentation Theory) There exists no relationship between short, medium, and long-term interest rates, as investors do not readily switch from one maturity to another.

(Liquidity Preference Theory) Investors prefer to preserve their liquidity and invest funds for short periods of time. Borrowers prefer to borrow at fixed rates for long periods of time. Hence forward rates are greater than expected future spot rates.

6.2 Forwards and Futures

Based on the book by John Hull (2021).

6.2.1 Pricing

To examine how forward prices and futures prices are related to spot price, we assume the following are true for some market participants:

i. No transaction costs
ii. Same tax rate on all net trading profits
iii. Money borrowed and lend are at the same risk-free rate
iv. Arbitrage opportunities are taken advantaged of as they occur

Let T be time until delivery date (in years), S_0 be price of underlying asset today, F_0 be price of forward or futures today, r be zero-coupon risk-free rate per annum in continuous compounding (maturing in T years).

Consider a forward contract on underlying asset with spot price S_0 that provides no income. Then

$$F_0 = S_0 e^{rT}$$

If $F_0 > S_0 e^{rT}$, long asset and short forward. If $F_0 < S_0 e^{rT}$, short asset and long forward.
If short sales are not possible, and arbitrage opportunities exist, then if $F_0 > S_0 e^{rT}$, investor may:

1. Borrow S_0 in cash at an interest rate r for T years
2. Buy 1 unit of asset
3. Enter forward contract to sell 1 unit of asset

At time T, asset is sold for F_0, investor to repay $S_0 e^{rT}$ loan, making profit $F_0 - S_0 e^{rT}$.
If $F_0 < S_0 e^{rT}$, investor may:

1. Sell asset for S_0
2. Invest proceeds at interest rate r for time T
3. Enter into forward contract to buy 1 unit of asset

At time T, cash has grown to $S_0 e^{rT}$. Investor repurchase asset for F_0, makes profit of $S_0 e^{rT} - F_0$.

If the underlying asset provide income with present value of I during life of forward, then

$$F_0 = (S_0 - I)e^{rT}$$

If $F_0 > (S_0 - I)e^{rT}$, investor may long asset and short forward. If $F_0 < (S_0 - I)e^{rT}$, investor may short asset and long forward. If short sales are not possible, investors owning the asset will sell the asset and long forward.

If the underlying asset provides a known yield rather than income, with q as the average yield, then the following strategy must generate zero profit to prevent arbitrage:

1. Borrow S_0 to buy one unit of asset at time 0
2. Enter into forward to sell e^{qT} units of asset at time T for F_0
3. Close the forward by selling e^{qT} units of the asset at time T

Hence we have

$$S_0 e^{rT} = e^{qT} F_0$$

or

$$F_0 = S_0 e^{(r-q)T}$$

When a forward contract is first entered, the value is close to zero. Let K be delivery price negotiated some time ago, with T years delivery date, r is T-year risk-free interest rate, F_0 is forward price if contract is negotiated today. Let f be value of forward contract today. Then

$$f = (F_0 - K)e^{-rT}$$

In the case of stock indices, if $F_0 > S_0 e^{(r-q)T}$, profit can be made by buying stocks underlying the index at spot price and shorting the index futures contract. If $F_0 < S_0 e^{(r-q)T}$, short the stocks and long futures. This is known as *index arbitrage*.

In the case of currencies, let r_f be foreign risk-free rate in a foreign-denominated bond, S_0 be spot price of local currency in foreign currency, F_0 be forward or future price of local currency in foreign currency. Then the *interest rate parity* relation persists in the form

$$F_0 = S_0 e^{(r-r_f)T}$$

For commodities, let U be present value of storage costs net of income during life of forward. Then

$$F_0 = (S_0 + U)e^{rT}$$

If storage costs are proportional to price of commodity, they can be treated as negative yield. Let u be storage costs per annum as proportion of spot price net of any yield earned on asset. Then

$$F_0 = S_0 e^{(r+u)T}$$

If $F_0 > (S_0 + U)e^{rT}$, then investor may take advantage of arbitrage:

1. Borrow $S_0 + U$ at risk-free rate, purchase one unit of commodity and pay storage costs
2. Short futures on one unit of commodity

