Análise Estatística de Dados Financeiros

Conceição Amado, Cláudia Nunes
CEMAT, Instituto Superior Técnico, Universidade de Lisboa

Alberto Sardinha
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa

Edições SPE

FICHA TÉCNICA:

Título: Análise Estatística de Dados Financeiros

Autores: Conceição Amado, Cláudia Nunes e Alberto Sardinha

Editora: Sociedade Portuguesa de Estatística

Conceção Gráfica da Capa: ESTG P.PORTO. BJS

Impressão: Instituto Nacional de Estatística

Tiragem: 250 exemplares

ISBN: 978-972-8890-43-8

Depósito Legal: 462496/19

Prefácio

O número de livros dedicados às finanças e estatística tem crescido a um rápido ritmo, em particular nos últimos anos. Por exemplo, se se procurar num dos usuais *sites* internacionais de venda online de livros, usando como palavras chave *statistical* e *finance*, resulta desta pesquisa mais de 2000 títulos em inglês! E quase todos deles editados neste século.

Então para quê mais um livro a juntar a esta lista? A maior parte destes livros são escritos para leitores que conhecem a parte financeira mas que precisam de aprender (e de preferência rapidamente!) os conceitos e técnicas estatísticas. Mas o que acontece quando se está do outro lado, isto é, o que acontece quando os leitores conhecem a parte estatística mas têm pouco conhecimento sobre terminologia e modelos financeiros?

Este livro não preenche esta lacuna, mas é uma pequena contribuição para que a comunidade estatística se sinta um pouco mais familiarizada com noções como taxas de juro, opções americanas, modelo de Black-Scholes, volatilidade implícita e outros termos do enorme e complexo jargão financeiro.

Na sociedade atual é raro quem consegue viver sem se deparar com decisões a nível financeiro, tendo de lidar com termos como recessão económica, ciclo de crescimento, incumprimento de crédito, taxas de juro, *spreads*, opções ...

Esta convivência mais ou menos forçada estende-se à nossa atividade profissional, quer em termos letivos quer em termos académicos. Um número significativo dos nossos alunos vai trabalhar em empresas do setor financeiro. Dados da Pordata indicam que em 2017, em Portugal, havia 17 743 empresas com atividades financeiras e de seguros, ao passo que em 1990 havia somente 1024 empresas deste setor. De acordo com a mesma fonte, em 2017 havia 93 261 trabalhadores nestas empresas, e que a soma do salário e outros encargos sociais médios anuais por trabalhador era de 41 642 EUR. Este valor pode

ser comparado com o correspondente ao de um engenheiro (33 629 EUR, em 2019).

O desafio lançado pela Comissão Organizadora do XXIV Congresso da Sociedade Portuguesa de Estatística era escrever um livro de apoio ao minicurso, versando sobre a análise estatística de dados financeiros. O desafio foi prontamente aceite, mas rapidamente se levantou a questão sobre a abordagem que iríamos fazer ao tema.

Dado que cada um de nós tem interesses distintos a nível da investigação e da lecionação, decidimos assumir estas diferenças. Assim cada um de nós deu o seu contributo numa área com maior afinidade. E por escolha, decidimos assumir o nosso estilo próprio e distinto de escrita e de organização de conteúdos.

Numa parte inicial os leitores encontram os conceitos que servem de pilar à matemática financeira. Para além da introdução de terminologia de índole financeira, definem-se também os produtos financeiros mais populares, e apresentam-se algumas das suas propriedades. São mencionados dois modelos para derivação de preço de opções americanas (o modelo binomial, em tempo discreto e o modelo de Black-Scholes, em tempo contínuo). A apresentação é informal, esquivando-se, sempre que possível, a mencionar resultados do cálculo estocástico. Ainda assim, apresentamos algumas definições chave, como o conceito de valor esperado condicionado e martingala, essenciais para a compreensão de alguns resultados.

Um dos sub-produtos do modelo de Black-Scholes é a volatilidade implícita, que pode ser usada para prever comportamentos futuros dos preços das ações, pelo que é usada por investidores para projetar os investimentos. A volatilidade implícita não é igual à volatilidade histórica, também conhecida como volatilidade ou volatilidade estatística. Dada a sua relevância nos mercados financeiros, dedicamos um capítulo à sua definição e propriedades, falando em particular na sua modelação por séries temporais.

Na última parte deste trabalho apresentamos uma contribuição mais orientada para modelos de inteligência artificial, em particular em modelos de redes neuronais.

Para ilustrar a abordagem computacional são apresentados pequenos programas/linhas de comandos do *software* utilizado: o R e Python, que os leitores podem reproduzir e/ou usar em aplicações futuras.

No final de tudo, foi um grande desafio, mas que esperamos que valha a

pena para os leitores. Este trabalho está longe de cobrir tudo (ou meramente uma parte significativa) do que existe sobre análise estatística de dados financeiros. De fora ficaram *clusters* em séries temporais, métodos numéricos para definição de superfícies de volatilidade, ou modelação de séries de preços de ativos, por exemplo. Da parte que nos toca, ficou a vontade de fazer mais. Talvez o futuro nos traga mais oportunidades destas!

Lisboa, 14 outubro de 2019

Conceição Amado

Cláudia Nunes

Alberto Sardinha

Agradecimentos

Agradecemos à Sociedade Portuguesa de Estatística a oportunidade que nos deu de publicar este livro. Agradecemos ainda o apoio institucional do Instituto Superior Técnico através dos centros de investigação CEMAT e INESC-ID, à FCT através dos projetos UID/MULTI/04621/2019, UID/CEC/50021/2019, PTDC/EGE-ECO/30535/2017, DSAIPA/DS/0089/2018 e à *Air Force Office of Scientific Research* através do projeto FA9550-19-1-0020.

Índice

Capítulo 1

Motivação e Introdução

1.1 Motivação

Nos dias de hoje todos nós somos confrontados com notícias sobre o mundo financeiro. O impacto que as subidas e descidas de cotações em bolsa, a dívida pública e os juros têm na nossa via quotidiana é inquestionável.

Depois de um período de recessão financeira com início nos anos 2008, com grande impacto na atividade económica, estamos neste momento a atravessar um período de acalmia financeira. Mas já surge no horizonte possibilidades de novas crises. Aliás, o passado (em particular o século 20) mostrou de que depois de crises/recessões financeiras seguem-se períodos de expansão económica. Mas como podemos nós *navegar* nestas águas turvas?

Uma das palavras chave no jargão financeiro é **prever**! Compram-se ações prevendo que daqui a uns tempos a sua cotação na bolsa seja superior; vendem-se ações porque se receia uma descida com perdas consideráveis; as empresas podem vir a distribuir dividendos daqui a uns tempos; a taxa de juro pode ser alterada... Estes são alguns exemplos sobre a necessidade de previsão de comportamento de produtos financeiros.

Existe uma panóplia grande de produtos financeiros, cada qual com as suas especificidades. Produtos financeiros são investimentos criados para que os compradores ou vendedores destes instrumentos tenham potenciais ganhos a curto/médio/longo prazo. Potenciais porque o seu retorno raramente é assegurado, havendo sempre risco de ao invés de ter ganhos, os investidores

venham a ter perdas. Uma das referências fundamentais nesta área é o livro de Hull Hull (2003), recomendado para o leitor que pretenda adquirir um maior conhecimento sobre este universo de produtos. Outras referências sobre este universo são, por exemplo, Stefanini (2010), Overhaus *et al.* (2011), Shiryaev (1999), entre outros.

Os desafios nesta área são assinaláveis, e cabe também à comunidade estatística dar a sua contribuição na análise dos dados. Para tal é necessário entender (minimamente...) como funciona o mercado financeiro, o tipo de produtos transacionados, a noção de risco. O risco está inerente aos mercados e produtos financeiros, pois há sempre a possibilidade de acionistas, investidores ou outras partes interessadas financeiras perderem dinheiro. Ou seja, os ganhos com investimentos financeiros podem ser vistos como variáveis aleatórias, mais ou menos complexas, mas às quais a comunidade estatística pode utilizar a sua panóplia de técnicas e modelos.

O interesse na previsão de dados financeiros é óbvio e indiscutível. Existem histórias sobre investidores - verdadeiras lendas! - e sobre os seus *truques* para prever o comportamento dos produtos financeiros, e assim aumentar as suas chances de um retorno do investimento.

Os investidores diferem amplamente nas estratégias e filosofias aplicadas nas suas negociações; alguns inventaram maneiras novas e inovadoras de analisar os seus investimentos, enquanto outros escolheram produtos quase inteiramente por instinto.

Os pequenos investidores geralmente pouco conhecem sobre o mercado e suas expectativas, investindo muitas vezes por simples *intuição* (ou até crença), como se de um jogo se tratasse. Na verdade podemos ver o processo de investimento como uma atividade semelhante a um jogo de sorte ou azar. E neste contexto a ajuda das probabilidades pode ser de grande importância.

A título de curiosidade, referimos a história de Marge e Jerry Selbee, proprietários de uma loja de conveniência em Evart, Michigan, que vendiam cigarros, bebidas e bilhetes de lotaria[1]. Dados recentes dizem que no ano de 2018 os americanos gastaram cerca de 80 biliões de dólares ($) em apostas em lotarias, o que corresponde a cerca de $ 250 dólares por americano. Muitas destas lotarias são promovidas pelos estados federais, o que se traduz num aporte financeiro por vezes superior ao correspondente ao pagamento de

[1] https://www.cbsnews.com/news/jerry-and-marge-selbee-how-a-retired-couple-won-millions-using-a-lottery-loophole-60-minutes-2019-06-09/

impostos federais. Um dos estados que promove lotarias é o estado de Michigan. Em 2003 o estado lançou uma nova lotaria, designada por *Winfall*. Ao contrário das lotarias mais usuais (como o Euromilhões, popular em Portugal), em que o valor do prémio vai acumulando até que algum apostador acerte em todos os números (o designado *Jackpot*), nesta lotaria se o Jackpot atingisse 5 milhões $ e ninguém acertasse na chave (com 6 dígitos), o prémio seria automaticamente distribuído pelos jogadores que tivessem acertado 5, 4 ou 3 dígitos, por ordem decrescente de atribuição.

Esta regra não levantou suspeitas a ninguém. Mas depois de vender lotarias a milhares de clientes, Jerry descobriu como *controlar* as probabilidades de ganhar esta lotaria, aumentando a probabilidade de retorno positivo.

Jerry percebeu que, jogando milhares de dólares em determinadas semanas, quando o prémio se acumula de certa forma, conseguia garantir um lucro de cinco ou seis dígitos com probabilidade quase um!

Então foi isso que os Selbees começaram a fazer, ao ponto de que jogar na lotaria se tornou um trabalho de tempo integral. Ao longo de nove anos, funcionou, a tal ponto que o total estimado de ganhos nesta atividade foi de cerca de 27 milhões $.

Note-se que não havia nada de ilegal nesta atividade, mas havia uma percepção de que eles estavam a *esgrimir* as probabilidades a seu favor - longe do pequeno e ocasional jogador, que poderia pegar alguns bilhetes de lotaria a caminho de casa ou do trabalho.

O que ajudou os Selbees a entender esta possibilidade de lucro foi a quantidade de informação que eles dispunham, pelo facto de eles próprios venderem muitos bilhetes de lotaria. Ou seja, os dados e sua análise são fundamentais para maximizar o retorno (esperado) dos investimentos!

1.2 Dados financeiros

Um conjunto de dados financeiros é geralmente constituído por um registo de negociações de ações ou títulos[2], negociados em alguma bolsa de valores, ou por taxas de juros cotadas ou taxas de câmbio. Adicionalmente podem

[2]Neste capítulo usaremos alguns termos que serão mais tarde devidamente formalizados, tais como a noção de obrigações/títulos, taxas de juro, portfólios/carteiras de títulos, opções, etc. Assume-se que o leitor tem um conhecimento básico dos produtos financeiros, sem para já ter preocupações de formulismo e definição.

também conter informação sobre preço de *commodities* (i.e, de matéria prima ou de bens, como o ouro, o alumínio, o petróleo, etc).

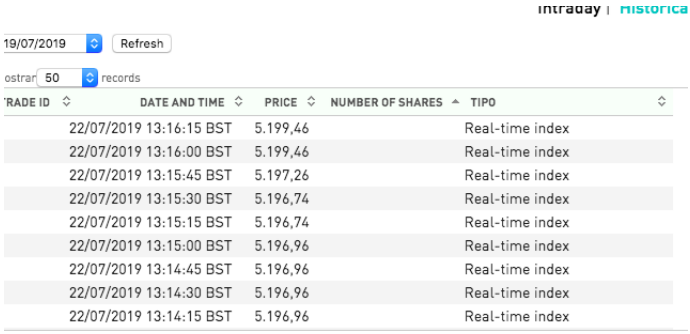
Para a análise de dados financeiros é relevante ter a indicação do intervalo temporal a que se referem, isto é, é importante saber se são relativos a transações horárias, diárias, semanais, etc. Usualmente estes dados contêm informação relativa à quantidade negociada, ao preço de oferta, ao preço de venda, a data a que se refere, a quantidade de ativos comercializados, entre outros tipos de informação.

Um dos indicadores relevantes sobre o comportamento financeiro de uma economia são os índices financeiros. Um índice financeiro é um indicator que varia ao longo do tempo e que reporta a evolução da cotação de um grupo específico de ações. Em Portugal, o índice PSI-20 agrega as 20 maiores empresas cotadas na Euronext Lisboa. É o principal índice de referência do mercado de capitais português.

Na Figura 1.1 mostramos a flutuação do índice entre os dias 15 e 22 de julho de 2019. Ao longo deste período o índice variou, mas regista uma tendência negativa, sinal de que a bolsa de Lisboa registou uma queda no período em questão.

Figura 1.1: Fonte: https://www.bolsadelisboa.com.pt/products/indices/PTING0200002-XLIS

Alguns investidores dedicam-se a comprar e vender ações e títulos no mesmo dia. É o que o mercado chama de operação *intraday*. Este tipo de operações geralmente reportam lucros quando simultaneamente se regista


Intraday | Histórica

19/07/2019 Refresh

ostrar 50 records

'RADE ID	DATE AND TIME	PRICE	NUMBER OF SHARES	TIPO
	22/07/2019 13:16:15 BST	5.199,46		Real-time index
	22/07/2019 13:16:00 BST	5.199,46		Real-time index
	22/07/2019 13:15:45 BST	5.197,26		Real-time index
	22/07/2019 13:15:30 BST	5.196,74		Real-time index
	22/07/2019 13:15:15 BST	5.196,74		Real-time index
	22/07/2019 13:15:00 BST	5.196,96		Real-time index
	22/07/2019 13:14:45 BST	5.196,96		Real-time index
	22/07/2019 13:14:30 BST	5.196,96		Real-time index
	22/07/2019 13:14:15 BST	5.196,96		Real-time index

Figura 1.2: Fonte: https://www.bolsadelisboa.com.pt/products/indices/PTING0200002-XLIS/quotes

uma grande volatilidade do mercado e alto volume de operações diárias. Consequentemente o preço das ações de uma empresa pode variar radicalmente em apenas um dia. Por isso, a operação *intraday* é muito utilizado pelos investidores que procuram especular rapidamente com o mercado no curtíssimo prazo. Na Figura 1.2 mostram-se os dados relativos ao PSI 20 no dia 22 de julho de 2019, referentes a períodos de 15 segundos.

Quando o intervalo de variação é muito curto, estamos em presença de HTF (*High Trade Frequency*). Como o próprio nome sugere, o HFT é uma estratégia de negociação de alta frequência (i.e., muitas transações por unidade de tempo). São utilizados algoritmos complexos e um altíssimo volume de negociações para explorar as micro variações de preços durante o dia. Por acontecer num ritmo e volume muito mais acelerado do que o intraday, o HTF é executado por investidores-robôs, que trabalham de forma automatizada de acordo com o que for programado. Normalmente, só tem acesso a esse tipo de operação as instituições financeiras ou grandes investidores institucionais. Este tipo de transações representa uma parte considerável das transações financeiras a nível mundial. O nível das negociações de alta frequência costumam girar entre 70% a 80% dos mercados financeiros.[3]

Os investidores que operam em *intraday* geralmente vendem as suas posições no final do dia, não levando em atenção o tipo de empresa a que ações se reportam. Pelo contrário, existem investidores que procuram investimentos mais a longo prazo, e para o qual é importante também avaliar o tipo de empresa a

[3]Fonte: https://www.bloomberg.com/news/articles/2011-01-24/high-frequency-trading-is-77-of-u-k-market-tabb-group-says

NOME	ISIN	SYMBOL	MERCADO	ÚLTIMA	%	DATE/TIME
ALTRI SGPS	PTALT0AE0002	ALTR	Euronext Lisbon	5,985	1,01%	22 Jul 2019 14:40
B.COM.PORTUGUES	PTBCP0AM0015	BCP	Euronext Lisbon	0,2707	-1,06%	22 Jul 2019 14:41
BENFICA	PTSLB0AM0010	SLBEN	Euronext Lisbon	2,87	-	22 Jul 2019 13:30
COFINA,SGPS	PTCFN0AE0003	CFN	Euronext Lisbon	0,468	0,65%	22 Jul 2019 14:41
COMPTA	PTCOM0AE0007	COMAE	Euronext Lisbon	0,119	8,18%	19 Jul 2019 11:30
CORTICEIRA AMORIM	PTCOR0AE0006	COR	Euronext Lisbon	9,80	0,10%	22 Jul 2019 14:34
CTT CORREIOS PORT	PTCTT0AM0001	CTT	Euronext Lisbon	2,018	0,40%	22 Jul 2019 14:27
EDP	PTEDP0AM0009	EDP	Euronext Lisbon	3,368	-0,06%	22 Jul 2019 14:43

Figura 1.3: Fonte: https://www.bolsadelisboa.com.pt/cotacoes/accoes-lisboa

que a ação se reporta. Neste caso é importante avaliar o comportamento das ações ao longo do tempo, recorrendo a dados históricos, por exemplo.

Dada a sua natureza, alguns dados financeiros são públicos, tal como o caso da cotação na bolsa de valores de Lisboa, que na data atual reporta o valor das ações relativas a 59 empresas, com informação sobre a última cotação e sua variação em relação à última transação, assim como a data da transação. Ver Figura 1.3.

Por outro lado, há alguns dados que não são acessíveis ao público em geral. Nesse caso ou são acessíveis apenas via plataformas de dados de acesso pago (como é o caso da Bloomberg ou GFD - Global Financial Data) ou são de acesso reservado (como é o caso de dados financeiros de contribuintes).

Na análise de dados financeiros intervêm duas perspectivas:

- Puramente técnica, que se baseia apenas no histórico das observações, na qual a análise estatística desempenha um papel fundamental
- Baseada no conhecimento sobre indicadores económicos e políticos (ou a análise de outros fatores específicos que podem influenciar o preço dos produtos financeiros). Esta perspectiva pode - e deve - ser combinada com a análise estatística.

Tipicamente os dados são relativos ao retorno dos investimentos, dos quais se salientam:

- *Retornos num período*: seja S_t o preço de uma ação no instante t. Caso não haja lugar à distribuição de dividendos no período $[t-1, t]$, o retorno

num período é dado por:

$$R_t = \frac{S_t - S_{t-1}}{S_{t-1}}$$

e que contabiliza a taxa do retorno por deter a ação durante este período.

- *Retornos em multi-períodos*: ao invés de se considerar apenas um período (da forma $[t-1, t]$, como no caso anterior), podemos definir o retorno em k períodos da seguinte forma:

$$R_t(k) = \frac{S_t - S_{t-k}}{S_{t-k}}$$

de onde resulta que:

$$1 + R_t(k) = \prod_{j=0}^{k-1} (1 + R_{t-j}) .$$

- *Log-returns*: resulta do cálculo relativo à evolução do logaritmo do preço, i.e.,

$$\ln \frac{S_t}{S_{t-1}}.$$

Uma das propriedades que faz com que os *log-returns* sejam populares é que à medida que o intervalo temporal, Δt, de um período se aproxima de 0, o *log-return* aproxima-se do retorno num período:

$$\ln \frac{S_t}{S_{t-1}} \to R_t$$

Outra propriedade é a aditividade dos retornos em multi-períodos: um k-período num *log-return* é a soma de k *log-returns* num único período:

$$\ln \frac{S_t}{S_{t-K}} = \sum_{j=0}^{k-1} \ln (1 + R_{t-j})$$

As fórmulas anteriores podem ser alteradas de forma a contemplar a distribuição de dividendos:

$$R_t = \frac{S_t + D_t - S_{t-1}}{S_{t-1}} \quad \text{(no caso de retornos)}$$

$$\ln \left[\prod_{j=0}^{k-1} \frac{S_{t-j} + D_{t-j}}{S_{t-j-1}} \right] = \sum_{j=0}^{k-1} \ln \left(\frac{S_{t-j} + D_{t-j}}{S_{t-k-1}} \right) \quad \text{no caso de } \textit{log-returns}$$

onde D_t designa o dividendo recebido no instante t.

No caso de um portfólio, constituído por p diferentes produtos, cada um com peso w_i, as fórmulas anteriores aplicam-se, com o fator de ponderação w_i para cada produto.

1.3 Risco

Os riscos estão presentes em qualquer tipo de aplicação financeira, inclusive nos fundos de investimento do qual tanto se fala atualmente. O comportamento futuro dos investimentos é uma incógnita, em face da aleatoriedade do mercados.

Paralelamente, diferentes indívíduos podem atribuir diferente utilidade ao dinheiro e ter um grau de aversão ao risco distinto (existem investidores neutros em relação ao risco, investidores avessos ao risco e aqueles investidores que preferem o risco a qualquer alternativa de certeza). A teoria da utilidade conclui que, de maneira geral, o investidor pode ser considerado avesso ao risco. Ou seja, entre duas condições, uma com certeza e outra com algum grau de risco, a preferência cairá na alternativa que apresenta o menor componente de risco possível. Ainda assim, entre os investidores avessos ao risco, vai haver maior ou menor grau de aversão, de acordo, mais uma vez, com as características individuais de cada um.

Ao longo desta secção assumimos que a função de utilidade é igual para todos os investidores, e que o investidor é avesso ao risco.

Como veremos na próxima secção, existem produtos financeiros mais complexos que as ações (usualmente designadas por *opções* ou *derivados financeiros*), cujo valor depende intrinsicamente das ações (os designados *ativos subjacentes*). As opções podem ser de diversos tipos, e de menor ou maior complexidade em termos de retorno do investimento.

A avaliação de risco e o cálculo do preço destes produtos são baseados em modelos que descrevem os preços dos ativos subjacentes. A qualidade das medidas de risco e a validade dos preços é fortemente dependente da qualidade do ajustamento do modelo estatístico considerado. Deficiências do modelo estatístico podem ser desastrosas, e não apenas para os especuladores financeiros.

Até a um passado muito recente, colocar as poupanças em contas bancárias

era considerado um bom investimento. Seguro (isto é, sem risco de perder o dinheiro), e com taxas de retorno positivas (ou seja, no final do contrato o dinheiro existente na conta era superior ao dinheiro investido). Qualidades que os pequenos investidores apreciam.

Mas nos dias de hoje a rentabilidade dos depósitos a prazo tem sido bastante baixa. Os investidores têm procurado outras alternativas de investimento. Entre as alternativas estão o imobiliário, obrigações de empresas e também fundos de investimento.

Estes produtos têm risco associado, ou seja, o retorno do investimento não é garantido. Mas em contrapartida, o retorno do investimento é, em média, superior ao dos depósitos bancários. Dados recentes [4] indicam que em Portugal há cerca de 213 fundos de investimento geridos por entidades portuguesas, com um retorno médio de 5%.

Os investidores estão mais dispostos a aceitar níveis de risco crescente a favor de hipóteses de retorno mais elevado. Um dos exemplos são os chamados *produtos multi-ativos*. Estes fundos tentam diversificar, investindo simultaneamente em ações, obrigações e outras aplicações.

As subscrições líquidas vão já em 270 milhões de euros em cinco meses[5]. Numa escala de 1 (menos risco) a 7 (maior risco), os fundos multi-ativos têm, na maior parte das vezes, risco de 3 ou 4. Renderam entre 4,86% e 10,68% no ano de 2018. Numa escala crescente de risco, encontramos os fundos de ações internacionais. Grande parte destes produtos têm risco 5; mas valorizam, em média, 11%.

Há risco porque existe uma probabilidade de perda do retorno do investimento. A gestão do risco é um processo crucial na tomada de decisões de investimento. O processo consiste em identificar a quantidade de risco envolvida num investimento e aceitar esse risco ou mitigá-lo. O risco pode ser classificado numa das três seguintes categorias:

- Risco comercial: este tipo de risco é assumido pelas próprias empresas, a fim de maximizar o valor e os lucros dos acionistas. Por exemplo, as empresas assumem riscos de alto custo em marketing para lançar novos

[4] Fonte: https://www.dn.pt/edicao-do-dia/24-jun-2019/interior/portugueses-apostam-em-fundos-de-investimento-mais-de-metade-rende-acima-de-5-11037782.html

[5] https://www.dn.pt/edicao-do-dia/24-jun-2019/interior/portugueses-apostam-em-fundos-de-investimento-mais-de-metade-rende-acima-de-5-11037782.html

produtos a fim de obter vendas maiores.

- Risco não comercial: este tipo de risco não está sob o controlo das empresas. Pode resultar de desequilíbrios políticos e económicos, por exemplo.

- Risco financeiro: é o risco que envolve perdas financeiras para as empresas. O risco financeiro geralmente surge devido à instabilidade e perdas no mercado financeiro causadas por movimentos nos preços das ações, moedas, taxas de juros e muito mais.

Para ajudar na gestão do risco, é usual *quantificar* o mesmo, através de alguma *medida de risco*. Uma medida de risco é de facto uma função do investimento. Mais formalmente, seja X uma variável aleatória definida no espaço de probabilidades $(\Omega, \mathcal{F}, P)$ (ou seja, X é o valor aleatório do retorno do investimento e para cada $\omega \in \Omega$, $X(\omega)$ designa o retorno quando o estado da economia é ω). Então define-se uma medida de risco ρ como sendo uma função:

$$\rho : \mathbb{R} \to \mathbb{R}$$

Como veremos há diversas medidas de risco. Mas seja qual for a medida de risco usada, deve-se ter em atenção que deve ser uma medida coerente, no sentido que caso não seja coerente, esta medida não nos dará informações adequadas sobre o risco de carteira de investimentos. Por esse motivo, existem quatro propriedades exigidas por qualquer medida de risco, e que descrevem as propriedades de uma medida coerente.

- *Monotonia*:
$$X_1 \leq X_2 \Rightarrow \rho(X_2) \leq \rho(X_1)$$
(i.e., se X_2 vale mais que X_1, seja qual for o eventual estado da economia $\omega \in \Omega$, então o risco de X_1 é maior que o risco de X_2)

- *Invariância de conversão*: se se adicionar um valor em dinheiro ao portfólio, então o risco será reduzido por esse valor.

- *Homogeneidade*: se um portfólio X for aumentado em k vezes (mantendo o peso de cada elemento do portfólio) então o risco será multiplicado pelo mesmo fator:
$$\rho(kX) = k\rho(X)$$
(i.e., o risco depende da dimensão da posição)

- *Subaditividade*: a medida de risco de duas carteiras incorporadas deve ser inferior à soma de suas medidas de risco individualmente.

$$\rho(X_1 + X_2) \leq \rho(X_1) + \rho(X_2)$$

Algumas medidas comuns de risco incluem:

- **Desvio Padrão**: O desvio padrão é usado na tomada de uma decisão de investimento para medir a quantidade de volatilidade histórica associada a um investimento em relação à sua taxa de retorno anual. Por exemplo, uma ação que possui um alto desvio padrão apresenta maior volatilidade e, portanto, um nível mais alto de risco. O problema do uso do desvio padrão é que não leva em linha de conta a direção da alteração. Se o desvio padrão for elevado devido a subidas no preço de uma ação, poderá ser bastante benéfico para o investidor.

- **Beta**: mede a quantidade de risco sistemático que um título individual ou setor industrial possui em relação a todo o mercado de ações. O mercado tem um valor beta de 1. Quanto maior for o valor de beta, mais volátil é o título. Estatisticamente, beta é a estimativa do declive do modelo de regressão linear simples dos valores históricos do título em função dos valores históricos do mercado onde o título é comercializado.

- **Valor em risco (VaR)**: O VaR mede a perda potencial máxima com um grau de confiança por um período especificado. O VaR pode ser utilizado quer avaliar o risco de um portfólio (constituído por diversos títulos) ou para calcular apenas o risco de um título.

  Matematicamente, podemos definir o VaR de nível $\alpha \in (0, 1)$ do produto X, que se denota por $\mathrm{VaR}[X, \alpha]$ da seguinte forma:

$$\mathrm{VaR}[X, \alpha] = \inf\{x \in \mathbb{R} : P(X \leq x) \geq \alpha\}$$

  pelo que de facto o VaR é um quantil de probabilidade.

  Na literatura encontramos referência fundamentalmente a três tipos de métodos para calcular o VaR, que aqui se descrevem sucintamente, e apenas para o caso uni-dimensional (i.e., um único título):

  – Dados históricos: usando os retornos observados até ao momento presente, é construído um histograma (ou, de forma equivalente, os dados são ordenados, começando pelo retorno mais baixo e terminando no retorno mais elevado). Por análise do histograma,

podemos obter uma estimativa para o VaR pretendido. Por exemplo, se $\chi_{0,05}$ for o quantil amostral de probabilidade 0,05, então diz-se que com 95% de confiança, se se investir 100 EUR, a perda esperada não é superior a $\chi_{0,05}$.

  - Variância-Covariância: com base nos dados históricos do retorno, são calculadas estimativas do valor esperado e da variância. Estes valores são então usados como parâmetros de uma distribuição normal, a qual é usada para calcular os quantis pretendidos.
  - Simulação de Monte-Carlo: usando valores históricos para estimação de parâmetros, é gerado um conjunto de valores de uma distribuição pré-especificada, a partir do qual calcula o quantil(is) desejados.

- Valor em Risco Condicional (CVaR): é a pior perda esperada devido ao incumprimento durante um certo período de tempo, com uma dada probabilidade. O período de tempo é conhecido como período de retenção e a probabilidade é conhecida como intervalo de confiança. O CVaR não é uma estimativa da pior perda possível, mas a maior perda provável.

A medida mais popular entre a comunidade financeira é o VaR. Esta medida satisfaz as três propriedades de coerência anteriormente enumeradas, mas não satisfaz necessariamente a subaditividade, excepto no caso do retorno ter distribuição normal. O exemplo que se segue ilustra precisamente esta situação.

Exemplo 1.1. *Para mostrar que VaR não é uma medida subaditiva, considere-se o seguinte exemplo Dowd (2003). Um investidor vende duas opções, independentes uma da outra (i.e, dependem de ativos subjacentes distintos e cujo comportamento é independente entre si). Cada opção será exercida com probabilidade 0,04; caso seja exercida, o investidor perde 100 EUR. Designando por X_i o retorno da opção da opção i, tem-se que $X_1, X_2 \in \{-100, 0\}$ (interpretando o valor -100 como uma perda no valor de 100 EUR), e:*

$$\begin{aligned} P(X_1 = -100) = P(X_2 = -100) &= 0,04 \\ P(X_1 = 0) = P(X_2 = 0) &= 0,94 \\ P(X_1 + X_2 = -200) &= 0,0016 \\ P(X_1 + X_2 = -100) &= 0,0768 \\ P(X_1 + X_2 = 0) &= 0,9216 \end{aligned}$$

Para um nível de confiança de 95%, para ambos as opções o VaR é zero. Porém, se combinarmos as duas opções num portfólio o resultado é distinto. Como a probabilidade de nenhuma opção ser exercida é $0,96^2 \leq 0,95$*, o VaR é positivo, pelo que a sub-aditividade não é verificada neste caso.*

O VaR está relacionado com o comportamento de valores da cauda da distribuição do retorno. Quando dizemos que o VaR de nível 99% de uma carteira é de 1 milhão de euros, dizemos que a probabilidade de perdas superiores a 1 milhão é de 1%. Mas o que acontece nos 1% restantes? Os valores extremos podem-se reportar a perdas enormes, incomportáveis para o investidor, e que não são de forma alguma tomadas em linha de conta quando se avalia o risco do investimento tendo em linha de conta apenas o VaR.

O CVaR foi proposto para combater esta deficiência do VaR, uma vez que representa a perda provável quando se está na cauda esquerda da distribuição das perdas. Na literatura anglo-saxónica encontramos também o termo *expected short-fall* para designar esta medida.

Se $\mathrm{ES}[X, \alpha]$ designar o CVaR do produto cujo retorno é X para um nível α, então pode ser calculado recorrendo à seguinte fórmula:

$$\mathrm{ES}[X, \alpha] = \frac{1}{1-\alpha} \int_{\alpha}^{1} \mathrm{VaR}[X, s] ds$$

O seguinte lema (Embrechts e Wang (2015)) mostra relações úteis entre o VaR e o CVaR.

Lema 1.1.

$$ES[X, \alpha] = VaR[X, \alpha] + \frac{1}{1-\alpha} \mathbb{E}\left[(X - VaR[X, \alpha])^{+} \right] \tag{1.1}$$

$$ES[X, \alpha] = \mathbb{E}\left[X I_{\{X > VaR[X,\alpha]\}} \right] + VaR[X, \alpha] \left(P(X \leq VaR[X, \alpha]) - p \right) \tag{1.2}$$

onde a^{+} *representa a parte positiva de* a *e* I_A *indica a função indicatriz do evento* A*.*

Demonstração. Para provar a igualdade 1.1, usamos a definição de quantil e

de CVaR:

$$\begin{aligned}
\mathrm{ES}[X,\alpha] &= \frac{1}{1-\alpha}\int_{\alpha}^{1} F_X^{-1}(q)dq \\
&= F_X^{-1}(\alpha) + \frac{1}{1-\alpha}\int_{\alpha}^{1}\left(F_X^{-1}(q) - F_X^{-1}(\alpha)\right)dq \\
&= \mathrm{VaR}[X,\alpha] + \frac{1}{1-\alpha}\mathbb{E}[\left(F_X^{-1}(U_X) - \mathrm{VaR}[X,\alpha]\right)^{+}] \\
&= \mathrm{VaR}[X,\alpha] + \frac{1}{1-\alpha}[\,(X - \mathrm{VaR}[X,\alpha])^{+}]
\end{aligned}$$

onde U_X designa uma v.a. uniforme em $(0,1)$, tal que $X = F_X^{-1}(U_X)$ quase certamente.

Para provar 1.2, recorremos à igualdade anterior:

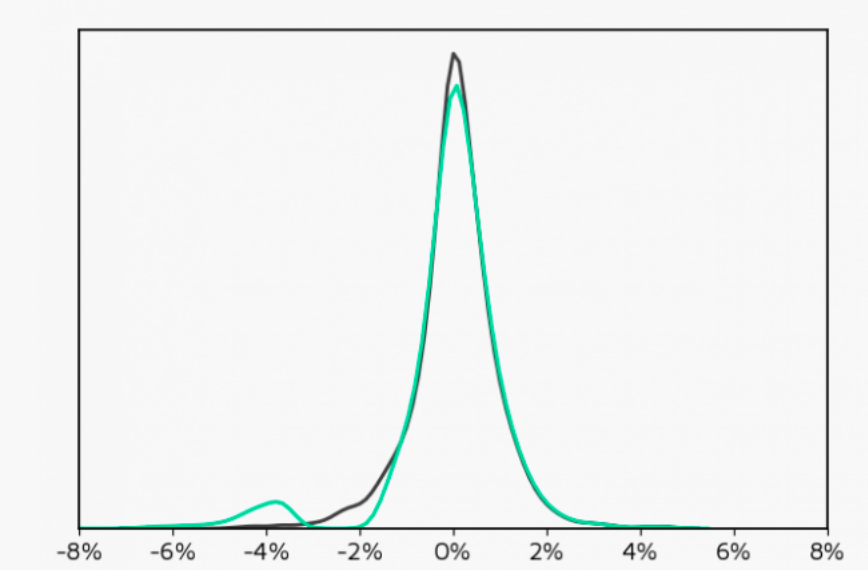
$$\begin{aligned}
(1-\alpha)\mathrm{ES}[X,\alpha] &= (1-\alpha)\mathrm{VaR}[X,\alpha] + \mathbb{E}[(X - \mathrm{VaR}[X,\alpha])\, I_{\{X>\mathrm{VaR}[X,\alpha]\}}] \\
&= \mathbb{E}[X I_{\{X>\mathrm{VaR}[X,\alpha]\}}] + \mathrm{VaR}[X,\alpha]P(X \leq \mathrm{VaR}[X,\alpha] - \alpha)
\end{aligned}$$

□

Nota: a expressão 1.1 traduz o que na gíria financeira se costuma referir como *ES captura a cauda do risco após o VaR.*

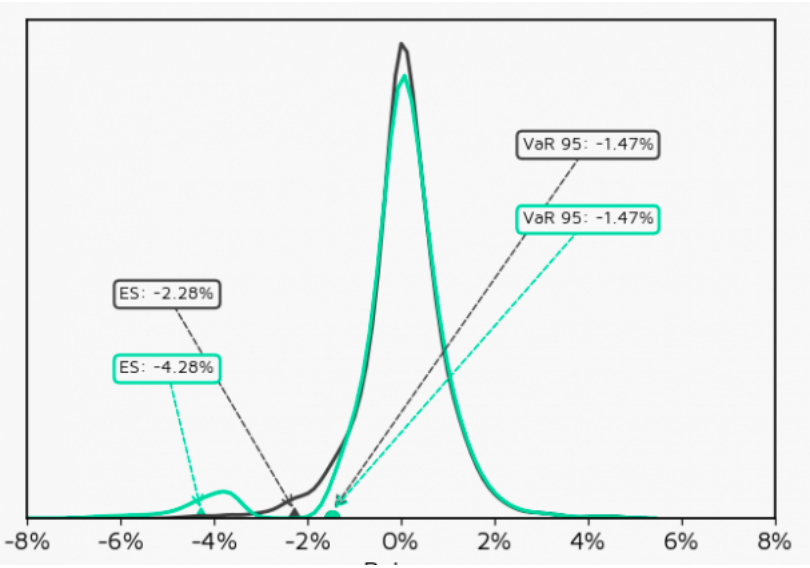
Pode acontecer que dois produtos financeiros tenham o mesmo VaR para algum(ns) nível(eis) mas diferente CVaRs, como o seguinte exemplo ilustra.

Exemplo 1.2. *Considerem-se dois produtos, X e Y, cujos retornos têm a seguinte distribuição:*



em que a linha com valor máximo designa o retorno do produto Y e a outra designa o retorno do produto X. Dada a forma das funções de densidade de probabilidade de X e Y, os produtos têm retornos idênticos excepto na cauda esquerda, que representa precisamente as perdas potenciais.

Se calcularmos os VaR's de nível 95%, por exemplo, concluímos que ambos os produtos têm o mesmo VaR.



Porém, se calcularmos o CVaR, concluímos que o produto X é mais arriscado que o Y pois leva a um CVaR mais elevado.

Ao contrário do VaR, o CVaR apresenta todas as propriedades de coerência pretendidas; em particular, é uma medida subaditiva. A demonstração de que esta propriedade é válida quando se considera o CVaR não é trivial e exige vários resultados auxiliares. Mas curiosamente há várias abordagens possíveis e muito distintas para a sua demonstração. Embrechts e Wang (2015) enumeram sete formas distintas de mostrar esta propriedade.

A título ilustrativo, de seguida apresenta-se a demonstração mais simples da subaditividade do CVaR, e que se baseia no seguinte resultado:

Lema 1.2. *Seja B_p o conjunto de todas as v.a. de Bernoulli de parâmetro $1-p$, e $A_X = I_{\{U_X \geq p\}} \in B_p$, onde U_X designa, como anteriormente, uma v.a. com distribuição uniforme em $(0,1)$. Então*

$$\mathbb{E}[XA_X] \geq \mathbb{E}[XB], \forall B \in B_p.$$

Demonstração. Por definição, para qualquer $B \in B_p$, tem-se

$$\mathbb{E}[X(A_X - B)] = \mathbb{E}[(X-m)(A_X - B)], \quad \forall m \in \mathbb{R}.$$

Tome-se então $m = F_X^{-1}(p)$. Se $F_X^{-1}(U_X) > m$, então $U_X > p, A_X = 1$ e $\mathbb{E}[(X-m)(A_X - B)] \geq 0$; se $F_X^{-1}(U_X) < m$, então $U_X < p, A_X = 0$

e $\mathbb{E}[(X-m)(A_X-B)] \geq 0$; finalmente, se $F_X^{-1}(U_X) = m$, então $\mathbb{E}[(X-m)(A_X-B)] = 0$, pelo que o resultado é válido. □

Finalmente apresenta-se o resultado sobre a subaditividade da medida CVaR:

Proposição 1. *CVaR é uma medida de risco subaditiva.*

Demonstração.

$$\begin{aligned}\mathrm{ES}[X,\alpha] &= \frac{1}{1-\alpha}\int_\alpha^1 F_X^{-1}(q)dq = \frac{1}{1-\alpha}\mathbb{E}[F_X^{-1}(U_X)I_{\{U_X\geq\alpha\}}] \\ &= \frac{1}{1-\alpha}\mathbb{E}[XA_X]\end{aligned}$$

Em face do lema anterior, segue-se que

$$\mathrm{ES}[X,\alpha] = \frac{1}{1-\alpha}\sup\{\mathbb{E}[XB] : B \in B_\alpha\}$$

i.e., CVaR é o supremo da aplicação $X \to \frac{1}{1-\alpha}\mathbb{E}[XB]$, para $B \in B_\alpha$, pelo que é sub-aditiva (uma vez que o lim sup é um operador sub-aditivo, desde que ambos os limites estejam bem definidos).

□

Capítulo 2

Produtos Financeiros

2.1 Produtos sem risco

O mercado financeiro está inundado de diversos tipos de produtos. Produtos como obrigações e ações são-nos mais familiares. Quando se fala em opções, os investidores não profissionais têm menor conhecimento acerca do assunto. Mas o setor financeiro tem vindo, nos últimos anos, a disponibilizar aos clientes produtos de investimento cada vez mais sofisticados.

A elevada complexidade técnica da maioria destas aplicações levou a que fossem classificados como *produtos financeiros complexos*. São produtos financeiros complexos alguns instrumentos financeiros como os (*Contracts for Differences (CFD)*, que nalguns países (como os E.U.A.) nem sequer são legais. Embora estes produtos possam gerar elevados retornos, os investidores podem incorrer também em perdas superiores ao capital aplicado, ao invés dos produtos mais conhecidos (como as ações, obrigações ou opções). Os CFDs não são objeto de análise no presente livro, pelo que se remete a sua definição para literatura apropriada, como por exemplo Kristiansen (2004).

Os produtos financeiros complexos são instrumentos financeiros cuja especial complexidade se encontra sobretudo associada à percepção dos riscos que o investimento neste tipo de produtos envolve. Essa dificuldade resulta do facto de, sob a aparência de um instrumento financeiro único, o produto financeiro complexo incorporar riscos e caraterísticas de dois ou mais instrumentos financeiros de diferente estrutura e natureza.

Alguns exemplos de produtos financeiros complexos, cujas principais características e riscos é possível conhecer no guia de produtos financeiros complexos da CMVM[1] são:

- obrigações estruturadas;
- valores mobiliários representativos de dívida com possibilidade de perda de capital;
- certificados;
- *warrants* autónomos;
- CFD ou *Contracts for difference*;
- soluções de proteção de taxas de juro ou *caps*;
- contratos de derivados sobre divisas ou *ForexForward*;
- fundos especiais de investimento mobiliários e imobiliários.

Não cabe aqui neste livro falarmos de todos estes produtos. Ao invés, focar-nos-emos nos que nos são mais familiares, e em particular no tratamento de opções (também conhecidas por derivados ou *contingent claims*). E mesmo sobre estes produtos apenas falaremos no essencial, de forma a providenciar algum conhecimento básico. Informação mais detalhada pode ser encontrada, por exemplo, em Hull (2003) ou Kwok (2008).

Qualquer produto financeiro é na verdade um contrato, no qual é fixada a sua duração (usualmente designada por *maturidade* do contrato). Estes contratos podem envolver risco ou não, o que leva a que os produtos financeiros possam ser classificados numa de duas categorias:

- Sem risco (por exemplo, obrigações, certificados de aforro, depósitos bancários)
- Com risco (por exemplo ações de empresas, opções)

Tal como o nome indica, nos produtos financeiros sem risco o retorno do investimento é assegurado. Tal facto não implica necessariamente que seja um investimento com retorno positivo. Por exemplo, dependendo da maturidade

[1] Comissão do Mercado de Valores Mobiliários.

do produto, a inflação pode levar a que no final do investimento o valor do retorno seja inferior ao valor do investimento.

De entre os produtos sem risco, destaca-se aqui as *obrigações* (*bonds*, na literatura anglo-saxónica). Uma obrigação é um título de crédito, que confere ao seu detentor o direito de receber periodicamente juros (*cupons*) e, numa determinada data, o reembolso do capital. No contrato de venda de uma obrigação é estabelecido: o valor nominal, o preço de emissão, o valor do reembolso e método de amortização (correspondente à entrega periódica dos juros). Há vários tipos de obrigações, sendo a mais simples a obrigação de cupão zero:

Definição 2.1. *Numa obrigação de cupão zero não são pagos juros periodicamente, pelo que na maturidade do contrato o investidor recebe o valor de reembolso igual ao valor nominal.*

Nos modelos mais simples é assumido que a taxa de juro, r, é constante ou função determinística do tempo. Neste contexto, se B_0 e B_n designarem, respectivamente, o investimento inicial e o valor do investimento ao fim de n períodos (anos, tipicamente), então a relação entre B_0 e B_n depende da forma como os juros são acumulados. Assim:

- Juros anuais: $B_n = B_0(1+r)^n$;
- Juros semestrais: $B_n = B_0(1+\frac{r}{2})^{2n}$;
- Juros capitalizados k vezes no ano: $B_n = B_0(1+\frac{r}{k})^{nk}$;
- Juros continuamente capitalizados: $B_n = B_0 e^{rn}$.

Caso os juros não sejam constantes, mas sim uma função do tempo, $\{r(t), t > 0\}$, então os cálculos anteriores são facilmente adaptados. Por exemplo, se os juros forem continuamente capitalizados, teremos:

$$B_n = B_0 e^{\int_0^t r(s)ds}$$

assumindo que o integral anterior está bem definido.

No caso de uma obrigação ter associada a distribuição de cupões, então o cálculo do seu valor é distinto. Por exemplo, se uma obrigação a n anos, com valor nominal B_n, com distribuição de cupões no valor de C_j no instante t_j

$(j = 1, 2 \ldots, J)$, com juros continuamente capitalizados, então o seu valor no instante 0 é:

$$B_0 = B_n e^{-rn} + \sum_{j=1}^{J} C_j e^{-rt_j}$$

Caso os juros sejam capitalizados m vezes por ano, o fator e^{-r} é substituído por $\left(1 + \frac{r}{m}\right)^{-m}$.

Para efeitos notacionais, seja $B(t, T)$ o valor no instante t de uma obrigação de valor nominal 1 e de maturidade T. Obviamente que $B(T, T) = 1$. A *spot rate*, aqui denotada por $R(t, T)$, no caso de haver capitalização contínua dos juros é dada por

$$R(t, T) = -\frac{\ln B(t, T)}{T - t}$$

ou, equivalentemente:

$$B(t, T) = e^{-(T-t)R(t,T)}.$$

A taxa de juro desempenha um papel muito relevante na avaliação dos investimentos e no valor dos produtos financeiros. Para comparar retornos de investimentos que ocorrem em instantes de tempo distintos, o seu valor tem de ser descontado para o mesmo instante temporal. Por exemplo, assumindo uma taxa de juro constante e continuamente capitalizada, se C_i denotar o retorno do investimento i no instante t_i, $i = 1, 2$, então estes valores deverão ser comparados no instante presente, pelo que os seus valores descontados são $C_1 e^{-rt_1}$ e $C_2 e^{-rt_2}$, respectivamente.

2.1.1 Taxas de juro estocásticas

Quando se investe numa obrigação, há dois tipos de taxas de retorno que podem ser usadas para determinar o valor do produto:

- O rendimento até o vencimento (*yield to maturity*) é a taxa total de retorno que será obtida na maturidade do contrato após pagar o investimento original.
- A taxa *spot* é o valor do retorno se se puder vender o título a qualquer momento (até à sua maturidade), após pagar o investimento original.

Enquanto que a primeira taxa é relevante para comparar produtos que são guardados até à maturidade, a segunda é usada para comparar produtos que poderão ser comercializados no mercado a qualquer instante.

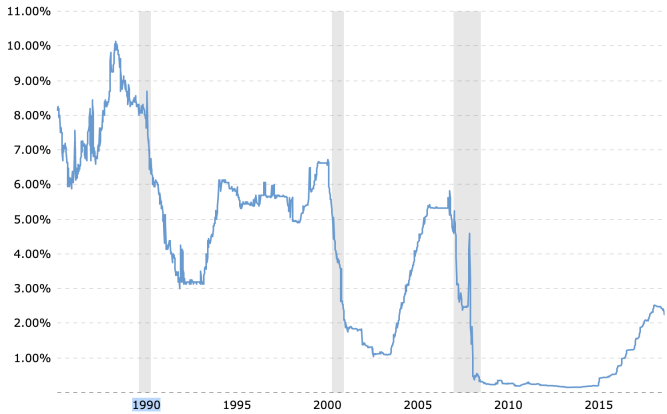


Figura 2.1: Taxa LIBOR a 1 mês

As taxas *spot* têm geralmente um comportamento muito variável, uma vez que o valor do retorno de um título que se pretenda vender no mercado depende fortemente da oferta e da procura do mesmo. Trata-se pois de uma taxa fortemente influenciada pelas condições do mercado, pelo que não é realista assumir que é constante ou até função determinística do tempo.

A Figura 2.1 ilustra precisamente a grande variabilidade das taxas de juro: reporta-se à taxa LIBOR [2] a um mês.

Estas taxas são intrínsecas ao tipo de produto que se está a negociar. Para além das obrigações ou títulos, também são muito relevantes nos mercados de transação de bens de consumo, de matérias primas e de câmbio de moeda.

Uma vez que não podemos realisticamente prever o comportamento futuro de taxas de juro, é natural que se veja o processo de evolução da taxa de juro como um processo estocástico.

Existem diversos modelos estocásticos para as taxas de juro. Estes modelos são descritos por equações diferenciais estocásticas da forma:

$$dr(t) = \mu(r(t), t)dt + \sigma(r(t), t)dW(t)$$

onde $\{r(t), t \geq 0\}$ representa o processo (estocástico) das taxas de juro (com $r(t)$ representando a taxa no instante t), $W = \{W(t), t \geq 0\}$ representa o

[2] LIBOR - *London InterBank Offered Rate* - é a taxa média interbancária contra a qual um grupo representativo de bancos se propõe efetuar empréstimos mutuamente no mercado monetário de Londres.

movimento Browniano padrão, e μ e σ são funções com determinadas características e que dependem do modelo que se assume para a taxa de juro [3].

De seguida apresentamos dois dos modelos mais frequentemente usados na modelação das taxas (estocásticas) de juro.

- *Modelo de Vasicek*

  Sob este modelo, o processo da taxa de juro estocástica, $\{r(t), t \geq 0\}$, segue a seguinte dinâmica:

  $$dr(t) = \alpha(\beta - r(t))dt + \sigma dW(t). \tag{2.1}$$

  O termo $\alpha(\beta - r(t))$ é o fator de tendência e descreve a alteração na taxa de juro por unidade de tempo; β representa a *média a longo prazo* (i.e, a longo prazo todas as trajetórias de r convergem para este valor). α é usualmente designado por *taxa de reversão à média* (e caracteriza a velocidade a que as trajetórias se aproximam do valor médio β), e σ representa a volatilidade.

  Na Figura 2.2 ilustra-se o comportamento típico de uma simulação de um processo cuja dinâmica é descrita por 2.1.

  Trata-se de um modelo com propriedades interessantes; em particular é um modelo com *tendência de reversão à média*, o que significa que embora as taxas de juro possam oscilar, não podem aumentar ou diminuir indefinidamente (uma vez que a longo prazo deverão aproximar-se do valor médio). Em particular, os mercados financeiros e económicos mostram que quando as taxas de juro estão *excessivamente* elevadas, há pressão do mercado e da situação económica para as forçar a diminuir, o que se traduz num comportamento de reversão à média.

  Adicionalmente permite que as taxas de juro tomem valores negativos.

  A equação diferencial estocástica 2.1 tem solução única (no sentido probabilístico), e é dada por:

  $$r(t) = r(s)e^{-\alpha(t-s)} + \beta(1 - e^{-\alpha(t-s)} + \sigma \int_s^t e^{-\alpha(t-u)} dW(u), \quad s < t \tag{2.2}$$

[3] Para mais informações sobre equações diferenciais estocásticas, refere-se Braumann (Braumann, 2005).

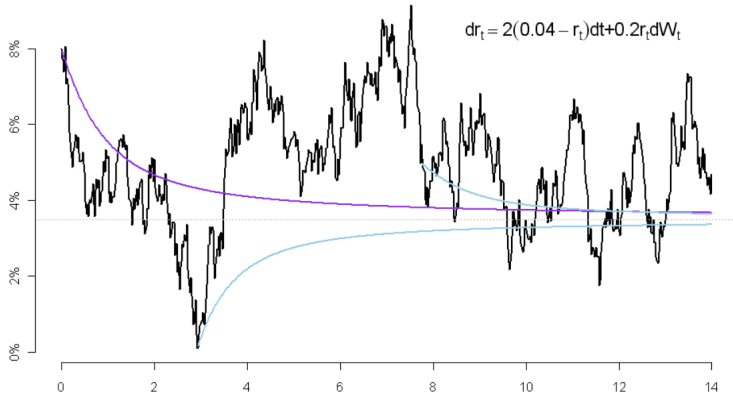


Figura 2.2: Trajetória de um modelo de Vasicek, com $\alpha = 2$, $\beta = 4\%$ e $\sigma = 0.2$. A curva central designa a taxa de retorno no instante $t = 0$ e as curvas posteriores representam as taxas de retorno em dois instantes de tempo posteriores. Fonte: https://en.wikipedia.org/wiki/Vasicekmodel#/media/File:Zins-Vasicek.png

onde o integral em 2.2 é um integral de Itô (Shreve, 2004). Além disso:

$$\mathbb{E}[r(t)|r(s)] = r(s)e^{-\alpha(t-s)} + \beta\left(1 - e^{-\alpha(t-s)}\right) \xrightarrow{t\to\infty} \beta \tag{2.3}$$

$$Var[r(t)|r(s)] = \frac{\sigma^2}{2\alpha}\left(1 - e^{-2\alpha(t-s)}\right) \tag{2.4}$$

Finalmente, é possível ainda derivar uma fórmula para o preço de uma obrigação de cupão nulo quando a taxa de juro segue o modelo de Vasicek: recordando que $B(t,T)$ designar o preço de uma obrigação de valor unitário, com maturidade T no instante $t \leq T$, então:

$$B(t,T) = A(t,T)e^{-r(t)P(t,T)} \tag{2.5}$$

com

$$P(t,T) = \frac{1 - e^{-\alpha(T-t)}}{\alpha} \tag{2.6}$$

$$A(t,T) = \exp\left(\beta - \frac{\sigma^2}{2\alpha^2}\left(B(t,T-t) - \frac{\sigma^2}{4\alpha}B^2(t,T)\right.\right. \tag{2.7}$$

(vide Mamon (2004))

- *Modelo de Cox-Ingersoll-Ross (CIR)*

Nesse caso a correspondente equação diferencial estocástica é:

$$dr(t) = \alpha(\beta - r(t))dt + \sigma\sqrt{r(t)}dW(t) \tag{2.8}$$

em que os fatores $\alpha(\beta - r(t))$, β e σ têm a mesma interpretação que no modelo anterior. Ao contrário do modelo de Vasicek, neste caso o processo apenas toma valores não negativos. Tal como o modelo anterior, também é um modelo de reversão à média, com:

$$\mathbb{E}[r(t)|r(s)] = r(s)e^{-\alpha(t-s)} + \beta(1 - e^{-\alpha(T-t)} \xrightarrow{t\to\infty} \beta \tag{2.9}$$

$$Var[r(t)|r(s)] = r(s)\left(e^{-\alpha(T-t)} - e^{-2\alpha(T-t)}\right) + \tag{2.10}$$

$$\frac{\beta\sigma^2}{2\alpha}(1 - e^{-\alpha(T-t)})^2 \tag{2.11}$$

Adicionalmente, a taxa de juro condicionada num instante anterior, $r(t)|r(s)$, tem igual distribuição à variável aleatória $\frac{Y}{2c}$, onde Y é uma v.a. com distribuição Qui-quadrado não central, com $\frac{4\alpha\beta}{\sigma^2}$ graus de liberdade, e parâmetro de não-centralidade $2cr(s)e^{-\alpha(t-s)}$, onde

$$c = \frac{2\alpha}{(1 - e^{-\alpha(t-s)})}\sigma^2.$$

2.2 Produtos com risco

Em contraponto com os produtos sem risco, existem os produtos com risco, no qual o retorno do investimento não é assegurado. No âmbito deste trabalho, estaremos especialmente interessados em ações e em *derivados financeiros*. Um derivado (*derivative* ou *contingent claim*, em inglês) é um produto financeiro cujo valor depende do valor de outros produtos (designado(s) por ativo(s) subjacente(s)). Nesta categoria de derivados, destacam-se os contratos *forward*, os futuros e as opções, assim como combinações destes produtos.

Os contratos de *forward* e de futuros são produtos com uma estrutura muito simples, que envolvem um comprador e um vendedor, no qual existe um acordo sobre a venda futura de um produto por um preço acordado. Os contratos podem envolver ações, moeda estrangeira ou obrigações, mas também podem envolver outro tipo de produtos (agrícolas, como rações e carne, ou outros, como petróleo, eletricidade, etc). Nestes contratos, T representa a maturidade e K o valor acordado de preço. No instante $t < T$, $V_{K,T}(S(t), T - t)$ representa o valor deste contrato quando o valor do ativo

subjacente no instante presente é $S(t)$ e o tempo para a maturidade do contrato é $\tau = T - t$. A diferença entre um contrato *forward* e futuro reside na forma como são transacionados: os contratos futuros são transacionados em mercados existentes para este fim, enquanto que os contratos *forward* podem ser transacionados em ambiente arbitrário.

Como referido anteriormente, estes contratos comportam risco no sentido que o seu retorno não é determinístico, como o exemplo 2.1 ilustra.

Exemplo 2.1. *Um investidor compra um contrato de futuros, no qual se compromete a comprar 1 milhão de dólares daqui a 90 dias, pagando a taxa de câmbio de 1 EUR para 1,13 \$. O investidor tem lucro se e só se na maturidade do contrato a taxa de câmbio do dólar for inferior à acordada no contrato. Mas seja qual for a taxa de câmbio na maturidade, o investidor é obrigado a cumprir o contrato de compra.*

Um contrato de futuro ou de *forward* não tem preço inicial, isto é, no instante em que o contrato é redigido não há lugar a pagamento de qualquer quantia. Na maturidade o contrato tem de ser honrado, havendo lucro sempre para uma e só para uma das partes.

Em ambos os casos, os contratos de *forward* e de futuros obrigam ambas as partes a cumprir o contrato. Pelo contrário, as opções dão o direito ao comprador da opção de exercer ou não, dependendo do preço do ativo subjacente na altura.

Definição 2.2. *Uma opção de compra europeia é um contrato que dá ao comprador da opção (detentor da opção) o direito de comprar na maturidade T o ativo subjacente pelo preço K (preço de exercício). Caso não exerça a opção, o contrato expira, sem mais nenhuma consequência.*

Caso a opção seja de venda, então o detentor da opção tem o direito de vender na maturidade o ativo subjacente pelo preço de K. Nesse caso a opção diz-se ser uma opção europeia de venda. Caso estas opções possam ser exercidas **até** *à maturidade, então designam-se como opções americanas.*

Existem outros tipos de opções (exóticas, asiáticas, etc) mas as opções europeias e americanas (de compra e de venda) são as mais comercializadas. Por esse motivo este tipo de opções são frequentemente designadas na literatura e no meio financeiro por opções *plain vanilla*.

O retorno de uma opção de compra europeia é dado por:

$$\max\{S(T) - K, 0\} \tag{2.12}$$

onde $S(T)$ designa o preço do ativo subjacente na maturidade do contrato e K o preço de exercício. Para uma opção de venda europeia o retorno é dado por:

$$\max\{K - S(T), 0\} \tag{2.13}$$

As funções definidas por 2.12 e 2.13 denominam-se *funções contrato*. Qualquer opção tem associada uma função contrato, que assim a define. Por exemplo se $X = I_{S(T)\leq K}$, então X é na verdade uma opção cujo retorno (unitário) apenas ocorre caso o preço do ativo subjacente na maturidade tiver um preço inferior a K.

Em muitas situações os investidores investem em vários produtos financeiros. Uma carteira ou *portfólio* é uma combinações de dois ou mais produtos financeiros, e o seu valor resulta da soma dos valores de cada produto que o compõe. Cada elemento do portfólio é designado por *posição*. Um investidor tem uma *posição longa* quando compra um produto, e tem uma *posição curta* quando vende. Consequentemente, um investidor tem uma posição longa numa opção de compra se comprar uma opção de compra; tem uma posição curta se vender um contrato de futuros.

Há certas combinações de opções de compra e de venda que têm uma designação particular. Por exemplo, um *bull call spread* resulta da compra de uma opção de compra e da venda de uma opção de venda com a mesma maturidade, com o mesmo ativo subjacente, mas com preço de exercício superior. Este tipo de estratégia é usada quando o investidor pensa que provavelmente o valor do ativo subjacente vai subir moderadamente nos próximos tempos. O retorno desta estratégia está representado na Figura 2.3.

Ao invés, quando o investidor espera uma diminuição do preço de um ativo subjacente, investe num *bear call spread*, que resulta da compra de uma opção de compra e outra de venda sobre o mesmo ativo subjacente, com a mesma maturidade, sendo o preço de exercício da opção de compra superior ao da venda; ver Figura 2.4.

Quando as duas opções têm o mesmo ativo subjacente, a mesma maturidade e mesmo preço de exercício, então diz-se que um investidor que detém uma de cada tem um *straddle call spread*. Neste caso o investidor tem um retorno positivo independentemente da variação do preço do ativo subjacente. O retorno é tão mais significativo quanto maior for a variação (quer para valores superiores quer para valores inferiores) do preço do ativo subjacente. O retorno desta estratégia tem a forma apresentada na Figura 2.5.

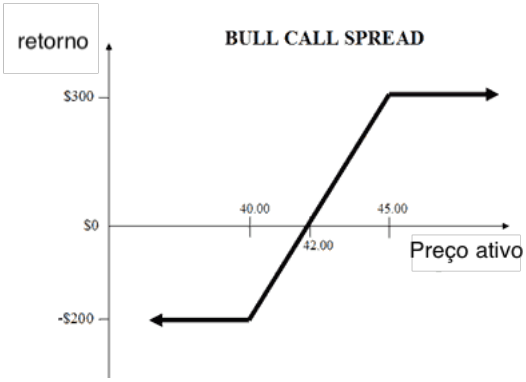


Figura 2.3: Retorno de um *bull call spread*.

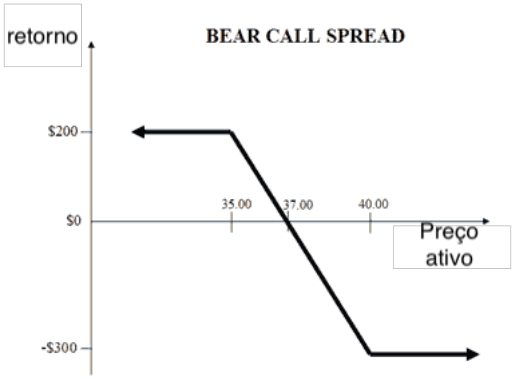


Figura 2.4: Retorno de uma *bear call spread*

Várias outras combinações de opções podem ser consideradas, dando origem a diferentes estruturas de retorno, e que refletem frequentemente as expectativas dos investidores. Para mais exemplos de estratégias, recomenda-se a consulta de Hull (2003).

Frequentemente estas estratégias têm por objectivo proteger o investidor face a potenciais perdas. Ao investir numa opção de compra, por exemplo, um investidor limita as suas perdas (no máximo perde o valor do contrato, o preço inicial que paga ao comprar a opção), sendo os seus ganhos hipoteticamente ilimitados. Mas se em vez de comprar apenas uma opção de compra, investir também numa opção de venda, de acordo com a estratégia de *straddle* , então

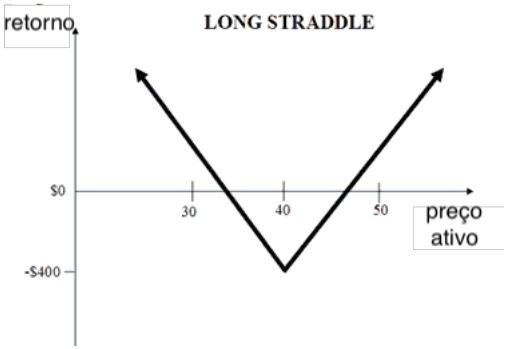


Figura 2.5: Retorno de uma *straddle*.

poderá ter retorno positivo do investimento quer o preço do ativo subjacente suba quer desça. Caso o preço do ativo subjacente suba, o retorno é inferior ao que obteria caso tivesse investido apenas numa opção de compra. Este exemplo ilustra que as estratégias de proteção ao risco são eficientes, no sentido que limitam as perdas, mas têm, em contraponto, a faculdade de reduzir (ou até limitar) os ganhos.

Para finalizar esta secção, terminamos com o conceito de *short selling*. Trata-se de uma estratégia financeira em que o investidor negocia ações que não detém. Para o efeito, pede *emprestadas* ações a um corretor, passando imediatamente a dispôr destas ações como se fossem suas; por exemplo, pode vende-las ao preço atual de mercado. No fim do contrato terá de cobrir a sua posição curta, comprando o mesmo número de ações no mercado, de forma a devolver ao corretor. Caso durante a vigência do contrato os detentores das ações tenham direito a dividendos, o investidor terá de pagar o valor correspondente ao dividendo ao corretor.

Short selling é usualmente usado quando o investidor acredita que o preço das ações vai descer. Trata-se de uma estratégia arriscada, usual em épocas de especulação mas legal. Por exemplo, após a crise financeira de 2008, os EUA suspenderam, temporariamente, a atividade de *short-selling*, impondo várias restrições, numa tentativa de controlar os danos da crise financeira.

2.3 Preço de derivados financeiros

Ao contrário dos contratos de futuros ou de *forward*, o preço de uma opção não é nulo (note-se que o retorno de uma opção é uma função não simétrica). Este preço é geralmente função de vários elementos, dos quais se destaca:

- A *volatilidade* (ou incerteza) do ativo subjacente, que impacta desde o início do contrato até à maturidade do mesmo.
- A taxa de juro, assim como o seu comportamento (determinística, estocástica, sazonal, etc).
- Existência (ou não) de dividendos, custos de manutenção ou outro tipo de custos inerentes à ação e passíveis de serem imputados até à maturidade do contrato.
- Comissões e custos de transação.

Para além destes fatores, é também de esperar que o preço da opção dependa do valor de exercício K (por exemplo, quando maior for K, mais cara deve ser uma opção de venda europeia; ao invés, quanto menor for K, maior deverá ser o preço de uma opção de compra europeia), da maturidade do contrato T (sendo de esperar que quanto maior for T, mais caro será uma opção americana; no caso de uma opção europeia, esta relação já não é tão clara).

Mas como se determina o preço de uma opção? Este assunto tem inspirado inúmeros trabalhos, sendo o resultado de inúmeros trabalhos científicos. A famosa fórmula de Black-Scholes, que fez valer a Robert C. Merton e Myron Scholes o prémio Nobel da economia em 1997, é um dos exemplos mais famosos.

Esta fórmula foi inicialmente apresentada no final da década de 60 por Fischer Black e Myron Scholes. Estes dois investigadores, após terem tentado, sem êxito, aplicar a fórmula nos mercados, resolveram concentrar os seus esforços no meio académico. A fórmula seria publicada em 1973 num artigo intitulado "The Pricing of Options and Corporate Liabilities". Paralelamente Robert Merton publicou um trabalho, (Merton, 1973) sobre a análise matemática do modelo que Black e Scholes propuseram. Por altura do Prémio Nobel, Black já tinha falecido (em 1995). Ainda que inelegível para o prémio devido à sua morte, Black foi mencionado como contribuidor pela academia sueca.

Na secção anterior apresentámos alguns conceitos básicos sobre produtos financeiros, sem nunca nos debruçarmos sobre o custo destes produtos. Nesta secção iremos apresentar alguns resultados preliminares e relações entre preços de opções, ações e obrigações. Posteriormente, no Capítulo 4, apresentaremos uma fórmula para o preço de uma opção, baseada precisamente no trabalho de Black, Merton e Scholes.

Ao descrevermos uma opção europeia de compra, por exemplo, vemos que o seu retorno para o comprador da opção é sempre positivo, uma vez que a opção só é exercida caso o preço do ativo subjacente seja superior ao preço de exercício. A questão óbvia é determinar o retorno para o vendedor da opção. Claramente que uma opção, assim como qualquer derivado, tem de ter um preço. Caso contrário o vendedor do derivado teria seguramente ou lucro nulo (no caso da opção não ser exercida) ou prejuízo.

E efetivamente é isto que acontece: os derivados financeiros têm um preço. No momento da compra da opção, o comprador paga o *devido* preço ao vendedor da opção. No momento do exercício, caso a opção seja exercida, o comprador tem retorno do investimento. Caso não seja exercida, o comprador não tem retorno e por isso este investimento acarretou um prejuízo para o comprador da opção e um lucro para o vendedor da mesma.

Mas qual é o preço correto de uma opção? Esta questão é bastante complexa em geral, havendo porém alguns resultados conhecidos para certo tipo de opções. Mas antes de abordarmos como este preço é calculado, começamos por definir um conceito fundamental em finanças: *oportunidade de arbitragem*. Uma oportunidade de arbitragem surge quando é possível fazer um investimento com risco nulo. Do ponto de vista matemático, se $V(t)$ designar o valor de um investimento no instante t e se $V(0)$ designar o investimento inicial, então existe uma oportunidade de investimento se

$$P(V(t) \geq 0 | V(0) = 0) = 1$$

isto é, sem necessidade de investimento inicial (pois $V(0) = 0$), o retorno no investimento é não negativo com probabilidade um. Desta forma significa que caso haja uma oportunidade de arbitragem, o investidor nunca perde dinheiro.

As oportunidades de arbitragem traduzem estados da economia não aceitáveis, e que não são admitidos em mercados regulados. Doravante consideram-se as seguintes condições de mercado.

Hipótese 1. *O mercado financeiro é perfeito, i.e, não há oportunidades de arbitragem, os custos de transação são nulos, não há restrições no número*

de vendas ou compras, e short-selling é sempre possível. A taxa de juro de empréstimos é igual à taxa de juro de depósitos e é possível vender ou comprar um número fracionário de ações.

Com estas hipóteses, é possível determinar relações de alguns derivados financeiros. Como se verá, neste caso as ferramentas matemáticas são muito básicas, mas em contraponto não se consegue determinar preços de derivados financeiros muito relevantes na economia mundial.

Os primeiros resultados estão relacionados com o retorno de qualquer produto financeiro e sua relação com o investimento inicial. A título ilustrativo, esboça-se a sua prova, baseada em simples argumentos de *não arbitragem.*

- Se dois portfólios, A e B, tiverem o mesmo valor num dado instante T, então têm o mesmo valor para qualquer $t < T$ (*princípio da não-dominância*).

  Demonstração. Seja $V_A(t)$ e $V_B(t)$ o valor do portfólio A e B, respectivamente, no instante t. Sem perda de generalidade, suponhamos que $V_A(t) < V_B(t)$. Então um investidor compra uma unidade de A e vende uma unidade de B (recorrendo para tal ao *short selling*). Dessa transação resulta $V_B(t) - V_A(t)$, que o investidor coloca no banco, a uma taxa de juro r. No instante T, o investidor tem o valor do portfólio A, tem $(V_B(t) - V_A(t))\, e^{r(T-t)} > 0$[4] no banco e precisa de devolver o portfólio B ao *short-seller* (e pelo qual terá de pagar $V_B(T)$). Então o seu lucro é:

  $$V_A(T) - V_B(T) + (V_B(t) - V_A(t))\, e^{r(T-t)} > 0$$

  pelo que existe uma oportunidade de arbitragem.

  Se se assumir, pelo contrário, que $V_A(t) > V_B(t)$, então argumento semelhante leva à conclusão de que existe uma oportunidade de arbitragem. Consequentemente, como se assume ausência de arbitragem, $V_A(t) = V_B(t), \forall t \leq T$. □

  Este resultado pode ser assim utilizado para definir o preço de um produto à custa de outro, caso se saiba que na maturidade dos contratos o retorno é igual.

[4]Sem perda de generalidade, assumimos nesta prova que existe capitalização contínua dos juros.

- Seja $V_F(t)$ o valor de um contrato *forward* (para o qual a maturidade é T e K é o valor do exercício) no instante t, dado que atualmente o preço do ativo subjacente é S_t. Então se o ativo não pagar dividendos nem custos ao longo da vigência do contrato, o valor do contrato é:

$$V_F(t) = S(t) - Ke^{-r(T-t)}$$

Demonstração. Consideremos os seguintes portfólios:
Portfólio A: constituído por uma posição longa de um contrato *forward* (com maturidade em T e preço de exercício K), e uma posição longa de uma obrigação de cupão zero, com valor K e maturidade T.
Portfólio B: uma posição longa de uma ação.

Ambos os portfólios têm na maturidade o valor igual ao da ação, $S(T)$. Logo têm de ter o mesmo valor para qualquer instante, e igual a $S(t)$, i.e.,

$$V_A(t) = V_B(t) \Leftrightarrow V_F(t) + Ke^{-r(T-t)} = S(t)$$

□

- O preço de opções de compra (venda) americanas e europeias é decrescente (crescente) no preço de exercício K. A demonstração segue a linha de raciocínio das anteriores, usando um argumento de arbitragem ao comparar os preços de duas opções com a mesma maturidade, o mesmo ativo subjacente mas diferente preço de exercício.

- O preço de opções de compra e venda (europeias e americanas) é convexo no preço de exercício K.

Doravante utilizamos a seguinte notação: os preços de opções de compra americanas e europeias no instante t, com preço de exercício K, maturidade T e com ativo subjacente S, denotam-se, respectivamente, por:

$$C(t, S(t), T, K) \text{ e } c(t, S(t), T, K).$$

De forma análoga, para opções de venda:

$$P(t, S(t), T, K) \text{ e } p(t, S(t), T, K).$$

O resultado que se segue é conhecido na literatura por *put-call parity*, e estabelece uma relação entre os preços de opções de compra e venda europeias. A sua demonstração baseia-se no princípio da não dominância, anteriormente descrito.

Proposição 2. *As seguintes relações, designadas por* put-call parity*, são válidas:*

- *Se o ativo subjacente não pagar dividendos, os preços das opções de compra e venda estão relacionados da seguinte forma:*

$$p\left(t, S(t), T, K\right) + S(t) = c\left(t, S(t), T, K\right) + Ke^{-r(T-t)}. \tag{2.14}$$

- *Caso haja lugar a pagamento de dividendos* $\{D_i, i = 1, \ldots, n\}$ *nos instantes* $t < t_1 < t_2 \ldots < t_n$ *então a relação é da seguinte forma:*

$$p\left(t, S(t), T, K\right) + S(t) - \sum_{i=1}^{n} D_i e^{-r(T-t_i)} = c\left(t, S(t), T, K\right) + Ke^{-r(T-t)}. \tag{2.15}$$

Demonstração. Mostramos a prova de 2.14, uma vez que a de 2.15 é semelhante. Para tal usamos o princípio da não-dominância, atrás enunciado. Consideremos então dois portfólios, constituídos pelos seguintes componentes:

- Portfólio A: uma opção de compra europeia e Ke^{-rT} dinheiro.
- Portfólio B: uma opção de venda europeia e uma ação (do ativo subjacente).

Cálculos triviais mostram que na maturidade o valor de cada um dos portfólios é igual:

$$\max(S(T), K)$$

Então, pelo princípio da não-dominância, o seu valor em qualquer instante t terá de ser igual, de onde resulta 2.14.

□

Claramente que há certos limites superiores e inferiores para os preços das opções. De seguida apresentam-se estes limites. As demonstrações não são apresentadas, mas todas se baseiam em argumentos de não arbitragem semelhantes aos atrás apresentados.

- Se o ativo subjacente não pagar dividendos antes da maturidade, então:

$$S(t) - Ke^{-r(T-t)} \leq c(t, S(t), T, K) \leq C(t, S(t), T, K)$$

- Uma opção de compra americana com um ativo subjacente que não pague dividendos até à maturidade nunca deve ser exercida antes da maturidade.

- Para opções de compra europeias:

$$-e^{-r(T-t)} \leq \frac{c(t, S(t), T, K_2) - c(t, S(t), T, K_1)}{K_2 - K_1} \leq 0$$

- Opções de compra e venda americanas têm preços crescentes com a maturidade.

Estes resultados, cujas demonstrações assentam num princípio simples e intuitivo (o princípio de ausência de arbitragem), não fornecem o preço de uma opção, mas sim estabelecem relações e desigualdades a que este preço deve obedecer. Por exemplo, decorre da *put-call parity* que apenas é necessário estabelecer o preço de uma opção de compra, por exemplo, dada a relação obrigatória entre este e o preço de uma opção de venda. Mas a questão fundamental continua por resolver:

Como determinar o preço de uma opção num mercado perfeito?

Esta questão será o tema central dos próximos dois capítulos. No Capítulo 3 veremos um modelo em tempo discreto e no Capítulo 4 abordaremos um modelo em tempo contínuo, o famoso modelo de Black-Scholes.

Capítulo 3

Modelo Binomial

3.1 Introdução

No capítulo anterior vimos alguns conceitos e terminologia da gíria financeira. A base dos produtos financeiros com risco são as ações, pelo que o estudo do comportamento do preço das ações é fundamental, uma vez que irá certamente influenciar o retorno do investimento. Como influencia o retorno do investimento, é também de esperar que tenha impacto no preço dos derivados financeiros, tais como as opções de compra ou venda que vimos no capítulo anterior.

Neste capítulo vamos explorar um modelo muito básico: o modelo binomial, o qual serve de motivação para o modelo mais conhecido da literatura (o modelo de Black-Scholes). O modelo binomial é um modelo com tempo discreto. Para evidenciar as diferenças em relação a modelos em tempo contínuo, usaremos a notação S_t para indicar o valor de um processo estocástico no instante t, com $t \in \mathbb{N}$, em vez de $S(t)$ (que reservamos para modelos em tempo contínuo).

Na próxima secção introduzimos alguma notação, a qual será também utilizada posteriormente, quando abordarmos o modelo em tempo contínuo.

Na maior parte do capítulo consideram-se apenas variáveis aleatórias e processos estocásticos uni-dimensionais. No contexto destes capítulos e do ponto de vista matemático, o caso multi-dimensional tem um tratamento semelhante mas com notação mais complexa.

3.2 Notação e hipóteses

Seja $(\Omega, \mathcal{F}, P)$ um espaço de probabilidades, e $T \in \mathbb{N}$. Considera-se ainda $\{\mathcal{F}_t, t \in \{0, 1, \ldots, T\}\}$, uma filtragem que descreve a informação disponível.

Uma filtragem é uma família não-decrescente de σ-álgebras (i.e., $\mathcal{F}_T = \mathcal{F}$ (onde T denota a maturidade dos contratos), com $\mathcal{F}_t \subseteq \mathcal{F}_{t+1}$), definidas em $\mathcal{F}$, onde usualmente $\mathcal{F}_0$ é a σ-álgebra trivial. Informalmente, $\mathcal{F}_t$ descreve a *informação* disponível até ao instante t. Desta forma podemos considerar que no instante zero não há informação disponível (pelo que nesse instante apenas os conjuntos triviais $\emptyset$ e Ω são mensuráveis, i.e., $\mathcal{F}_0 = \{\emptyset, \Omega\}$), e que a informação vai aumentando até ao instante T.

Assume-se que o mercado financeiro é composto por um ativo sem risco (tipicamente dinheiro no banco ou obrigações), e um número finito de ativos com risco. Os ativos com risco deste mercado constituem os títulos que são transacionáveis no mesmo.

O ativo sem risco é definido pelo seu processo de preço

$$S^0 = \left\{S_t^0, t \in \{0, 1, \ldots, T\}\right\}$$

com $S_0^0 = 1$ (sem perda de generalidade, assume-se que o valor nominal inicial é unitário) e S_t^0 sendo uma variável aleatória $\mathcal{F}_{t-1}$-mensurável. Esta última hipótese significa que o preço do ativo no instante t é conhecido logo no instante $t-1$, pelo que o investimento não comporta risco, uma vez que se sabe qual o retorno no instante prévio. Um processo com estas características diz-se ser um *processo previsível*. Pode acontecer que S_t^0 seja $\mathcal{F}_0$ mensurável, o que significa que o valor do ativo sem risco é determinístico; tal é o caso de aplicações em depósitos com taxas de juro definidas logo no início do contrato, pelo que o retorno do investimento é conhecido no instante em que o contrato (depósito) é celebrado.

Definição 3.1. *Dado um espaço de probabilidades equipado com uma filtragem, $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\in\mathbb{N}}, \mathbb{P})$, um processo estocástico $(X_t)_{t\in\mathbb{N}}$ é previsível se X_{t+1} for mensurável com respeito à σ-álgebra $\mathcal{F}_t$, para todo o $t \in \mathbb{N}$.*

A dinâmica do ativo sem risco depende apenas da taxa de juro. Se $r_{t,t+1}$ representar a taxa de juro entre os instantes t e $t+1$ então:

$$S_{t+1}^0 = (1 + r_{t,t+1})S_t^0, \quad t = 0, 1 \ldots, T-1.$$

Note-se que se se admitir que, embora estejamos com um modelo em tempo discreto, os juros são capitalizados em tempo contínuo, então a relação anterior toma a seguinte forma:

$$S^0_{t+1} = e^{r_{t,t+1}} S^0_t, \quad t = 0, 1 \ldots, T-1.$$

Para que S^0 seja um processo previsível, então $r_{t,t+1}$ tem de ser $\mathcal{F}_t$ mensurável, para qualquer $t = 0, 1 \ldots, T-1.$, i.e., o processo das taxas de juro é também um processo previsível.

Pelo contrário, a evolução dos ativos com risco comporta aleatoriedade.

Definição 3.2. *Um processo estocástico $(X_t)_{t\in\mathbb{N}}$ definido num espaço de probabilidades equipado com uma filtragem, $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\in\mathbb{N}}, \mathbb{P})$, é um processo adaptado se e só se X_t é mensurável com respeito à σ-álgebra $\mathcal{F}_t$, para todo o t.*

Seja

$$S^i = \left\{ S^i_t, t = \{0, 1 \ldots, T\} \right\}$$

o processo do preço do i-ésimo ativo com risco. Ao contrário de S^0, S^i não é previsível, i.e., dada a informação até ao instante $t-1$, quantificada em $\mathcal{F}_{t-1}$, S_t não é conhecido. Ao invés, S^i é um processo adaptado. Por simplicidade, frequentemente diremos apenas que um dado processo estocástico é previsível ou adaptado sem referir em relação a que espaço de probabilidades e filtragem. Salvo indicação contrária, caso deveremos interpretar como a filtragem natural.

Definição 3.3. *Dado um processo estocástico em tempo discreto, $(X_t)_{t\in\mathbb{N}}$, com $X_t : \Omega \to S$, com (S, Σ) sendo um espaço mensurável e $(\Omega, \mathcal{F}, \mathbb{P})$ um espaço de probabilidades, a filtragem natural de $\mathcal{F}$ em relação ao processo estocástico é definida como $(\mathcal{F}^X_t)_{t\in\mathbb{N}}$, onde*

$$F^X_t = \sigma \left\{ X_j^{-1}(A) \middle| \, j \leq t, A \in \Sigma \right\},$$

i.e., para cada $t \in \mathbb{N}$, $\mathcal{F}^X_t$ é a menor σ-álgebra em Ω que contém todas as pré-imagens de subconjuntos de S que são Σ-mensuráveis, para todos os instantes até t.

Um dos conceitos frequentemente referidos em problemas financeiros, já informalmente mencionado no capítulo anterior, é o de *portfólio* (ou carteira). No presente contexto, um portfólio é um processo estocástico

$$(\eta, \theta) = \{(\eta_t, \theta_t), t = 0, 1 \ldots, T\}$$

adaptado à filtragem $\{\mathcal{F}_t, t = 0, 1 \ldots, T-1\}$, e que toma valores em $\mathbb{R} \times \mathbb{R}^d$, onde d designa o número de tipos distintos de ativos com risco. Neste caso, η_t denota o número de ativos sem risco (por exemplo, se $\eta_2 = 1000$, então significa que o portfólio no instante $t = 2$ detém 1000 EUR em obrigações ou, de forma equivalente, 1000 EUR em depósitos[1]). Por seu turno, se $\theta_2 = (\theta_2^1, \theta_2^2, \theta_2^3) = (1; 0,3; 0,4)$ então significa que no mesmo instante o portfólio detém 1 ação de um certa empresa, 0,3 de outra empresa e 0,4 de uma terceira empresa.

Note-se que não há restrições nos valores que o portfólio pode assumir. Em particular:

- $\eta_t < 0$ significa que o dono do portfólio tem um empréstimo no instante t no valor de η_t;
- $\theta_t^i < 0$ significa que o dono do portfólio tem uma posição curta do ativo i no instante t.

Além disso quer o número de posições em ativos sem risco quer com risco pode ser fracionário (o que significa que se pode deter um número fracionário de ações). Por exemplo, se no instante 1, o portfólio for constituído por $(\eta_1, \theta_1) = (-10; 0,5; 2,5; -0,3)$, significa que neste instante há um empréstimo de 10 unidades, que detém 0.5 ações da primeira empresa, 2.5 ações da segunda empresa, e que tem *emprestada* (por *short-selling*, por exemplo) 0,3 ações da terceira empresa. Ao fechar o portfólio (i.e, quando os contratos terminarem, na maturidade), o detentor deste portfólio terá de pagar o empréstimo (sendo o valor atualizado face à taxa de juros) e devolver 0,3 ações da terceira empresa ao *short-seller*; caso esta empresa tenha pago dividendos neste período, terá ainda de devolver o valor dos dividendos devidamente capitalizado (tendo em conta a taxa de juro, o valor dos dividendos e quando foram atribuídos tais dividendos).

Para cada instante t, define-se a seguinte variável:

$$V_t^{(\eta,\theta)} = \eta_t S_t^0 + \sum_{i=1}^{d} \theta_t^i S_t^i \tag{3.1}$$

que designa o valor do portfólio (η, θ) no instante t.