If $F_0 < (S_0 + U)e^{rT}$, then investor may take advantage of arbitrage:

1. Sell commodity, save storage costs, invest proceeds at risk-free interest rate
2. Long futures contract

Benefits from holding physical assets are *convenience yields*, such as by crude oil manufacturers. Let y be the convenience yield, then

$$F_0 e^{yT} = (S_0 + U)e^{rT}$$

If storage costs per unit are a constant proportion of spot price, then y is defined such that

$$F_0 e^{yT} = S_0 e^{(r+u)T}$$

Convenience yield reflects market expectation on future availability of the commodity. The greater the possibility that shortages will occur, the higher the commodity yield.

The *cost of carry* measures storage cost plus interest paid to finance the asset less income earned:

i. Non-dividend paying stock: r, as no storage and income is earned
ii. Stock index: $r - q$, as income is earned at rate q on asset
iii. Currency: $r - r_f$
iv. Commodity: $r - q + u$, where it provides income at rate q and requires storage costs at rate u

6.2.2 Hedging with Futures

The fundamentals of hedging with futures are *hedge-and-forget* strategies, where no changes is made to adjust the hedge once it has been put in place.

Definition 6.2.1. *(Basic Principles of Futures Hedging)*
The objective is to take a position that neutralises the risk as far as possible.

i. *Short Hedge*: short position on futures.
Used when hedger already owns an asset and will sell the asset at some time in the future; or when asset is not owned right now but will be owned and ready for sale sometime in the future.
ii. *Long Hedge*: long position on futures.
Used when hedger will purchase an asset in the future and wants to lock in the price now.

In practice, hedging is not perfect due to factors as follows:

1. Asset being hedged is not exactly the same as the asset underlying the futures contract.

	Short Hedge	**Long Hedge**
May 15	Spot: 50 Futures: 49	Spot: 50 Futures: 49
August 15 Scenario 1	Spot: 45 Gain from hedge: 4	Spot: 45 Loss from hedge: 4
August 15 Scenario 2	Spot: 55 Loss from hedge: 6	Spot: 55 Gain from hedge: 6

2. Uncertainty as to exact date in which the asset will be bought or sold.
3. Hedge may require the futures contract to be closed out before its delivery month.

These lead to *basis risk*.

Definition 6.2.2. The *basis* in a hedging situation is defined as

$$\text{Basis} = \text{Spot Price} - \text{Futures Price}$$

An increase/decrease in basis is a strengthening/weakening of the basis.

Definition 6.2.3. Let S_i be spot price at time t_i, F_i be futures price at time t_i, b_i be basis price at time t_i. Assume hedge is placed at time t_1, closed at time t_2. Price realised for asset is S_2, profit from futures position is $F_1 - F_2$. Effective price obtained for asset hedging is therefore $S_2 + F_1 - F_2 = F_1 + b_2$.
If b_2 is known, perfect hedge will result. The *basis risk* is the hedging risk from uncertainty associated with b_2.

Definition 6.2.4. *Cross Hedging* occurs when the asset underlying the futures contract is not the same as the asset whose price is being hedged.

Cross hedging is often used when futures of the original asset being hedged are not actively traded on the market, and the hedger seeks an alternative asset to hedge the original asset.

Definition 6.2.5. *Hedge Ratio* is the ratio of size of position taken in futures contract to the size of exposure.

Assuming no daily settlement of futures contracts, hedger seeks a hedge ratio that minimises variance of hedged position value. Let ΔS be change in spot price, ΔF change in futures price. Assuming linear relationship,

$$\Delta S = a + b\Delta F + \epsilon$$

where a, b are constants, ϵ is an error term. Suppose hedge ratio is h. Change in value of position per unit of exposure to S is:

$$\Delta S - h\Delta F = a + (b - h)\Delta F + \epsilon$$

Standard deviation is minimised by setting $h = b$. Let minimum variance hedge ratio be h^*. Then

$$h^* = \rho \frac{\sigma_S}{\sigma_F}$$

where σ_S, σ_F is standard deviation of $\Delta S, \Delta F$ respectively, ρ is coefficient of correlation.