[1]Sem perda de generalidade, assume-se que o valor nominal de uma obrigação ou o valor de um depósito é unitário.

Definição 3.4. *Um portfólio* (η, θ) *diz-se ser* auto-financiado *se*

$$V_t^{(\eta,\theta)} - V_{t-1}^{(\eta,\theta)} = \eta_t(S_t^0 - S_{t-1}^0) + \sum_{i=1}^{d} \theta_t^i (S_t^i - S_{t-1}^i) \tag{3.2}$$

i.e., não existe nem injeção nem retirada de dinheiro (uma vez que o valor do portfólio só se altera devido às alterações dos valores dos ativos que constituem o mesmo).

Consequentemente, num portfólio auto-financiado não há nem injeção de dinheiro nem utilização do lucro resultante nalgum instante. A compra de novos ativos tem de ser totalmente financiada pela venda de ativos disponíveis no portfólio.

O problema fundamental na área da matemática financeira é a determinação do preço de derivados financeiros. Ao longo deste trabalho usar-se-á as designações derivados e opções indistintamente.

Definição 3.5. *Um derivado financeiro é um processo estocástico* X *da forma* $X = g(S)$*, em que* S *designa o processo estocástico do preço do(s) ativo(s) subjacente(s) e* g *é a função contrato.*

A interpretação de um derivado financeiro é a seguinte: é um contrato que estabelece que o seu comprador recebe um valor, função do(s) ativo(s) subjacente(s) no seu término. Estes contratos podem depender apenas do valor do(s) ativo(s) no instante T (i.e., $X = g(S_T)$), onde S_T é o valor do ativo no instante T, ou podem depender do valor do(s) ativo(s) em instantes anteriores (por exemplo, $X = g(S_t, t \in \{1, 2, \ldots, T\})$.

Por exemplo, se $g(s) = \max[s - K, 0]$, então $X = g(S_T)$ representa o contrato relativo a uma opção de compra europeia. Por outro lado, se:

$$X = \max[\frac{1}{n} \sum_{i=1}^{T} S_i - K, 0]$$

então X representa um derivado para o qual na maturidade o retorno depende de todos os valores registados para o ativo subjacente. Não se trata de uma opção europeia, embora a função contrato tenha algumas semelhanças.

Para um derivado financeiro X, seja $\Pi(t; X)$ o preço no instante t. Em consequência do princípio da não arbitragem, temos a seguinte relação trivial:

$$\Pi(T; X) = X$$

onde T designa a maturidade do derivado, i.e., o preço de um contrato na sua maturidade é exatamente igual ao seu retorno. A questão é então determinar $\Pi(t;X), t \leq T-1$, i.e., determinar o preço do contrato X para qualquer instante anterior à maturidade.

Uma das estratégias possíveis para determinar o preço de uma opção é encontrar um portfólio (uma combinação de um ativo sem risco com ações) cujo retorno na maturidade seja igual ao retorno da opção, com probabilidade um. Se tal acontecer então, pelo princípio da não dominância, o preço deste derivado é exatamente igual ao preço do portfólio. Mas nesse caso a questão relevante é: *Será que é possível construir portfólios que retornem* **exatamente** *o mesmo valor que o derivado X?*

De forma a responder a esta questão, considera-se a seguinte definição:

Definição 3.6. *Um derivado financeiro X é* replicável *se existir um portfólio (η, θ) tal que:*

$$V_T^{(\eta,\theta)} = X$$

com probabilidade 1. Se todos os derivados financeiros forem replicáveis, então o mercado diz-se ser completo.

Consequentemente, se um derivado for replicável, tal significa que do ponto de vista estritamente financeiro não existe nenhuma diferença entre comprar o derivado ou investir no portfólio. Na verdade:

Proposição 3. *Se um derivado financeiro X for replicável pelo portfólio (η, θ), então:*

$$\Pi(t;X) = V_t^{(\eta,\theta)}, \quad t \in [0,T]$$

Qualquer outro valor levará a oportunidades de arbitragem.

A demonstração é trivial, invocando apenas o princípio da não dominância apresentado no Capítulo 2.

3.3 Modelo binomial uni-dimensional

Nesta secção apresentamos o modelo binomial mais simples, que envolve apenas o preço do ativo sem risco (que pode ser uma obrigação ou empréstimo ou depósito bancário) e um ativo com risco (uma ou uma fração de ação). Mais

tarde veremos como este modelo pode ser estendido a um portfólio mais complexo, com vários tipos de ações, por exemplo. Além disso admitimos apenas opções cuja função contrato depende apenas do valor do ativo subjacente na maturidade, i.e.,

$$X = g(S_T)$$

Este tipo de contratos é usualmente designado por *contrato simples*.

O ativo sem risco, $S^0 = \{S_t^0, t = 0, 1, \ldots, T\}$ tem uma evolução determinística, com:

$$S_0^0 = 1, \quad S_1^0 = 1 + r$$

onde r representa a taxa de juro, que neste contexto se assume determinística e constante. Além disso, considera-se que a capitalização do juro decorre ao mesmo ritmo que a unidade temporal considerada. Tal significa, por exemplo, se T designar o número de meses do contrato, r é a taxa de juro mensal.

Pelo contrário, o preço da ação, $S^1 = \{S_t^1, t = 0, 1, \ldots, T\}$, varia aleatoriamente ao longo do tempo, e o modelo que se considera é o seguinte:

$$S_0^1 = s, \quad S_{t+1}^1 = S_t^1 Z_{t+1}$$

onde $\{Z_t, t = 1, \ldots, T\}$ é um conjunto de v.a. independentes e identicamente distribuídas, com $P(Z_t = z) = p1_{\{Z=u\}} + (1-p)1_{\{Z=d\}}$, onde $u, d \in \mathbb{R}$. Consequentemente, o valor deste portfólio nos instante $t = 0$ e $t = 1$ é dado por:

$$V_0^{(\eta,\theta)} = \eta_0 + \theta_0 s, \quad V_t^{(\eta,\theta)} = \eta_0(1+r) + \theta_0 s Z_1$$

Proposição 4. *O modelo binomial não tem portfólios de arbitragem se e só se:*

$$d \leq 1 + r \leq u. \tag{3.3}$$

A equação 3.3 tem a seguinte leitura económica: o retorno do investimento num produto com risco não pode dominar (caso contrário não haveria interesse em investir em aplicações sem risco) nem ser dominado pelo retorno de uma obrigação (caso contrário não haveria interesse em investir em ações).

Demonstração. Comecemos por provar que ausência de arbitragem implica 3.3. Sem perda de generalidade, assumimos que $T = 1$. Suponhamos, por contradição, que 3.3 não se verifica, e que, por exemplo, $\theta_0(1 + r) > \theta_0 u$. Então $\theta_0(1 + d) > \theta_0 d$, pelo que o seguinte portfólio leva a uma situação de

arbitragem: $(\eta_0, \theta_0) = (\theta_0, -1)$ (de forma a que o valor inicial do investimento seja nulo), i.e., short-sell uma ação, vendê-la e investir o dinheiro resultante no banco. Então embora o valor do investimento seja zero, o valor deste portfólio no instante $T = 1$ é:

$$V_1^{(\eta,\theta)} = \theta_0(1+r) - \theta_0 Z_1 > \theta_0(1+r) - \theta_0 d > 0$$

com probabilidade 1, o que mostra uma oportunidade de arbitragem.

No sentido contrário, suponhamos que 3.3 é satisfeita. Seja ainda (η, θ) um portfólio, composto inicialmente por θ_0 ações e $\eta_0 = -\theta_0 s$ em ativo sem risco, a que corresponde o valor inicial $\eta_0 + \theta_0 s = 0$. No instante $T = 1$ o valor deste portfólio é:

$$V_1^{(\eta,\theta)} = \theta_0 s \times \begin{cases} u - (1+r), & Z_1 = u \\ d - (1+r), & Z_1 = d \end{cases}$$

pelo que caso $\theta_0 > 0$, o portfólio só seria um portfólio de arbitragem se $u - (1+r) > 0$ e $d - (1+r) > 0$, o que contraria a hipótese. Se $\theta_0 < 0$, o portfólio só seria um portfólio de arbitragem se $u-(1+r) < 0$ e $d-(1+r) < 0$, o que contraria a hipótese. □

Em face do resultado anterior, decorre que o sistema de equações a um passo:

$$(1+r)\eta_0 + su\theta_0 = g(su) \tag{3.4}$$
$$(1+r)\eta_0 + sd\theta_0 = g(sd) \tag{3.5}$$

tem uma e uma só solução, a qual é dada por:

$$\eta_0 = \frac{1}{1+r}\frac{ug(d) - dg(u)}{u-d} \tag{3.6}$$
$$\theta_0 = \frac{1}{s}\frac{g(u) - g(d)}{u-d} \tag{3.7}$$

pelo que o mercado é completo, i.e., qualquer derivado financeiro X pode ser replicado por um portfólio consistindo num investimento inicial de $\eta_0 = \frac{1}{1+r}\frac{ug(d)-dg(u)}{u-d}$ obrigações e $\theta_0 = \frac{1}{s}\frac{g(u)-g(d)}{u-d}$ ações. Além disso, decorre de considerações anteriores que o preço deste derivado, $\Pi(0; X)$, é igual a:

$$\begin{aligned} V_0^{\eta,\theta} &= \frac{1}{1+r}\frac{ug(d) - dg(u)}{u-d} + \left(\frac{1}{s}\frac{g(u) - g(d)}{u-d}\right) s \\ &= \frac{1}{1+r}\left(\frac{1+r-d}{u-d}g(u) + \frac{u-(1+r)}{u-d}g(d)\right) \end{aligned}$$

Este argumento pode ser repetido para um instante arbitrário $t \leq T$. Note-se que para um tempo arbitrário t, a construção do processo estocástico do preço do ativo subjacente com risco leva a que:

$$S_t = su^Y d^{t-Y}, \quad Y = 0, 1, \ldots, t$$

onde Y denota o número de subidas que ocorreram até ao instante t. E em face das hipóteses sobre a evolução do preço do ativo, vem que Y tem distribuição binomial, de parâmetros t e u, o que motiva o nome deste modelo (*modelo binomial*).

Cálculos semelhantes aos do caso $t = 1$ levam ao seguinte resultado para construção do portfólio de réplica para um instante arbitrário $t \leq T$ (a demonstração desta proposição segue as linhas do caso $t = 1$, podendo ser encontrada em Björk (2009)).

Proposição 5. *O portfólio de réplica* $(\eta, \theta) = \{(\eta_t, \theta_t), t = 0, 1, \ldots, T\}$ *do derivado financeiro* X *é dado por:*

$$\eta_t(k) = \frac{1}{1+r} \frac{uV_t(k) - dV_t(k+1)}{u-d} \tag{3.8}$$

$$\theta_t(k) = \frac{1}{S_{t-1}} \frac{V_t(k+1) - V_t(k)}{u-d} \tag{3.9}$$

onde $\eta_t(k)$ *(*$\theta_t(k)$*) designa o número de ativos sem (com) risco no instante* t *quando ocorreram* k *subidas do preço do ativo subjacente até esse instante, e* $V_t(k)$ *designa o valor do portfólio no instante* t*, tendo ocorrido* k *subidas. Os valores de* $V_t(k)$ *podem ser calculados recursivamente da seguinte forma:*

$$V_t(k) = \frac{1}{1+r} \left(q_u V_{t+1}(k+1) + (1-q_u) V_{t+1}(k)\right) \tag{3.10}$$

$$V_T(k) = g(su^k d^{T-k}) \tag{3.11}$$

com

$$q_u = \frac{1+r-d}{u-d}; \quad q_d = 1 - q_u = \frac{u-(1+r)}{u-d}. \tag{3.12}$$

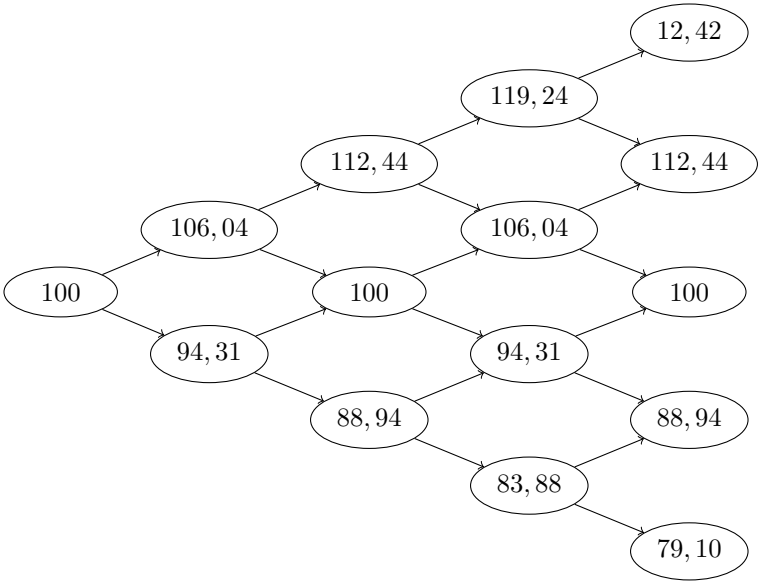
Em particular o preço do derivado X *no instante* 0 *é dado por:*

$$\Pi(0; X) = \left(\frac{1}{1+r}\right)^T \sum_{k=0}^{T} q_u^k q_d^{T-k} g(su^k d^{T-k})$$

No exemplo 3.1 ilustra-se a aplicação destes cálculos.

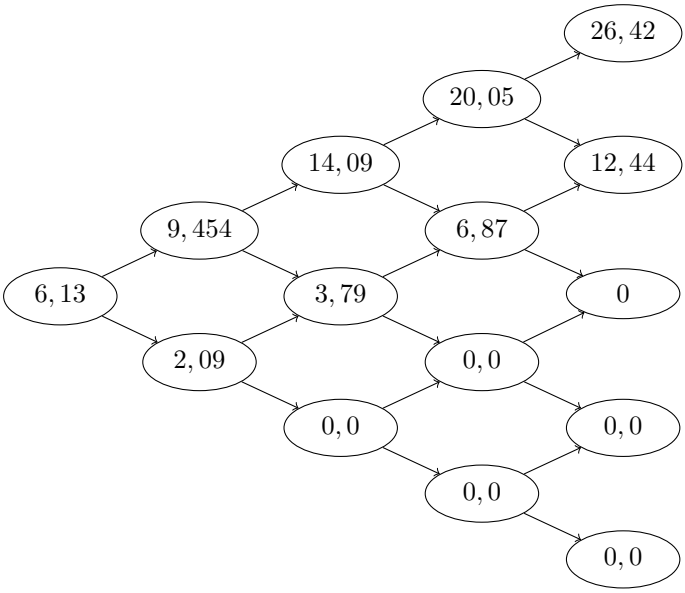
Exemplo 3.1. *Considere-se uma opção de compra europeia, com preço de exercício de 100 EUR , com maturidade de 4 meses. O preço atual do ativo subjacente é de 100 EUR, a taxa de juro é de 0% e o ativo pode subir 6,04% ou descer 5,69%. Considera-se um modelo binomial, com intervalo temporal de um mês.*

Neste caso o preço do ativo subjacente toma os seguintes valores possíveis:



Este tipo de estrutura designa-se usualmente por árvore recombinante. *Se o número de intervalos temporais considerado for W, então na maturidade o número de resultados possíveis para o preço do ativo subjacente é $W + 1$.*

Usando as fórmulas anteriores, o preço deste derivado em cada um dos nós da árvore é o seguinte:



De onde se conclui que no momento presente, quando o preço da ação é de 100 EUR, o valor desta opção daqui a 4 meses é de 6,13 EUR.

Obviamente que o número de períodos que se consideram (que no caso anterior foi 4), assim como o tipo de capitalização da taxa de juros, tem impacto no valor que se atribuiu à opção. Por exemplo, se em vez de se considerar 4 instantes de avaliação (um por cada mês) se considerasse 24 períodos, o valor desta mesma opção seria aproximadamente 6,24 EUR.

3.3.1 Medida de risco neutro

Uma das consequências da equação 3.3 é que o fator $(1 + r)$ pode ser visto como combinação linear convexa de u e de d, i.e.,

$$1 + r = qu + (1 - q)d$$

com $q > 0$, pelo que se define uma função de probabilidade:

$$Q(Z = u) = q_u, \quad Q(Z = d) = 1 - q_u$$

com q_u dado por 3.12.

Calculando o valor esperado de S_1 (o valor do ativo com risco no instante 1) em relação a esta medida de probabilidade, temos que:

$$\frac{1}{1+r}\mathbb{E}^Q[S_1] = \frac{1}{1+r}(qsu + (1-q)sd) = s$$

pelo que obtemos a seguinte relação:

$$s = \frac{1}{1+r}\mathbb{E}^Q[S_1].$$

Esta fórmula mostra que o preço atual da ação pode ser visto como o valor esperado descontado do preço da ação amanhã. A medida de probabilidade Q designa-se por **medida de martingala**.

Usando esta nova medida de martingala, podemos calcular o preço de um derivado financeiro, usando uma fórmula alternativa mas equivalente à que resulta da aplicação da Proposição 5:

Proposição 6. *O preço de um derivado financeiro X com função contrato g no instante inicial 0 é dado por:*

$$\Pi(0;X) = \left(\frac{1}{1+r}\right)^T \mathbb{E}^Q[g(S(T))]$$

onde Q designa a medida de martingala anteriormente definida. Em particular,

$$\Pi(0;X) = \left(\frac{1}{1+r}\right)^T \sum_{k=0}^{T} \binom{T}{k} q_u^k (1-q_u)^{T-k} g(su^k d^{T-k})$$

3.3.2 Opções americanas

O modelo binomial atrás apresentado refere-se a opções europeias, isto é, opções em que o detentor do contrato decide sobre o exercício da mesma no instante da maturidade. Mas será que se pode adaptar alguns dos resultados quando a opção é americana, podendo ser exercida em qualquer instante até à maturidade?

Proposição 7. *Para uma opção americana de compra cujo ativo subjacente não paga dividendos nunca é óptimo exercer antes da maturidade.*

Demonstração. Para demonstrar este resultado, vamos provar que o preço de uma opção europeia de compra é igual ao preço de uma opção americana de

compra quando não há lugar a distribuição de dividendos, o que prova que não deve ser exercida antes da maturidade. Para isso vamos admitir, por contradição, que o preço da opção americana é superior ao preço da correspondente europeia, i.e., vamos admitir que

$$c(0, S(0), T, K) < C(0, S(0), T, K)$$

que, por simplicidade de notação, vamos designar simplesmente por c e C. Então considere-se a seguinte estratégia, assumindo que o exercício da opção americana ocorre antes da maturidade.

- Vender a opção americana e comprar a europeia. Investir $C - c$ no banco, a uma taxa de juro r;
- Quando o comprador da opção americana decidir exercer num instante $t \leq T$, *short-sell* uma ação (do ativo subjacente), que será dada ao comprador da opção, recebendo em troca K. Colocar este valor recebido no banco.
- No instante T decidir sobre a opção de compra europeia:
  - se $S(T) > K$, exercer a opção, retornar a ação ao short-seller. Nesse caso o lucro é:

    $$(C - c)e^{rT} + K(e^{T-t} - 1) > 0$$

  - Se $S(T) \leq K$, a opção não é exercida. Nesse caso terá de comprar uma ação, por um valor inferior a K, de forma a devolver ao *short-seller*. O lucro é então:

    $$(C - c)e^{rT} + Ke^{T-t} - S(T) > (C - c)e^{rT} + K(e^{T-t} - 1) > 0$$

  Em ambos os casos há hipótese de arbitragem pelo que não é possível considerar exercício antecipado.

□

Outra forma de enunciar o resultado da Proposição 7 encontra-se na próxima proposição.

Proposição 8. *O preço de uma opção americana de compra em que não há pagamento de dividendos é igual ao correspondente opção europeia.*

No caso de haver lugar a pagamento de dividendos, o argumento anterior pode falhar porque o valor do dividendo pode ser tal que seja ótimo que o exercício ocorra antes da maturidade.

No caso de opções de compra, pode ser ótimo exercer antes da maturidade, mesmo no caso de não haver direito a pagamento de dividendos.

Exemplo 3.2. *Considere-se um modelo binomial para uma opção de venda americana com preço de exercício 100 EUR, para uma maturidade de $T = 3$ meses, cujo ativo subjacente atualmente vale 100 EUR. O preço do ativo pode subir por um fator de 25% ou pode descer, por um fator de 20%. A taxa de juro mensal é 0,01.*

Para esta opção, a seguinte estratégia é ótima: não exercer ao fim do primeiro mês; exercer no segundo mês caso se registe uma descida do preço da ação (e nesse caso o retorno será de 36 EUR). O preço deste derivado é 15,91 EUR.

É possível estender estes resultados ao caso de que a opção seja escrita como dependente de vários ativos. A abordagem e os resultados são semelhantes aos aqui apresentados para o caso unidimensional. Sugere-se a leitura de Björk (2009).

3.4 Abordagem probabilística

Antes de apresentar a abordagem probabilística dos preços dos derivados financeiros, apresenta-se um pequeno resumo dos conceitos mais relevantes sobre martingalas em tempo discreto.

3.4.1 Valor esperado condicionado

O conceito de valor esperado condicionado é central em Teoria da Probabilidade. Na determinação do preço de um derivado financeiro desempenha um papel fundamental. Nesta secção apresentamos os conceitos e propriedades mais relevantes, indicando algumas referências bibliográficas onde o assunto é explicado com mais detalhe.

Seja Z uma variável aleatória definida num espaço de probabilidades $(\Omega, \mathcal{F}, P)$, $\mathcal{F}$-mensurável, i.e.,

$$\{\omega \in \Omega : Z(\omega) \leq z\} \in \mathcal{F}, \quad \forall z \in \mathbb{R}.$$

De forma similar, define-se a σ-álgebra gerada pela variável Z, que se denota por $\mathcal{F}^Z$, como sendo a menor σ-álgebra para a qual todos os acontecimentos $\{\omega \in \Omega : Z(\omega) \leq z\}$ são $\mathcal{F}^Z$-mensuráveis. Por esta razão, a σ-álgebra gerada pela v.a. Z contém toda a informação sobre Z.

Uma v.a. Z, $\mathcal{F}$-mensurável, é *integrável* se $\mathbb{E}[|Z|] < \infty$, o que equivale a exigir que $\mathbb{E}[Z^+] < \infty$ e que $\mathbb{E}[Z^-] < \infty$ (onde a^+ e a^- representam a parte positiva e negativa de a, respetivamente). Desta forma:

$$\mathbb{E}[Z] = \mathbb{E}[Z^+] - \mathbb{E}[Z^-]$$

está bem definido e toma valores em $\mathbb{R}$. Doravante assumiremos que as v.a. referidas no texto são mensuráveis e integráveis, exceto menção em contrário.

Definição 3.7. *Dada uma v.a. Z integrável e mensurável com respeito à sigma-álgebra $\mathcal{F}$, e $\mathcal{G} \subseteq \mathcal{F}$, o valor esperado condicionado de Z dado $\mathcal{G}$, que se denota por $Y = \mathbb{E}[Z|\mathcal{G}]$ é tal que:*

- *$\mathcal{F}^Y \subset \mathcal{F}$ (i.e., o valor esperado condicionado de uma v.a. não contém mais informação que a contida na σ-álgebra $\mathcal{F}$);*
- *Y é $\mathcal{G}$-mensurável;*
- *Y satisfaz a seguinte relação:*

$$\mathbb{E}[Z1_A] = \mathbb{E}[Y1_A], \quad \forall A \in \mathcal{G}$$

A questão que naturalmente se levanta é a existência e unicidade de valor esperado condicionado. A proposição que se segue assegura que a resposta a ambas as questões é afirmativa.

Proposição 9. *Nas condições da definição anterior, a v.a. Y é única, no sentido que caso exista uma v.a. W nas mesmas condições, então*

$$P\left(\omega \in \Omega : Y(\omega) = W(\omega)\right) = 1$$

A demonstração deste resultado assenta em resultados clássicos de Teoria da Probabilidade, mas relevamos a sua prova para referência apropriada, por exemplo, Mikosch (1998).

O valor esperado condicionado de uma v.a. Z pode ser interpretado como uma versão *grosseira* da própria v.a., no sentido que ambas coincidem nalguns

conjuntos A (conjuntos esses que têm de ser elementos da σ-álgebra), mas não em todos.

Mas esta definição não é construtiva, e nalguns casos *calcular* o valor esperado condicionado é complexo e não se reveste de interesse. Obviamente que caso $\mathcal{G} = \mathcal{F}^X$, então:

$$\mathbb{E}[X|\mathcal{F}^X] = X$$

Na verdade, para efeitos do cálculo estocástico relevante na determinação de preços de derivados, as propriedades do valor esperado condicionado são mais relevantes que o seu cálculo. De seguida enumeram-se estas propriedades, omitindo a sua demonstração. Para um tratamento mais aprofundado de valores esperados condicionados, referem-se os seguintes textos: Williams (1991), Rao (2005), Bogachev (2007), por exemplo.

Proposição 10. *Seja $(\Omega, \mathcal{F}, P)$ um espaço de probabilidades, e $\mathcal{G} \subseteq \mathcal{F}$ uma σ-álgebra. Assumindo que todas as variáveis aleatórias de seguida são integráveis e $\mathcal{F}$-mensuráveis, e que as igualdades e desigualdades são interpretadas no sentido de quase-certamente com respeito à medida de probabilidades P, então:*

- *Aditividade:* $\mathbb{E}[X + Y|\mathcal{G}] = \mathbb{E}[X|\mathcal{G}] + \mathbb{E}[Y|\mathcal{G}]$
- *Monotonia: se $X \leq Y$ então* $\mathbb{E}[X|\mathcal{G}] \leq \mathbb{E}[Y|\mathcal{G}]$
- *Se α é $\mathcal{G}$-mensurável, então* $\mathbb{E}[\alpha X|\mathcal{G}] = \alpha\mathbb{E}[X|\mathcal{G}]$. *Em particular, se X for $\mathcal{G}$-mensurável, então* $\mathbb{E}[X|\mathcal{G}] = X$.
- *Independência: se $\mathcal{G}$ for independente de $\mathcal{F}^X$, então* $\mathbb{E}[X|\mathcal{G}] = \mathbb{E}[X]$[2].
- *Valor esperado iterado (Tower property): Se $\mathcal{H} \subseteq \mathcal{G}$ então*

$$\mathbb{E}[\mathbb{E}[X|\mathcal{G}]|\mathcal{H}] = \mathbb{E}[X|\mathcal{H}]$$

- *Desigualdade de Jensen: se $f : \mathbb{R} \to \mathbb{R}$ for uma função convexa, então* $f(\mathbb{E}[X|\mathcal{G}]) \leq \mathbb{E}[f(X)|\mathcal{G}]$.

[2]Duas σ-álgebras $\mathcal{F}$ e $\mathcal{G}$ são independentes se para qualquer $F \in \mathcal{F}$ e $G \in \mathcal{G}$, $P(F \cap G) = P(F)P(G)$, i.e., dados dois elementos quaisquer de cada uma das σ-álgebras a probabilidade da interseção é igual ao produto das probabilidades individuais.

3.4.2 Martingalas em tempo discreto

Um dos conceitos fundamentais em matemática financeira, e que está intimamente ligado à ausência de arbitragem, é a definição de martingala.

Doravante considera-se um espaço de probabilidades $(\Omega, \mathcal{F}, P)$ e uma filtragem $\{\mathcal{F}_t, t \leq T\}$, com $\mathcal{F}_t \subseteq \mathcal{F}$. Além disso, exceto em menção contrária, assume-se que todos os processos estocásticos mencionados no texto estão adaptados à filtragem em questão.

Definição 3.8. *Um processo $M = \{M_t, t = 0, 1 \ldots, T\}$ é uma $(\mathcal{F}, P)$-martingala (respectivamente $(\mathcal{F}, P)$ sub martingala $(\mathcal{F}, P)$ super martingala) se M_t for P-integrável para qualquer $t \leq T$, e se:*

$$\mathbb{E}[M_t|\mathcal{F}_{t-1}] = (\leq, \geq)M_{t-1}, \quad \forall t \leq T. \tag{3.13}$$

Em termos formais, a propriedade de martingala diz sempre respeito a uma filtragem, mas frequentemente quando se classifica um processo como sendo martingala, omite-se em relação a que filtragem se considera.

Uma martingala é pois um processo estocástico com valor esperado condicionado constante. Os seguintes casos são exemplos de martingalas.

- Seja Y uma variável aleatória integrável no espaço de probabilidades equipado com uma filtragem $(\Omega, \mathcal{F}, P, \{\mathcal{F}_t\})$. Considere-se agora a seguinte construção do processo estocástico X:

  $$X_t = \mathbb{E}[Y|\mathcal{F}_t], t \geq 0$$

  Então $X = \{X_t, t \geq 0\}$ é uma martingala em relação à filtragem $\{\mathcal{F}_t, t \geq 0\}$.

- Se X for um processo com incrementos independentes num espaço de probabilidades equipado com uma filtragem, $(\Omega, \mathcal{F}, P, \{\mathcal{F}_t\})$, com valor esperado constante, então X é uma martingala.

Em resultado da desigualdade de Jensen para valores esperados condicionados, o seguinte resultado ocorre:

Proposição 11. *Seja X uma martingala e f uma função convexa (côncava) tal que $f(X_t)$ é integrável para todo o t. Então o processo $\{f(X_t), t \geq 0\}$ é uma submartingala (supermartingala).*

Se X for uma submartingala e se f for uma função convexa, não decrescente, com $f(X_t)$ integrável para todo o t, o processo $\{f(X_t), t \geq 0\}$ é uma submartingala.

Num intervalo finito $[0, T]$ qualquer martingala é da seguinte forma:

$$X_t = \mathbb{E}[X_T | \mathcal{F}_t]$$

No caso do intervalo ser infinito, o resultado não é necessariamente verdadeiro.

3.4.3 Determinação do preço via martingalas

No modelo binomial o preço de um derivado financeiro foi obtido recorrendo a um portfólio de réplica. Ou seja, no caso do modelo ser completo (que é o caso do modelo binomial), qualquer derivado financeiro pode ser replicado por um portfólio de ativo sem risco e de ativos com risco, no sentido que o processo de ganhos e/ou perdas obtido no investimento no portfólio é exatamente igual ao obtido caso se invista no derivado financeiro. Um simples argumento de arbitragem implica então que o preço de ambos terá de ser o mesmo.

A condição:

$$d \leq 1 + r \leq u$$

equivale a ausência de arbitragem. Para além disso, decorre trivialmente que

$$Q = (\frac{1 + r - d}{u - d}, \frac{u - (1 + r)}{u - d})$$

define uma distribuição de probabilidades. O seguinte resultado, muito simples de provar, na verdade revela-se essencial no tratamento de derivados financeiros com uma abordagem probabilística, permitindo usar resultados referentes a martingalas.

Proposição 12. *O processo $\{\frac{S_t}{(1+r)^t}, t \leq T\}$ é uma martingala em relação à distribuição de probabilidades Q.*

Demonstração. Claramente que $\{\frac{S_t}{(1+r)^t}, t \leq T\}$ é adaptado à σ-álgebra gerada por $\{Z_t, t \leq T\}$ (onde Z_t, tal como definido anteriormente, toma os valores u ou d, com probabilidades q_u e $q_d = 1 - q_u$, sob a medida Q, respetivamente), com $\mathbb{E}\left[\left|\frac{S_t}{(1+r)^t}\right|\right] \leq S_0 \left(\frac{u}{1+r}\right)^t < \infty$, uma vez que $S_t \leq S_0 u^t$

com probabilidade 1. Por isso resta apenas verificar a propriedade do valor esperado condicionado.

$$\begin{aligned}\mathbb{E}^Q\left[\frac{S_{t+1}}{(1+r)^{t+1}}|\mathcal{F}_t\right] &= \mathbb{E}^Q\left[\frac{S_t}{(1+r)^t}\frac{Z_{t+1}}{1+r}|\mathcal{F}_t\right] \\ &= \frac{S_t}{(1+r)^t}\mathbb{E}^Q\left[\frac{Z_{t+1}}{1+r}|\mathcal{F}_t\right] = \frac{S_t}{(1+r)^t}\mathbb{E}^Q\left[\frac{Z_{t+1}}{1+r}\right] \\ &= \frac{S_t}{(1+r)^t}\left(\frac{u}{1+r}\frac{1+r-d}{(u-d)} + \frac{d}{1+r}\frac{u-(1+r)}{u-d}\right) \\ &= \frac{S_t}{(1+r)^t}\end{aligned}$$

onde $\mathbb{E}^Q$ designa o valor esperado em relação à distribuição de probabilidades Q. □

A distribuição de probabilidades Q é designada por *medida de martingala* ou *medida de risco neutro*. E nas condições enunciadas, esta medida existe e é única.

Logo a seguinte relação é válida:

$$\frac{S_t}{(1+r)^t} = \mathbb{E}^Q\left[\frac{S_{t+1}}{(1+r)^{t+1}}|\mathcal{F}_t\right]. \tag{3.14}$$

Esta relação tem uma interpretação muito clara: assumindo a medida de probabilidade Q, o valor esperado descontado do preço do ativo do próximo instante é igual ao valor descontado do preço atual.

Mas a propriedade de martingala estende-se também ao próprio derivado financeiro, X, como a seguir se refere.

Proposição 13. *Seja N o número de passos de um modelo binomial. Então o valor descontado de um derivado financeiro X, $\left\{\frac{\Pi(n;X)}{(1+r)^n}, n \leq N\right\}$ é uma martingala, para a qual:*

$$\frac{\Pi(n;X)}{(1+r)^n} = \mathbb{E}^Q\left[\frac{\Pi(n+1;X)}{(1+r)^{n+1}}|\mathcal{F}_n\right]$$

pelo que, em particular, o preço do derivado X no instante n é dado por:

$$\Pi(n;X) = \frac{1}{(1+r)^{N-n}}\mathbb{E}^Q[g(S_N)|S_0] \tag{3.15}$$

onde g designa a função contrato e S_N é o valor do ativo subjacente na maturidade.

Corolário 1. *Tendo em conta a dinâmica assumida para o processo do preço do ativo subjacente, que, sob a medida de probabilidade $Q = (q_u, q_d)$, é tal que $P(S_{n+1} = S_n u) = q_u$ e $P(S_{n+1} = S_n d) = q_d$, o preço de um derivado X no instante inicial, quando o valor do ativo subjacente é S_0, é igual a:*

$$\Pi(0; X) = \frac{1}{(1+r)^N} \sum_{k=0}^{N} \binom{N}{k} q_u^k (1-q_u)^{N-k} g\left(S_0 u^k d^{N-k}\right). \tag{3.16}$$

Exemplo 3.3. *Considere-se o exemplo anterior mas agora supondo que se trata de uma opção de compra europeia.*

Para esta situação, a distribuição de probabilidades de martingala é igual a $Q = (\frac{1,01-0,8}{1,25-0,8} = 0,4667; 1-0,4667)$, pelo que

$$\begin{aligned}
\Pi(0; X) &= \left(\frac{1}{1,01}\right)^3 \mathbb{E}^Q[g(S_3)] \\
&= \frac{95,31225 \times 0,4667^3 + 25 \times 3 \times 0,4667^2 \times (1-0,4667)}{1.01^3} \\
&= 17,859
\end{aligned}$$

Capítulo 4

Modelo de Black-Scholes

4.1 Introdução

Neste capítulo abordamos o modelo de Black-Scholes. É um modelo matemático de mercados de derivados financeiros, no qual a fórmula de Black-Scholes é utilizada para determinar o preço correspondente. Esta fórmula permite derivar os preços das opções de compra e venda cuja função contrato seja da forma:

$$X = g(S(T)) \tag{4.1}$$

onde, à semelhança do assumido nos capítulos anteriores, X designa o retorno da opção, g é a função contrato e $S(T)$ o valor do ativo subjacente na maturidade, T. Contratos da forma 4.1 são denominados por *contratos simples*.

Ao contrário da notação usada no capítulo anterior, usaremos

$$\{S(t), t \leq T\}$$

onde t designa o instante temporal, aqui assumido contínuo. O processo estocástico relativo ao preço do ativo subjacente decorre em tempo contínuo.

A fórmula de Black-Scholes foi a primeira fórmula matemática amplamente adotada para determinação do preço de opções. Antes desta fórmula, a transação de opções não usava métodos matemáticos consistentes para avaliar as opções. Porém a análise empírica mostrou que as estimativas de preço produzidas por essa fórmula estão próximas dos preços observados.

Na sua formulação inicial, Fischer Black e Myron Scholes (os economistas que originalmente formularam o modelo) apresentaram uma equação diferencial parcial conhecida como a equação de Black-Scholes Black e Scholes (1973). A derivação desta equação segue princípios intrinsecamente associados à noção de arbitragem e de portfólios de réplica. Mais tarde Robert Merton publicou um artigo que mostra uma abordagem do mesmo resultado mas baseada em cálculo estocástico. Tanto Myron Scholes quanto Robert Merton dividiram o Prémio Nobel de 1997 em Economia. Uma vez que Fisher Black já tinha falecido na altura, e não sendo possível atribuir prémios Nobel a título póstumo, a academia sueca mencionou Fisher Black para a contribuição do resultado.

Como veremos em mais detalhe nas próximas secções, este modelo determina o preço de uma opção (europeia) calculando o retorno do investimento, subtraindo o valor do investimento, e assumindo que o preço do ativo subjacente segue uma distribuição log-normal.

Este resultado ajudou a legitimar o comércio de opções, fazendo com que parecesse menos como jogo e mais como ciência. Hoje, a fórmula Black-Scholes é ainda amplamente usada. Uma vez que as condições de admissibilidade da fórmula são hoje em dia raramente satisfeitas, tem havido nas últimas décadas muito trabalho de investigação, quer a nível empírico quer a nível analítico, de forma a estender este tipo de análise a condições do mercado mais realistas.

Uma grande parte das ferramentas para entender a derivação da fórmula é o cálculo estocástico. Para se entender na sua plenitude o resultado, é essencial saber as propriedades do movimento Browniano, o que são equações diferenciais estocásticas e difusões, a definição e propriedades do integral em relação ao movimento Browniano (como o integral de Itô), a fórmula de Feynman-Kacs, e o teorema de Radon-Nikodym, por exemplo. Mas tal não é esse o objectivo deste capítulo e nem o do presente livro, pelo que se optará por uma versão mais *aligeirada* da fórmula de Black-Scholes. Caso o leitor pretenda entender melhor a formulação estocástica do resultado, recomenda-se a leitura, por exemplo, de Björk (2009).

4.2 Notação e hipóteses

No modelo de Black-Scholes há um conjunto de hipóteses necessárias para a sua aplicabilidade, sendo que algumas são menos realistas que outras. De seguida apresentam-se estas hipóteses, com uma discussão parcial sobre a sua

validade em mercados financeiros e até que ponto podem ser relaxadas de forma a que a fórmula de Black-Scholes ainda seja válida.

- *O preço do ativo subjacente segue uma distribuição log-normal*: na sua essência mais simples, o modelo de Black-Scholes assume que o mercado consiste em dois produtos financeiros: um produto sem risco (obrigação ou investimento em banco, por exemplo) e um produto com risco, que representa o ativo subjacente (ações de uma empresa, por exemplo).

  O preço do ativo sem risco, $B = \{B(t), t \geq 0\}$, tem a seguinte dinâmica:

  $$dB(t) = r(t)dB(t) \tag{4.2}$$

  onde $\{r(t), t \geq 0\}$ é um processo determinístico, função conhecida do tempo. No caso de ser constante, o modelo tem uma análise mais simples, sendo esse precisamente o caso aqui apresentado:

  $$r(t) = r, \quad \forall t. \tag{4.3}$$

  Nesse caso r é designada por *taxa de juro sem risco* (*risk-free interest rate*). No Capítulo 2 já abordámos o processo das taxas de juro, mencionando alguns modelos para o caso estocástico.

  Embora doravante se assuma o caso mais simples 4.3, é possível estender o modelo ao caso em que $\{r(t), t \geq 0\}$ é um processo previsível. Recordamos a sua definição no Capítulo 3 para processos em tempo discreto. A caracterização de processos estocásticos previsíveis em tempo contínuo é mais complexa. Informalmente um processo previsível é um processo estocástico cujo valor é conhecido num momento anterior. Os processos previsíveis formam a menor classe que é fechada para limites de sucessões e contém todos os processos adaptados contínuos à esquerda.

  Regressando às condições aqui assumidas no modelo de Black-Scholes, assume-se ainda que a taxa de juro relativa a empréstimos é igual à taxa de juro relativa a depósitos.

  A solução da equação 4.2 é:

  $$B(t) = B(0)e^{\int_0^t r(s)ds}$$

  O preço do ativo com risco, $S = \{S(t), t \geq 0\}$, é solução da seguinte equação diferencial estocástica:

  $$dS(t) = \mu S(t)dt + \sigma S(t)dW(t)$$

na qual $W = \{W(t), t \geq 0\}$ é o movimento Browniano padrão, e μ e σ são constantes. Esta equação é sobejamente conhecida, e simples aplicação das regras fundamentais do cálculo estocástico (em particular da fórmula de Itô, Shiryaev (1999)) resultam na seguinte solução da equação:

$$S(t) = S(0)e^{(\mu - 0.5\sigma^2)t + \sigma W(t)} \tag{4.4}$$

pelo que $S(t)$ tem distribuição log-normal, com parâmetros:

$$\mathbb{E}[S(t)] = S(0)e^{\mu t}; \quad Var[S(t)] = S^2(0)e^{2\mu t}\left(e^{\sigma^2 t} - 1\right) \tag{4.5}$$

Na equação 4.4, o parâmetro σ é designado por *volatilidade*. Quanto maior σ, maior será a variabilidade do preço do ativo subjacente. No caso de σ não ser constante mas sim ser uma função determinística do tempo, i.e., no caso de S ser solução da equação diferencial estocástica:

$$dS(t) = \mu S(t)dt + \sigma(t)S(t)dW(t)$$

é ainda possível aplicar o modelo de Black-Scholes.Se σ depender também do preço do ativo (i.e., $\sigma(t, S(t))$) é ainda possível algum tratamento analítico mas não é possível determinar uma solução explícita, pelo que será necessário recorrer a métodos numéricos para encontrar o preço de uma opção europeia. No contexto deste livro, assumimos que o preço do ativo subjacente segue a lei 4.4.

O parâmetro μ da equação 4.4 designa-se por *tendência* (drift), e impacta quer no valor esperado quer na variância de $S(t)$. Porém, como se verá mais tarde, não tem qualquer influência no valor do preço de uma opção.

- *Dividendos*: Não há atribuição de dividendos relativos ao ativo subjacente. Caso esta hipótese não seja verificada, é possível ainda desenvolver uma fórmula para o caso em que existe atribuição de dividendos ou de forma contínua ou de forma programada, em instantes específicos. Porém apenas é possível encontrar uma fórmula fechada no caso dos dividendos serem uma proporção conhecida do preço do ativo subjacente.

- *Cobertura contínua de risco*: na avaliação de um portfólio, uma das medidas avaliadas é a sensibilidade do valor do portfólio ao preço do ativo subjacente. Se $V^h(t, s)$ designar o valor portfólio no instante t

quando o preço do ativo subjacente é s, então define-se a seguinte quantidade:

$$\Delta = \frac{\partial V^h}{\partial s}$$

usualmente designada por *Delta*. O Delta é pois uma medida da exposição ao risco do portfólio (que pode ser constituído por ações e derivados) às alterações dos preços das ações. Se $\Delta = 0$, então o portfólio é classificado como *Delta neutro*.

Esta medida é uma das medidas usualmente designadas por *Gregos*; no contexto presente, apenas nos focaremos no Delta, mas a título informativo referimos também que existem outras, como *Vega*, *Theta*, *Rho* e *Gamma* (vide Hull e White (1990)).

Quando ao portfólio é adicionada uma opção, é possível encontrar a proporção *ideal* de número de opções que leva a um Delta nulo. Assim, suponhamos que ao portfólio, de valor $V^h(t,s)$, são adicionadas x opções do mesmo tipo, com ativo subjacente igual ao inicialmente considerado no portfólio, com valor $\Pi(t;s)$. Então o valor ajustado deste novo portfólio, $V(t,s)$, é igual a:

$$V(t,s) = V^h(t,s) + x\Pi(t;s)$$

pelo que para que o Delta seja nulo, teremos que adicionar ao portfólio o seguinte número de opções:

$$x = -\frac{\frac{\partial V^h}{\partial s}}{\frac{\partial \Pi}{\partial s}}. \tag{4.6}$$

A esta estratégia dá-se o nome de *cobertura de risco* (*Delta hedging*).

Na derivação da fórmula de Black-Scholes que se apresenta neste capítulo (e que assenta no conceito de portfólios de réplica) é assumido que é possível efetuar cobertura de risco em tempo contínuo. Isto é, assume-se que seja possível controlar o portfólio, comprando ou vendendo ativos e ou opções, de forma a que o valor Delta seja nulo, pelo que em cada instante o número de opções detidas, dado o valor do ativo subjacente, é dado por 4.6. De facto tal não é possível, pois as transações inerentes à compra e venda de opções e de ações decorrem em tempo discreto, e sendo o modelo de Black-Scholes em tempo contínuo, esta hipótese significa que a cobertura de risco deve ser feita em tempo contínuo.

- *Custos de transação*: o modelo aqui apresentado assume que não há custos de transação, i.e., a compra e venda de produtos é isentada de qualquer comissão de venda. Esta hipótese não é realista, e o seu impacto é especialmente importante quando se admite que há venda e compra de ativos e opções ao longo do tempo para balancear o risco (como explicado no ponto anterior, sobre a cobertura de risco).

- *Arbitragem*: assume-se que arbitragem não é possível. Porém, embora legalmente tal não seja previsto, há situações que escapam na malha e que prefiguram hipóteses de arbitragem. Investidores atentos podem comprar um ativo num mercado, e aproveitando-se de taxas de câmbio ou mesmo de deficiente informação entre os mercados, vender quase imediatamente noutro mercado, por um preço superior.

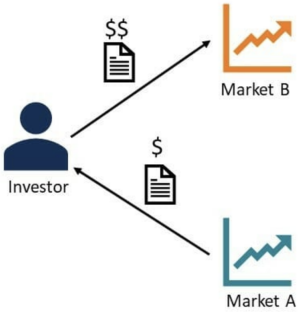


Figura 4.1: Embora se assuma que arbitragem não é possível, investidores com acesso a tecnologia podem ter hipóteses de arbitragem ao negociarem em mercados distintos, com pequenas frações de tempo entre transações

  Se os mercados estivessem todos perfeitamente sincronizados, no sentido que todos os bens transacionados teriam exatamente o mesmo preço, não haveria possibilidade de arbitragem. Este conceito equivale ao conceito de mercados perfeitamente eficientes.

- *Mercados líquidos*, i.e., é possível comprar ou vender um número arbitrário de ações (incluindo frações).

Apresentadas e brevemente discutidas as hipóteses subjacentes ao modelo de Black-Scholes, apresentamos na secção seguinte a derivação do preço de uma opção europeia.

4.3 Derivação do preço sob ausência de arbitragem

Seguindo a notação e hipóteses anteriormente introduzidas, consideremos um mercado com dois ativos (um sem risco e outro com risco):

$$\begin{aligned} dB(t) &= rB(t)dt \\ dS(t) &= \mu S(t)dt + \sigma S(t)dW(t) \end{aligned}$$

e um derivado financeiro para o qual a função contrato é da forma:

$$X = g(S(T))$$

para o qual o preço é $\Pi = \{\Pi(t; X), t \leq T\}$, onde T é a maturidade do contrato. Para facilitar a notação, usamos apenas $\Pi(t)$. Adicionalmente, o preço deve ser uma função não só do tempo t, mas também do preço do ativo subjacente, $S(t)$, pelo que Π deve ser da forma:

$$\Pi(t) = F(t, S(t))$$

onde F é uma função a determinar. Decorre da fórmula de Itô (Shiryaev, 1999) que:

$$d\Pi(t) = dF(t, S(t)) = \mu_\Pi(t)\Pi(t)dt + \sigma_\Pi(t)\Pi(t)dW(t)$$

onde

$$\mu_\Pi(t) = \frac{F_t + \mu S F_S + \frac{1}{2}\sigma^2 S^2 F_{SS}}{F} \tag{4.7}$$

$$\sigma_\Pi(t) = \frac{\sigma S F_S}{F} \tag{4.8}$$

Nesta fórmula, F_t e F_S representam as derivadas parciais de F em relação a t e a S, respetivamente, e F_{SS} a segunda derivada parcial de F em relação S, assumindo que estas derivadas existem[1]. Para aliviar a notação, omite-se ainda os argumentos das funções. Por exemplo

$$\frac{\sigma S F_S}{F} = \frac{\sigma S(t) F_S(t, S(t))}{F(t, S(t))}$$

[1]Embora S seja uma v.a., no cálculo das derivadas parciais F_S e F_{SS} utilizamos as regras usuais de derivação.

Suponhamos agora que temos um portfólio h formado por um ativo subjacente e uma opção, com

$$h = \{(u_S(t), u_\Pi(t)), t \leq T\}$$

onde $u_S(t)$ e $u_\Pi(t))$ designam a proporção relativa de ações e opções no instante t, respectivamente, com

$$u_S(t) + u_\Pi(t) = 1, \forall t. \tag{4.9}$$

Admitindo que o portfólio h é auto-financiado, então o seu valor, V^h, em face de 3.2, obedece à seguinte dinâmica:

$$\begin{aligned} dV^h(t) &= V^h(t)\left(u_S(t)\frac{dS(t)}{S(t)} + u_\Pi(t)\frac{d\Pi(t)}{\Pi(t)}\right) \\ &= V^h(t)\left[u_S(t)\left(\mu dt + \sigma dW(t)\right) + u_\Pi(t)\left(\mu_\Pi + \sigma_\Pi dW(t)\right)\right] \\ &= V^h(t)\left[u_S(t)\mu + u_\Pi(t)\mu_\Pi\right]dt + \\ &+ V^h(t)\left[u_S(t)\sigma + u_\Pi(t)\sigma_\Pi\right]dW(t) \end{aligned} \tag{4.10}$$

Note-se que os dois fatores que estão entre parentesis retos são lineares quer em u_S quer em u_Π.

Assumindo que este portfólio não tem risco, então teremos que o fator relativo a $dW(t)$ é nulo:

$$u_S(t)\sigma + u_\Pi(t)\sigma_\Pi = 0 \tag{4.11}$$

pelo que o valor do portfólio é:

$$dV^h(t) = V^h(t)\left[u_S\mu + u_\Pi\mu_\Pi\right]dt$$

Adicionalmente, de forma a que não haja arbitragem, a taxa de retorno do portfólio tem de ser igual à taxa de juro, pelo que:

$$u_S\mu + u_\Pi\mu_\Pi = r \tag{4.12}$$

Resolvendo as equações 4.9 e 4.11, obtêm-se os seguintes valores para a proporção de ações e opções no portfólio em cada instante t:

$$u_S(t) = \frac{\sigma_\Pi(t)}{\sigma_\Pi(t) - \sigma} = \frac{S(t)F_S(t, S(t))}{S(t)F_S(t, S(t)) - F(t, S(t))} \tag{4.13}$$

$$u_\Pi(t) = \frac{-\sigma}{\sigma_\Pi(t) - \sigma} = \frac{-F(t, S(t))}{S(t)F_S(t, S(t)) - F(t, S(t))} \tag{4.14}$$

Usando as equações 4.7, 4.13 e 4.14 em 4.12, obtemos a seguinte equação diferencial:

$$F_t(t,S(t))+rS(t)F_S(t,S(t))+ \\ + \frac{1}{2}\sigma^2 S^2(t)F_{SS}(t,S(t)) - rF(t,S(t)) = 0, \quad t \leq T. \tag{4.15}$$

Em resultado da condição terminal

$$\Pi(T) = g(S(T))$$

e por ter de ser válida para qualquer valor de t e de $S(t)$, temos o seguinte resultado.

Proposição 14. *Nas condições atrás enunciadas, o preço de um derivado X com função contrato $g(S(T))$ é solução da seguinte equação diferencial parcial com valor terminal:*

$$F_t(t,s) + rsF_s(t,s) + \frac{1}{2}\sigma^2 s^2 F_{ss}(t,s) - rF(t,s) = 0 \tag{4.16}$$

$$F(T,s) = g(s) \tag{4.17}$$

A equação 4.16, com a condição terminal 4.17, é a famosa **equação de Black-Scholes**.

Note-se que em face desta equação, que não depende de μ (o parâmetro da tendência do preço do ativo subjacente), se conclui que o preço de uma opção é indiferente à tendência μ. Resultado semelhante ocorre no modelo binomial, uma vez que a probabilidade atribuída à subida de preço não intervém no preço da opção.

Re-escrevendo a equação 4.16:

$$F_t(t,s) + \frac{1}{2}\sigma^2 s^2 F_{ss}(t,s) = rF(t,s) - rsF_s(t,s) \tag{4.18}$$

podemos interpretar a fórmula da seguinte forma:

- O lado esquerdo da fórmula representa a alteração no preço da opção devido a dois fatores: ao tempo e à convexidade (segunda derivada) do valor do preço em relação ao valor do ativo subjacente;
- O lado direito representa o retorno de uma posição longa na opção e uma posição curta de $F_s(t,s)$ ações.

A unicidade de solução da equação 4.16 decorre da não-arbitragem: qualquer opção tem de ter um preço único, caso contrário existe possibilidade de arbitragem. Mas para garantir a unicidade da solução, é necessário impôr um número suficiente de condições fronteira. A equação 4.16 é uma equação parabólica, regressiva. Daí a necessidade de impor uma condição fronteira terminal, que surge naturalmente por argumentos de não-arbitragem: $F(T,s) = g(s)$. Mas porque se trata de uma equação de segunda ordem em s e de primeira ordem em t, é necessário impôr mais restrições.

Mas estas condições dependem intrinsecamente da opção que se está a analisar. Por exemplo, se for uma opção de compra, para a qual:

$$g(S(T)) = \max(S(T) - K, 0) \tag{4.19}$$

então a primeira restrição é natural:

$$F(t,0) = 0, \forall t$$

i.e., o preço de uma opção para a qual o ativo subjacente vale zero tem de ser nulo (como S segue um movimento geométrico Browniano, se o preço alguma vez atingir o valor zero, então ficará para sempre igual a zero, pelo que a opção nunca será exercida para $K > 0$).

Por outro lado, quando $s \to \infty$, a opção deve ser exercida seguramente, pelo que o valor desta opção deverá ser da mesma ordem que o retorno da mesma, i.e.,

$$F(t,s) \sim s - Ke^{-r(T-t)}, \quad s \to \infty$$

Pelo contrário, se a opção for de venda, para a qual

$$g(S(T)) = \max(K - S(T), 0) \tag{4.20}$$

então se o valor do ativo subjacente for nulo, a opção será necessariamente exercida, pelo que neste caso:

$$F(t,0) = Ke^{-r(T-t)}$$

Pelo contrário, se o preço do ativo for muito elevado, então a opção não será exercida, pelo que

$$F(t,s) \to 0, \quad s \to \infty$$

É possível encontrar a solução da equação de Black-Scholes para alguns casos particulares da função contrato. É o caso das opções europeias de compra ou venda, para as quais as funções contrato são dadas por 4.19 e 4.20, respetivamente.

Proposição 15. *O preço de uma opção de compra europeia com preço de exercício K e maturidade T, no instante t quando o preço do ativo subjacente é s é dado por:*

$$F(t,s) = s\Phi\left(d_1(t,s)\right) - e^{-r(T-t)}K\Phi\left(d_2(t,s)\right) \tag{4.21}$$

onde Φ designa a função de distribuição de uma normal reduzida, e:

$$d_1(t,s) = \frac{1}{\sigma\sqrt{T-t}}\left\{\ln\left(\frac{s}{K}\right) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)\right\} \tag{4.22}$$

$$d_2(t,s) = d_1(t,s) - \sigma\sqrt{T-t} \tag{4.23}$$

No caso de ser uma opção de venda, o preço é dado por

$$F(t,s) = Ke^{-r(T-t)}\Phi\left(-d_2(t,s)\right) - s\Phi\left(-d_1(t,s)\right) \tag{4.24}$$

Omite-se a prova desta proposição, uma vez que pode ser facilmente encontrada na literatura, como por exemplo em Björk (2009). Trata-se de um exercício simples mas tedioso: usar as propriedades da distribuição normal e calcular as derivadas de F dado por 4.21 ou por 4.24 para provar que efetivamente a equação 4.16 é satisfeita.

No caso da opção não ser uma opção de compra ou de venda, como determinar o seu preço? Nesse caso teríamos de encontrar suficientes condições de fronteira, à semelhança do que foi feito nas opções de compra e de venda, de forma a encontrar a solução única da equação diferencial.

O resultado que a seguir se apresenta é essencial para determinar uma fórmula geral para a solução de 4.16 para qualquer opção para a qual o contrato seja da forma $g(S(T))$.

Lema 4.1. *Dadas funções (determinísticas) $\mu(t,x)$, $\sigma(t,x)$ e $g(x)$, a solução do problema:*

$$\begin{aligned} \frac{\partial F}{\partial t}(t,x) + \mu(t,x)\frac{\partial F}{\partial x}(t,x) + \frac{1}{2}\sigma^2(t,x)\frac{\partial F^2}{\partial x^2}(t,x) &= 0 \\ F(T,x) &= g(x) \end{aligned} \tag{4.25}$$

admite a seguinte representação:

$$F(t,x) = \mathbb{E}[g(X(T))|X(t) = x] \tag{4.26}$$

onde X satisfaz a seguinte equação diferencial estocástica:

$$dX(s) = \mu(s, X(s))ds + \sigma(s, X(s))dW(s) \tag{4.27}$$

com $X(t) = x$*, e tal que* $\int_0^t \mathbb{E}[X^2(s)]ds < \infty$, $\forall t > 0$.

Nas mesmas condições, a solução do problema:

$$\frac{\partial F}{\partial t}(t,x) + \mu(t,x)\frac{\partial F}{\partial x}(t,x) + \frac{1}{2}\sigma^2(t,x)\frac{\partial F^2}{\partial x^2}(t,x) = rF(t,x) \qquad (4.28)$$
$$F(T,x) = g(x)$$

admite a seguinte representação:

$$F(t,x) = e^{-r(T-t)}\mathbb{E}[g(X(T))|X(t) = x]. \qquad (4.29)$$

Este lema é usualmente conhecido na literatura por *representação estocástica de Feynman-Kac* e tem ampla utilização no âmbito de aplicações financeiras, como é o caso da determinação do preço de uma opção.

Demonstração. A demonstração assenta essencialmente na fórmula de Itô: se X for solução da equação diferencial estocástica 4.27, e $F : \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}$ uma função de classe $C^1 \times C^2$, então

$$dF(t,X(t)) = [F_t(t,X(t)) + \mu(t,X(t))F_X(t,X(t)) + \frac{1}{2}\sigma^2(t,X(t))F_{XX}(t,X(t))]dt + \sigma(t,X(t))F_X(t,X(t))dW(t)$$

onde F_t, F_X e F_{XX} designam as derivadas parciais (à semelhança do definido em 4.13). Usando o facto de que F é solução da equação diferencial parcial 4.25, vem que:

$$dF(t,X(t)) = \sigma(t,X(t))F_X(t,X(t))dW(t)$$

Integrando e aplicando o valor esperado (condicionado em $X(t) = x$) na relação anterior. vem:

$$\mathbb{E}[F(T,X(T))|X(t) = x] - F(t,x) = 0$$

pois o valor esperado do integral de Itô é nulo (*primeira isometria de Itô*), pelo que, usando a condição fronteira, vem que

$$F(t,x) = \mathbb{E}[g(X(T))|X(t) = x]$$

A demonstração de 4.28 segue o mesmo tipo de argumento. □

Usando este resultado, temos a seguinte fórmula para o preço de um derivado com função contrato g:

Proposição 16. *O preço de um derivado cujo valor na maturidade é $g(S(T))$ no instante t, quando o valor do ativo subjacente é $S(t) = s$, $F(t, s)$ é igual a:*

$$F(t,s) = e^{-r(T-t)} E^Q[g(S(T))|S(t) = s]$$

onde, sob a distribuição de probabilidades Q, S é solução da seguinte equação diferencial estocástica:

$$dS(t) = rS(t)dt + \sigma S(t)dW(t)$$

A demonstração deste resultado pode ser encontrada, por exemplo, em Mikosch (1998). Na sua demonstração são utilizados resultados sobre mudança de distribuições de probabilidade (a designada derivada de Radon-Nikodym) e o teorema de Girsanov (que justifica a *substituição* do parâmetro de difusão μ por r, a taxa de juro).

Esta fórmula tem uma interpretação muito clara, do ponto de vista económico: o preço de uma opção no momento atual t é igual ao valor esperado do retorno na maturidade, descontado ao momento presente. Este valor esperado é tomado não em relação à medida de probabilidade P (sob a qual o preço do ativo subjacente é um movimento geométrico Browniano com parâmetro de difusão μ) mas sim em relação à medida de probabilidade Q, sob a qual o preço do ativo subjacente é um movimento geométrico Browniano com parâmetro de difusão r. E sob esta medida de probabilidade o preço descontado do ativo subjacente é uma martingala, como já vimos anteriormente.

Exemplo 4.1. *O preço de uma ação transacionada na bolsa de valores de Lisboa segue um movimento geométrico Browniano, com volatilidade 0.2. O preço atual desta ação é 40 €. Uma empresa de investimentos está a vender uma opção, pelo preço 10 €, a qual dá um retorno de 100 € ao fim de um ano se o preço da ação ao fim de um ano for superior a $(1+x)40$. Assumindo que a taxa anual de juro é 0,02, qual é o valor de x para o qual não há hipótese de arbitragem?*

Para responder a esta questão, usamos o modelo de Black-Scholes, para o qual o preço deste derivado é igual a:

$$\begin{aligned}\pi(0) &= e^{-0,02}100 \times Q\left(40e^{0,2W(1)} > (1+x)40\right) \\ &= 98,0199Q\left(W(1) > (1+x)/0,2\right) \\ &= 10\end{aligned}$$

onde $\pi(0)$ designa o preço do derivado no momento atual, e Q a medida de probabilidades, sob a qual o preço da ação segue um movimento geométrico Browniano de tendência 0,02 (o valor anual da taxa de juro). Logo não existe hipótese de arbitragem de o valor de x for igual a:

$$98,0199\left(1-\Phi(\frac{1+x}{0,2})\right)=0,10$$

Capítulo 5

Volatilidade

5.1 Introdução

Nas últimas décadas tem-se observado alterações muito significativas na área das finanças, quer na academia, quer na indústria. Existem várias razões para esta mudança, mas a mais importante é seguramente a disponibilidade de dados de acesso rápido e a maior capacidade de processamento. É fácil hoje em dia aceder a dados, mesmo os de alta frequência (na ordem dos segundos), sejam eles de ações, moedas, taxas de juro, ou de outros mercados, como o das opções, energia ou ambiente. A existência dessa informação permite que os modelos teóricos possam ser testados mas também a construção de novos modelos que melhor se adequam à realidade.

Um dos conceitos importantes em finanças é o de volatilidade. Em linguagem comum, volátil é algo incerto ou inconstante e a volatilidade um termo que se usa para descrever flutuações observadas em algum fenómeno ao longo do tempo. Em economia a volatilidade está associada à variabilidade de uma componente aleatória de uma série temporal. A volatilidade é, inerentemente, uma componente latente e estocástica, que evolui ao longo do tempo e constitui um parâmetro fundamental para os investidores que procuram investir em ativos de risco. A avaliação deste parâmetro é necessária para a execução da maioria das teorias económicas ou financeiras que norteiam os investimentos. A volatilidade é também importante para avaliar o desempenho dos mercados financeiros, uma vez que, em mercados muito voláteis o valor dos ativos pode

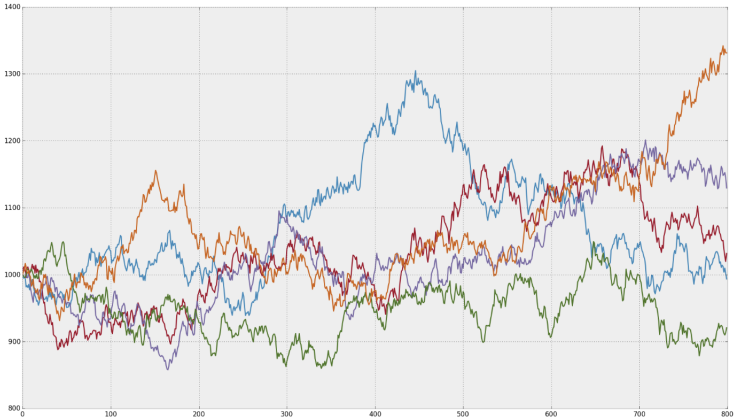


Figura 5.1: Ilustração da realização de diferentes caminhos de movimentos geométricos Brownianos, para diferentes valores de volatilidade

alterar-se drasticamente num curto espaço de tempo induzindo uma maior incerteza sobre os rendimentos.

Vimos anteriormente que o parâmetro de volatilidade, denotado por σ, é uma medida da incerteza em relação à evolução do preço do ativo, pelo que desempenha um papel fundamental na avaliação dos investimentos de risco.

Na Figura 5.1 apresenta-se a simulação de vários movimentos geométricos Brownianos com igual valor inicial, parâmetro de difusão mas diferente volatilidade.

Os mercados são especialmente atrativos para investidores quando a volatilidade é elevada. Para analisar o efeito da volatilidade, é importante considerar a maturidade, ou seja:

- Investimentos a curto prazo (i.e., maturidade próxima): o impacto da volatilidade depende da posição do investidor. Por exemplo, se se investir numa combinação de opções do tipo *straddle* (uma combinação de uma opção de compra e de uma opção de venda com o mesmo preço de exercício), então haverá lucro no caso do preço do ativo subjacente se alterar (quer para valores superiores quer para valores inferiores), pelo que neste caso uma volatilidade elevada potencia ganhos, enquanto que uma volatilidade baixa leva a perdas do investimento. Se, pelo contrário, vendermos uma *straddle* então volatilidades elevadas são sinónimo de perda

Figura 5.2: Notícia do *Economist*, do dia 11 de Maio de 2019

de valor.

- Investimentos a longo prazo (i.e., longas maturidades): neste caso a volatilidade é geralmente necessária para que haja algum retorno positivo do investimento. Uma volatilidade baixa significa que não há alteração significativa do preço do ativo, pelo que não há oportunidades relevantes de retornos interessantes.

Assim, embora volatilidade elevada potencie a probabilidade de perdas, também potencia a probabilidade de ganhos. E quando se tem uma posição longa numa opção de compra (i.e., quando se é o comprador), as perdas são limitadas, mas os ganhos são ilimitados, o que torna o produto especialmente interessante, Figura 5.2.

Sendo a volatilidade um parâmetro tão relevante, é natural que se questione como se procede à sua estimação.

Admitindo que a volatilidade é constante, podemos utilizar os estimadores usuais do desvio padrão. Por exemplo, tal como sugerido por Hull (2003), dado uma sequência de observações relativas ao preço de uma ação, $\{S_t, t \in \{1, 2 \ldots, n\}\}$, podemos calcular a série do logaritmo dos retornos, $\{u_t, t \in \{1, 2 \ldots, n\}\}$, com:

$$r_t = \log\left(\frac{S_t}{S_{t-1}}\right). \tag{5.1}$$

Nesse caso um estimativa do desvio padrão, $\widehat{\sigma}$, pode ser, por exemplo:

$$\widehat{\sigma} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(r_i - \bar{r})^2}$$

com $\bar{r} = \frac{1}{n}\sum_{i=1}^{n} r_i$.

Com este tipo de fórmulas simples, podemos hipoteticamente estimar a volatilidade em qualquer escala temporal. Por exemplo, se τ for o intervalo de tempo considerado, em anos, então:

$$\frac{\widehat{\sigma}}{\sqrt{\tau}}$$

é a estimativa da volatilidade anual. Se os dados relativos ao preço do ativo forem mensais, então $\tau = \frac{1}{12}$.

Mas esta forma de estimar volatilidades baseia-se no pressuposto que a volatilidade mantém-se constante em pelo menos algum período de tempo. Observando dados relativos a preços de ações, apercebemo-nos que os dados, em geral, não são compatíveis com esta hipótese.

Na Figura 5.3 apresentamos os preços diários das obrigações emitidas pelo estado Alemão, para uma maturidade de 10 anos. A amostra representa o logaritmo dos preços diários consecutivos (tal como definido em (5.1), relativo a preços de 3 de janeiro de 2000 até 30 dezembro de 2011 (painel da esquerda). No painel da direita representa-se a raiz quadrada destes mesmos valores.

Uma simples obervação deste gráfico indicia desde logo que a volatilidade começou a aumentar durante o período correspondente à última crise financeira de 2008/2009. A partir de 2010 a volatilidade aumentou ainda mais, evidenciando a crise da dívida pública da zona Euro.

Este exemplo é um de muitos que se podem encontrar quer na literatura sobre volatilidade em mercados financeiros, quer em notícias de jornais generalistas ou da especialidade.

Apesar da importância da volatilidade, esta não é fácil de medir, e existem muitas abordagens para fazer isso. Inúmeros trabalhos, quer académicos, quer na indústria se têm dedicado à avaliação da volatilidade uma vez que enquadrar qual o melhor modelo que se pode utilizar para descrever a evolução da volatilidade é uma tarefa bastante complexa.

De seguida apresentamos uma breve incursão à estimação da volatilidade. Incia-se pela volatilidade implícita, de seguida apresentam-se alguns estima-

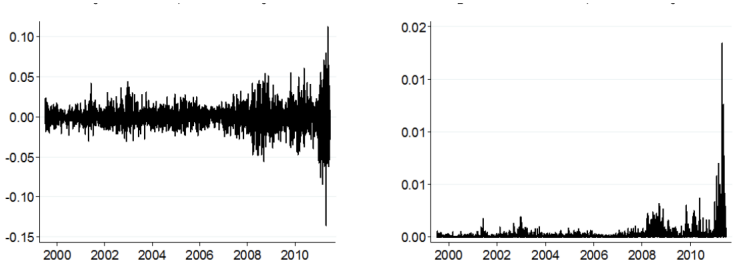


Figura 5.3: Logaritmo dos preços diários consecutivos relativo a preços de 3 de janeiro de 2000 até 30 dezembro de 2011 (painel da esquerda)

dores de volatilidade baseados em valores históricos e, por fim, uma breve descrição de dois modelos de séries temporais que permitem modelar a volatilidade condicionada.

5.2 Volatilidade implícita

Na derivação da volatilidade implícita, a fórmula de Black-Scholes é *olhada* ao inverso. Em vez de se obter o preço do derivado com base nos parâmetros relativos à taxa de juro, volatilidade, maturidade e preço atual da ação, a fórmula é invertida, de forma a responder à questão:

Que volatilidade deve ser usada de forma a obter o preço de mercado da opção?

Este valor é usualmente designado por *volatilidade implícita*. Por outras palavras, a volatilidade implícita é o valor da volatilidade do preço do ativo subjacente que o comprador usa para obter o preço, usando a fórmula de Black-Scholes.

Recordando a fórmula de Black-Scholes para uma opção de compra usual, por exemplo, facilmente que se levanta a questão de como resolver, do ponto de vista analítico, a determinação da volatilidade implícita. A inversão da fórmula é analiticamente impossível (recordemos que envolve a função de distribuição de uma normal, por si só não calculável analiticamente), pelo que é necessário recorrer a métodos numéricos. Para tal, é necessário argumentar para um dado preço de uma opção, uma maturidade, um preço de exercício, taxa de juro e preço do ativo subjacente, que existe um valor único para a vo-

latilidade implícita. De facto tal é verdade quando há ausência de arbitragem. Em particular tal implica que as desigualdades apresentadas no Capítulo 2 são satisfeitas.

Exemplo 5.1. *O preço de uma opção de compra europeia é uma função crescente da volatilidade. De facto, sendo o preço de tal contrato dado por*

$$\Pi(\sigma; t, S_t) = S_t\Phi(d_1(\sigma)) - Ke^{-r(T-t)}\Phi(d_2)\sigma))$$

(onde usamos uma notação diferente da apresentada aquando da fórmula de Black-Scholes, para evidenciar a dependência na volatilidade σ) tem-se que

$$\begin{aligned}\frac{\partial\Pi(\sigma; t, S_t)}{\partial\sigma} &= S_t\phi(d_1(\sigma))\frac{\partial d_1(\sigma)}{\partial\sigma} - Ke^{-r(T-t)}\phi(d_2(\sigma))\frac{\partial d_2(\sigma)}{\partial\sigma} \\ &= \left(S_t\phi(d_1(\sigma)) - Ke^{-r(T-t)}\phi(d_2(\sigma))\right)\frac{\partial d_1(\sigma)}{\partial\sigma} \\ &+ Ke^{-r(T-t)}\phi(d_2(\sigma))\sqrt{T-t}\end{aligned}$$

onde ϕ designa a função densidade de probabilidade de uma normal padrão. Cálculos simples levam a que a seguinte relação seja válida:

$$S_t\phi(d_1(\sigma)) - Ke^{-r(T-t)}\phi(d_2(\sigma)) = 0$$

donde resulta que

$$\frac{\partial\Pi}{\partial\sigma} = Ke^{-r(T-t)}\phi(d_2)\sqrt{T-t} > 0 \tag{5.2}$$

Usando propriedades da função de distribuição, resulta ainda que:

$$\lim_{\sigma\to 0^+}\Pi(\sigma; t, S_t) = \max(0, S_t - Ke^{-r(T-t)}, \quad \lim_{\sigma\to 0^+}\Pi(\sigma; t, S_t) = S_t$$

o que está de acordo com o princípio de não arbitragem.

O Exemplo 5.1 mostra que existe uma relação bijectiva entre o preço de uma opção e a volatilidade correspondente.

A *praxis* mostra que os investidores estão frequentemente mais preocupados com a volatilidade do que com o próprio preço, porque sabem que o último é uma consequência do primeiro. Portanto, é essencial descobrir que nível implícito de volatilidade tem sido usado para determinar o preço de opções, a fim de determinar qual a estratégia de negociação que deve ser adotada.

Mas esta relação bijectiva apenas pode ser resolvida numericamente, em ambos os sentidos. Um dos métodos usados para determinar a volatilidade implícita é o método de Newton-Raphson (Wilmott, 2013), recorrendo à fórmula:

$$\sigma_{i+1} = \sigma_i - \frac{c(\sigma_i) - c_m}{\frac{\partial c}{\partial \sigma}}$$

onde

- σ_i: valor da volatilidade no passo i
- $c(\sigma_i)$: preço da opção calculado com a volatilidade igual a σ_i
- c_m: preço da opção no mercado
- $\frac{\partial c}{\partial \sigma}$ é a derivada do preço em relação à volatilidade (que, no caso de ser uma opção europeia, é dada pela Equação5.2.

Este método é muito rápido e, se as estimativas iniciais forem boas, conduz a valores com grande precisão. No entanto a sua precisão pode ser afetada pela precisão do cálculo das funções de distribuição da normal envolvidas na fórmula de Black-Scholes e, com maior impacto, no valor de $\frac{\partial c}{\partial \sigma}$ (usualmente designado por *Vega*). No caso de ser uma opção europeia de compra, como a ilustrada no Exemplo 5.1, uma vez que se consegue uma expressão analítica para esta derivada, ver Equação 5.2, o seu cálculo numérico não apresenta dificuldade assinalável. Mas no geral pode ser bem mais complicado, pelo que a vantagem do método de Newton-Raphson pode ser comprometida. Outros métodos numéricos têm sido propostos na literatura, entre eles o método da bisseção, este será o usado mais à frente, numa pequena aplicação, onde se ilustra o cálculo da volatilidade implícita.

Seja qual for o método numérico utilizado, ao inverter a fórmula do preço de uma opção de forma a obter a volatilidade correspondente, a volatilidade implícita varia consoante o valor de exercício e a maturidade, para o mesmo ativo subjacente. Consequentemente, a volatilidade implícita *não é constante*. De acordo com o modelo de Black-Scholes, tal não deveria acontecer. A volatilidade é uma propriedade do ativo, e por isso duas opções arbitrárias com o mesmo ativo subjacente deveriam ter o mesmo valor da volatilidade. Mas os dados dos mercado mostram precisamente o contrário.

Na Figura 5.4 ilustra-se tal propriedade. Os dados são relativos a opções com maturidade de 1 mês, no dia 3 de abril de 1996, transacionadas no mercado alemão. Os dados provêm de Campa *et al.* (1997). Em ordenada encon-

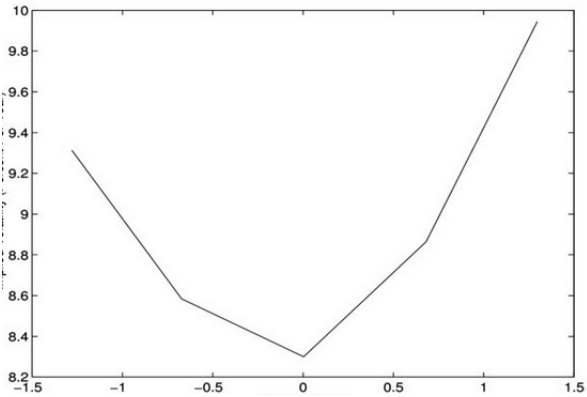


Figura 5.4: Ilustração da evolução da volatilidade face a diferentes valores de $\frac{K}{S_0}$

tramos o valor da volatilidade e em abcissa o valor de $\frac{K}{S_0}$. O gráfico exibe uma forma parabólica, que faz lembrar um sorriso. Por esse motivo é frequentemente denominado de *volatility smile*. Este exemplo mostra que a volatilidade é menor para opções cujo valor de exercício seja próximo do preço do ativo subjacente na altura do contrato. Pelo contrário, a volatilidade aumenta com preços de exercício elevados ou baixos.

A relação entre o preço de exercício e o preço do ativo subjacente é relevante e permite distinguir dois tipos de situações:

- Uma opção *Out The Money* (OTM) pode ser uma opção de compra cujo preço de exercício (*strike*) encontra-se acima do valor da cotação do ativo-objeto, ou uma opção de venda cujo preço de exercício (*strike*) encontra-se abaixo do valor da cotação do ativo-objeto. Este tipo de opções não apresentam valor intrínseco. Muito provavelmente, o titular de uma opção OTM não irá exercer o seu direito de compra ou de venda do ativo, pois tal condição de negociação é muito pior do que a compra ou venda deste ativo no mercado. Por esta razão, o valor para aquisição de uma opção OTM é muito baixo. Este tipo de produto de investimento é especialmente interessante no caso de especulações. Porém, pode acontecer que devido a grandes volatilidades do mercado, a opção venha a ser exercida, traduzindo-se então num retorno do investimento bastante assinalável. Como a Figura 5.4 ilustra, a opções OTM correspondem valores elevados de volatilidade.

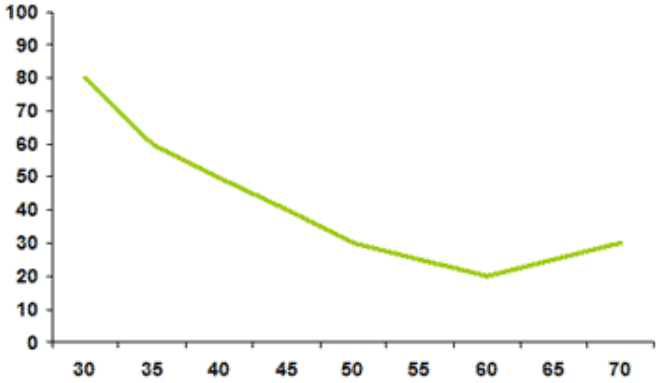


Figura 5.5: Comportamento da volatilidade em opções OTM com valor de exercício baixo

- Em contraponto, uma opção *At The Money* (ATM) o valor do exercício é igual ao do valor do ativo subjacente.

Este tipo de comportamento, que leva ao *volatility smile*, não se aplica a todas as opções. Por exemplo, no caso da volatilidade só ser elevada para opções OTM com valor de exercício baixo, teremos um comportamento do tipo que se apresenta na Figura 5.5. Os analistas económicos interpretam este tipo de comportamento como sendo a consequência do mercado, em geral, descer mais rapidamente que a sua subida.

No ambiente computacional R (R Core Team, 2019) não é difícil escrever uma função que permita calcular a volatilidade implícita. O código é apresentado na Figura 5.6. Os parâmetros `S`, `K` e `r` são os já denotados preço da opção, o preço de exercício da opção e a taxa de juro, respectivamente, que são usados na fórmula Black-Scholes. Adicionalmente, usamos a designação `sig` para representar a volatilidade, `TT` o tempo de vencimento (maturidade), `vmercado` o preço da opção no mercado e `tipo`, o tipo de opção, ou seja se é de compra `C`, do inglês *Call*, ou se é de venda `P`, do inglês *Put*.

O código R da Figura 5.7 ilustra uma aplicação das funções apresentadas na Figura 5.6. Na Figura 5.8 apresenta-se a mesma aplicação mas agora recorrendo ao pacote derivmkts (McDonald, 2019). A volatilidade implícita obtida, usando qualquer um desses códigos é o conjunto de valores $\{0,4954; 0,4791; 0,7316; 0,7122\}$.

Funções para o cálculo da volatilidade implícita

```
imp.vol.fun <-function(S, K, TT, r, vmercado, tipo,
 tol=0.00001, maxit=1000){
# valores iniciais
sig <- 0.1
sig.up <- 1
sig.down <- 0.001
count <- 0
err <- BS.fun(S, K, TT, r, sig, tipo) - vmercado
## Critério de paragem
while(abs(err) > tol && count<maxit){
if(err < 0){
sig.down <- sig
sig <- (sig.up + sig)/2
}else{
sig.up <- sig
sig <- (sig.down + sig)/2
}
err <- BS.fun(S, K, TT, r, sig, tipo) - vmercado
count <- count + 1
}
if(count==maxit){ return(NA)} else{return(sig)}
}
### A função BS.fun calcula os preços recorrendo
# à fórmula de Black-Scholes. C = Call; P = Put
BS.fun <-
function(S, K, TT, r, sig, tipo="C"){
d1 <- (log(S/K) + (r + sig^2/2)*TT) / (sig*sqrt(TT))
d2 <- d1 - sig*sqrt(TT)
if(tipo=="C"){
valor <- S*pnorm(d1) - K*exp(-r*TT)*pnorm(d2)
}
if(tipo=="P"){
valor <- K*exp(-r*TT)*pnorm(-d2) - S*pnorm(-d1)
}
return(valor)
}
```

Figura 5.6: Código R para o cálculo da volatilidade implícita usando o método da bissecção

Exemplo de cálculo da volatilidade implícita

```
S <- 5290.36
TT <- 20/365
r <- 0.03294
K<-c(5350,5500,3700,3800)
vmercado<-c(221.6,154.2,4.9,6.4)
tipo<-c("C","C","P","P")
vol.imp<-NULL
for (i in 1:4)
{
vol.imp[i]<-imp.vol.fun(S, K[i], TT, r, vmercado[i], tipo[i])
}
vol.imp
```

Figura 5.7: Código R que exemplifica a aplicação da função do R, imp.vol.fun, definida pelo código apresentado na Figura 5.6

Exemplo de cálculo da volatilidade implícita usando o pacote derivmkts

```
library(derivmkts)
S <- 5290.36
TT <- 20/365
r <- 0.03294
K<-c(5350,5500,3700,3800)
vmercado<-c(221.6,154.2,4.9,6.4)
tipo<-c("C","C","P","P")
vol.imp.call<-NULL
vol.imp.put<-NULL
for (i in 1:2)
{
vol.imp.call[i]<-bscallimpvol(s=S, k=K[i], r=r, tt=TT, d=0,
price=vmercado[i]))
}
for (i in 3:4)
{
vol.imp.put[i]<-bsputimpvol(s=S, k=K[i], r=r, tt=TT, d=0,
price=vmercado[i]))
}
vol.imp.call
vol.imp.put
```

Figura 5.8: Código R de aplicação das funções bscallimpvol e bsputimpvol do pacote derivmkts

5.3 Volatilidade histórica

Como referido anteriormente, na prática, quando se pretendem implementar várias estratégias financeiras são necessárias estimativas da volatilidade e que um estimador simples não é mais do que o desvio-padrão da amostra de retornos durante um certo período de tempo. Um dos problemas associado a este estimador é saber qual o melhor período a considerar para o seu cálculo, pois se for muito longo poderá não ser muito relevante para a estimativa de hoje, por outro lado se o período for muito curto poderá não conter a informação necessária. No entanto, a curto prazo as estimativas da volatilidade histórica poderão ser relevantes para a previsão dos seus valores futuros.

Figura 5.9: Ilustração da evolução do preço de ativos num dia de mercado financeiro

Vários estimadores baseados nos valores passados têm sido propostos. A seguir apresentam-se vários desses estimadores, iniciando-se por introduzir alguma notação relevante para a sua descrição.

Considere-se que:

σ, como até aqui, corresponde à volatilidade de um ativo financeiro, e é o parâmetro que se pretende estimar;

C_t é o valor de fecho do preço desse ativo no período t;

O_t é o valor de abertura do preço desse ativo no período t;

H_t é o maior valor do preço desse ativo no período t;

L_t é o menor valor do preço desse ativo no período t;

n é o número de períodos usados para o cálculo da estimativa da volatilidade;

N é o número de períodos por ano.

No gráfico da Figura 5.9 ilustra-se esta informação usando os valores de preços do índice NASDAQ Composite[1] para um período de um dia.

Close-to-Close

A medida de volatilidade mais simples é a volatilidade padrão (desvio padrão) que usa um preço perto do fecho do mercado.

$$\hat{\sigma}_{Cl} = \sqrt{\frac{N}{n-2}\sum_{i=1}^{n-1}(r_i - \bar{r})^2},$$

$$\text{onde } r_i = \log\left(\frac{C_i}{C_{i-1}}\right)$$

$$\text{e } \bar{r} = \frac{r_1 + r_2 + \ldots + r_{n-1}}{n-1}.$$

Volatilidade Alta-Baixa: Parkinson

A expressão do estimador de Parkinson (Parkinson, 1980) para estimar a volatilidade histórica é baseada no maior e menor preço do dia.

$$\hat{\sigma}_P = \sqrt{\frac{N}{4n \times \log 2}\sum_{i=1}^{n}\left(\log\frac{H_i}{L_i}\right)^2}.$$

[1]Este índice é constituído por ações e títulos listados na bolsa de valores da NASDAQ. Juntamente com o Dow Jones Average e o S&P 500, é um dos três índices com maior relevo nas bolsas de valores dos EUA.