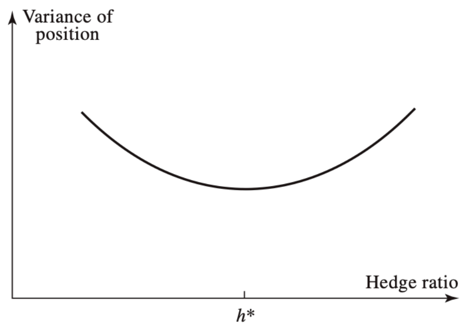


Figure 16: Dependence of variance of position on hedge ratio.

Hedge effectiveness is the proportion of variance eliminated by hedging. This is R^2 from regression of ΔS against ΔF, and equals ρ^2. Parameters ρ, σ_S, σ_F are estimated from historical data on ΔS and ΔF.

The optimal number of futures to be used in hedging is

$$N^* = \frac{h^* Q_A}{Q_F}$$

where Q_A is size of potion being hedged (units), Q_F is size of one futures contract (units). The futures contract should be on $h^* Q_A$ units of the asset.

If daily settlement is used, there are a series of one-day hedges, and thus let $\hat{\sigma}_S, \hat{\sigma}_F$ be standard deviation of percentage one-day changes in spot and future price respectively, $\hat{\rho}$ be correlation between percentage one-day changes in spot and future prices. The optimal one day hedge is then

$$h^* = \hat{\rho}\frac{\hat{\sigma}_S S}{\hat{\sigma}_F F}$$

and the optimal number of futures to be used is then

$$N^* = \hat{\rho}\frac{\hat{\sigma}_S S Q_A}{\hat{\sigma}_F F Q_F}$$

If an interest $r\%$ per annum is earned or paid over the remaining life of the hedge, then the optimal number of futures is $N^*/(1+0.01r)$; this is *tailing the hedge*.

Stock index futures may be used to hedge a well diversified equity portfolio. Let V_A, V_F be the current value of portfolio and one futures contract respectively.
If portfolio mirrors the index, the optimal hedge ratio is then 1.0, and number of futures contracts to be shorted is then $N^* = \frac{V_A}{V_F}$. If portfolio do not mirror the index, then capital asset pricing model (CAPM) should be used to determine beta (β), and the number of futures contracts to be shorted is then $N^* = \beta\frac{V_A}{V_F}$, assuming maturity of futures contract is close to the maturity of the hedge.

If instead, the hedger wishes to change the beta of portfolio where $\beta > \beta^*$, a short position $(\beta - \beta^*)\frac{V_A}{V_F}$ is required. If $\beta < \beta^*$, then a long position $(\beta^* - \beta)\frac{V_A}{V_F}$ is required.

Stock index hedging is typically used when the portfolio manager is uncertain about performance of market, but is confident that the stocks in the portfolio will outperform the market. The hedger may also be planning to hold a portfolio for a long period of time and requires short-term protection in an uncertain market situation.

If expiration date of hedge is later than delivery dates of all futures contracts that may be used, then the hedger may *stack and roll* by closing out one futures contract and taking the same position in a futures contract with a later delivery date.

6.2.3 Interest Rate Futures

The *day count* defines the way in which interest accrues, and is expressed as X/Y, where X is the way in which number of days between two dates is calculated, Y is total number of days in reference period.

$$\text{interest earned between two dates} = \frac{X}{Y} \times \text{interest earned in reference period}$$

The three common day count conventions used in United States are:

i. Actual/Actual (in period): for Treasury Bond
ii. 30/360: for corporate and municipal bonds, assumes 30 days a month and 360 days a year
iii. Actual/360: for money market instruments

Prices of money market instruments are quoted using discount rate, which is interest earned as a percentage of final face value. Let P be quoted price, Y be cash price, n remaining life of Treasury bill in calendar days:

$$P = \frac{360}{n}(100 - Y)$$

Prices of treasury bonds are quoted in dollars and thirty-seconds of a dollar, i.e., 120-05 or $120\frac{5}{32}$, which is a bond value of $\$120,156.25$. The quoted price is *clean price*, and the cash paid by purchaser is *dirty price*:

$$\text{cash price} = \text{quoted price} + \text{accrued interest since last coupon date}$$