Como este estimador só usa os preços mais alto e o mais baixo, é menos sensível às diferenças das horas de negociação. Por exemplo, o fecho nos mercados da União Europeia e dos Estados Unidos têm uma diferença de aproximadamente metade de um dia de negociação. Isto significa que poder-se-á ter retornos muito diferentes. Mas ao usar o valor mais alto e mais baixo significa que a negociação durante todo o dia é examinada, e os dias sobrepõem-se. No entanto, este estimador não lida com saltos, ou seja, em média, subestima a volatilidade, pois não tem em conta valores altos e baixos quando a negociação não ocorre (por exemplo, nos fins de semana, entre o fechar e abrir dos mercados). Embora não lide com tendência (do inglês *drift*), não é um facto muito relevante pois o valor da mesma é geralmente pequeno. O estimador de Parkinson é até 5,2 vezes mais eficiente do que o estimador *Close-to-Close*. Embora outros estimadores sejam mais eficientes, alguns estudos baseados em dados simulados, mostraram que este estimador é uma das melhores medidas em aplicações reais.

Garman e Klass

O estimador de Garman e Klass (Garman e Klass, 1980) assume um movimento Browniano com valor de tendência nulo e sem saltos de abertura (ou seja, o valor de abertura é igual ao valor de fecho no período anterior). Este estimador é 7,4 vezes mais eficiente que o estimador *Close-to-Close* mas tal como o estimador Parkinson também subestima a volatilidade.

$$\hat{\sigma}_{GK} = \sqrt{\frac{N}{n}\sum_{i=1}^{n}\left[\tfrac{1}{2}\left(\log\frac{H_i}{L_i}\right)^2 - (2\log 2 - 1)\left(\log\frac{C_i}{O_i}\right)^2\right]}.$$

OHLC Volatilidade: Rogers e Satchell

O estimador de Roger e Satchell (Rogers e Satchell, 1991) assume um movimento Browniano com tendência não nula mas assume, como no estimador de Garman e Klass, que não há saltos de abertura. A eficiência deste estimador é similar à do estimador Garman-Klass, porém beneficia-se de poder ter um valor de tendência não nulo. Mais uma vez ao não lidar com os saltos de abertura faz com que também subestime a volatilidade.

$$\hat{\sigma}_{RS} = \sqrt{\frac{N}{n}\sum_{i=1}^{n}\left[\log\frac{H_i}{C_i}\times\log\frac{H_i}{O_i} + \log\frac{L_i}{C_i}\times\log\frac{L_i}{O_i}\right]}.$$

OHLC Volatilidade: Garman e Klass - Yang e Zhang

O estimador proposto por Yang e Zhang (2000) é uma modificação do estimador de Garman e Klass que permite a existência de saltos na abertura. A sua eficiência é 8 vezes superior à do estimador *Close-to-Close*.

$$\hat{\sigma}_{GKYZ} = \sqrt{\frac{N}{n}\sum_{i=1}^{n}\left[\left(\log\tfrac{O_i}{C_{i-1}}\right)^2 + \tfrac{1}{2}\left(\log\tfrac{H_i}{L_i}\right)^2 - (2\times\log 2 - 1)\left(\log\tfrac{C_i}{O_i}\right)^2\right]}.$$

OHLC Volatility: Yang e Zhang

Por último, apresentamos o estimador de volatilidade histórica proposto também por Yang e Zhang (2000). Estes autores mostram que este estimador é centrado (assumindo continuidade nos valores dos preços), é independente da tendência, de saltos de abertura e tem uma variância mínima dentro dos estimadores com propriedades semelhantes. Pode ser interpretado como uma média pesada do estimador de Rogers e Satchell.

$$\hat{\sigma}^2_{YZ} = \hat{\sigma}^2_o + k\hat{\sigma}^2_c + (1-k)\hat{\sigma}^2_{RS},$$

onde

$$\hat{\sigma}^2_o = \frac{N}{n-1}\sum_{i=1}^{n}\left(\log\frac{O_i}{C_{i-1}} - \hat{\mu}_o\right)^2, \quad \text{com} \quad \hat{\mu}_o = \frac{1}{n}\sum_{i=1}^{n}\log\frac{O_i}{C_{i-1}}$$

$$\hat{\sigma}^2_c = \frac{N}{n-1}\sum_{i=1}^{n}\left(\log\frac{C_i}{O_i} - \hat{\mu}_c\right)^2, \quad \text{com} \quad \hat{\mu}_c = \frac{1}{n}\sum_{i=1}^{n}\log\frac{C_i}{O_i},$$

$$\hat{\sigma}^2_{RS} = \frac{N}{n}\sum_{i=1}^{n}\left(\log\tfrac{H_i}{C_i}\times\log\tfrac{H_i}{O_i} + \log\tfrac{L_i}{C_i}\times\log\tfrac{L_i}{O_i}\right)$$

e

$$k = \frac{\alpha - 1}{\alpha + \frac{n+1}{n-1}}.$$

O valor de α é, em geral, considerado igual a $1,34$, como proposto por Yang e Zhang (2000).

Como se pode verificar, alguns destes estimadores históricos da volatilidade incorporam ponderação, ao atribuírem maior peso aos preços de ativos mais recentes e menor peso aos mais antigos. Para além dos preços de fecho do mercado são também utilizados os preços de abertura, valores máximos e mínimos por forma a refletir o mercado no período em análise.

No Exemplo 5.2 ilustra-se a estimação da volatilidade histórica para o índice FTSE 100 utilizando os vários estimadores descritos. Para isso recorre-se ao ambiente computacional R e ao pacote TTR (Ulrich, 2018). De referir que o índice FTSE 100 é uma média ponderada do valor dos preços das ações das 100 maiores empresas de ações negociadas na Bolsa de Londres.

A Figura 5.10 ilustra a evolução da cotações deste índice nos seus preços diários de abertura, de fecho, o maior e o menor no período compreendido entre 2 janeiro de 2019 e 30 agosto de 2019. Na Figura 5.11 mostra-se a evolução durante esse período dos log-retornos.

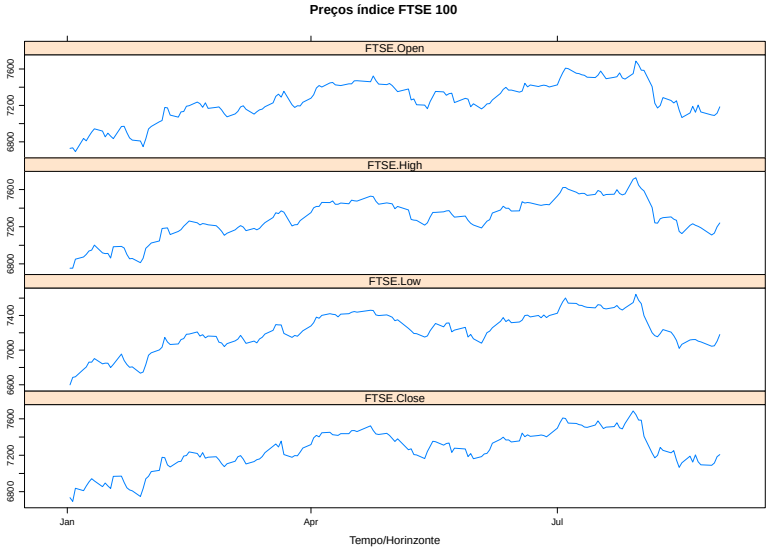


Figura 5.10: Evolução dos preços diários do índice FTSE 100 no período de 2 janeiro de 2019 a 30 agosto de 2019. De cima para baixo: preço diário de abertura, maior valor diário, menor valor diário e preço diário de fecho

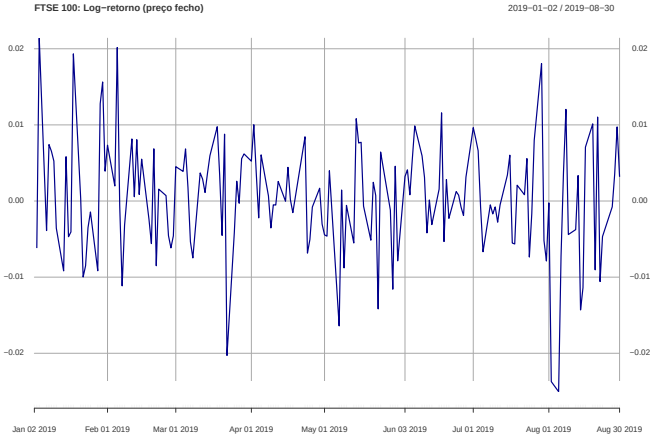


Figura 5.11: Evolução dos retornos do índice FTSE 100 no período de 2 janeiro de 2019 a 30 agosto de 2019

Tabela 5.1: Estimativas da volatilidade histórica para o índice FTSE 100 a 30 de agosto de 2019 baseadas nos vários estimadores apresentados

CC	P	GK	RS	GK.YZ	YZ
0,1622	0,1458	0,1325	0,1258	0,1324	0,1336

Exemplo 5.2. *Consideram-se os preços diários de abertura, fecho, maior e menor do índice FTSE 100 no período de 2 janeiro de 2019 a 30 agosto de 2019. Usando o pacote do R, quantmod (Ryan e Ulrich, 2019) é muito fácil obter estes dados diretamente de* `https://finance.yahoo.com`*, o site Yahoo Finanças. Estimaremos a volatilidade história para este índice no final de agosto de 2019, considerando 22 observações anteriores. O número de períodos por ano, N, será de 252 para obtermos uma estimativa anual. Na prática, se os preços são diários usa-se $N = 252$, se os dados usados forem semanais então $N = 52$, e se forem mensais $N = 12$. O código R é o apresentado na Figura 5.12.*

A 30 de agosto de 2019 os valores estimados da volatilidades histórica para cada um dos estimadores são os apresentados na Tabela 5.1.

No período em análise é interessante observar a evolução das estimativas

Estimação da volatilidade histórica

```
library(quantmod)
FTSE100<-getSymbols("^FTSE", from = "2019-01-02",
+ to = "2019-08-30")
library(TTR)
vol.C<-volatility(FTSE100, n = 22, N=252,
calc = "close")
vol.GK <- volatility(FTSE100, n = 22,N=252,
calc="garman.klass")
vol.P <- volatility(FTSE100, n = 22, N=252,
calc="parkinson")
vol.R <- volatility(FTSE100, n = 22, N=252,
calc="rogers.satchell")
vol.GK.YZ <- volatility(FTSE100, n = 22, N=252,
calc="gk.yz")
vol.YZ <- volatility(FTSE100, n = 22, N=252,
calc="yang.zhang")
```

Figura 5.12: Código R recorrendo à função volatility do pacote TTR para estimação da volatilidade histórica relativa ao índice FTSE100, usando cada um dos seis estimadores apresentados

da volatilidade histórica segundo os vários estimadores numa janela de 22 dias. A Figura 5.13 mostra esta evolução, e como podemos observar as estimativas *Close-to-Close* são as que apresentam, em geral, maiores valores de volatilidade, seguidas das estimativas obtidas pela estimador de Parkinson. As restantes estimativas parecem seguir o mesmo padrão de valores.

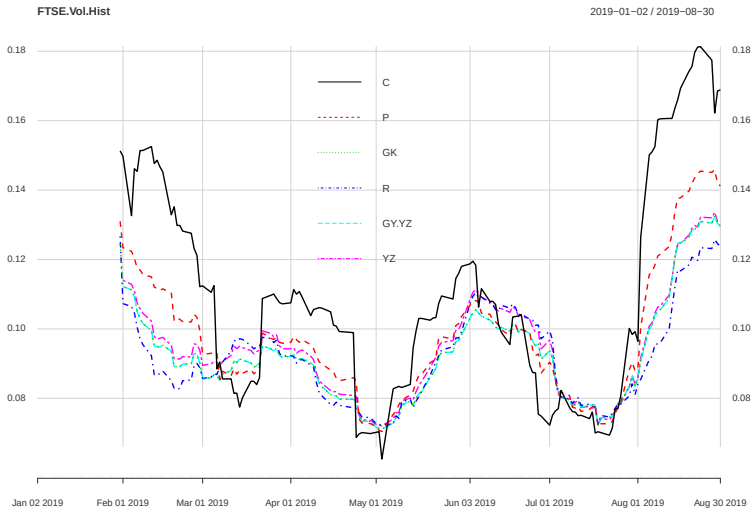


Figura 5.13: Evolução das estimativas da volatilidade histórica anual, para o índice FTSE 100 no período compreendido entre 2 janeiro de 2019 a 30 agosto de 2019. Considerou-se para o cálculo de cada estimativa os valores dos preços de 22 dias anteriores. Cada linha representa as várias estimativas relativamente a cada um dos seis estimadores apresentados

5.4 Volatilidade condicionada

No mundo das finanças, o modelo de avaliação de opções de Black-Scholes é, sem dúvida, um dos mais populares. Todavia, neste modelo a variável que representa o retorno financeiro assume uma distribuição normal com desvio padrão (volatilidade) constante ao longo do tempo. E, em geral, esta suposição é inadequada pois não reflete o dinamismo do mercado atual, onde claramente a variabilidade não é constante, nomeadamente, em períodos de

inquietação financeira em que a volatilidade dos retornos é maior comparada com períodos serenos. Nestas últimas décadas vários modelos têm sido propostos para melhor refletir a realidade do mercado, entre eles os modelos da família ARCH, ou seja de heteroscedasticidade condicionada autorregressiva, acrónimo de *Auto-Regressive Conditional Heterocedasticity*, introduzidos por Engle (1982). Esta classe de modelos para séries temporais permite incorporar a evolução temporal da variância condicionada (volatilidade).

5.4.1 Modelo ARCH

A seguir serão apresentados alguns destes modelos, iniciando-se pela definição de um modelo ARCH(p).

Definição 5.1. *Uma série temporal* $\{X_T, t \in T\}$ *diz-se ser do tipo ARCH(p) se*

$$X_t = \sqrt{\sigma_t^2} W_t,$$

onde $\{W_T, t \in T\}$ *é um ruído branco Gaussiano (ou seja uma sequência de variáveis aleatórias independentes e identicamente distribuídas N(0,1)) e* σ_t^2 *é uma função (positiva) de* $X_{t-1}, X_{t-2}, \ldots$ *da forma:*

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i X_{t-i}^2,$$

com $\alpha_0 > 0$ *e* $\alpha_i \geq 0$ *para* $i = 1, 2, \ldots, p-1$ *e* $\alpha_p > 0$.

Na forma geral de um modelo ARCH(p) não é necessário que o ruído branco $\{W_T, t \in T\}$ seja gaussiano, mas por simplicidade admite-se aqui esta suposição. Outras distribuições que têm sido consideradas na análise de dados financeiros são a distribuição t-Student com baixo número de graus de liberdade, a distribuição de erro generalizada, GED, (*Generalized Error Distribution*) ou alguma outra distribuição de probabilidade que melhor descreva as caudas pesadas típicas em séries financeiras. Quando se especifica a distribuição para W_t esta irá permitir definir a distribuição condicional $F(X_t|I_{t-1})$, em que $I_{t-1} = (X_{t-1}, X_{t-2}, \ldots)$ representa o conjunto de informação[2] até o instante $t-1$, o qual contem X_{t-1} assim como todas as realizações anteriores do processo X_{t-1}.

[2]Na linguagem probabilística podemos interpretar o I_{t-1} como a filtragem natural no instante $t-1$, $\mathcal{F}_{t-1}$.

O modelo mais simples desta família, o ARCH(1), no qual a variância condicionada σ_t^2 depende apenas do momento imediatamente anterior, vai-nos permitir explorar um pouco as propriedades deste modelo. Neste caso o modelo toma a seguinte forma:

$$X_t = \sqrt{\sigma_t^2} W_t \quad \text{e} \quad \sigma_t^2 = \alpha_0 + \alpha_1 X_{t-1}^2. \tag{5.3}$$

O valor médio não condicionado de X_t é zero, pois

$$\begin{aligned}\mathbb{E}[X_t] &= \mathbb{E}[\mathbb{E}[X_t|I_{t-1}]] \\ &= \mathbb{E}[\mathbb{E}[\sigma_t W_t|I_{t-1}]] \\ &= \mathbb{E}[\sigma_t \mathbb{E}[W_t|I_{t-1}]] = \mathbb{E}[\sigma_t \times 0] = 0.\end{aligned}$$

Note-se que como $\sigma_t^2 = \alpha_0 + \alpha_1 X_{t-1}^2$ depende apenas da informação no tempo $t-1$ pode ser tratado como uma constante quando se condiciona nesse tempo. Para além disso W_t é não correlacionado com X_{t-1}. Tem-se ainda que:

$$\begin{aligned}\mathbb{E}[X_t X_{t-1}] &= \mathbb{E}[\mathbb{E}[X_t X_{t-1}|I_{t-1}]] \\ &= \mathbb{E}[X_{t-1} \mathbb{E}[X_t|I_{t-1}]] \\ &= \mathbb{E}[X_{t-1} \times 0] = 0,\end{aligned}$$

logo a covariância entre X_t e X_{t-1} é

$$\text{Cov}(X_t, X_{t-1}) = \mathbb{E}[X_t X_{t-1}] - \mathbb{E}[X_t]\mathbb{E}[X_{t-1}] = 0.$$

De forma similar se mostra que $\text{Cov}(X_t, X_{t-k}) = 0, \forall k \geq 1$. Uma vez que a covariância é zero não se pode prever X_t com base na sua história I_{t-1}. Nos mercados financeiros, este facto é uma evidência que se designa por *hipótese do mercado eficiente*. Segundo esta hipótese os mercados financeiros são "eficientes em relação à informação", assim os preços negociados para os ativos refletem toda a informação histórica disponível (Jesen, 1978; Malkiel, 1992).

No entanto X_t^2 pode ser previsto. Para vermos isto calcule-se a variância condicionada de X_t.

$$\begin{aligned}\text{Var}(X_t|I_{t-1}) &= \mathbb{E}\left[X_t^2|I_{t-1}\right] \\ &= \mathbb{E}\left[\sigma_t^2 W_t^2|I_{t-1}\right] \\ &= \sigma_t^2 \mathbb{E}\left[W_t^2|I_{t-1}\right] = \mathbb{E}[\sigma_t \times 1] = \sigma_t^2.\end{aligned}$$

Logo σ_t^2 representa a variância condicionada, a qual por definição é função da história do processo,

$$\sigma_t^2 = \alpha_0 + \alpha_1 X_{t-1}^2,$$

e deste modo pode ser prevista usando X_{t-1}^2. Como se tem

$$\mathbb{E}\left[X_t^2 | I_{t-1}\right] = \alpha_0 + \alpha_1 X_{t-1}^2,$$

é possível então estimar α_0 e α_1 usando o método de mínimos quadrados para modelar X_t^2 em função de X_{t-1}^2. Deste modo, X_t^2 pode ser visto como um modelo $AR(1)$ onde o seu ruído não é i.i.d.

Vejamos agora qual a variância não condicionada de X_t.

$$\begin{aligned}
\mathrm{Var}(X_t) =& \mathbb{E}\left[X_t^2\right] - \mathbb{E}^2\left[X_t\right] = \mathbb{E}\left[X_t^2\right] \\
=& \mathbb{E}\left[\mathbb{E}\left[X_t^2 | I_{t-1}\right]\right] \\
=& \mathbb{E}\left[\alpha_0 + \alpha_1 X_{t-1}^2\right] \\
=& \alpha_0 + \alpha_1 \mathbb{E}\left[X_{t-1}^2\right] \\
\Longrightarrow \mathbb{E}\left[X_t^2\right] =& \frac{\alpha_0}{1-\alpha_1} \text{ (se } 0 < \alpha_1 < 1).
\end{aligned}$$

Tendo em conta as propriedades vistas para o modelo ARCH(1) e, se $\alpha_1 < 1$, então este é um processo estacionário. Para além disso, é fácil de verificar que a variância condicionada inclui a variância não condicionada. Se denotarmos por $\sigma^2 = \mathrm{Var}\left(X_t\right)$, pode mostrar-se que

$$\sigma_t^2 = \sigma^2 + \alpha_1 \left(X_{t-1}^2 - \sigma^2\right), \tag{5.4}$$

pois

$$\sigma^2 = \frac{\alpha_0}{1-\alpha_1},$$

o que implica que $\alpha_0 = \sigma^2\left(1-\alpha_1\right)$, se substituirmos em $\sigma_t^2 = \alpha_0 + \alpha_1 X_{t-1}^2$ obtemos o resultado apresentado em (5.4).

Nos modelos ARCH constatámos que a variância não condicionada é fixa, enquanto que a variância condicionada depende do histórico variando no tempo. Outra propriedade destes modelos é que a distribuição não condicionada (marginal) de X_t não é normal mesmo que a distribuição de W_t o seja, sendo assim apropriados para modelar os retornos pois estes tendem a apresentar distribuições com caudas pesadas com valor de curtose superior

a três. Continuando com a exemplificação para um modelo ARCH(1) com ruído branco gaussiano verificamos que a curtose é dada por

$$\frac{\mathbb{E}\left[X_t^4\right]}{\left(\mathrm{Var}(X_t)\right)^2} = 3\left(\frac{1-\alpha_1^2}{1-3\alpha_1^2}\right) > 3,$$

que será igual a 3 se $\alpha_1 = 0$.

Uma outra característica importante dos modelos ARCH é a de conseguir capturar a tendência de grupos de volatilidade, ou seja, grandes (pequenas) mudanças de valores de ativos podem ser seguidas por outras grandes (pequenas) mudanças desses valores, mas com sinal imprevisível.

As propriedades deste modelo são uma vantagem, uma vez que permitem refletir o que se passa nos mercados financeiros, em particular quando se pretende modelar séries temporais de retornos.

Para ilustrarmos o comportamento deste modelo vamos fazer uma simulação usando o R.

Exemplo 5.3. *Considere-se um modelo ARCH(1) para modelar os retornos, X_t, de um certo ativo financeiro:*

$X_t = \sigma_t W_t$, *com* $W_t \sim_{iid} N(0,1)$,

e $\sigma_t^2 = 5 + 0,4X_{t-1}^2$

Simularemos, usando códigos diferentes, este modelo. Geram-se 1200 valores de um ruído branco gaussiano e fixamos o valor de X_1 em zero. Usamos uma fase burn-in de 200 simulações para que os resultados da simulação não fiquem dependentes da condição inicial $X_1 = 0$.

Na Figura 5.14 apresenta-se um código simples da linguagem R. Na Figura 5.15 é apresentado o código recorrendo ao pacote rugarch, Ghalanos (2019). O resultado, em ambos as situações, é o apresentado na Figura 5.16.

Um outro facto já referido sobre os processos ARCH é que a sua distribuição marginal não é normal e que tem caudas pesadas. O gráfico Q-Q normal comparando a distribuição dos retornos gerados segundo o processo ARCH(1) do Exemplo 5.3 com a distribuição normal encontra-se no painel inferior da Figura 5.18. Observa-se a não linearidade dos pontos, o que sugere que as observações não são normalmente distribuídos. Assim, embora tenhamos uma sequência não correlacionada, ver Figura 5.17, é possível que existam formas significativas de dependência entre os sucessivos termos da série. A

Simulação de um modelo ARCH(1)

```
n.sim<-1200
alfa0 <-5
alfa1 <-0.4
x<-NULL
x[1]<-0
w<-rnorm(nsim)
sigma2<-NULL
for(t in 2:n.sim){
sigma2[t]<-alfa0+alfa1*x[t-1]^2
x[t]<-w[t]*sqrt(sigma2[t])
}
```

Figura 5.14: Código R com comandos básicos para simulação de um modelo ARCH(1)

Simulação de um modelo ARCH(1)

```
library(rugarch)
ex.arch1 <-ugarchspec(variance.model =
+ list(garchOrder = c(1, 0)),
mean.model = list(armaOrder = c(0,0)),
fixed.pars = list(mu = 0, omega = 5,
alpha1 = 0.4))
ex.arch1.sim <- ugarchpath(ex.arch1, n.sim = 1200)
```

Figura 5.15: Código R usando o pacote rugarch para simulação de um modelo ARCH(1)

existência de dependência é confirmada pela função de auto-correlação parcial do quadrado dos valores da série temporal, que é reproduzido no painel inferior da Figura 5.17. Como se pode ver, a sua forma não se parece com a da função de autocorrelação de um ruído branco.

Como já referido, a estimativa da função de autocorrelação associado aos valores gerados é apresentada na Figura 5.17 (no painel em cima) e verifica-se que a série aparenta não apresentar autocorrelação. No painel inferior da Figura 5.17 mostra-se a estimativa da função de autocorrelação parcial, PACF, do quadrado dos valores simulados segundo o processo ARCH(1). Podemos

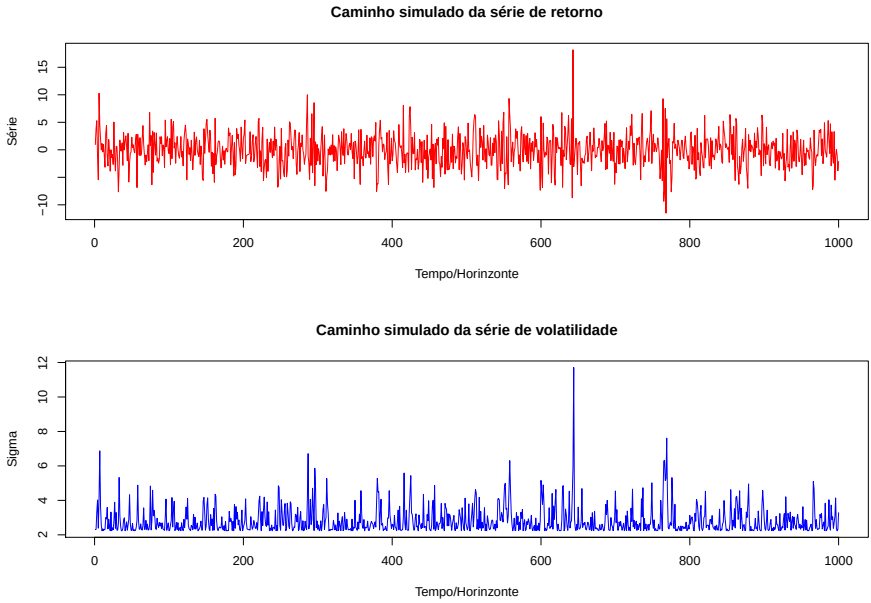


Figura 5.16: Resultado da simulação do modelo ARCH(1) de acordo com o Exemplo 5.3. Apresentam-se os caminhos associado à série de retornos (em cima) e à série de volatilidade (em baixo)

ver um pico no primeiro desfasamento da PACF, o que parece sugerir a presença do efeito ARCH(1).

O modelo ARCH é de facto um modelo simples mas muitas vezes requer muitos parâmetros para descrever adequadamente o processo de volatilidade de um retorno de ativos financeiros. É comum, em várias aplicações de dados financeiros, que o valor de p tenha de ser bastante elevado por forma a capturar todas as correlações do quadrado dos retornos. Para ultrapassar esta questão foram propostos outros modelos, e um dos mais populares é o modelo que falaremos de seguida, o modelo GARCH.

5.4.2 Modelo GARCH

O modelo GARCH (*Generalized Auto-Regressive Conditional Heteroscedasticity*) é uma extensão do modelo ARCH que foi desenvolvida por Bollerslev

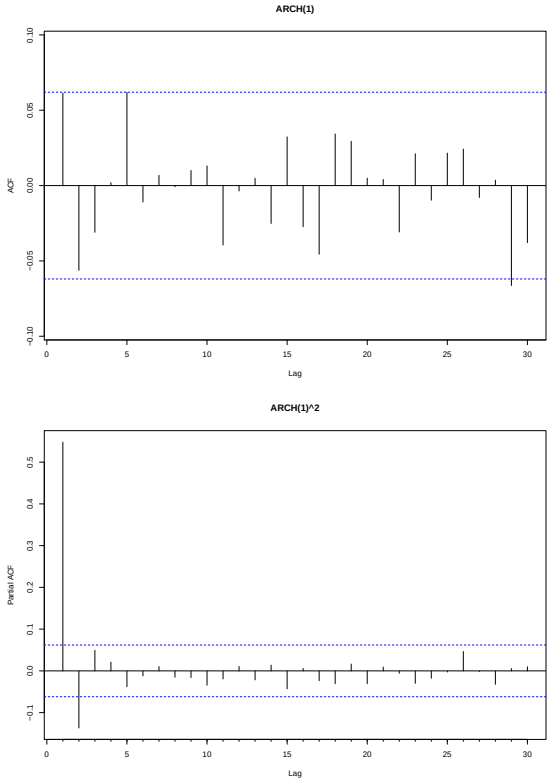


Figura 5.17: Em cima: Estimativa da função de autocorrelação da série simulada para os retornos segundo um processo ARCH(1). Em baixo: Estimativa da função de autocorrelação parcial para os quadrados dos retornos da série simulada segundo um processo ARCH(1)

(1986, 1987). Este modelo introduz na equação ARCH os valores anteriores da variância e permite descrever a volatilidade de forma mais parcimoniosa, ou seja, com menor número de parâmetros quando comparado com a modelação por um modelo ARCH. De facto, o modelo GARCH é um dos modelos mais comuns quando se pretende modelar séries temporais de dados financeiros, tendo inspirado mais de uma dezena de modelos mais sofisticados.

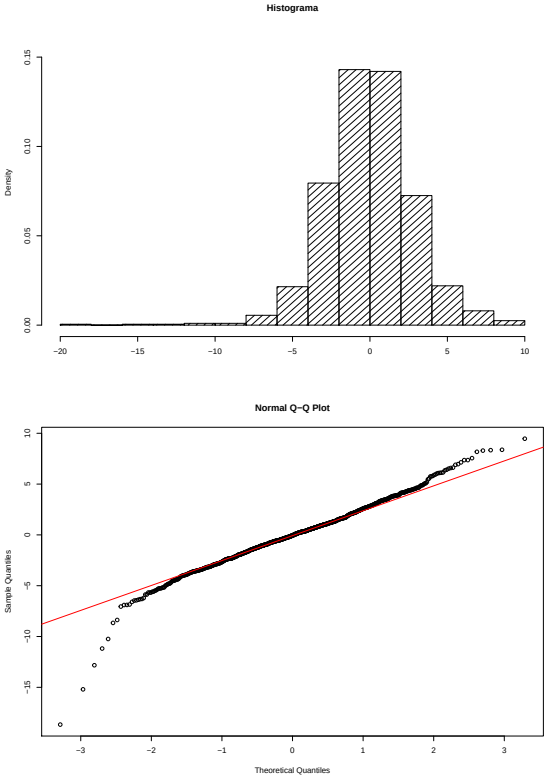


Figura 5.18: Em cima: Histograma da série simulada para os retornos segundo um processo ARCH(1). Em baixo: Gráfico QQnormal da série simulada para os retornos segundo um processo ARCH(1)

Definição 5.2. *Uma série temporal* $\{X_T, t \in T\}$ *diz-se ser do tipo GARCH(p,q) se*

$$X_t = \sqrt{\sigma_t^2} W_t,$$

onde $\{W_T, t \in T\}$ *é um ruído branco Gaussiano onde a equação da evolução de* σ_t^2 *é dada por*

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i X_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2,$$

com $\alpha_0 > 0$, $\alpha_i \geq 0$ *para* $i = 1, 2, \ldots, p-1$ *e* $\alpha_p > 0$, $\beta_j \geq 0$ *para* $j = 1, 2, \ldots, q-1$ *e* $\beta_q > 0$.

Nos modelos GARCH para que haja uma solução única e estacionária (Bougerol e Picard, 1992; Straumann, 2005) as restrições impostas sobre os parâmetros são dadas por

$$\sum_{i=1}^{k} (\alpha_i + \beta_i) < 1, \text{ com } k = max(p, q).$$

Observar que essa definição generaliza a noção dos modelos ARCH(p), que podem ser recuperados pela configuração $q = 0$.

A ideia principal é que σ_t^2, a variância condicionada de X_t dada a informação disponível até ao tempo $t-1$, tenha uma estrutura auto-regressiva e positivamente correlacionada com o seu próprio passado e com os valores recentes dos quadrados dos retornos. Isto captura a ideia da volatilidade (variância condicionada) como sendo persistente, i.e valores elevados (baixos) de X_t^2 têm maior propensão para serem seguidos de elevados (baixos) valores.

Como no caso de modelos ARCH, a suposição de ruído branco gaussiano pode ser relaxada podendo ser usadas as distribuições t-Student, GED ou outra de caudas pesadas.

De seguida apresentam-se algumas das propriedades básicas dos modelos GARCH. Como acontecia no modelo ARCH o valor esperado não condicionado de X_t é zero. E a variância não condicionada de X_t é dada por

$$\mathbb{E}\left[X_t^2\right] = \frac{\alpha_0}{1 - \sum_{i=1}^{max(p,q)} (\alpha_i + \beta_j)}.$$

De seguida vamos verificar algumas das vantagens do uso dos modelos GARCH exemplificando com o modelo GARCH(1, 1),

$$\sigma_t^2 = \alpha_0 + \alpha_1 X_{t-1}^2 + \beta_1 \sigma_{t-1}^2, \quad 0 \leq \alpha_1, \beta_1 \leq 1, (\alpha_1 + \beta_1) < 1.$$

que é frequentemente usado no ajuste de séries temporais financeiras.

Podemos observar que, para um valor elevado de X_{t-1}^2 ou σ_{t-1}^2 conduz a um valor elevado de σ_t^2 o que levará a que $X_t^2 = \sigma_t^2 W_t^2$ também será elevado. Como visto atrás, este comportamento também é verificado nos modelos ARCH e é consistente com a existência de grupos de volatilidade, que como se sabe é um comportamento característico das séries temporais financeiras. Outra propriedade partilhada com os modelos ARCH é a da sua distribuição não condicionada de X_t não ser normal mesmo que a distribuição de W_t o seja. Aliás, prova-se que a sua distribuição tem caudas mais pesadas que as da distribuição normal (Tsay, 2010). Por exemplo, se W_t seguir uma distribuição normal a curtose de um processo GARCH(1,1) é

$$\frac{\mathbb{E}\left[X_t^4\right]}{\left(\mathrm{Var}(X_t)\right)^2} = \frac{3\left[1-(\alpha_1+\beta_1)^2\right]}{1-(\alpha_1+\beta_1)^2-2\alpha_1^2} > 3,$$

quando $(\alpha_1+\beta_1)^2 + 2\alpha_1^2 < 1$.

Uma outra propriedade interessante do modelo GARCH é que X_t^2 admite uma representação ARMA$(\max(p,q),q)$. Esta característica é bastante útil pois permite identificar a ordem (p,q) do modelo GARCH usando os critérios habituais para os modelos ARMA.

A estimação dos parâmetros quer dos modelos ARCH ou GARCH é, geralmente, efetuada pelo método da máxima verosimilhança. A função de verosimilhança dependendo naturalmente da distribuição suposta para W_t.

Exemplo 5.4. *Considere-se um modelo GARCH(1,1) para modelar os retornos, X_t, de um certo ativo financeiro:*

$X_t = \sigma_t W_t$, *com* $W_t \sim_{iid} N(0,1)$,

e $\sigma_t^2 = 5 + 0,3X_{t-1}^2 + 0,1\sigma_{t-1}^2$.

Geram-se 1200 valores de um ruído branco gaussiano e fixamos o valor de X_1 em zero. Como no exemplo anterior, usamos uma fase burn-in de 200 simulações para que os resultados da simulação não fiquem dependentes da condição inicial $X_1 = 0$.

O resultado da simulação, cujo código R se apresenta na Figura 5.19, é o apresentado na Figura 5.20.

As estimativas das funções de autocorrelação associados aos valores gerados e aos seus quadrados são as apresentadas na Figura 5.21.

Simulação de um modelo GARCH(1,1)

```
library(rugarch)
ex.garch11<-ugarchspec(variance.model=list(garchOrder=c(1,1)),
+ mean.model = list(armaOrder=c(0,0)),
+ fixed.pars=list(mu = 0, omega=5, alpha1=0.3,beta1 = 0.1))
ex.garch11.sim<-ugarchpath(ex.garch11, n.sim=1200)
```

Figura 5.19: Código R usando o pacote rugarch para simular o modelo GARCH(1,1) definido no Exemplo 5.4

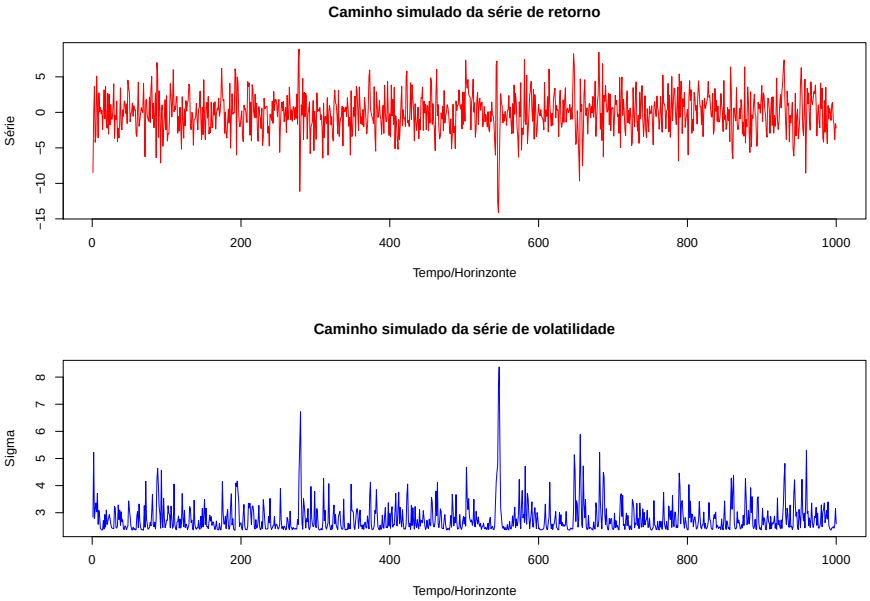


Figura 5.20: Resultado da simulação do modelo GARCH(1,1) de acordo com o Exemplo 5.4. Apresentam-se os caminhos associado à série de retornos (em cima) e à série de volatilidade (em baixo)

Observe-se que o processo GARCH(1,1) gerado tem a mesma variância (não condicionada) que o processo ARCH(1) apresentado na Figura 5.16. No entanto, a variância condicionada do processo GARCH parecer ser mais volátil; comparar a Figura 5.16 com a Figura 5.20. Além disso, o pro-

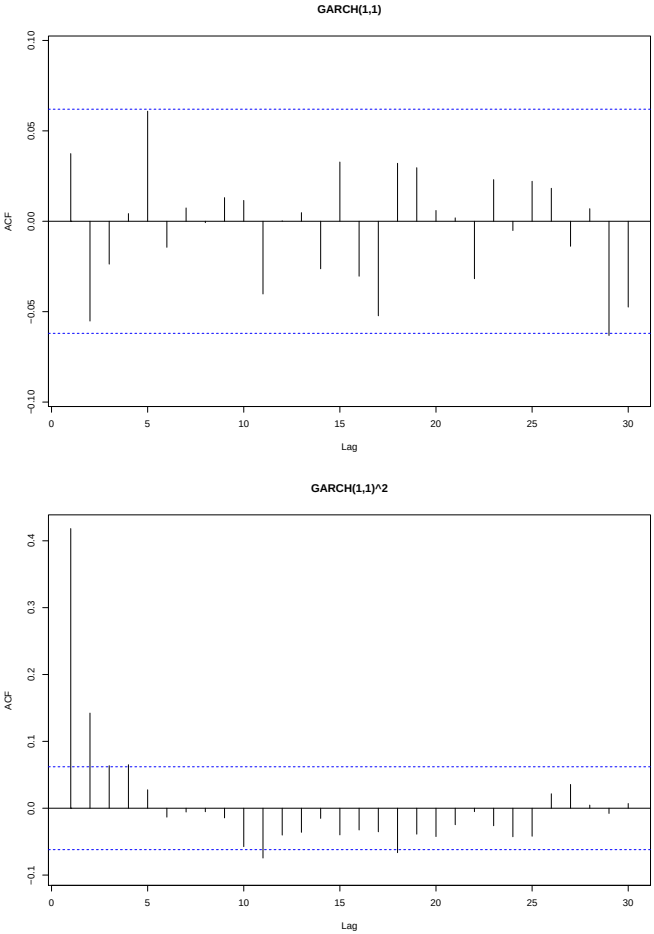


Figura 5.21: Em cima: Estimativa da função de autocorrelação da série simulada para os retornos segundo um processo GARCH(1,1). Em baixo: Estimativa da função de autocorrelação da série simulada para os quadrados dos retornos segundo um processo GARCH(1,1)

cesso GARCH parece apresentar grupos de volatilidade mais persistentes. Os gráficos apresentados na Figura 5.22 mostram que a distribuição marginal de X_t é leptocúrtica. As estimativas das funções de autocorrelação indicam alguma autocorrelação significativa no quadrado de X_t mas não em X_t, Figura 5.21.

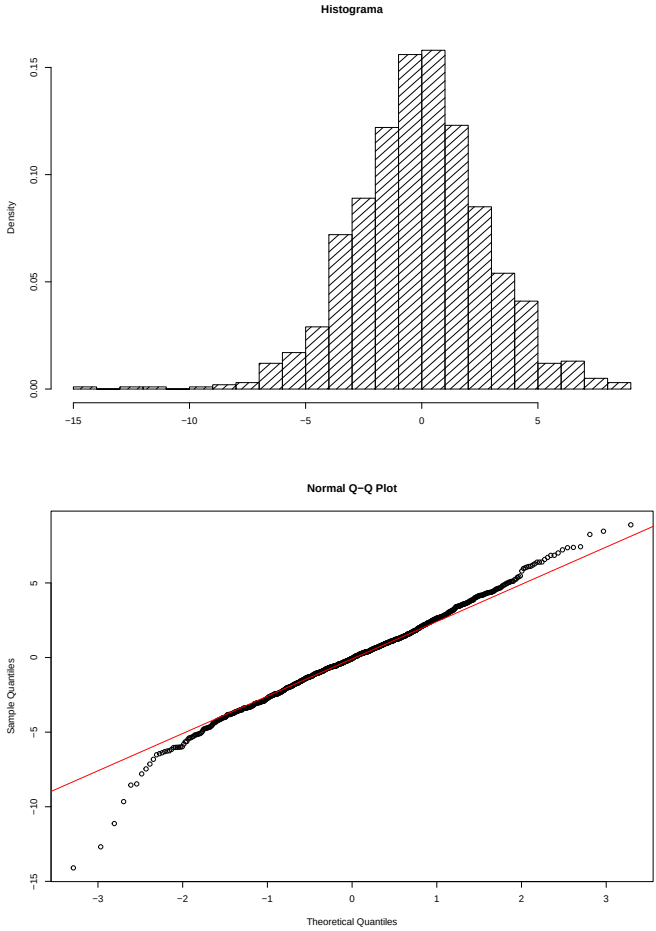


Figura 5.22: Em cima: Histograma da série simulada para os retornos segundo um processo GARCH(1,1). Em baixo: Gráfico QQnormal da série simulada para os retornos segundo um processo GARCH(1,1)

Os modelos ARCH e GARCH foram estendidos em muitas direções distintas, e a quantidade de modelos disponíveis é enorme. Devido à complexidade dos aspetos técnicos dessas generalizações, não se discute aqui as várias formas de GARCH que têm vindo a ser propostas para acomodar as características perdidas pelo modelos padrão ARCH e GARCH. Apresentamos de seguida uma lista breve de algumas das variações mais populares.

- LGARCH *Leverage* GARCH
- PGARCH *Power* GARCH
- EGARCH *Exponential* GARCH
- TGARCH Threshold GARCH
- CGARCH Component GARCH
- GARCH-M GARCH in the Mean
- FIGARCH Fractionally Integrated GARCH

Para uma leitura mais aprofundada sobre estes modelos consultar, por exemplo, Gouriéroux (1997), Fan e Yao (2003), Straumann (2005), Andersen *et al.* (2014) e Ruppert (2015).