7 Currency Trading

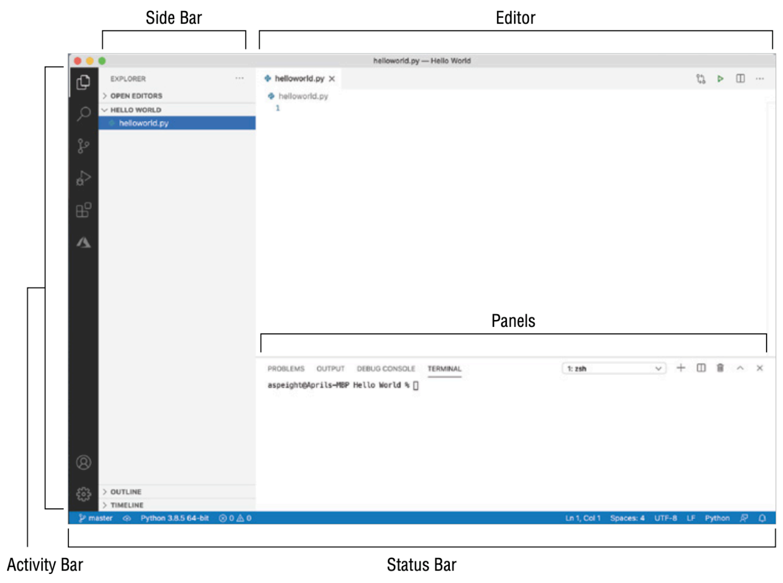
8 Commodities Trading

9 Appendix

For everything that cannot be classified under algorithmic trading, but is absolutely necessary for executing the concepts.

9.1 Visual Studio Code

Based on the book by April Speight (2021).



```
aspeight@Aprils-MBP Hello World % ▯
```

Figure 17: The visual studio code interface.

Activity bar on the far-left side allows switching between views, with quick access to tasks such as:

i. Explorer: file and folder management
ii. Search: global search and replace, can use plain text or regex
iii. Source Control: Git source control
iv. Run: for debugging, such as variables, call stacks, breakpoints
v. Extensions: browsing, installation, management of extensions

Activity bar may also provide custom views from extensions installed from Extension Marketplace. Views may be hidden, dragged and dropped around.

Side bar displays active view. Editor where files may be edited, may be resized, and top editor region changes depending on the type of file active in the editor; the top bar may also have a Source Control view if project is connected to git; files may be pinned and grouped by tabs.

Figure 18: Source control button

Panels are for debugging information, errors and warnings; and also for opening an integrated terminal on the root of the project. A REPL terminal (Python standard shell) may also be opened.

The status bar contains information about the opened project and files being edited. This includes: source control management with Git, total number of problems for the opened programs, line/column, indentation settings for space or tabs, encoding setting, end-of-line sequence setting, language mode, VS Code feedback mechanism, and notifications.

The command palette at the very top of the UI can be used to run commands to execute editor tasks in addition to extension commands.

9.1.1 Setting Up Python

To set default interpreter path in VS Code, in settings editor, search for *python.pythonPath*. In Python: Default Interpreter Path setting, enter the path to the interpreter.

To enable Quick Fix which help fix issues identified by warnings or errors (with lightbulb popping up), in settings editor, search for *python.jediEnabled*, then set it to false.

IntelliSense is a variety of tools to assist wth programming, such as code completion, object definition, location of object or variable declarations. These are triggered by either pressing Ctrl+Spacebar, or by tying a trigger character (i.e., a dot character in Python).

If linter detects any errors, these will be present in the Panels' Problem tab.