5.4.3 Aplicação

No que se segue vamos analisar uma aplicação dos modelos ARCH e GARCH a um conjunto de dados reais de retornos dos ativos S&P 500 abreviação de Standard & Poor's 500 e também conhecido por S&P. Este índice trata-se de um índice composto por quinhentos ativos cotados nas bolsas de NYSE ou NASDAQ, qualificados devido ao seu tamanho de mercado, sua liquidez e sua representação de grupo industrial. Os dados usados neste exemplo foram retirados da página `https://finance.yahoo.com` e correspondem à cotação diária deste índice entre o dia 3 de janeiro de 2000 e o dia 23 de setembro de 2019. Foi considerado o preço de fecho ajustado[3] (ajustado para os dividendos).

É muito fácil a recolha deste tipo de dados usando o pacote pdfetch (Reinhart, 2019) do R, como se pode comprovar pelos comandos apresentados na Figura 5.23.

Recolha de dados - S&P 500

```
library(pdfetch)
dataSP500<-pdfetch_YAHOO(c("^gspc"),
+from = as.Date("2000-01-03"),
+ to=as.Date(as.Date("2019-09-23")))
```

Figura 5.23: Código R usando o pacote pdfetch que permite recolher dados do índice S&P 500 diretamente do site da Yahoo! Finanças

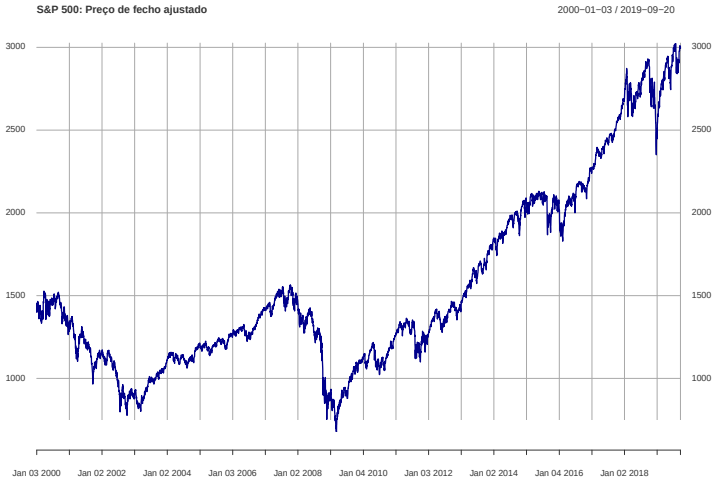


Figura 5.24: Valores dos preços de fecho ajustado diário do índice S&P 500, desde o dia 3 de janeiro de 2000 a 23 de setembro de 2019

Os retornos diários, R_t, são calculados pela primeira diferença do logaritmo dos preços ajustados, Y_t, de fecho, ou seja

$$R_t = \log(Y_t) - \log(Y_{t-1}), \quad t = 2, 3 \cdots,$$

Na Figura 5.24 apresenta-se a evolução histórica dos preços de fecho ajustados do índice S&P 500. O retorno diário desse ativo é ilustrado na Figura 5.25. Pode observar-se um período conturbado em 2008. Devemos recordar que em 15 setembro de 2008, marco da crise financeira, um dos tradicionais bancos

[3]https://help.yahoo.com/kb/SLN28256.html

de investimento dos Estados Unidos, o Lehman Brothers, foi à falência, e as Bolsas de todo o mundo foram afetadas. A data ficou conhecida como segunda-feira negra. Em seguida, outros bancos anunciam perdas bilionárias e os meses que se seguiram foram de muita instabilidade no mercado.

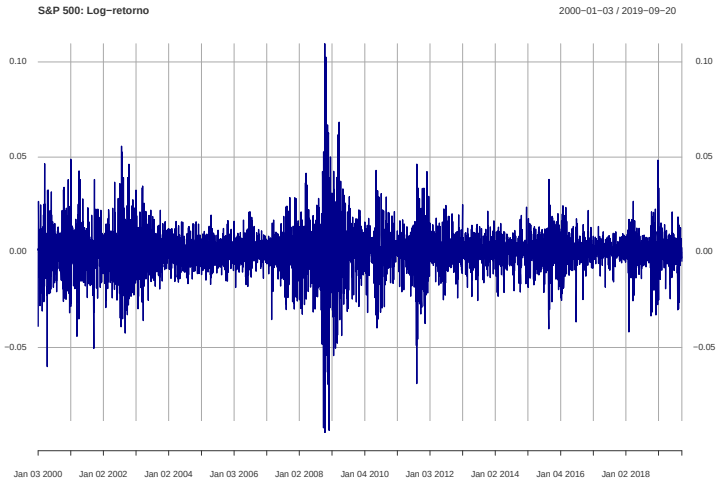


Figura 5.25: Evolução histórica dos retornos diários do índice S&P 500

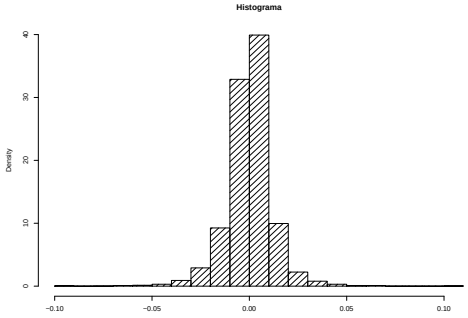
A Figura 5.26 apresenta o histograma e um sumário descritivo desses retornos. Vemos que a sua média é próxima de zero e há um problema de assimetria e curtose, que indica a presença de caudas mais pesadas que a distribuição normal.

Para modelar a heterocedasticidade condicionada vamos iniciar o ajuste por um modelo GARCH(1,1) com ruído branco normal. Consequentemente a especificação do modelo a ajustar será:

$$R_t = \mu + X_t, \;\; X_t = \sigma_t W_T \;\; e \;\; \sigma_t^2 = \alpha_0 + \alpha_1 X_{t-1}^2 + \beta_1 \sigma_{t-1}^2$$

com $W_t \sim_{iid} N(0,1)$.

O código R recorrendo ao pacote rugarch (Ghalanos, 2019) é o apresentado na Figura 5.27. Neste código apresenta-se a especificação do modelo, por defeito esta função considera a distribuição normal para os W_t, pelo que não é necessário especificá-la na função ugarchspec.



	Valor
Dimensão	4960
Mínimo	-0.0947
Máximo	0.1096
Média	0.0001
Mediana	0.0005
Desvio padrão	0.0119
Coef. Assimetria	-0.2254
Curtose	8.5536

Figura 5.26 & Tabela 5.2: Histograma dos retornos S&P 500 e tabela com valores de estatísticas sumárias

Ajuste de modelo GARCH(1,1) com ruído normal - S&P 500

```
library(PerformanceAnalytics)
SP500.ret<-CalculateReturns(dataSP500$`^gspc.adjclose`,
+ method="log")
SP500.ret <-SP500.ret[-1,]
library(rugarch)
garch11.par<-ugarchspec(variance.model=list(
+ garchOrder=c(1,1)), mean.model=list(armaOrder=c(0,0)))
garch11.out<-ugarchfit(spec=garch11.par, data=SP500.ret)
garch11.out
```

Figura 5.27: Código R usando o pacote rugarch para ajustar o modelo GARCH(1,1) com ruído normal aos retornos S&P 500

O resultado da execução do código R é o que se apresenta nas Figuras 5.28, 5.29 e 5.30. O resultado obtido pela aplicação desta função é um longo relatório onde consta a informação sobre as estimativas dos parâmetros e testes estatísticos relevantes para a avaliação do ajuste do modelo especificado.

Assumindo $W_t \sim_{iid} N(0,1)$ o modelo ajustado foi o seguinte:

$$\hat{R}_t = 0,00056 + x_t, \;\; \hat{\sigma}_t^2 = 0,1130x_{t-1}^2 + 0,8710\hat{\sigma}_{t-1}^2.$$

Para este modelo, o valor de AIC= -6,4822 e BIC= -6,4769. Em relação à análise dos resíduos, observamos que o teste de Ljung Box, que permite testar o comportamento de ruído branco, tem associados valores-p pequenos,

Resultado do ajuste de modelo GARCH(1,1) com ruído normal - S&P 500

```
*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model : sGARCH(1,1)
Mean Model : ARFIMA(0,0,0)
Distribution : norm

Optimal Parameters
------------------------------------
         Estimate  Std. Error  t value Pr(>|t|)
mu      0.000561    0.000112   5.0104 0.000001
omega   0.000002    0.000001   2.7743 0.005533
alpha1  0.113004    0.010414  10.8514 0.000000
beta1   0.871038    0.010862  80.1932 0.000000

Robust Standard Errors:
         Estimate  Std. Error  t value Pr(>|t|)
mu      0.000561    0.000093  6.02586 0.000000
omega   0.000002    0.000005  0.42571 0.670319
alpha1  0.113004    0.038582  2.92895 0.003401
beta1   0.871038    0.051072 17.05508 0.000000

LogLikelihood : 16079.77

Information Criteria
------------------------------------
Akaike       -6.4822
Bayes        -6.4769
Shibata      -6.4822
Hannan-Quinn -6.4803
```

Figura 5.28: Resultado do ajuste do modelo GARCH(1,1) com ruído normal aos retornos S&P 500

havendo evidência para a rejeição da hipótese nula (que é a de não existir autocorrelação nos resíduos). Daqui, podemos concluir que os resíduos mostram evidência de um comportamento distinto ao do ruído branco.

Olhando para a parte dos resultados relativos aos resíduos quadrados estandardizados e para os valores dos testes LM[4] para os efeitos ARCH, vemos que os valores-p são superiores aos níveis de significância usuais, pelo que não há evidência da correlação da série dos quadrados dos resíduos.

Relativamente ao teste de bondade de ajustamento (Palm, 1996), que compara a distribuição empírica dos resíduos padronizados com os teóricos da densidade escolhida, vemos que os valores-p são inferiores aos níveis de significância usais, ou seja há evidência para que a suposição de distribuição normal seja rejeitada.

Uma vez que a série dos retornos do preço de fecho ajustado do índice S&P 500 parece apresentar autocorrelação e ruído não normal, a análise destes dados prossegue tentando ajustar um modelo, por exemplo, autorregressivo para captar a dependência na média e assim remover qualquer tipo de autocorrelação e conjuntamente alterar a distribuição para uma com caudas pesadas.

Na Figura 5.31 apresenta-se o código R onde se especifica um modelo AR(2) + GARCH(1,1) e uma reparametrização da distribuição Johnson SU (Rigby e Stasinopoulos, 2005) para o ajuste deste conjunto de dados. Podemos ver o resultado deste ajuste nos gráficos da Figura 5.32. No painel superior apresentam-se os retornos com os limites dos intervalos de confiança e no painel inferior a evolução dos valores estimados da volatilidade condicionada.

[4]Teste Lagrange Multiplier (LM) para autocorrelação, Engle (1982).

Resultado do ajuste de modelo GARCH(1,1) com ruído normal - S&P 500

```
Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                        statistic  p-value
Lag[1]                      9.133 0.002511
Lag[2*(p+q)+(p+q)-1][2]    10.042 0.001864
Lag[4*(p+q)+(p+q)-1][5]    11.899 0.002960
d.o.f=0
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                         statistic p-value
Lag[1]                       1.096  0.2951
Lag[2*(p+q)+(p+q)-1][5]      4.250  0.2245
Lag[4*(p+q)+(p+q)-1][9]      5.603  0.3464
d.o.f=2

Weighted ARCH LM Tests
------------------------------------
            Statistic Shape Scale P-Value
ARCH Lag[3]  0.002015 0.500 2.000  0.9642
ARCH Lag[5]  1.562475 1.440 1.667  0.5760
ARCH Lag[7]  2.147461 2.315 1.543  0.6861

Nyblom stability test
------------------------------------
Joint Statistic:  34.9535
Individual Statistics:
mu     0.2923
omega  3.2641
alpha1 0.5320
beta1  0.9889

Asymptotic Critical Values (10% 5% 1%)
Joint Statistic:       1.07 1.24 1.6
Individual Statistic:  0.35 0.47 0.75
```

Figura 5.29: Resultado do ajuste do modelo GARCH(1,1) com ruído normal aos retornos S&P 500

Resultado do ajuste de modelo GARCH(1,1) com ruído normal - S&P 500

```
Sign Bias Test
------------------------------------
                     t-value   prob  sig
Sign Bias           2.9070 3.665e-03 ***
Negative Sign Bias  0.9263 3.543e-01
Positive Sign Bias  2.6373 8.384e-03 ***
Joint Effect       32.6038 3.904e-07 ***


Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
    group statistic p-value(g-1)
1    20     178.0    7.573e-28
2    30     192.7    4.322e-26
3    40     217.5    1.223e-26
4    50     250.3    8.370e-29


Elapsed time : 0.2204602
```

Figura 5.30: Resultado do ajuste do modelo GARCH(1,1) com ruído normal aos retornos S&P 500

Ajuste do modelo AR(2)+GARCH(1,1) - S&P 500

```
garch11.jsu.spec=ugarchspec(variance.model=
+list(garchOrder=c(1,1)), mean.model=list(armaOrder=c(2,0)),
+distribution.model = "jsu")
garch11.jsu.fit=ugarchfit(spec=garch11.jsu.spec,
+data=SP500.ret)
```

Figura 5.31: Código R para o ajuste do modelo AR(2)+ GARCH(1,1) e distribuição Johnson SU aos dados dos retornos S&P 500. Foi usado o pacote rugarch

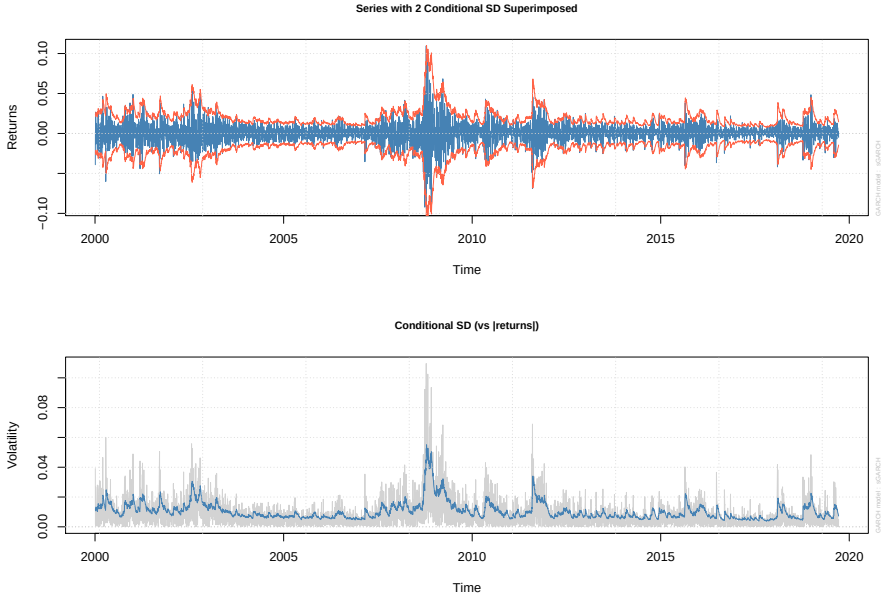


Figura 5.32: Retornos diários S&P 500 e a volatilidade estimada pelo modelo AR(2)+GARCH(1,1) e distribuição Johnson SU

Capítulo 6

Redes Neuronais e o Perceptrão

6.1 Introdução às redes neuronais

Uma rede neuronal artificial (RNA) é um método prático para aprender funções reais, discretas e vetoriais a partir de dados. A inspiração deste método vem das estruturas neuronais dos seres humanos, que aprendem com a experiência. Tal como o cérebro é composto de unidades básicas, denominadas neurónios, uma RNA é composta de várias unidades de processamento, baseados num modelo simples do neurónio. Em particular, estas unidades de processamento recebem um conjunto de valores reais como entrada e produzem um valor real como saída. É importante salientar que o modelo de funcionamento da RNA é bem mais simples que o cérebro.

Estas unidades de processamento podem ser conectadas de tal forma que uma saída de uma unidade pode ser a entrada de outra unidade, formando assim uma rede com várias unidades simples de processamento. As RNAs são a base para uma área da inteligência artificial chamada *Deep Learning*. Esta área tem ganho uma grande popularidade devido às áreas de aplicação, tais como a classificação de milhões de imagens (e.g., Google Images), reconhecimento de voz (e.g., A Siri da Apple), recomendação de vídeos para milhões de utilizadores (e.g., YouTube), e jogos em que se aprende a ganhar de campeões mundiais (e.g., AlphaGo da DeepMind).

6.2 O neurónio biológico e a unidade de processamento de uma rede neuronal

De uma forma simplificada, um neurónio biológico recebe impulsos elétricos vindos de outros neurónios através dos dendritos (ver Figura 6.1). Estes impulsos podem estimular o corpo do neurónio, tanto em termos da sua intensidade ou da sua natureza (i.e., inibitória ou excitatória). Se estes estímulos conjuntamente ultrapassarem um certo limiar, o neurónio dispara um sinal elétrico pelo axônio e outros neurónios recebem este impulso elétrico através das sinapses.

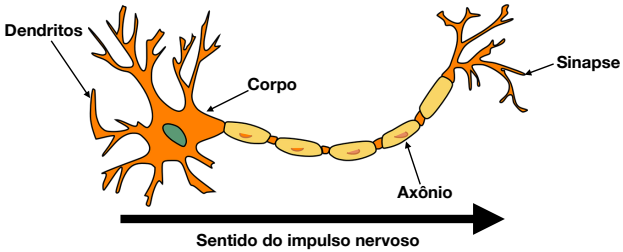


Figura 6.1: O neurónio biológico

A analogia entre o neurónio biológico e uma unidade de processamento de uma rede neuronal é a seguinte. Na Figura 6.2, as entradas $x_1, x_2, ..., x_n$ são valores que são recebidos de outras unidade de processamento, imitando os impulsos elétricos recebidos de outros neurónios. Estas entradas são multiplicadas por pesos $w_1, w_2, ..., w_n$, tentando imitar os estímulos que ocorrem dentro do corpo do neurónio. Por último, a soma ponderada $\sum_{i=0}^{n} w_i x_i$ que passa por uma função de ativação é um modelo simples do disparo elétrico que ocorre no neurónio.

6.3 Perceptrão

O Perceptrão foi um dos primeiros modelos de aprendizagem a partir do qual as redes neuronais surgiram. Esta rede neuronal foi concebida por Rosenblatt (1958) para resolver o problema de classificação binária (e.g., se aparece gato ou um cão numa imagem). Além disso, Rosenblatt utilizou o Perceptrão numa

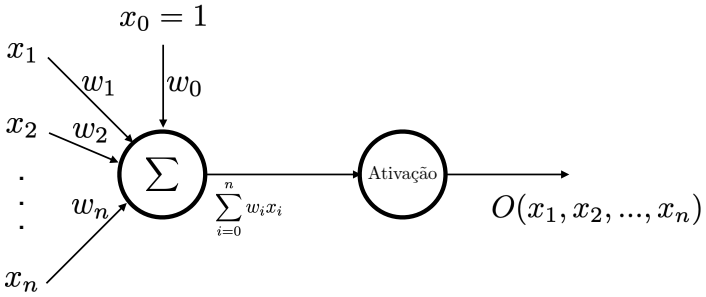


Figura 6.2: A unidade de processamento de uma rede neuronal

máquina capaz de reconhecer imagens, chamada de *Mark 1 perceptron*.

O modelo do Perceptrão utiliza a estrutura da Figura 6.2. É importante salientar que esta rede neuronal não possui unidades que se ligam a outras unidades de processamento. Em particular, o Perceptrão tem as seguintes características:

- As entradas $x_1, x_2, ..., x_n$ da unidade de processamento são valores reais que representam alguma característica ou atributos dos dados (e.g., informação sobre cada pixel da imagem);
- Cada entrada x_i é multiplicada por um peso w_i que representa o quanto esta entrada deve influenciar a unidade de processamento;
- A unidade de processamento possui um viés que é o resultado da multiplicação de uma constante $x_0 = 1$ pelo peso w_0;
- Todas as expressões $w_i x_i$ são somadas e o resultado é uma soma ponderada igual à $\sum_{i=0}^{n} w_i x_i$. Se consideramos que $x_0, x_1, x_2, ..., x_n$ é um vetor x e os pesos $w_0, w_1, w_2, ..., w_n$ é um vetor w, então a soma ponderada pode ser calculada através da seguinte multiplicação de vetores $w^T x$;
- Uma função de ativação determina qual a classe (e.g., gato ou cão) da entrada $x_1, x_2, ..., x_n$. Em particular, a função ativação do perceptrão é uma função degrau (ver Figura 6.3), onde:

$$O(x_1, x_2, ..., x_n) = \begin{cases} 1, \text{se } w^T x > 0 \\ 0, \text{se } w^T x \leq 0 \end{cases} \tag{6.1}$$

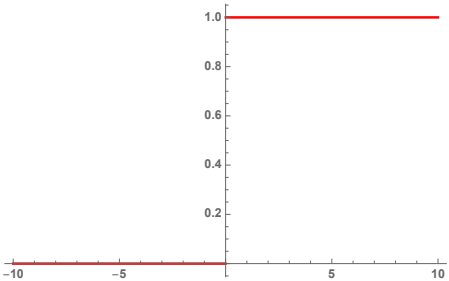


Figura 6.3: A função de ativação do Perceptrão: função degrau

Note que para aprender a função de classificação do perceptrão (i.e., a função $O(x_1, x_2, ..., x_n)$) é preciso um conjunto de exemplos (e.g., imagens de gatos e cães), onde cada exemplo d possui uma entrada $x_1^d, x_2^d, ..., x_n^d$ e sua respectiva classe $y^d \in \{0, 1\}$. Consequentemente, é preciso encontrar os pesos $w_0, w_1, w_2, ..., w_n$ da função $O(x_1, x_2, ..., x_n)$ que permita classificar corretamente todos os exemplos. A seção seguinte apresenta o algoritmo para determinar os pesos da função de classificação através de um conjunto de exemplos.

6.4 O algoritmo de aprendizagem

O algoritmo de aprendizagem do Perceptrão proposto por Rosenblatt foi inspirado no trabalho de Hebb (1949), onde Hebb sugere que a conexão entre neurónios biológicos cresce quando um neurónio dispara um sinal para outro neurónio. Em particular, o algoritmo de aprendizagem do Perceptrão recebe um exemplo por vez e faz uma predição da classe deste exemplo. Ao comparar com a classe pretendida, o algoritmo altera os pesos do Perceptrão quando há um erro entre a classe prevista e a pretendida. Consequentemente, o algoritmo de aprendizagem do Perceptrão atualiza os pesos para cada exemplo com a Equação 6.2 até que todos os exemplos sejam corretamente classificados.

$$w_i = w_i + \eta(y^d - \hat{y}^d)x_i^d \tag{6.2}$$

onde:

- w_i é um peso de uma entrada;

- x_i^d é uma entrada do exemplo d do conjunto de exemplos;
- y^d é a classe correta (ou pretendida) do exemplo d;
- $\hat{y}^d$ é a classe prevista com a função $O(x_1^d, x_2^d, ..., x_n^d)$ do exemplo d;
- η é a taxa de aprendizagem que determina o quanto os pesos devem ser alterados em cada passo do algoritmo de aprendizagem.

Na próxima seção, implementa-se o algoritmo de aprendizagem do Perceptrão em Python.

6.5 Um exemplo do Perceptrão

Esta seção apresenta um exemplo simples para mostrar como Perceptrão aprende uma função de classificação. A função que se pretende aprender funciona com uma porta lógica *AND*.

A Tabela 6.1 apresenta os quatro exemplos de treino para o Perceptrão com duas entradas, nomeadamente x_1^d e x_2^d. O objetivo é implementar o algoritmo de aprendizagem do Perceptrão e assim encontrar os valores dos pesos que permita classificar corretamente todos os exemplos.

Tabela 6.1: Exemplos de treino para o Perceptrão

Exemplos de treino	x_1^d	x_2^d	y^d
$d = 1$	0	0	0
$d = 2$	0	1	0
$d = 3$	1	0	0
$d = 4$	1	1	1

A Figura 6.4 apresenta o código em Python para criar variáveis de configuração do Perceptrão. Em particular, este Perceptrão tem duas entradas e três pesos (i.e., um peso para cada entrada e um para o viés). Além disso, define-se a taxa de aprendizagem e um número máximo de iterações para o algoritmo de treino.

Na Figura 6.5 cria-se uma função que calcula a saída do Perceptrão tal como se vê na Figura 6.2. Note que primeiro se calcula a soma ponderada

Exemplo do Perceptrão

```
import numpy as np

# incializar variaveis de configuracao do Perceptrao
num_entradas = 2
pesos = np.zeros(num_entradas + 1)
tx_aprendizagem = 0.01
iteracoes_max = 1000
```

Figura 6.4: Código em Python para Configurar o Perceptrão

$w^T x$ e depois este resultado passa pela função de ativação (i.e., a função degrau).

Exemplo do Perceptrão

```
# calcular a saida do Perceptrao
def calcular_saida(exemplo):
   somatorio = np.dot(exemplo, np.transpose(pesos[1:]))\
                      + pesos[0]
   if somatorio > 0:
       return 1
   else:
       return 0
```

Figura 6.5: Código em Python para criar uma função que calcula a saída do Perceptrão

A Figura 6.6 apresenta o código que implementa o algoritmo de aprendizagem do Perceptrão da Seção 6.4. A função recebe um conjunto de exemplos e suas respectivas saídas pretendidas (i.e., as classes de cada exemplo). Para cada exemplo, compara-se primeiro a saída do Perceptrão com a saída pretendida e depois altera-se os pesos do Perceptrão quando há um erro, conforme a Equação 6.2. O código termina quando todos os exemplos são corretamente classificados ou um número máximo de iterações é atingido.

Na Figura 6.7 cria-se os exemplos da Tabela 6.1 e chama-se a função *treinar* da Figura 6.6. Em seguida, dois testes são feitos na Figura 6.8 para ver se o Perceptrão é capaz de classificar corretamente dois casos, através da função *calcularSaida*.

Exemplo do Perceptrão

```
# treinar o Perceptrao
def treinar(exemplos_treino, saidas_pretendidas):
   i = 0
   erro = 0
   while True:
       for exemplo, saida_pretendida in \
       zip(exemplos_treino, saidas_pretendidas):
           saida = calcular_saida(exemplo)
           erro += (saida_pretendida - saida)**2
           pesos[1:] += tx_aprendizagem * \
                    (saida_pretendida - saida) * exemplo
           pesos[0] += tx_aprendizagem * \
                    (saida_pretendida - saida)
           i += 1
           if i >= iteracoes_max:
               print("Treino parou por ter atingido o \
                      numero maximo de iteracoes")
               erro = 0
               break
       if erro == 0:
           print("Numero de iteracoes para treinar o \
                  perceptrao = ", i)
           print("Pesos = ", pesos)
           break
       erro = 0
```

Figura 6.6: Código em Python para criar uma função que treina o Perceptrão

Na Figura 6.9 apresenta-se o resultado do treino do Perceptrão, que teve que iterar 24 vezes pelo conjunto de treino para encontrar os pesos que classificam corretamente todos os exemplos. Por fim, os dois casos de teste demonstram que o Perceptrão é capaz de acertar as respectivas classificações dos exemplos.

Por último, a Figura 6.10 mostra um gráfico com os quatros exemplos de treino e uma reta que representa a fronteira de decisão do Perceptrão. A equação da reta desta fronteira é $w_0 + w_1x_1 + w_2x_2 = 0$, onde os pesos são calculados com o método *treinar* da Figura 6.6 . Nesta fronteira, todo ponto abaixo da reta deve ser classificado com $y^d = 0$ e todo ponto acima da reta

Exemplo do Perceptrão

```
#exemplos de treino
exemplos_treino = np.array([[0,0],[0,1],[1,0],[1,1]])
saidas_pretendidas = np.array([[0],[0],[0],[1]])

treinar(exemplos_treino, saidas_pretendidas)
```

Figura 6.7: Código em Python para criar os exemplos de treino e chamar a função de treino do Perceptrão

Exemplo do Perceptrão

```
# Testar o Perceptrao treinado com exemplos
exemplo_teste1 = np.array([1, 1])
print("Exemplo 1 de teste: ", exemplo_teste1)
print("Saida = ", calcular_saida(exemplo_teste1) )

exemplo_teste2 = np.array([0, 1])
print("Exemplo 2 de teste: ", exemplo_teste2)
print("Saida = ", calcular_saida(exemplo_teste2) )
```

Figura 6.8: Código em Python para testar o Perceptrão

Exemplo do Perceptrão

```
Numero de iteracoes para treinar o perceptrao =  24
Pesos =  [-0.02  0.02  0.01]

Exemplo 1 de teste:  [1 1]
Saida =  1
Exemplo 2 de teste:  [0 1]
Saida =  0
```

Figura 6.9: O resultado do treino e teste do Perceptrão

deve ser classificado com $y^d = 1$. Note que no caso de haver mais entradas no Perceptrão, a fronteira de decisão torna-se um hiperplano (i.e., $w^T x = 0$) e é por isso que o Perceptrão é apenas capaz de classificar problemas que são

linearmente separáveis.

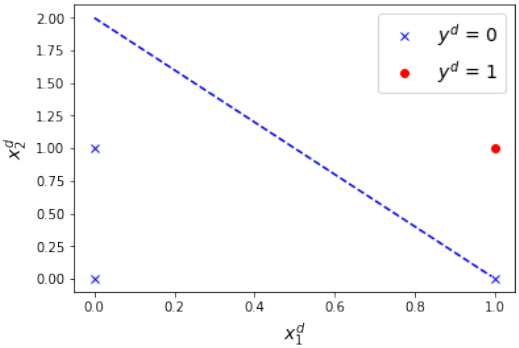


Figura 6.10: A fronteira de decisão do Perceptrão

Capítulo 7

Perceptrão Multicamadas

7.1 Introdução

Este capítulo tem como objetivo apresentar o Perceptrão multicamadas. Esta é uma das redes neuronais mais utilizadas para aprender funções complexas, que são capazes de resolver problemas com um alto grau de não linearidade. Além disso, estas redes neuronais e o seu método de aprendizagem são a base para a área chamada *Deep Learning*.

Na primeira seção é feito uma introdução à arquitetura do Perceptrão multicamadas. Nas duas seções seguintes, o algoritmo de aprendizagem do Perceptrão multicamadas é apresentado junto com um exemplo em Python. Por último, apresenta-se um resumo sobre a área *Deep Learning*.

7.2 Perceptrão multicamadas

O Perceptrão multicamadas, ou rede neuronal *feedforward*, é uma rede com vários Perceptrões que estão densamente ligadas. A única diferença entra a unidade de processamento do Perceptrão e do Perceptrão multicamadas é a escolha da função de ativação. Enquanto que no Perceptrão utiliza-se a função degrau, no Perceptrão multicamadas pode-se utilizar diferentes funções que introduzem uma componente não-linear nesta rede neuronal.

A Figura 7.1 apresenta a arquitetura do Perceptrão multicamadas com as

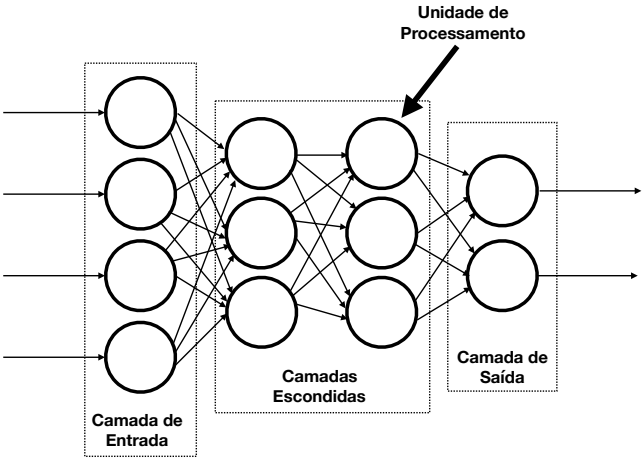


Figura 7.1: A arquitetura do Perceptrão multicamadas

seguintes características:

- A camada de entrada é a primeira camada da rede, onde cada unidade recebe uma entrada. A saída de cada unidade é então ligada a todas as unidades da camada seguinte.

- As camadas escondidas têm uma ou mais camadas que estão totalmente ligadas, ou seja, a saída de uma unidade de processamento está sempre ligada a todas as unidades da camada seguinte. Note no exemplo da Figura 7.1 que há duas camadas nas camadas escondidas.

- A camada de saída é a última camada e pode conter várias unidades de processamento. No entanto, cada unidade de processamento nesta camada tem apenas uma saída, que é uma saída da rede neuronal.

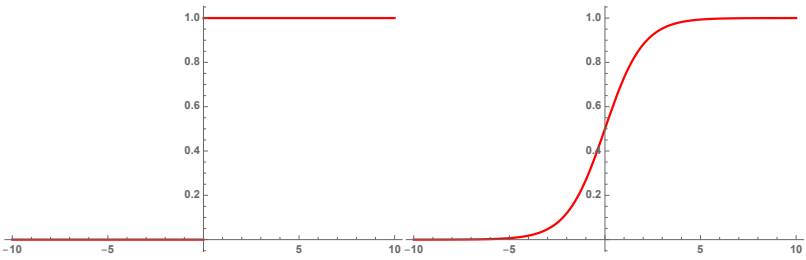
Esta rede neuronal é capaz de aprender funções bem mais complexas que o Perceptrão. Por exemplo, devido às várias camadas, a rede é capaz de aprender funções não lineares, como por exemplo, classificar problemas que não são linearmente separáveis.

7.3 O algoritmo de aprendizagem

O algoritmo de aprendizagem do Perceptrão multicamadas utiliza um método chamado *Backpropagation* (Rumelhart *et al.*, 1986) para aprender os pesos da rede. O *Backpropagation* utiliza o método do gradiente descendente, com o objetivo de minimizar o erro entre a saída da rede e a saída pretendida. Esta seção apresenta o algoritmo de aprendizagem do Perceptrão multicamadas[1], onde *Backpropagation* utiliza uma função de erro igual à soma dos erros quadráticos e uma função de ativação igual à função logística. No entanto, outras funções de erro e de ativação podem ser utilizadas também com o mesmo método.

7.3.1 Unidade de processamento com a função logística

O Perceptrão multicamadas também utiliza a mesma unidade de processamento da Figura 6.2. No entanto, para tornar esta rede neuronal capaz de aprender funções não lineares com o método do gradiente descendente, é preciso escolher uma função de ativação diferente da função degrau (ver Figura 7.2a). Uma alternativa é a função logística (ver Figura 7.2b) que parece com a função degrau mas é diferenciável em todos os pontos do seu domínio.



(a) Função degrau **(b)** Função logística

Figura 7.2: Funções de ativação

Tal como o Perceptrão, a unidade de processamento calcula primeiro a soma ponderada $\sum_{i=0}^{n} w_i x_i$ e depois utiliza este resultado como entrada da função logística. A função logística é definida na Equação 7.1 e sua derivada

[1] Esta seção utiliza uma notação muito parecida com a notação do livro *Machine Learning* do Mitchell (1997)

é apresentada na Equação 7.2 . Estas equações serão utilizadas no método do gradiente descendente.

$$\sigma(y) = \frac{1}{1 + e^{-y}} \tag{7.1}$$

$$\frac{d\sigma(y)}{dy} = \sigma(y)(1 - \sigma(y)) \tag{7.2}$$

Há outras funções de ativação sendo utilizadas atualmente nas redes neuronais com bastante sucesso, tais como a função $ReLU$. No entanto, por questões didáticas, esta seção apresenta apenas a derivação do método do gradiente descendente com a função logística.

7.3.2 O método *Backpropagation*

O método *Backpropagation* aprende os pesos do Perceptrão multicamadas através do método do gradiente descendente. O objetivo é encontrar os pesos que conseguem minimizar o erro entre a saída da rede e a saída pretendida de todos os exemplos de um conjunto de treino.

Na prática, utiliza-se uma versão do método do gradiente descendente chamada gradiente descendente estocástico (Mitchell, 1997) . Enquanto que o gradiente descendente original apenas atualiza os pesos depois de calcular o erro com todos os exemplos, o gradiente descendente estocástico faz esta atualização de forma incremental, onde os pesos são atualizados após o cálculo do erro de cada exemplo. Desta forma, o gradiente descendente estocástico costuma ser mais eficiente em termos computacionais e tende a convergir mais rápido para um mínimo local.

Em particular, o método do gradiente descendente estocástico tenta resolver o problema de otimização utilizando o seguinte procedimento. Para cada exemplo, este método atualiza o vetor de pesos w_u de cada unidade de processamento u na direção da maior descida ao longo da curva de erro. Esta direção é calculada com negativo do gradiente[2] em relação ao vetor de pesos, onde o gradiente é definido pela Equação 7.3.

[2]O gradiente indica a direção que produz o maior incremento do erro. Como é preciso a direção com o maior decremento do erro, utiliza-se o negativo do gradiente

$$\nabla E(w_u) \equiv \left[\frac{\partial E}{\partial w_{u,0}}, \frac{\partial E}{\partial w_{u,1}}, ..., \frac{\partial E}{\partial w_{u,n}}\right] \tag{7.3}$$

A atualização dos pesos é feita com as Equações 7.4 e 7.5.

$$w_u = w_u + \Delta w_u \tag{7.4}$$

$$\Delta w_u = -\eta \nabla E(w_u) \tag{7.5}$$

Note na Equação 7.5 que o negativo do gradiente é multiplicado por uma constante positiva η que se chama a taxa de aprendizagem e determina o tamanho do passo na atualização dos pesos. As Equações 7.6 e 7.7 mostram como é feito a atualização de cada peso $w_{u,i}$ da unidade u.

$$w_{u,i} = w_{u,i} + \Delta w_{u,i} \tag{7.6}$$

$$\Delta w_{u,i} = -\eta \frac{\partial E}{\partial w_{u,i}} \tag{7.7}$$

O restante desta seção apresenta a derivação do método do gradiente descendente estocástico. Para isso, utiliza-se algumas variáveis que estão na Tabela 7.1.

O primeiro passo para implementar o método do gradiente descendente estocástico é o cálculo do gradiente. Em particular, é preciso calcular cada componente do gradiente:

$$\frac{\partial E_d}{\partial w_{u,i}}$$

onde E_d é o erro entre a saída da rede e a saída pretendida do exemplo d, para todas as saídas da rede. A Equação 7.8 apresenta a definição do erro E_d.

$x_{u,i}$	a entrada i da unidade u
$w_{u,i}$	o peso associado a entrada i da unidade u
$soma_u$	a soma ponderada $\sum_i w_{u,i}x_{u,i}$ da unidade u
o_u	a saída da unidade u
t_u	a saída pretendida da unidade u
σ	a função logística
$saidas$	as unidades de processamento na camada de saída
$seguinte(u)$	as unidades de processamento na camada seguinte à unidade u, i.e., as unidades que estão ligadas à saída da unidade u

Tabela 7.1: Variáveis utilizadas na derivação do método do gradiente descendente estocástico

$$E_d \equiv \frac{1}{2} \sum_{k \in saidas} (t_k - o_k)^2 \tag{7.8}$$

Note que o peso $w_{u,i}$ apenas influencia a rede através da soma ponderada $soma_u = \sum_i w_{u,i}x_{u,i}$, logo é possível utilizar a regra da cadeia para calcular o componente do gradiente:

$$\frac{\partial E_d}{\partial w_{u,i}} = \frac{\partial E_d}{\partial soma_u} \frac{\partial soma_u}{\partial w_{u,i}} \tag{7.9}$$

A última derivada parcial da Equação 7.9 pode ser calculada da seguinte maneira:

$$\frac{\partial soma_u}{\partial w_{u,i}} = \frac{\partial}{\partial w_{u,i}} \Big(\sum_i w_{u,i}x_{u,i} \Big) = x_{u,i} \tag{7.10}$$

Logo a Equação 7.9 pode ser simplificada:

$$\frac{\partial E_d}{\partial w_{u,i}} = \frac{\partial E_d}{\partial soma_u} x_{u,i} \tag{7.11}$$

Em resumo, apenas falta calcular a derivada parcial $\frac{\partial E_d}{\partial soma_u}$ para calcular o gradiente do erro. No entanto esta derivada parcial não é igual para todas as unidades de processamento da rede. Consequentemente, considera-se dois casos, um onde a unidade pertence a camada de saída e outra onde a unidade faz parte das camadas escondidas.

Primeiro caso: Calculando o gradiente para uma unidade da camada de saída

Da mesma forma que $w_{u,i}$ apenas influencia a rede através da soma ponderada $soma_u$, $soma_u$ apenas influencia a rede através da saída da unidade o_u. Logo pode-se utilizar a regra da cadeia para calcular a derivada parcial $\frac{\partial E_d}{\partial soma_u}$:

$$\frac{\partial E_d}{\partial soma_u} = \frac{\partial E_d}{\partial o_u}\frac{\partial o_u}{\partial soma_u} \tag{7.12}$$

A primeira derivada parcial da Equação 7.12 pode ser calculada substituindo E_d pela Equação 7.8:

$$\frac{\partial E_d}{\partial o_u} = \frac{\partial}{\partial o_u}\frac{1}{2}\sum_{k \in saidas}(t_k - o_k)^2 \tag{7.13}$$

Pode-se simplificar a Equação 7.13 porque todas as derivadas de $(t_k - o_k)^2$ são iguais a zero, exceto quando $k = u$.

$$\frac{\partial E_d}{\partial o_u} = \frac{\partial}{\partial o_u}\frac{1}{2}(t_u - o_u)^2 = -(t_u - o_u) \tag{7.14}$$

A segunda derivada parcial da Equação 7.12 (i.e., $\frac{\partial o_u}{\partial soma_u}$) é a derivada da função logística porque o_u é igual à $\sigma(soma_u)$. Logo utilizando a Equação 7.2, que é a derivada da função logística, pode-se encontrar esta derivada parcial da seguinte maneira:

$$\frac{\partial o_u}{\partial soma_u} = \frac{\partial}{\partial soma_u}\sigma(soma_u) = \sigma(soma_u)(1 - \sigma(soma_u)) \tag{7.15}$$

$$\frac{\partial o_u}{\partial soma_u} = o_u(1 - o_u) \tag{7.16}$$

Agora é possível calcular a derivada parcial $\frac{\partial E_d}{\partial soma_u}$ da Equação 7.12 com as Equações 7.14 e 7.16:

$$\frac{\partial E_d}{\partial soma_u} = \frac{\partial E_d}{\partial o_u}\frac{\partial o_u}{\partial soma_u} = -(t_u - o_u)o_u(1 - o_u) \tag{7.17}$$

Consequentemente, a Equação 7.7, que atualiza os pesos, pode ser substituída da seguinte forma para as unidades da camada de saída:

$$\Delta w_{u,i} = -\eta\frac{\partial E}{\partial w_{u,i}} = -\eta\frac{\partial E_d}{\partial soma_u}\frac{\partial soma_u}{\partial w_{u,i}} \tag{7.18}$$

Com o resultado da Equação 7.11, é possível simplificar a Equação 7.18:

$$\Delta w_{u,i} = -\eta\frac{\partial E}{\partial w_{u,i}} = -\eta\frac{\partial E_d}{\partial soma_u}x_{u,i} \tag{7.19}$$

E com o resultado da Equação 7.17, a Equação 7.19 pode ser simplificada para gerar o seguinte resultado:

$$\Delta w_{u,i} = -\eta\frac{\partial E_d}{\partial soma_u}x_{u,i} = \eta(t_u - o_u)o_u(1 - o_u)x_{u,i} \tag{7.20}$$

Logo, para cada exemplo do conjunto de treino, a Equação 7.20 é utilizada para atualizar os pesos de uma unidade da camada de saída para que se possa diminuir o erro E_d. O valor $(t_u - o_u)o_u(1 - o_u)$ é conhecido como termo do erro da unidade u. Além disso, este termo de erro é representado pela variável δ_u e o seu valor é igual à $-\frac{\partial E_d}{\partial soma_u}$. Logo é possível reescrever a Equação 7.20 desta maneira:

$$\Delta w_{u,i} = \eta - \frac{\partial E_d}{\partial soma_u}x_{u,i} = \eta\delta_u x_{u,i} \tag{7.21}$$

$$\delta_u = (t_u - o_u)o_u(1 - o_u) \tag{7.22}$$

Segundo caso: Calculando o gradiente para uma unidade de uma camada escondida

Neste segundo caso, o objetivo é calcular a derivada parcial $\frac{\partial E_d}{\partial soma_u}$ para uma unidade de uma camada escondida e consequentemente a Equação que atualiza os pesos. Através da Figura 7.1, note que uma unidade de uma camada escondida apenas influencia as unidades da camada seguinte; em outras palavras, a saída de uma unidade de uma camada escondida liga-se apenas às entradas das unidades da camada seguinte. Este grupo de unidades é chamado de $seguinte(u)$. Além disso, o valor de $soma_u$ da unidade u apenas influencia a rede, e consequentemente o erro E_d, através das entradas das unidades da camada seguinte. Logo é possível calcular a derivada parcial da seguinte maneira:

$$\frac{\partial E_d}{\partial soma_u} = \sum_{k \in seguinte(u)} \frac{\partial E_d}{\partial soma_k} \frac{\partial soma_k}{\partial soma_u} = \sum_{k \in seguinte(u)} -\delta_k \frac{\partial soma_k}{\partial soma_u} \quad (7.23)$$

A derivada parcial $\frac{\partial E_d}{\partial soma_k}$ é substituída por $-\delta_k$, onde δ_k é o termo de erro de uma unidade k da camada seguinte. Além disso, $soma_u$ apenas influencia a rede através da saída da unidade o_u. Logo pode-se utilizar a regra da cadeia para calcular a derivada parcial $\frac{\partial soma_k}{\partial soma_u}$:

$$\frac{\partial E_d}{\partial soma_u} = \sum_{k \in seguinte(u)} -\delta_k \frac{\partial soma_k}{\partial soma_u} = \sum_{k \in seguinte(u)} -\delta_k \frac{\partial soma_k}{\partial o_u} \frac{\partial o_u}{\partial soma_u} \quad (7.24)$$

Lembre-se que $soma_k$ é igual à $\sum_i w_{k,i} x_{k,i}$, onde os valores $x_{k,i}$ vem das saídas das unidades da mesma camada da unidade u. Logo, ao calcular $\frac{\partial soma_k}{\partial o_u}$, a derivada de todos os termos $w_{k,i} x_{k,i}$ são iguais a zero, exceto quando $x_{k,i}$ for igual à saída o_u. Consequentemente é possível simplificar a Equação 7.24:

$$\frac{\partial E_d}{\partial soma_u} = \sum_{k \in seguinte(u)} -\delta_k w_{k,u} \frac{\partial o_u}{\partial soma_u} \quad (7.25)$$

Por último a derivada parcial $\frac{\partial o_u}{\partial soma_u}$ é a derivada da função logística, logo é possível simplificar a Equação 7.25:

$$\frac{\partial E_d}{\partial soma_u} = \sum_{k \in seguinte(u)} -\delta_k w_{k,u} o_u (1 - o_u) \tag{7.26}$$

Reorganizando os termos e usando δ_u no lugar de $-\frac{\partial E_d}{\partial soma_u}$:

$$\delta_u = o_u(1 - o_u) \sum_{k \in seguinte(u)} \delta_k w_{k,u} \tag{7.27}$$

Note que os exemplos de treino apenas possuem a saída pretendida da rede (i.e., as saídas pretendidas da camada de saída) e não uma saída pretendida para uma unidade da camada escondida. Consequentemente, não é possível calcular o termo de erro tal como foi feito na unidade da camada de saída. No entanto, o termo de erro de uma unidade escondida pode ser calculado através de uma soma ponderada entre os termos de erros das unidades da camada seguinte δ_k e os pesos $w_{k,u}$ entre as unidades u e k. O peso $w_{k,u}$ pode ser interpretado como grau que o erro da unidade u é responsável pelo erro na saída k. Isto é uma interpretação intuitiva da Equação 7.27.

Tal como no caso da camada de saída, os pesos de uma camada escondida são atualizados com a Equação 7.28.

$$\Delta w_{u,i} = \eta - \frac{\partial E_d}{\partial soma_u} x_{u,i} = \eta \delta_u x_{u,i} \tag{7.28}$$

O Algoritmo 1 mostra como se implementa o *Backpropagation* para uma rede com duas camadas, assumindo que as funções de ativação são funções logísticas. Normalmente, uma rede com uma camada de entrada, J camadas escondidas e uma camada de saída é dita ter $J+1$ camadas, porque a unidade de entrada não é considerada. Por isso que a rede no Algoritmo 1 tema duas camadas.

No Algoritmo 1, cada exemplo é utilizado para que se possa calcular todas as saídas das unidades de processamento. Note que isto só é possível se este cálculo for feito uma camada por vez, indo da esquerda para direita (i.e., da camada de entrada para a saída). Consequentemente, esta etapa chama-se a propagação para frente ou *forward propagation*. Depois é feito o cálculo dos erros. Nesta etapa, também só é possível fazer o cálculo se for feito

uma camada por vez, indo da direita para esquerda (i.e., da camada de saída para a entrada). Por isso, esta etapa chama-se propagação para trás ou *back propagation*. Note que as linhas 7 e 9 do Algoritmo 1 vem das Equações 7.22 e 7.27 respectivamente. Depois destas etapas é possível atualizar os pesos da rede.

Apesar do Algoritmo 1 ser específico para uma rede de duas camadas, é fácil apresentar uma extensão que pode ser generalizada para redes de qualquer número de camadas. A única alteração a ser feita está relacionada com as linhas que calculam o δ nas linhas 8 e 9, onde é feito o cálculo do erro da camada escondida. Em um caso geral, uma unidade r de uma camada m deve ter o seu δ_r calculado com a Equação 7.29, onde é necessário já se ter calculado os valores de δ das unidades da camada $m+1$. Logo a linha 9 deve ser substituída pela Equação 7.29. Além disso, as linhas 8 e 9 (atualizada com a Equação 7.29) devem ser repetidas para todas as camadas da rede, no sentido da última camada escondida até a primeira.

$$\delta_r = o_r(1 - o_r) \sum_{s \in \text{ camada } m+1} w_{s,r}\delta_s \tag{7.29}$$

7.4 Um exemplo do Perceptrão multicamadas

Esta seção apresenta um exemplo simples de como Perceptrão multicamadas aprende uma função capaz de gerar os mesmos resultados de uma porta lógica *XOR*. A Tabela 7.2 apresenta os quatro exemplos de treino com duas entradas x_1^d e x_2^d e uma saída y^d. A porta lógica *XOR* retorna 1 (ou verdade) apenas quando x_1^d e x_2^d possuem valores diferentes.

Tabela 7.2: Exemplos de treino para o Perceptrão multicamadas

Exemplos de treino	x_1^d	x_2^d	y^d
$d = 1$	0	0	0
$d = 2$	0	1	1
$d = 3$	1	0	1
$d = 4$	1	1	0

Algoritmo 1 O algoritmo Backpropagation para duas camadas

Entrada: D: conjunto de treino. Cada exemplo $d \in D$ possui um par $< x, t >$, onde x é um vetor com as entradas da rede e t é um vetor com os valores pretendidos
η: taxa de aprendizagem (e.g., $\eta = 0.01$)
n_{ent}: o número de entradas da rede (ou número de unidades na camada de entrada)
n_s: o número de saídas da rede (ou número de unidades na camada de saída)
n_{esc}: o número de unidades na camada escondida

início

criar uma rede com n_{ent} unidades na camada de entrada, n_{esc} unidades na camada escondida e n_s unidades na camada de saída;
inicialize os valores dos pesos com valores aleatórios pequenos (por exemplo, entre -0.05 e 0.05);
repita
  para cada $< x, t > \in D$ **faça**
    `// propagar a entrada pela rede para frente`
    `// (da camada de entrada para a saída)`
    Com a entrada x, calcule as saídas o_u de cada unidade da rede;
    `// propagar os erros pela rede para trás`
    `// (da camada de saída para a entrada)`
    para cada *unidade k da camada de saída* **faça**
      $\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$
    fim
    para cada *unidade h da camada escondida* **faça**
      $\delta_h \Leftarrow o_h(1 - o_h) \sum_{k \in saidas} w_{k,h}\delta_k$
    fim
    para cada *peso* $w_{j,i}$ **faça**
      $w_{j,i} \leftarrow w_{j,i} + \eta\delta_j x_{j,i}$
    fim
  fim
até *a condição de paragem ser verdadeira*;
fim

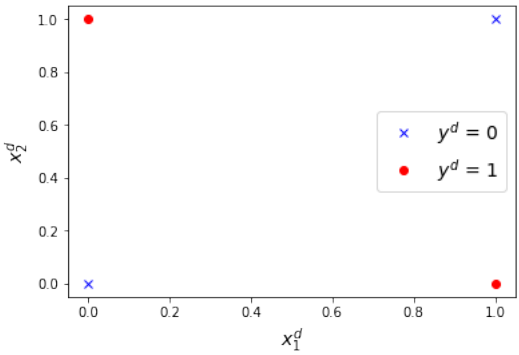


Figura 7.3: O problema de classificação de uma porta lógica *XOR*

Para este problema, o objetivo da rede neuronal é aprender a classificar as entradas de acordo com a Tabela 6.1, muito parecido com o problema da Seção 6.5. Então por que não se utiliza o Perceptrão para este problema? A resposta é simples; este problema de classificação não é linearmente separável, tal como se pode ver na Figura 7.3. Note que não é possível traçar uma reta que seja capaz de separar as duas classes, tal como foi feito na Figura 6.10.

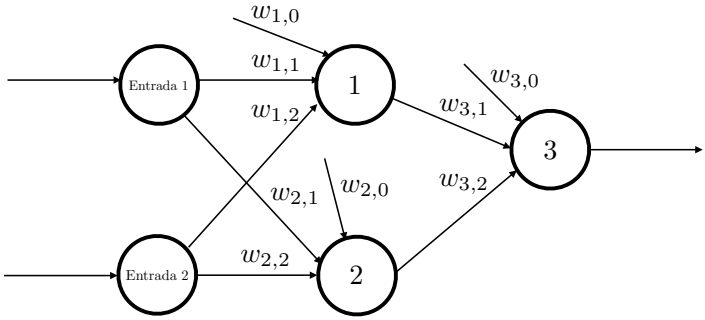


Figura 7.4: A arquitetura da rede para o problema de classificação de uma porta lógica *XOR*

Consequentemente, para resolver este problema, é preciso utilizar uma rede com duas camadas, tal como se pode ver na Figura 7.4. Em particular, há duas unidades na camada de entrada, duas unidades na camada escondida e uma unidade na camada de saída.

A Figura 7.5 apresenta o código para criar variáveis que guardam os exem-

Exemplo do Perceptrão multicamadas

```
import numpy as np

# funcoes uteis para o Backpropagation
def flogistica (x):
    return 1/(1 + np.exp(-x))

def derivada_flogistica(x):
    return x * (1 - x)

#exemplos de treino
entradas = np.array([[0,0],[0,1],[1,0],[1,1]])
saidas_pretendidas = np.array([[0],[1],[1],[0]])
entradas_com_vies = np.concatenate( \
                   (np.ones((entradas.shape[0],1)), \
                    entradas), axis=1)
```

Figura 7.5: Código em Python para criar variáveis com os exemplos de treino e funções úteis para o *Backpropagation*

plos da Tabela 7.2, além de criar a função logística e sua derivada para o *Backpropagation*. Já a Figura 7.6 mostra como configurar a rede com duas unidades de entrada, duas unidades na camada escondida e uma unidade na camada de saída. Note que os pesos de todas as unidades começam com valores aleatórios e a taxa de aprendizagem é 0,1.

No código da Figura 7.7, o *Backpropagation* é implementado em Python. Em cada iteração, o código faz a propagação da entrada para frente, para que se possa calcular a saída de todas as unidades de processamento. Note que por questões de eficiência, a propagação para frente é feita com todos os exemplos ao mesmo tempo, utilizando cálculo matricial[3]. Nas restantes linhas de código, os termos de erro (i.e. δ) são calculados e atualizam-se os pesos com cálculo matricial.

O código da Figura 7.8 imprime os valores finais dos pesos da rede e a saída da rede para cada exemplo da Tabela 7.2. Na Figura 7.9, apresenta-se as saídas da rede neuronal para os exemplos [0,0], [0,1], [1,0] e [1,1], que são [0.01316838], [0.98875665], [0.98874267] e [0.01157947] respectivamente. Note

[3]O código em Python utiliza uma biblioteca para vetorização chamada *NumPy* para estes cálculos matriciais.

Exemplo do Perceptrão multicamadas

```
#configuracao da rede
num_uentrada = 2
num_uescondida = 2
num_usaida = 1

#pesos unidade 1
pesos1 = np.random.uniform(size=(num_uentrada+1,1))
#pesos unidade 2
pesos2 = np.random.uniform(size=(num_uentrada+1,1))
#pesos unidade 3
pesos3 = np.random.uniform(size=(num_uescondida+1,1))

iteracoes = 100000
tx_aprendizagem = 0.1
```

Figura 7.6: Código em Python para configurar o Perceptrão multicamadas

que os resultados não são exatamente iguais aos resultados pretendidos, mas estão muito próximos. Isto ocorre por causa da função logística e na prática aplica-se um arredondamento para que se obtenha os resultados pretendidos.

Exemplo do Perceptrão multicamadas

```
#Backpropagation
for _ in range(iteracoes):
    #Propagacao para frente
    ativacao_u1 = np.dot(entradas_com_vies,pesos1)
    saida_u1 = flogistica(ativacao_u1)
    ativacao_u2 = np.dot(entradas_com_vies,pesos2)
    saida_u2 = flogistica(ativacao_u2)
    entradas_u3 = np.concatenate((saida_u1,saida_u2),axis=1)
    entradas_u3_com_vies = np.concatenate( \
                    (np.ones((entradas_u3.shape[0],1)), \
                     entradas_u3),axis=1)
    ativacao_u3 = np.dot(entradas_u3_com_vies,pesos3)
    saida_u3 = flogistica(ativacao_u3)

    #Propagacao do erro para tras
    termo_erro_u3 = derivada_flogistica(saida_u3) * \
                    (saidas_pretendidas - saida_u3)
    termo_erro_u2 = derivada_flogistica(saida_u2) * \
                    (pesos3[2] * termo_erro_u3)
    termo_erro_u1 = derivada_flogistica(saida_u1) * \
                   (pesos3[1] * termo_erro_u3)

    #Atualizacao dos pesos
    delta_pesos3 = tx_aprendizagem * \
              np.dot(termo_erro_u3.T,entradas_u3_com_vies)
    pesos3 += delta_pesos3.T
    delta_pesos2 = tx_aprendizagem * \
              np.dot(termo_erro_u2.T,entradas_com_vies)
    pesos2 += delta_pesos2.T
    delta_pesos1 = tx_aprendizagem * \
              np.dot(termo_erro_u1.T,entradas_com_vies)
    pesos1 += delta_pesos1.T
```

Figura 7.7: Código em Python para treinar o Perceptrão multicamadas

Exemplo do Perceptrão multicamadas

```
print("Pesos finais da Unidade 1: ",end='')
print(*pesos1)
print("Pesos finais da Unidade 2: ",end='')
print(*pesos2)
print("Pesos finais da Unidade 3: ",end='')
print(*pesos3)

print("\nSaida depois de 100,000 iteracoes: ", end='')
print(*saida_u3)
```

Figura 7.8: O código em Python para imprimir o resultado do *Backpropagation*

Exemplo do Perceptrão multicamadas

```
Pesos finais da Unidade 1:
[-7.36430869] [4.79787826] [4.80242532]
Pesos finais da Unidade 2:
[-3.02615447] [6.72370628] [6.74273792]
Pesos finais da Unidade 3:
[-4.78601251] [-11.02146821] [10.29661965]

Saida da rede neuronal depois de 100,000 iteracoes:
[0.01316838] [0.98875665] [0.98874267] [0.01157947]
```

Figura 7.9: Resultado do *Backpropagation*

7.5 *Deep Learning*

Aprendizagem (*machine learning* em inglês) é uma área da inteligência artificial que cria sistemas que são capazes de adquirir conhecimento a partir de dados. Por exemplo, a técnica *naive Bayes* (Mitchell, 1997) é um classificador capaz de aprender como separar e-mails legítimos de e-mails *spam*.

No entanto, muitas técnicas de aprendizagem dependem de uma boa representação dos dados. Consequentemente, durante muitas décadas, as técnicas de aprendizagem tinham dificuldade de aprender a partir de dados não tratados (i.e., sem nenhum pré-processamento). Em muitos casos, as técnicas de aprendizagem só obtinham uma boa performance se houvesse um bom conhecimento do domínio e se fosse feita uma boa engenharia para selecionar entradas e transformar os dados. Assim, este trabalho de engenharia era capaz de gerar uma representação razoável para a técnica de aprendizagem.

Os métodos *Deep Learning* (LeCun *et al.*, 2015) são redes neuronais que são capazes de aprender esta representação de forma automática. Desta forma, podem aprender diretamente de dados não tratados. Por exemplo, a *convolutional neural network* é um tipo de rede neuronal *feedforward* que é capaz de aprender vários níveis de representações através de camadas *convolutional* e camadas de um Perceptrão multicamadas. Cada camada pode ser responsável por amplificar aspectos importantes da entrada e suprimir os aspectos irrelevantes como se fossem filtros. Tal como no Perceptrão multicamadas, estas camadas tem pesos que são aprendidos com o método *Backpropagation*.

Também há outras redes neuronais que fazem parte dos métodos *Deep Learning*. Por exemplo, o Perceptrão multicamadas com um número grande de camadas e unidades de processamento também é chamado atualmente de *deep feedforward networks* por muitos trabalhos científicos (e.g., LeCun *et al.* (2015)). Também há um grupo de redes neuronais chamado *recurrent neural networks* que tem sido utilizado com dados sequenciais (e.g., linguagem natural, preço de ações) com bastante sucesso.

Os métodos *Deep Learning* também ganharam grande popularidade nos últimos anos por serem capazes de melhorar a performance de muitos sistemas na área da inteligência artificial, inclusive conseguindo obter melhores resultados que os humanos. Por exemplo, Cireşan *et al.* (2012) desenvolveram em 2012 uma *deep neural network* capaz reconhecer placas de trânsito com uma taxa de reconhecimento de 99,46%. Esta rede neuronal foi capaz de ter uma taxa de reconhecimento melhor que um humano numa competição. Em

2015, o sistema *AlphaGo* (Silver *et al.*, 2016) da *Deep Mind* usou uma *deep neural network* com um método de busca em árvores e foi capaz de vencer um jogador profissional do jogo *Go*.

Capítulo 8

Funções de Ativação e Erro

8.1 Introdução

Esta capítulo apresenta as funções de ativação e funções de erro mais utilizadas nas redes neuronais. Como há várias opções de funções de ativação e funções de erro, este capítulo discute as vantagens e desvantagens de cada opção.

8.2 Funções de ativação

A função de ativação de uma unidade de processamento é inspirada no funcionamento de um neurónio biológico. Lembre-se que um neurónio biológico recebe impulsos elétricos vindos de outros neurónios (ver Figura 8.1) e se estes estímulos conjuntamente ultrapassarem um certo limiar, o neurónio dispara um sinal elétrico para outros neurónios. Em particular, a função de ativação modela o disparo elétrico quando os sinais recebidos estão acima de um limiar.

Inspirado nesta analogia, Rosenblatt utilizou a função degrau (ver Figura 8.2) para o Perceptrão porque é um função que imita um disparo, ou seja uma saída igual à 1, quando a soma ponderada está acima de um limiar, ou $\sum_{i=0}^{n} w_i x_i > 0$. Quando a soma ponderada está abaixo do limiar, ou seja $\sum_{i=0}^{n} w_i x_i \leq 0$, o disparo não ocorre, ou uma saída igual à 0. No entanto, a função degrau não pode ser utilizada com o método do gradiente descendente

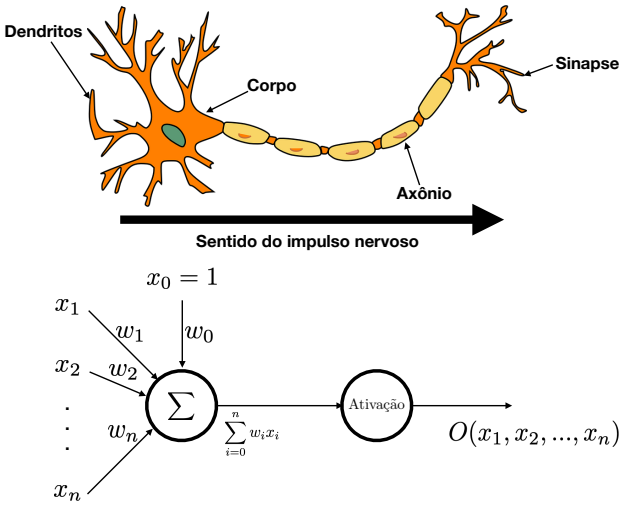


Figura 8.1: O neurónio biológico e uma unidade de processamento de uma rede neuronal

do *Backpropagation* porque a função não é diferenciável no ponto x igual à 0 e a derivada nos outros pontos é igual 0. Consequentemente, o método do gradiente descendente não conseguiria atualizar os pesos da rede.

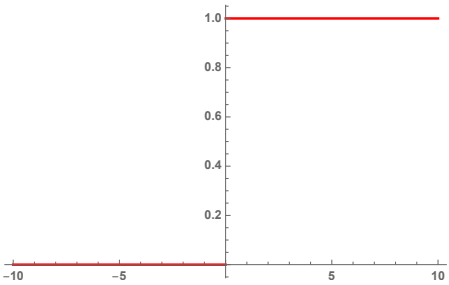


Figura 8.2: A função de ativação do Perceptrão: função degrau

A função logística é uma opção interessante para substituir a função degrau por vários motivos. Primeiro, o formato (ver Figura 8.3) é parecido com a função degrau, então continua a ter um comportamento semelhante ao disparo de um neurónio. Segundo, a função é diferenciável em todos os pontos do seu domínio. Terceiro, quando a soma ponderada é maior que 2 ou menor que -2,

os valores da função estão muito próximos de 1 ou 0, respectivamente. Isso gera previsões mais precisas.

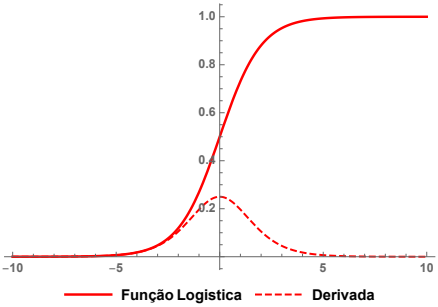


Figura 8.3: Função de ativação: função logística e derivada

No entanto, a função logística também apresenta alguns problemas. Primeiro, para redes com muitas unidades de processamento e muitas camadas, o cálculo das saídas e o processo de aprendizagem são ineficientes em termos computacionais porque a função *exp()* das linguagens de programação não é eficiente. Segundo, esta função de ativação funciona bem para problemas de classificação mas não diretamente para problemas de regressão[1]. Terceiro, esta função de ativação pode sofrer de um problema chamado *Vanishing Gradient Problem* quando há um número grande de camadas. Este problema ocorre por causa da derivada da função logística que é muito próxima de zero para valores maiores do que 4 ou menores do que -4 (ver Figura 8.3). Como o *Backpropagation* utiliza estas derivadas e regra da cadeia para calcular os gradientes do erro para cada camada, a multiplicação de muitos números pequenos gera uma atualização muito pequena dos pesos. Esta atualização muito pequena dos pesos é mais agravante nas primeiras camadas porque são as últimas a serem atualizadas com o *Backpropagation*. Consequentemente, a rede pode ter problemas para aprender.

Outra função de ativação utilizada nas redes neuronais é a tangente hiperbólica (ver Figura 8.4), que é definida da seguinte maneira:

[1]Para as técnicas de aprendizagem, um problema de regressão é quando um modelo tem que aprender uma função que tem como saída um valor numérico (sem limites). O problema de classificação é quando um modelo tem que aprender uma função que tem como saída uma classe ou um rótulo.

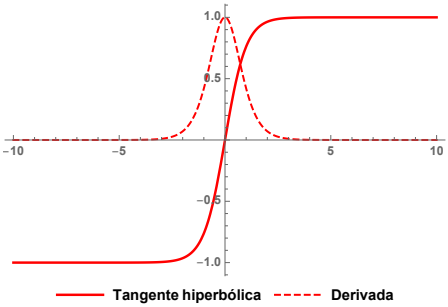


Figura 8.4: Função de ativação: tangente hiperbólica e derivada

$$tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{8.1}$$

Comparando com a função logística, a tangente hiperbólica tem um formato parecido mas agora os valores variam entre -1 e 1. Uma vantagem em relação à função logística é que esta função tem um gradiente mais forte por causa das derivadas maiores, tal como se pode ver na Figura 8.4. Isto torna o *Backpropagation* mais eficiente no processo de aprendizagem. As outras vantagens e desvantagens da tangente hiperbólica são muito parecidas com a função logística, incluindo o *Vanishing Gradient Problem*.

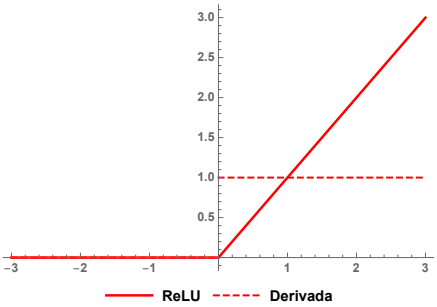


Figura 8.5: Função de ativação: *ReLU* e derivada

A função *Rectified Linear Unit* (*ReLU*) é uma função de ativação que se tornou muito popular com o *Deep Learning*. A Figura 8.5 apresenta a função *ReLU* que é definida pela seguinte equação:

$$ReLU(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{8.2}$$

Uma vantagem da função *ReLU* é que ela é muito eficiente em termos computacionais, então as redes com muitas camadas são mais rápidas para treinar e para serem utilizadas. Note que a derivada da função é igual à 0 para valores de $x < 0$, igual à 1 para valores de $x > 0$, e inexistente para $x = 0$. Para resolver este problema da derivada no ponto $x = 0$, utiliza-se um valor arbitrário para a derivada neste ponto, com por exemplo igual à 1. Desta forma é possível calcular o gradiente e utilizar o *Backpropagation*, com a vantagem de ser bem mais eficiente do que a função logística e a tangente hiperbólica. Além disso, esta função de ativação não tem o *Vanishing Gradient Problem*, o que torna esta função muito útil para redes com muitas camadas.

Outra vantagem desta função ativação é que a rede tende a ter uma ativação esparsa, onde as unidades de processamento que tem a soma ponderada com valores negativos não são ativadas (i.e., a saída é igual à 0). Isto quer dizer que dependendo da entrada (i.e., problema) apenas algumas unidades de processamento são ativadas por vez, tornando as unidades mais especializada ao problema que a rede tem que resolver. Esta propriedade normalmente cria modelos com um melhor poder de predição e menos *overfitting*[2]. Além disso, uma rede com ativação mais esparsa consome menos recursos computacionais.

No entanto, a função *ReLU* pode ter problemas com o *Backpropagation*. Este problema é tipicamente chamado de *Dying ReLU problem*. Note que os pesos de uma unidade de processamento não são atualizados quando a soma ponderada tem valores negativos. Isto ocorre porque a derivada da função *ReLU* é 0 para valores negativos. Se houver muitas unidades com esta situação, a rede pode ter dificuldades para aprender. Mesmo assim, a função *ReLU* tem sido muito utilizada nas camadas escondidas com resultados muito bons, principalmente para redes com muitas camadas e muitas unidades de processamento.

Para tentar resolver este problema do *Dying ReLU problem*, algumas variantes da função *ReLU* foram criadas. Uma destas variantes chama-se *Leaky ReLU* (ver Figura 8.6) com a seguinte definição:

[2] *Overfitting* ocorre tipicamente quando um modelo de predição ajusta-se muito bem ao conjunto de treino mas se mostra pouco eficiente para prever novos resultados.

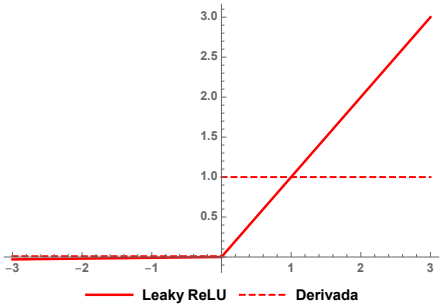


Figura 8.6: Função de ativação: *Leaky ReLU* e derivada

$$LeakyReLU(x) = \begin{cases} 0,01x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{8.3}$$

Note que o *Leaky ReLU* agora tem uma derivada igual à 0,01 quando $x < 0$. Logo, os pesos de uma unidade de processamento agora são atualizados com um valor pequeno quando a soma ponderada tem valores negativos.

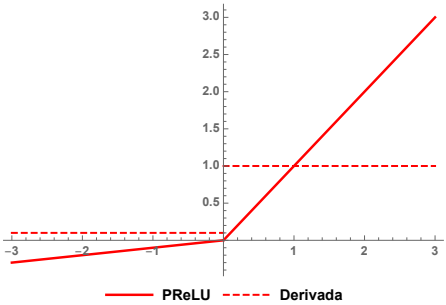


Figura 8.7: Função de ativação: *PReLU* e derivada ($a = 0,1$)

Caso esta função ainda não seja suficiente para resolver o problema, há uma outra variante da função *ReLU* chamada *PReLU* (ver Figura 8.7) com a seguinte definição:

$$PReLU(x) = \begin{cases} ax & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{8.4}$$

onde a é um parâmetro para se escolher a inclinação da reta quando $x < 0$. Este parâmetro a pode ser aprendido com o *Backpropagation*, junto com os pesos, para que se encontre o valor mais apropriado.

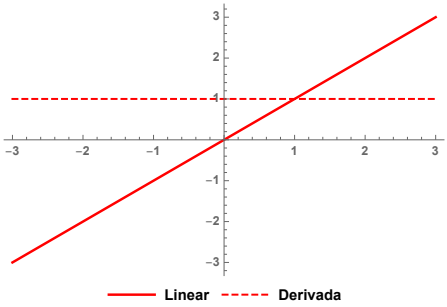


Figura 8.8: Função de ativação: Função linear e derivada

Para a camada de saída, a função linear é tipicamente utilizada para problemas de regressão porque a saída desta função é sem limites (i.e., valores entre $-\infty$ e $+\infty$), tal como se pode ver na Figura 8.8. Um ponto importante sobre esta função de ativação é que ela não pode ser utilizada em toda rede neuronal, porque uma combinação linear de funções lineares é uma outra função linear. Isto tornaria a rede num modelo de regressão linear, enquanto que as funções logística, tangente hiperbólica, $ReLU$ e variantes são não lineares e permitem aprender funções bem mais complexas. Outro motivo que tornaria impossível de utilizar a função linear em toda a rede é que a derivada desta função é uma constante, logo não há relação com a entrada e fica impossível de utilizar o *Backpropagation*.

Outra função de ativação utilizada na camada de saída chama-se *Softmax*. Esta função tem a seguinte definição para calcular a saída i da rede neuronal:

$$Softmax(y_i) = \frac{e^{y_i}}{\sum_{j=0}^{n_{saidas}} e^{y_j}} \tag{8.5}$$

onde y_i é a soma ponderada de uma unidade de processamento i da camada de saída. Note que o denominador da equação tem a finalidade de normalizar a saída, logo a soma das saídas é igual à 1. Outra propriedade interessante é que as unidades de saída com y_i mais altos ficam com valores de $Softmax(y_i)$ mais altos por causa da exponencial na equação. Esta função de ativação é

utilizada, por exemplo, quando a rede quer aprender uma função que tem como saída um distribuição de probabilidade.

8.3 Funções de erro

Para treinar uma rede neuronal, um dos primeiro passos deste processo é a definição de um problema de otimização. O objetivo deste problema de otimização é minimizar o erro entre a saída da rede e a saída pretendida, tal como foi apresentado na Seção 7.3.2. Após esta definição do problema de otimização, é possível utilizar métodos como o gradiente descendente estocástico para aprender os pesos da rede que minimizam a função de erro.

A função que minimiza o erro entre a saída da rede e a saída pretendida é chamada de função de erro. Dependendo do problema que a rede tem que aprender (i.e., um problema de regressão ou classificação), há funções de erros mais adequadas para cada opção. Esta seção apresenta algumas das funções de erro mais relevantes para as redes neuronais.

8.3.1 Regressão

Um problema de regressão consiste em prever um valor real. Por exemplo, a previsão do preço de uma ação no futuro pode ser considerada um problema de regressão. O objetivo desta seção é apresentar algumas das funções de erros mais utilizadas no Perceptrão multicamadas para problemas de regressão e o seu respectivo código em Python.

Funções de erro

```
from sklearn.datasets import make_regression
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import SGD
from matplotlib import pyplot
```

Figura 8.9: O código em Python para importar as bibliotecas

O código nesta seção utiliza uma biblioteca chamada *Keras*[3] para implementar o Perceptrão multicamadas, a biblioteca *scikit-learn*[4] para gerar e tratar os dados, e a biblioteca *Matplotlib*[5] para criar os gráficos. Consequentemente, o código na Figura 8.9 importa todas estas bibliotecas.

Funções de erro

```
# gerar dados para regressao
X, y = make_regression(n_samples=1000, n_features=1, \
                       noise=10, random_state=1)
pyplot.title('Pontos para regressao')
pyplot.plot(X,y,'ro')
pyplot.show()
```

Figura 8.10: O código em Python para gerar dados para regressão

A Figura 8.11 apresenta um conjunto de dados que foi gerado com o código da Figura 8.10. Note que o código utiliza a função *make_regression* da biblioteca *scikit-learn* para gerar 1000 exemplos aleatórios.

Figura 8.11: Pontos para regressão

Para facilitar o processo de aprendizagem do Perceptrão multicamadas, utiliza-se uma função da biblioteca *scikit-learn* para padronizar os dados. O código da Figura 8.12 é utilizado para padronizar os dados e a Figura 8.13 apresenta o resultado. Note que a função *StandardScaler()* transforma os dados para que a média seja próxima de 0 e o desvio padrão seja próximo de 1.

[3]https://keras.io/
[4]https://scikit-learn.org/
[5]https://matplotlib.org

Funções de erro

```
# padronizar os dados
X = StandardScaler().fit_transform(X)
y = StandardScaler().fit_transform(y.reshape(len(y),1))[:,0]

pyplot.title('Pontos padronizados para regressao')
pyplot.plot(X,y,'ro')
pyplot.show()
```

Figura 8.12: O código em Python para padronizar os dados

Figura 8.13: Pontos padronizados para regressão

Funções de erro

```
# dividir dados para treino e teste
n_dados_treino = 800
treino_X,teste_X = X[:n_dados_treino,:],X[n_dados_treino:,:]
treino_y,teste_y = y[:n_dados_treino],y[n_dados_treino:]
```

Figura 8.14: O código em Python para dividir os dados

Após o processo de padronização, o código da Figura 8.14 divide os dados em duas partes, nomeadamente o conjunto de treino e o conjunto de teste. Note que 800 pontos são separados para treinar o Perceptrão multicamadas e 200 pontos para testar a rede neuronal.

Em seguida, utiliza-se a biblioteca *Keras* para definir o modelo do Perceptrão multicamadas. O código da Figura 8.15 cria uma rede neuronal com

Funções de erro

```
# definir modelo do Perceptrao multicamadas
model = Sequential()
model.add(Dense(5, input_dim=1, activation='relu', \
                kernel_initializer='he_uniform'))
model.add(Dense(1, activation='linear'))
opt = SGD(lr=0.01, momentum=0.9)
model.compile(loss='mean_squared_error', optimizer=opt)
```

Figura 8.15: O código em Python para definir o modelo do Perceptrão multicamadas utilizando o erro quadrático médio

a seguinte especificação:

- uma camada de entrada com 1 unidade de processamento;
- uma camada escondida com 5 unidades de processamento que utilizam a função de ativação $ReLU$;
- uma camada de saída com 1 unidade de processamento que utiliza a função de ativação linear.

O código da Figura 8.15 também define que o método do gradiente descendente estocástico (i.e., a função *SGD* do *Keras*) deve ser utilizado para treinar a rede neuronal. Além disso, a última linha define a função de erro (i.e., *loss='mean_squared_error'*) utilizada no método do gradiente descendente estocástico. Neste caso, a função de erro chama-se erro quadrático médio e é definido pela seguinte equação:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{8.6}$$

onde y_i é a saída da rede neuronal e $\hat{y}_i$ é a saída pretendida do dado i.

A primeira linha do código na Figura 8.16 utiliza a função do *Keras* para treinar a rede durante 100 iterações. Em outras palavras, esta função encontra os pesos que minimizam a função de erro. O restante do código calcula o erro quadrático médio para o conjunto de treino e o conjunto de teste. Para os

Funções de erro

```
# treinar o modelo
historia = model.fit(treino_X, treino_y, \
                     validation_data=(teste_X, teste_y), \
                     epochs=100, verbose=0)

# avaliar o model
treino_mse = model.evaluate(treino_X, treino_y, verbose=0)
teste_mse = model.evaluate(teste_X, teste_y, verbose=0)
print('Treino: %.3f, Teste: %.3f' % (treino_mse, teste_mse))

# criar grafico do erro
pyplot.title('Erro / Erro quadratico medio')
pyplot.plot(historia.history['loss'], label='treino')
pyplot.plot(historia.history['val_loss'], label='teste')
pyplot.legend()
pyplot.show()
```

Figura 8.16: O código em Python para treinar e avaliar o Perceptrão multicamadas

dados apresentados na Figura 8.13, o erro para o conjunto de treino é 0,072 e o erro para o conjunto de teste é 0,071. Além disso, as últimas linhas criam o gráfico da Figura 8.17 com o cálculo do erro para cada iteração durante a fase de treino da rede. Note que a rede consegue diminuir o erro com o conjunto de treino e teste conforme o número de iterações aumenta até convergir para um valor próximo de 0,07.

O erro quadrático médio costuma ser a primeira opção quando se escolhe uma função de erro para problemas de regressão. No entanto, esta função de erro tem a propriedade de penalizar a rede neuronal quando os erros entre $(y_i - \hat{y_i})$ são grandes por causa do quadrado na Equação 8.6. Uma alternativa para estes casos é a função de erro chamada erro logarítmico quadrático médio com a seguinte definição:

$$MSLE = \frac{1}{n}\sum_{i=1}^{n}(log(y_i) - log(\hat{y_i}))^2 \tag{8.7}$$

onde y_i é a saída da rede neuronal e $\hat{y_i}$ é a saída pretendida do dado i.

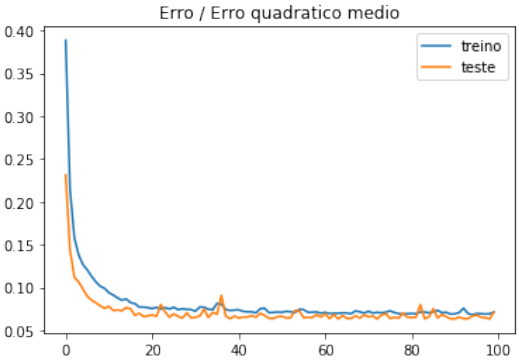


Figura 8.17: Erro para cada iteração de treino da rede neuronal utilizando o erro quadrático médio

Note que $(log(y_i) - log(\hat{y}_i))$ é igual à $log\frac{y_i}{\hat{y}_i}$ e este resultado pode ser interpretado como um medida do rácio entre a saída da rede e a saída pretendida.

A única alteração ao código Python que se precisa fazer é na definição do modelo, tal como se pode ver no código da Figura 8.18. Em particular, a última linha deve ser alterada para que a definição da função de erro seja o erro logarítmico quadrático médio (i.e., *loss='mean_squared_logarithmic_error'*).

Funções de erro

```
# definir modelo do Perceptrao multicamadas
model = Sequential()
model.add(Dense(5, input_dim=1, activation='relu', \
                kernel_initializer='he_uniform'))
model.add(Dense(1, activation='linear'))
opt = SGD(lr=0.01, momentum=0.9)
model.compile(loss='mean_squared_logarithmic_error', \
              optimizer=opt, metrics=['mse'])
```

Figura 8.18: O código em Python para criar o modelo do Perceptrão multicamadas utilizando o erro logarítmico quadrático médio

A Figura 8.19 apresenta o cálculo do erro para cada iteração durante a fase de treino da rede. O resultado final dos erros é ligeiramente inferior quando comparado com a Figura 8.17. Isto pode sugerir que o erro logarítmico quadrático médio é mais adequado para os dados da Figura 8.13.

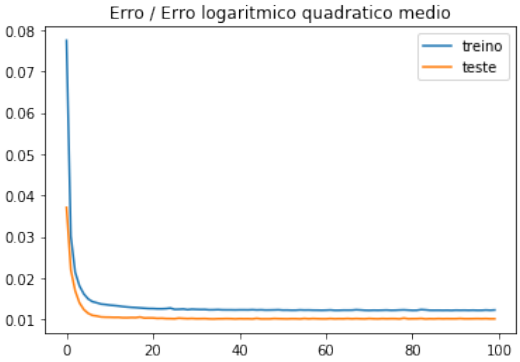


Figura 8.19: Erro para cada iteração de treino da rede neuronal utilizando o erro logarítmico quadrático médio

Outra função de erro que se pode utilizar é o erro médio absoluto com a seguinte definição:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{8.8}$$

onde y_i é a saída da rede neuronal e $\hat{y}_i$ é a saída pretendida do dado i.

Tal como no caso do erro logarítmico quadrático médio, a única alteração que se precisa fazer ao código Python é na última linha da definição do modelo. Note no código da Figura 8.20 que a última linha agora tem a definição da função de erro igual à *loss='mean_absolute_error'*.

Na Figura 8.21, pode-se ver os erros de treino e teste para cada iteração da fase de treino da rede. Este gráfico dos erros é superior quando se compara com a Figura 8.17 e a Figura 8.19. Consequentemente, isto pode sugerir que o erro quadrático médio ou o erro logarítmico quadrático médio são mais adequados para os dados da Figura 8.13.

8.3.2 Classificação

Nesta seção, apresenta-se as funções de erro utilizadas para problemas de classificação binária. Como existem apenas duas opções de saída na classificação binária, a rede neuronal deve ter uma saída que indica uma das classes ou

Funções de erro

```
# definir modelo do Perceptrao multicamadas
model = Sequential()
model.add(Dense(5, input_dim=1, activation='relu', \
                kernel_initializer='he_uniform'))
model.add(Dense(1, activation='linear'))
opt = SGD(lr=0.01, momentum=0.9)
model.compile(loss='mean_absolute_error', \
              optimizer=opt, metrics=['mse'])
```

Figura 8.20: O código em Python para criar o modelo do Perceptrão multicamadas utilizando o erro médio absoluto

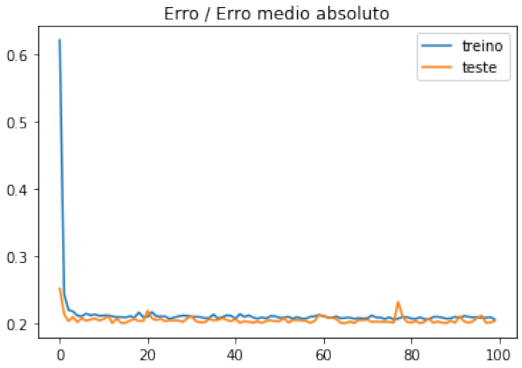


Figura 8.21: Erro para cada iteração de treino da rede neuronal utilizando o erro médio absoluto

rótulos. Muitas vezes a primeira classe é associada ao valor 0 e a segunda classe ao valor 1.