Refactoring is used to maintain functionality while improving the internal structure or architecture of a program. This should be a routine task that occurs before any updates or new features are added to a program. VS Code can help with refactoring via the following commands in Command Palette: Extract Variable, Extract Method, Sort Imports. Refactoring requires the *Rope* library.
Extract variable command allows extracting all similar occurrences of the same constant value of expression in multiple places, and replaces it with a variable. May be accessed via Python Refactor: Extract Variable.
Extract method command extracts all similar occurrences of selected expression or block, creates a new method, and replaces the expression with a method call. May be accessed via Python Refactor: Extract Method.
Sort imports method uses *isort* packages to consolidate specific imports from the same module into a single import statement, and organises 'import' statements in alphabetical order.

If a code pattern is repeated within a file or across multiple files, code snippets may be used, by searching in Command Palette for Snippets: Configure User Snippets.

9.1.2 Debugging

After starting debug session, the Run view opens. As the debugger runs, the current state of variables is reflected in Variables panel, which organises variables into local and global scopes. In the editor, the Debug toolbar will appear with the following functionalities:

i. Continue (A): Runs all the code after the breakpoint up to the next breakpoint or end of program.
ii. Step Over (B): Step line by line at the current scope. If current line is a function call, debugger runs the function entirely and then pauses at the next line after function call.
iii. Step Into (C): Steps over each line within the function scope and any additional function calls.
iv. Step Out (D): To exit from within a function to the scope that called it.
v. Restart (E); Stops execution and restarts the debugging session.
vi. Stop (F): Stop all execution without finishing the program.

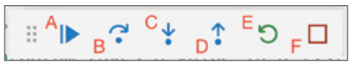
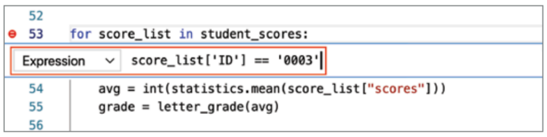


Figure 19: Debug toolbar on Editor

The side bar also has a call stack section, which shows the whole chain of function calls leading up to the current point of execution. Useful if calls go through other files in the project.

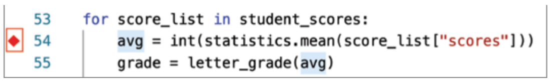
Conditional breakpoint allows for stopping of the code during specific conditions to evaluate and debug.



```
for score_list in student_scores:
Expression   score_list['ID'] == '0003'
    avg = int(statistics.mean(score_list["scores"]))
    grade = letter_grade(avg)
```

Figure 20: Adding a condition in a breakpoint

A logpoint outputs message to Debug Console without breaking the debugger. Expressions can be evaluated within curly braces.



```
for score_list in student_scores:
    avg = int(statistics.mean(score_list["scores"]))
    grade = letter_grade(avg)
```

Figure 21: Adding a logpoint

If program has a lot of variables which is hard to keep track of in variables pane, the variables may be added to the Watch panel which keep tracks of the variables.

The Debug Console allows access and modification of all the program's variables, call functions, evaluate expressions, and whatever code in the program's current state, rather than modifying the code and restarting. Different scenarios may be tested in the Debug Console and copy the fix into the program while the debugger is paused. The Debug Console also shows suggestions as code is typed.

To enable a test framework, *pytest* or *unittest* must be installed. Run the command Python: Configure Test to select the framework, directory that contains the test, and pattern to identify test files.

i. unittest: Looks for any Python file with 'test' in the name in top-level project folder. All test files must be importable modules or packages. To specify discovery pattern, change the pattern in *python.testing.unitTestArgs*.
ii. pytest: Looks for any Python file beginning with 'test_' or ends with '_test' located anywhere within current folder and all subfolders.

References

Hamilton, J. D. (1994). *Time Series Analysis.* Princeton University Press.

Hull, J. C. (2021). *Options, Futures, and Other Derivatives.* Pearson.

Narang, R. K. (2013). *Inside the Black Box: The Simple Truth About Quantitative Trading.* Wiley Finance Series. Wiley.

Prado, M. L. D. (2018). *Advances in Financial Machine Learning.* Wiley Finance Series. Wiley.

Speight, A. (2021). *Visual Studio Code for Python.* Wiley.

Thierry, A. and G. Helyette (2000, October). Order flow, transaction clock, and normality of returns. *The Journal of Finance 55*(5), 2259–2284.

Velu, R. (2020). *Algorithmic Trading and Quantitative Strategies.* CRC Press.