Tal como na seção de funções de erro para problemas de regressão, utiliza-se um conjunto de dados que são gerados com a biblioteca *scikit-learn* (ver código para importar bibliotecas na Figura 8.22). O código da Figura 8.23 gera os dados para um problema de classificação binária e a Figura 8.24 mostra um gráfico com estes dados.

Note que os dados da Figura 8.24 possuem um vetor de entrada (i.e., variável X no código da Figura 8.23) com duas dimensões e uma saída com o valor do rótulo (i.e., variável y no código da Figura 8.23). Na Figura 8.24,

Funções de erro

```
# importar bibliotecas para gerar dados para classificacao
# e gerar graficos
from sklearn.datasets import make_circles
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import SGD
from numpy import where
from matplotlib import pyplot
```

Figura 8.22: O código em Python para importar as bibliotecas

Funções de erro

```
# gerar dados para problema de classificacao
X, y = make_circles(n_samples=1000,noise=0.1,random_state=1)

# gerar grafico dos dados
rotulo_0_ix = where(y == 0)
pyplot.scatter(X[rotulo_0_ix, 0],X[rotulo_0_ix, 1],label=0)
rotulo_1_ix = where(y == 1)
pyplot.scatter(X[rotulo_1_ix, 0],X[rotulo_1_ix, 1],label=1)
pyplot.title('Pontos para classificacao')
pyplot.legend()
pyplot.show()
```

Figura 8.23: O código em Python para gerar dados para um problema de classificação

os círculos representam os pontos com rótulo 0 e as cruzes representam os pontos com rótulo 1.

Como o vetor de entrada possui coordenadas com valores aproximadamente entre -1 e 1, não há necessidade de padronizar os dados. O código da Figura 8.25 separa os dados em dois conjuntos, um de treino e outro de teste. Os primeiros 800 pontos são utilizados para treinar o Perceptrão multicamadas e os restantes 200 pontos para testar a rede neuronal.

O código da Figura 8.26 define um Perceptrão multicamadas com 3 camadas, nomeadamente uma camada de entrada com duas unidades, uma camada

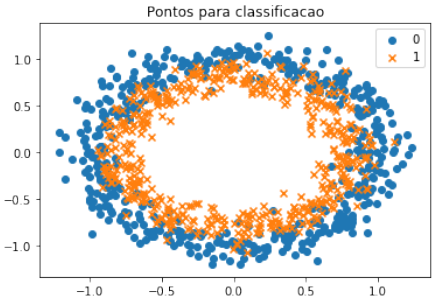


Figura 8.24: Gráfico com os pontos para um problema de classificação binária

Funções de erro

```
# dividir dados para treino e teste
n_dados_treino = 800
treino_X,teste_X = X[:n_dados_treino,:], X[n_dados_treino:,:]
treino_y,teste_y = y[:n_dados_treino], y[n_dados_treino:]
```

Figura 8.25: O código em Python que separa os dados em conjunto de treino e teste

escondida com 20 unidades que utilizam uma função de ativação *ReLU*, e uma camada de saída com uma unidade que tem uma função de ativação logística.

Funções de erro

```
# definir modelo do Perceptrao multicamadas
model = Sequential()
model.add(Dense(20, input_dim=2, activation='relu',\
                kernel_initializer='he_uniform'))
model.add(Dense(1, activation='sigmoid'))
opt = SGD(lr=0.01, momentum=0.9)
model.compile(loss='binary_crossentropy', optimizer=opt, \
              metrics=['accuracy'])
```

Figura 8.26: O código em Python para definir o modelo do Perceptrão multicamadas

Além disso, o código da Figura 8.26 define a função de erro na última

linha. Em particular, a função de erro utilizada por esta rede neuronal é a entropia cruzada. Esta função de erro costuma ser a primeira opção quando se tem um problema de classificação binária. A entropia cruzada tem a seguinte definição:

$$CE = -\frac{1}{n}\sum_{i=1}^{n} \hat{y}_i \log(y_i) + (1-\hat{y}_i)\log(1-y_i) \tag{8.9}$$

onde y_i é a saída da rede neuronal e $\hat{y}_i$ é a saída pretendida do dado i.

A variável y_i na Equação 8.9 toma valores no intervalo (0,1). Consequentemente, a função logística é um boa opção para utilizar como função de ativação na camada de saída. Além disso, como $\hat{y}_i$ assume os valores 0 ou 1, note que a expressão $\hat{y}_i \log(y_i) + (1-\hat{y}_i)\log(1-y_i)$ aproxima-se de 0 quando y_i e $\hat{y}_i$ são muito próximos.

Funções de erro

```
# treinar o modelo
historia = model.fit(treino_X, treino_y, \
                     validation_data=(teste_X, teste_y), \
                     epochs=200, verbose=0)

# avaliar o model
_,accuracy_treino = model.evaluate(treino_X, treino_y, \
                                   verbose=0)
_,accuracy_teste = model.evaluate(teste_X, teste_y, \
                                  verbose=0)
print('Treino: %.3f', accuracy_treino)
print('Teste: %.3f', accuracy_teste)
```

Figura 8.27: O código em Python para treinar e avaliar o modelo do Perceptrão multicamadas

Tal como a rede para resolver problemas de regressão, o código da Figura 8.27 utiliza a função *model.fit* do *Keras* para treinar a rede durante 200 iterações. Esta função é responsável por encontrar os pesos que minimizam a entropia cruzada. As linhas seguintes do código calculam a percentagem de exemplos corretamente classificados (também conhecida como *accuracy* em inglês) para o conjunto de treino e o conjunto de teste. Para os dados apre-

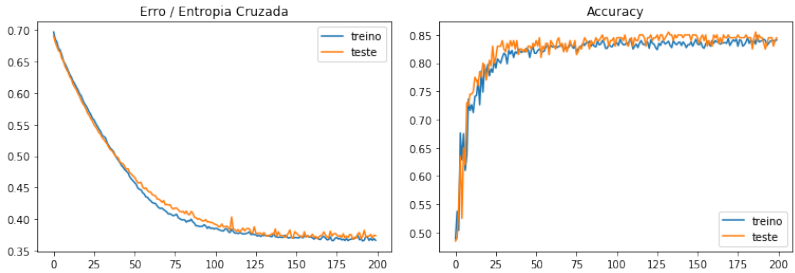
sentados na Figura 8.24, a *accuracy* para o conjunto de treino é 84,1% e a *accuracy* para o conjunto de teste é 84,5%.

Funções de erro

```
# gerar o grafico da funcao de erro durante o treino
pyplot.title('Erro / Entropia Cruzada')
pyplot.plot(historia.history['loss'], label='treino')
pyplot.plot(historia.history['val_loss'], label='teste')
pyplot.legend()
pyplot.show()

# gerar o grafico da accuracy durante o treino
pyplot.title('Accuracy')
pyplot.plot(historia.history['acc'], label='treino')
pyplot.plot(historia.history['val_acc'], label='teste')
pyplot.legend()
pyplot.show()
```

Figura 8.28: O código em Python para gerar os gráficos do erro e *accuracy* do modelo com entropia cruzada



(a) Erro **(b)** *Accuracy*

Figura 8.29: Gráficos do erro e *accuracy* da rede com a entropia cruzada

O código da Figura 8.28 cria os gráficos da Figura 8.29 com o cálculo do erro e da *accuracy* para cada iteração durante a fase de treino da rede. Ao observar a Figura 8.29a, pode-se notar que a rede consegue diminuir o erro em cada iteração, tanto para o conjunto de treino quanto para o conjunto de teste, até convergir para um valor próximo de 0,37. Na Figura 8.29b, a

accuracy aumenta em cada iteração até convergir para um valor próximo de 84%.

A função *Hinge* é uma outra opção para função de erro para problemas de classificação binária. Esta função é muito utilizada em um classificador chamado de *Support Vector Machine* ou SVM. Em particular, a função de erro *Hinge* tem a seguinte definição:

$$H = \sum_{i=1}^{n} max(0, 1 - \hat{y}_i y_i) \tag{8.10}$$

onde y_i é a saída da rede neuronal e $\hat{y}_i$ é a saída pretendida do dado i.

Para a função de erro *Hinge*, a saída pretendida da rede precisa ser 1 ou -1. Consequentemente, a função de ativação tangente hiperbólica é provavelmente a opção mais adequada para se utilizar na camada de saída com esta função de erro. Com uma saída pretendida igual a 1 ou -1, note que a Equação 8.10 é igual a 0 quando y_i e $\hat{y}_i$ são ambos iguais.

Funções de erro

```
# alterar y de {0,1} para {-1,1}
y[where(y == 0)] = -1

# dividir dados para treino e teste
n_dados_treino = 800
treino_X,teste_X = X[:n_dados_treino,:], X[n_dados_treino:,:]
treino_y,teste_y = y[:n_dados_treino], y[n_dados_treino:]
```

Figura 8.30: O código em Python para alterar os rótulos dos dados

Para testar a função de erro *Hinge* no Perceptrão multicamadas, é preciso alterar os rótulos dos dados criados com a função *make_circles* no código da Figura 8.23. O código da Figura 8.30 altera os exemplos com rótulos iguais à 0 para que eles tenham rótulos iguais à -1. Depois, divide-se os dados em conjunto de treino e teste.

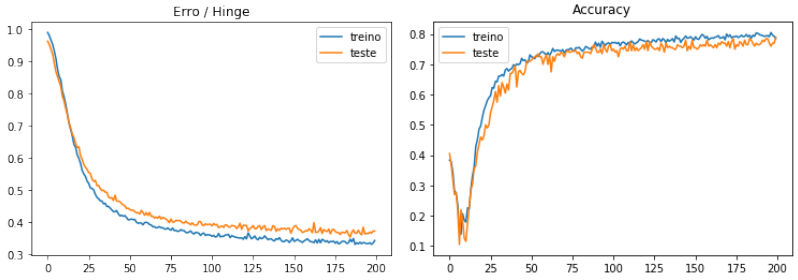
O código da Figura 8.31 é praticamente igual ao código da Figura 8.26 em termos da definição do Perceptrão multicamadas. As únicas diferenças são que o código da Figura 8.31 cria uma rede com um camada de saída que utiliza a função de ativação tangente hiperbólica e a função de erro é a *Hinge*.

Funções de erro

```
# definir modelo do Perceptrao multicamadas
model = Sequential()
model.add(Dense(20, input_dim=2, activation='relu',\
                kernel_initializer='he_uniform'))
model.add(Dense(1, activation='tanh'))
opt = SGD(lr=0.01, momentum=0.9)
model.compile(loss='hinge', optimizer=opt, \
              metrics=['accuracy'])
```

Figura 8.31: O código em Python para definir o modelo do Perceptrão multicamadas com a função de erro *Hinge*

Ao utilizar um código semelhante ao código da Figura 8.28, obtêm-se os gráficos da Figura 8.32 com o cálculo do erro e da *accuracy* para cada iteração durante a fase de treino da rede. A Figura 8.32a mostra que a rede consegue diminuir o erro em cada iteração até convergir para um valor próximo de 0,38 para o conjunto de teste. No entanto, a Figura 8.32b apresenta a *accuracy* da rede em cada iteração até convergir para um valor próximo de 79%, o que é ligeiramente inferior aos resultados da Figura 8.29b. Isto pode sugerir que a função de erro entropia cruzada e a função de ativação logística são mais adequados para este conjunto de dados.



(a) Erro **(b)** *Accuracy*

Figura 8.32: Gráficos do erro e *accuracy* da rede com a função *Hinge*

Para os problemas de classificação binária, ainda há outras opções para serem utilizadas no Perceptrão multicamadas e que estão disponíveis no *Keras*. Por exemplo, a função de erro *Squared Hinge* é uma variante da *Hinge*, onde

a equação é igual à $SH = \sum_{i=1}^{n}(max(0, 1 - \hat{y}_i y_i))^2$. Note que a expressão $max(0, 1 - \hat{y}_i y_i)$ está sendo elevada ao quadrado. Normalmente, o problema de classificação fica mais fácil de resolver numericamente pois esta função de erro tem uma superfície mais regular.

Para problemas de classificação com múltiplos rótulos ou classes, deve-se definir um modelo do Perceptrão multicamadas ligeiramente diferente. Normalmente, a rede tem uma unidade de processamento na camada de saída para cada rótulo. Por exemplo, se os dados para o problema de classificação possuem 3 rótulos então a rede vai ter 3 saídas (ou três unidades na camada de saída). Além disso, para um dado exemplo, as saídas devem representar a previsão das probabilidades para cada rótulo. Desta forma, a função de ativação *Softmax* é a mais utilizada para estes problemas.

Em relação às funções de erro para problemas de classificação com múltiplas classes, há várias opções também. A função mais utilizada é a entropia cruzada categórica e também está disponível no *Keras*.

Capítulo 9

Redes Neuronais e Dados Financeiros

9.1 Introdução

Este capítulo tem a finalidade de mostrar como as redes neuronais podem ser utilizadas para prever características importantes do mercado de ações. Estas previsões são utilizadas muitas vezes por investidores para auxiliar o processo de decisão de compra ou venda de ações.

A primeira seção está focada em apresentar alguns exemplos de trabalhos científicos que utilizaram as redes neuronais para prever características do mercado de ações. Na segunda seção, um exemplo prático é desenvolvido para que o leitor possa ver como é possível construir uma rede neuronal para prever a tendência do preço de uma ação.

9.2 Trabalhos científicos

Há muitos trabalhos científicos que utilizaram dados financeiros para prever propriedades do mercado de ações. Em particular, esta seção tem o objetivo de apresentar alguns artigos que utilizaram redes neuronais para os seguintes problemas do mercado financeiro:

- Previsão do preço de uma ação - Cada ação no mercado tem um preço que varia ao longo do tempo. Este preço sobe ou desce dependendo do número de ações que são compradas e vendidas no mercado. Os autores deste trabalhos utilizam modelos para prever tanto o preço de uma ação no futuro como o movimento da ação (i.e., se a ação sobe, desce ou mantem-se constante no futuro).

- Previsão de índices financeiros - Um índice financeiro (e.g., NASDAQ Composite, Dow Jones Industrial Average e S&P 500) é uma medida de uma parte do mercado de ações. Para calcular um índice, primeiro é feito uma seleção de ações que fazem parte deste índice e depois calcula-se normalmente uma soma ponderada que leva em consideração o preço das ações selecionadas. Consequentemente, um índice pode subir ou descer conforme a mudança dos preços das ações selecionadas. Investidores gostam de utilizar estes índices como uma ferramenta para descrever uma parte do mercado de ações e permitir uma comparação com investimentos específicos. Muitos dos trabalhos científicos nesta área tentam prever o valor do índice no futuro, o movimento do índice (i.e., se o índice vai subir, descer ou manter-se constante no futuro), e a volatilidade do índice no mercado futuro.

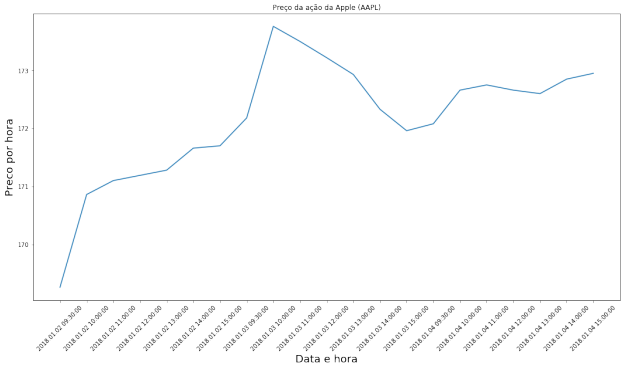


Figura 9.1: Gráfico do preço da ação da Apple (AAPL) por hora

Outro fator importante que pode influenciar o modelo de uma rede neuronal é a frequência dos índices financeiros e preços de ações. Esta frequência pode ser diária, por hora, por minuto, por segundo ou ainda por milissegundo. Por exemplo, a Figura 9.1 mostra os valores do preço da ação da Apple provenientes de dados com uma frequência por hora.

9.2.1 Previsão de preços de ações

A previsão de preço de ações pode ser muito útil para um investidor decidir a sua estratégia de negociação no mercado financeiro. A Figura 9.2 mostra um gráfico com os preços diários da ação da Microsoft de 13-03-1986 à 10-11-2017, apenas para que se possa ver um exemplo de como estes preços variam ao longo do tempo.

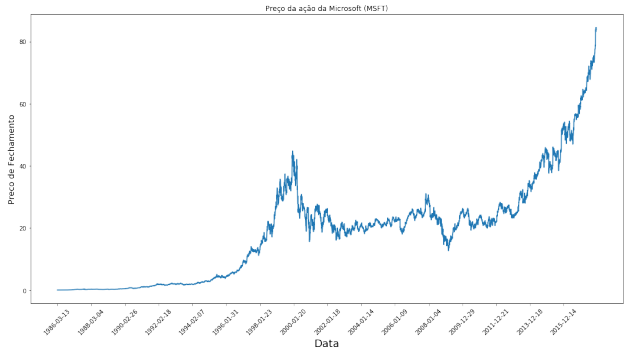


Figura 9.2: Gráfico do preço diário da ação da Microsoft (MSFT) de 13-03-1986 à 10-11-2017

Note na Figura 9.2 que os preços da ação nos primeiros anos parecem variar pouco. No entanto, isto não é verdade como podemos ver na Figura 9.3 onde o gráfico apresenta a variação dos preços de 13-03-1986 à 31-01-1996. Como os preços da ação subiram muito ao longo do tempo, os primeiros valores do preço podem parecer que variaram pouco. Este efeito ocorre principalmente para empresas que são cotadas numa bolsa durante duas ou mais décadas. Consequentemente, os trabalhos científicos precisam levar em consideração esta característica dos preços nos modelos de previsão ou utilizar dados que não tenham uma janela de tempo tão grande.

Há várias trabalhos científicos que tentam prever o preço da ação ou a sua tendência no futuro. Por exemplo, Ticknor (2013) utiliza um Perceptrão multicamadas para prever o movimento dos preços das ações da Microsoft e Goldman Sachs. Esta rede neuronal foi treinada com preços diários entre 4-01-2010 à 31-12-2012. A configuração da rede tem 3 camadas, onde cada camada escondida tem 5 unidades de processamento. Em uma segunda experiência, o autor também utilizou um Perceptrão multicamadas para prever o preço da ação no próximo dia com dados da IBM e Apple entre 10-02-2003 à 21-01-

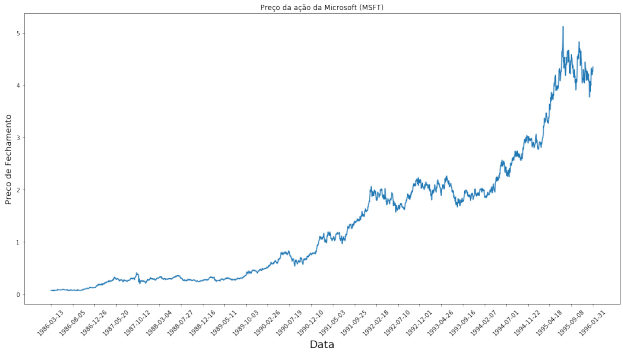


Figura 9.3: Gráfico do preço diário da ação da Microsoft (MSFT) de 13-03-1986 à 31-01-1996

2005. A configuração da rede neste caso utiliza 20 unidades em cada camada escondida. Os resultados mostram que estas redes neuronais são eficientes para prever o movimento e preço de uma ação no futuro.

O trabalho de Arévalo *et al.* (2016) utiliza um Perceptrão multicamadas para prever o preço por minuto das ações da Apple para ser utilizado em estratégias de negociação de alta frequência. A janela de tempo dos dados é de 2-9-2008 à 7-11-2008. Como os autores testaram vários conjuntos de entradas, a configuração desta rede possui 6 camadas, onde cada camada escondida tem um número de unidades que varia com o número de entradas. Os resultados mostram que esta rede consegue obter uma *accuracy* entre 63% e 66% para a direção do movimento da ação.

9.2.2 Previsão de índices de mercado

Para exemplificar a variação de um índice financeiro, a Figura 9.4 apresenta um gráfico com os valores diários de fecho do índice financeiro S&P 500. Este índice americano foi criado em 1950 e seleciona as 500 maiores empresas cotadas no mercado financeiro do Estados Unidos da América. Atualmente é considerado um dos índices financeiros mais representativos do mercado americano.

Nos próximos parágrafos desta seção, apresenta-se alguns trabalhos científicos para prever índices financeiros. Por exemplo, Moghaddam *et al.* (2016) utiliza um Perceptrão multicamadas para prever o índice financeiro da

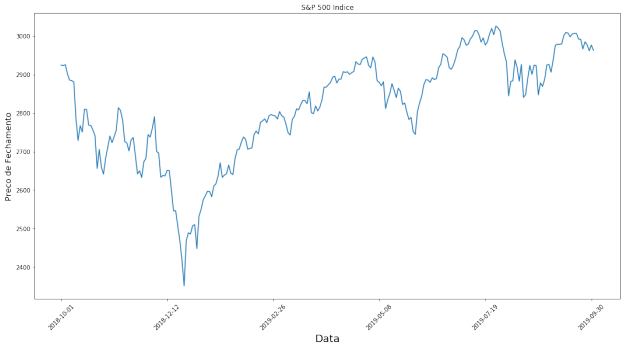


Figura 9.4: Gráfico do índice do S&P 500

NASDAQ. As entradas desta rede são os últimos n valores do índice financeiro e o dia da semana. Os autores testaram várias configurações da rede; em outras palavras, testaram configurações de redes com diferentes números de camadas escondidas e diferentes números de unidades de processamento nestas camadas. Alem disso, a precisão das previsões destas várias configurações foram analisadas para um $n = 4$ e um $n = 9$. No entanto, os autores concluem que não há grande diferença nos resultados para valores diferentes de n.

O trabalho de Qiu e Song (2016) utiliza um Perceptrão multicamadas para prever a direção diária do movimento de um índice financeiro do mercado de ações Japonês (i.e., Nikkei 225). Os autores testaram dois conjuntos de entradas com diferentes indicadores do mercado. Além disso, a rede neuronal tem uma camada escondida com 10 unidades de processamento. Os resultados mostram que o segundo conjunto de entradas atinge uma *accuracy* de 81,27% para prever a direção do movimento do Nikkei 225 no próximo dia.

Por último, Hamid e Iqbal (2004) utilizam um Perceptrão multicamadas para prever a volatilidade dos preços futuros do índice financeiro S&P 500. Esta rede neuronal tem uma camada de entrada com 13 unidades, duas camadas escondidas com 26 unidades por camada e uma camada de saída com uma unidade. Os resultados da previsão da volatilidade são comparados com uma volatilidade implícita que é extraída do modelo Barone-Adesi e Whaley (1987) para opções americanas. Os resultados mostram que esta rede neuronal pode ser utilizada com uma boa ferramenta para prever a volatilidade dos preços futuros.

9.3 Exemplo prático

Esta seção tem o objetivo de apresentar um exemplo prático em Python de como utilizar um Perceptrão multicamadas para prever a direção do movimento de um preço de uma ação. Os dados utilizados são os preços diários da ação da Apple entre 07-09-1984 e 07-07-2004. A Figura 9.5 apresenta um gráfico com os preços da ação da Apple para a janela de tempo escolhida.

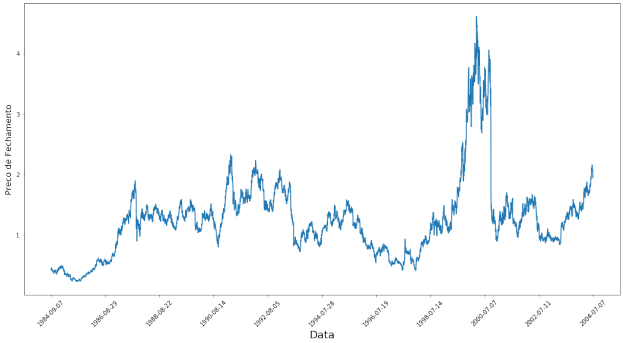


Figura 9.5: Gráfico do preço diário da ação da Apple (AAPL) de 07-09-1984 à 07-07-2004

O código da Figura 9.6 é utilizado para importar as seguintes bibliotecas: (i) o *Keras*[1], para que se possa implementar o Perceptrão multicamadas neste exemplo; (ii) o *Pandas*[2], que é utilizado para importar dados diretamente para uma estrutura de dados e possui ferramentas para fazer análise destes dados; (iii) o *scikit-learn*[3], que possui funções para tratar os dados; (iv) o *Matplotlib*[4], que é utilizado para criar os gráficos; (v) o *TA-Lib*[5], que possui várias funções para obter indicadores financeiros.

No código da Figura 9.7, a primeira linha utiliza a biblioteca *Pandas* para ler os dados de um ficheiro e criar uma estrutura de dados. Este ficheiro foi obtido de um repositório chamado *Kaggle*[6] e possui os preços diários da ação da Apple. Na linha seguinte, imprime-se as primeiras 5 linhas dos dados importados e a Tabela 9.1 mostra este resultado. Note que o ficheiro possui

[1] https://keras.io/
[2] https://pandas.pydata.org
[3] https://scikit-learn.org/
[4] https://matplotlib.org
[5] http://ta-lib.org
[6] https://www.kaggle.com

Perceptrão multicamadas com dados financeiros

```
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import SGD
import pandas as pd
import numpy as np
import talib
import os
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
```

Figura 9.6: O código em Python para importar as bibliotecas

4 informações importantes para cada data: (i) o preço da ação na abertura do mercado (i.e., coluna *Open*), (ii) o preço máximo que a ação atingiu neste dia (i.e., coluna *High*), (iii) o preço mínimo que a ação atingiu neste dia (i.e., coluna *Low*), e (iv) (i) o preço da ação no momento que o mercado fecha (i.e., coluna *Close*).

Tabela 9.1: As primeiras 5 linhas do preços diários da ação da Apple no ficheiro

	Date	**Open**	**High**	**Low**	**Close**
0	1984-09-07	0.42388	0.42902	0.41874	0.42388
1	1984-09-10	0.42388	0.42516	0.41366	0.42134
2	1984-09-11	0.42516	0.43668	0.42516	0.42902
3	1984-09-12	0.42902	0.43157	0.41618	0.41618
4	1984-09-13	0.43927	0.44052	0.43927	0.43927

Na penúltima linha do código 9.7, seleciona-se as primeiras 5000 linhas, que correspondem aos preços entre 07-09-1984 e 07-07-2004. Além disso, na última linha do código elimina-se qualquer linha da estrutura de dados que não possua algum preço.

O código da Figura 9.8 adiciona colunas na estrutura de dados que serão utilizadas posteriormente como entradas para o Perceptrão multicamadas. Em particular, as entradas para esta rede neuronal são as seguintes:

- *Max-Min*: a diferença entre o preço máximo e mínimo de cada data;

Perceptrão multicamadas com dados financeiros

```
# ler dados de um ficheiro com os precos
# da acao da Apple (AAPL)
dados = pd.read_csv( \
              os.path.join('Data/Stocks','aapl.us.txt'), \
              delimiter=',',\
              usecols=['Date','Open','High','Low','Close'])

# imprime as primeiras cinco linhas
print(dados.head(5))

# seleciona apenas uma parte do conjunto original
dados =  dados.loc[:5000]

#elimina linhas que nao possuem algum valor
dados = dados.dropna()
```

Figura 9.7: O código em Python para ler dados de um ficheiro com o *Pandas*

Perceptrão multicamadas com dados financeiros

```
# Cria as entradas para a rede neuronal
dados['Max-Min'] = dados['High'] - dados['Low']
dados['F-A'] = dados['Close'] - dados['Open']
dados['Media Movel 3 dias'] = dados['Close'].shift(1).\
                              rolling(window = 3).mean()
dados['Media Movel 10 dias'] = dados['Close'].shift(1).\
                               rolling(window = 10).mean()
dados['Media Movel 30 dias'] = dados['Close'].shift(1).\
                               rolling(window = 30).mean()
dados['Desvio Padrao']= dados['Close'].rolling(5).std()
dados['RSI'] = talib.RSI(dados['Close'].values, \
                         timeperiod = 9)
dados['Williams %R'] = talib.WILLR(dados['High'].values, \
            dados['Low'].values, dados['Close'].values, 7)
```

Figura 9.8: O código em Python para criar as entradas para a rede neuronal

- *F-A*: a diferença entre o preço de fecho e abertura de cada data;

- *Media Movel 3 dias*: a média dos últimos 3 preços de fecho em relação à data de cada linha;
- *Media Movel 10 dias*: a média dos últimos 10 preços de fecho em relação à data de cada linha;
- *Media Movel 30 dias*: a média dos últimos 30 preços de fecho em relação à data de cada linha;
- *Desvio Padrao*: o desvio padrão dos últimos 5 preços de fecho em relação à data de cada linha;
- *RSI*: um indicador financeiro chamado *Relative Strength Index*[7] que calcula a rácio entre o movimento para cima do preço e o valor absoluto do movimento do preço;
- *Williams %R*: um indicador financeiro chamado *Williams Percent Range*[8] que compara o preço de fecho com os preços máximos e mínimos dos últimos n dias. Como este indicador é possível saber se o preço de uma ação está no estado *overbought* (i.e., o preço está próximo dos máximos) ou *oversold* (i.e., preço está próximo dos mínimos);

O código da Figura 9.9 cria mais uma coluna na estrutura de dados com a saída pretendida da rede neuronal. Esta saída pretendida tem apenas dois valores. O valor da saída pretendida é '1' quando o preço de fecho do dia d é maior do que o preço de fecho do dia $d-n$ (i.e., o movimento do preço subiu), onde n o número de dias antes da data d. Em todos os outros casos, valor é '0' e isso representa um movimento constante ou para baixo do preço. Desta forma, esta rede vai resolver um problema de classificação binária. No caso particular do código da Figura 9.9, compara-se o preço de fecho de cada linha com o preço de fecho do dia anterior (i.e., o $n = 1$) para se determinar se o preço subiu.

A Figura 9.10 apresenta alguns exemplos de linhas na estrutura de dados depois de adicionar as colunas com as entradas e a saída pretendida da rede neuronal.

Nas duas primeiras linhas do código da Figura 9.11, cria-se duas estruturas de dados a partir da estrutura de dados original da Figura 9.10. A estrutura de

[7] A fórmula para calcular o RSI está disponível em https://www.fmlabs.com/reference/

[8] A fórmula para calcular o Williams %R também está disponível em https://www.fmlabs.com/reference/

Perceptrão multicamadas com dados financeiros

```
# cria a saida pretendida da rede neuronal
# com o nome Preco_Subiu
day_diff = 1
preco_subiu = np.where( \
                dados['Close'].shift(-1*day_diff).values - \
                dados['Close'].values > 0, 1, 0)
a = np.empty((day_diff,))
a[:] = np.nan
preco_subiu = np.append(a,preco_subiu)
for i in range(day_diff):
    preco_subiu = np.delete(preco_subiu,-1,axis = 0)
dados['Preco_Subiu'] = preco_subiu

#elimina linhas que nao possuem algum valor
dados = dados.dropna()
```

Figura 9.9: O código em Python para criar a saída pretendida da rede neuronal

dados X possui as entradas da rede, ou seja as colunas *Max-Min*, *F-A*, *Media Movel 3 dias*, *Media Movel 10 dias*, *Media Movel 30 dias*, *Desvio Padrao*, *RSI* e *Williams %R*. Já a estrutura de dados y tem a última coluna da Figura 9.10, ou seja a coluna *Preco_Subiu*. Após a criação destas estruturas de dados, os dados são divididos em conjuntos de treino e teste. Por último, os dados com as entradas da rede são padronizados para facilitar o processo de aprendizagem do Perceptrão multicamadas.

As primeiras linhas do código da Figura 9.12 utilizam a biblioteca *Keras* para definir o modelo do Perceptrão multicamadas com a seguinte especificação:

- uma camada de entrada com 8 unidades de processamento;
- duas camadas escondidas com 128 unidades de processamento que utilizam a função de ativação $ReLU$;
- uma camada de saída com 1 unidade de processamento que utiliza a função de ativação logística.

Na última linha do código da Figura 9.12, a rede neuronal é treinada

	Date	Open	High	Low	Close	Max-Min	F-A	Media Movel 3 dias	Media Movel 10 dias	Media Movel 30 dias	Desvio Padrao	RSI	Williams %R	Preco_Subiu
4991	2004-06-23	2.1130	2.1657	2.1066	2.1576	0.0591	0.0446	2.096367	2.02270	1.855473	0.031609	78.742146	-3.551074	1.0
4992	2004-06-24	2.1564	2.1576	2.1117	2.1247	0.0459	-0.0317	2.113367	2.04408	1.871297	0.031824	70.781305	-19.505233	0.0
4993	2004-06-25	2.1130	2.1576	2.1130	2.1576	0.0446	0.0446	2.131767	2.06319	1.884187	0.036567	73.765190	-7.431193	1.0
4994	2004-06-28	2.1708	2.1886	2.0630	2.0810	0.1256	-0.0898	2.146633	2.08213	1.897843	0.032364	58.197871	-81.576952	0.0
4995	2004-06-29	2.0555	2.1130	2.0119	2.0810	0.1011	0.0255	2.121100	2.09737	1.909153	0.038376	58.197871	-60.894171	0.0
4996	2004-06-30	2.0810	2.1104	2.0425	2.0836	0.0679	0.0026	2.106533	2.10890	1.920760	0.034508	58.573441	-59.422750	1.0
4997	2004-07-01	2.0542	2.0798	2.0425	2.0682	0.0373	0.0140	2.081867	2.10762	1.933353	0.035904	55.264863	-68.138087	0.0
4998	2004-07-02	1.9492	1.9965	1.9044	1.9901	0.0921	0.0409	2.077600	2.10428	1.944533	0.039966	41.795360	-69.845179	0.0
4999	2004-07-06	1.9914	2.0119	1.9722	1.9810	0.0397	-0.0104	2.047300	2.09263	1.954393	0.050856	40.501415	-73.047150	0.0
5000	2004-07-07	1.9722	2.0079	1.9286	1.9465	0.0793	-0.0257	2.013100	2.08378	1.963393	0.059157	35.777254	-85.186488	0.0

Figura 9.10: Exemplos de linhas na estrutura de dados que contem as entradas e saída da rede neuronal

Perceptrão multicamadas com dados financeiros

```
# Cria-se duas estruturas de dados para armazenar
# as entradas e saidas da rede neuronal
X = dados.iloc[:, 5:-1]
y = dados.iloc[:, -1]

# divide os dados em conjunto de treino e teste
divisao_dados = int(len(dados)*0.8)
X_treino, X_teste = X[:divisao_dados], X[divisao_dados:]
y_treino, y_teste = y[:divisao_dados], y[divisao_dados:]

# Padronizar dados para a rede neuronal
sc = StandardScaler()
X_treino = sc.fit_transform(X_treino)
X_teste = sc.transform(X_teste)
```

Figura 9.11: O código em Python para criar estruturas de dados com os conjuntos de treino e teste

durante 200 iterações. O código da Figura 9.13 avalia a *accuracy* desta rede neuronal com o conjunto de treino e teste. Para o conjunto de teste, obteve-se uma *accuracy* de classificação de 84,7%.

Por último, este mesmo código foi testado para alguns valores de n (ou

Perceptrão multicamadas com dados financeiros

```
# define modelo do Perceptrao multicamadas
modelo = Sequential()
modelo.add(Dense(128, input_dim = X.shape[1], \
    activation = 'relu', kernel_initializer = 'he_uniform'))
modelo.add(Dense(128, activation = 'relu', \
    kernel_initializer = 'he_uniform'))
modelo.add(Dense(1, activation = 'sigmoid', \
    kernel_initializer = 'he_uniform'))
opt = SGD(lr=0.01, momentum=0.9)
modelo.compile(loss='binary_crossentropy', optimizer=opt, \
               metrics = ['accuracy'])

# treinar o modelo
historia = modelo.fit(X_treino, y_treino, \
                      validation_data=(X_teste, y_teste),\
                      batch_size = 10, epochs=200)
```

Figura 9.12: O código em Python para criar o modelo do Perceptrão multicamadas e treinar a rede neuronal

Perceptrão multicamadas com dados financeiros

```
# avaliar o modelo
_, accuracy_treino = modelo.evaluate(X_treino, y_treino, \
                                     verbose=0)
_, accuracy_teste = modelo.evaluate(X_teste, y_teste, \
                                    verbose=0)
print('Teste: %.3f', accuracy_treino)
print('Teste: %.3f', accuracy_teste)
```

Figura 9.13: O código em Python para avaliar o Perceptrão multicamadas

Tabela 9.2: A *accuracy* para o problema de identificar o movimento da ação.

n = 1	**n = 3**	**n = 7**	**n = 30**
84,7%	82,2%	88,2%	81,4%

day_diff no código da Figura 9.9), onde n é o número de dias antes da data d na estrutura de dados. Lembre-se que d e n são utilizados para calcular a saída pretendida; ou seja, a saída pretendida é igual à '1' quando o preço de fecho do dia d é maior do que o preço de fecho do dia $d - n$. Os resultados da Tabela 9.2 mostram que os valores da *accuracy* são todos acima de 80%, sendo que um $n = 7$ obteve a maior *accuracy* de 88,2%.

Referências Bibliográficas

Andersen, T., Bollerslev, T. e Hadi, A. (2014). ARCH and GARCH Models. In *Wiley StatsRef: Statistics Reference Online*. American Cancer Society.

Arévalo, A., Niño, J., Hernández, G. e Sandoval, J. (2016). High-frequency trading strategy based on deep neural networks. In D.-S. Huang, K. Han, e A. Hussain (Eds.), *Intelligent Computing Methodologies*, Cham, pp. 424–436. Springer International Publishing.

Barone-Adesi, G. e Whaley, R. E. (1987). Efficient analytic approximation of american option values. *The Journal of Finance* **42**(2), 301–320.

Björk, T. (2009). *Arbitrage Theory in Continuous Time*. Oxford University Press.

Black, F. e Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of political economy* **81**(3), 637–654.

Bogachev, V. I. (2007). *Measure Theory*, Volume 1. Springer Science & Business Media.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics* **31**(3), 307 – 327.

Bollerslev, T. (1987). A conditionally heteroskedastic time series model for speculative prices and rates of return. *The Review of Economics and Statistics* **69**(3), 542–547.

Bougerol, P. e Picard, N. (1992). Stationarity of garch processes and of some nonnegative time series. *Journal of Econometrics* **52**(1), 115 – 127.

Braumann, C. A. (2005). *Introdução às Equações Diferenciais Estocásticas e Aplicações*. Edições SPE, Lisboa.

Campa, J. M., Chang, P. K. e Reider, R. L. (1997, September). Implied exchange rate distributions: Evidence from OTC Option Markets. Working Paper 6179, National Bureau of Economic Research.

Cireşan, D., Meier, U., Masci, J. e Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural Networks* **32**, 333 – 338. Selected Papers from IJCNN 2011.

Dowd, K. (2003). *An Introduction to Market Risk Measurement*. John Wiley & Sons.

Embrechts, P. e Wang, R. (2015). Seven proofs for the subadditivity of expected shortfall. *Dependence Modeling* **3**(1). De Gruyter Open, obtido em 13 Jan. 2019 de doi:10.1515/demo-2015-0009.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica* **50**(4), 987–1007.

Fan, J. e Yao, Q. (2003). *Nonlinear Time Series: Nonparametric and Parametric Methods*. Springer Series in Statistics. New York: Springer-Verlag.

Garman, M. B. e Klass, M. J. (1980). On the estimation of security price volatilities from historical data. *The Journal of Business* **53**(1), 67–78.

Ghalanos, A. (2019). *rugarch: Univariate GARCH models.* R Foundation for Statistical Computing. R package version 1.4-1.

Gouriéroux, C. (1997). *ARCH Models and Financial Applications*. Springer Series in Statistics. New York: Springer.

Hamid, S. A. e Iqbal, Z. (2004). Using neural networks for forecasting volatility of S&P 500 index futures prices. *Journal of Business Research* **57**(10), 1116 – 1125. Selected Papers from the third Retail Seminar of the SMA.

Hebb, D. O. (1949, June). *The organization of behavior: A neuropsychological theory*. New York: Wiley.

Hull, J. e White, A. (1990). Pricing interest-rate-derivative securities. *The review of financial studies* **3**(4), 573–592.

Hull, J. C. (2003). *Options Futures and Other Derivatives*. Pearson Education India.

Jesen, M. C. (1978). Some anomalous evidence regarding market efficiency. *Journal of Financial Economics* **6**(2/3), 95–101.

Kristiansen, T. (2004). Pricing of contracts for difference in the nordic market. *Energy Policy* **32**(9), 1075–1085.

Kwok, Y.-K. (2008). *Mathematical Models of Financial Derivatives*. Springer.

LeCun, Y., Bengio, Y. e Hinton, G. (2015, May). Deep learning. *Nature* **521**, 436–444.

Malkiel, B. (1992). Efficient market hypothesis. In M. M. P. Newman e J. (Eds.), *New Palgrave Dictionary of Money and Finance*, Macmillan, London.

Mamon, R. S. (2004). Three ways to solve for bond prices in the vasicek model. *Advances in Decision Sciences* **8**(1), 1–14.

McDonald, R. (2019). *derivmkts: Functions and R Code to Accompany Derivatives Markets*. R Foundation for Statistical Computing. R package version 0.2.4.

Merton, R. C. (1973). Theory of rational option pricing. *The Bell Journal of Economics and Management Science* **4**(1), 141–183.

Mikosch, T. (1998). *Elementary Stochastic Calculus with Finance in View*, Volume 6 of *Advanced series on statistical science and applied probability*. World Scientific.

Mitchell, T. M. (1997). *Machine Learning* (1 ed.). New York, NY, USA: McGraw-Hill, Inc.

Moghaddam, A. H., Moghaddam, M. H. e Esfandyari, M. (2016). Stock market index prediction using artificial neural network. *Journal of Economics, Finance and Administrative Science* **21**(41), 89–93.

Overhaus, M., Ferraris, A., Knudsen, T., Mao, F., Nguyen-Ngoc, L. e Schindlmayr, G. (2011). *Equity Derivatives: Theory and Applications*, Volume 106. John Wiley & Sons.

Palm, F. (1996). 7 GARCH models of volatility. In *Statistical Methods in Finance*, Volume 14 of *Handbook of Statistics*, pp. 209 – 240. Elsevier.

Parkinson, M. (1980). The extreme value method for estimating the variance of the rate of return. *The Journal of Business* **53**(1), 61–65.

Qiu, M. e Song, Y. (2016, May). Predicting the direction of stock market index movement using an optimized artificial neural network model. *PLOS ONE* **11**(5), 1–11.

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

Rao, M. M. (2005). *Conditional Measures and Applications*. Chapman and Hall/CRC.

Reinhart, A. (2019). *pdfetch: Fetch Economic and Financial Time Series Data from Public Sources*. R Foundation for Statistical Computing. R package version 0.2.4.

Rigby, R. A. e Stasinopoulos, D. M. (2005). Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* **54**(3), 507–554.

Rogers, L. C. G. e Satchell, S. E. (1991). Estimating variance from high, low and closing prices. *The Annals of Applied Probability* **1**(4), 504–512.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65–386.

Rumelhart, D. E., Hinton, G. E. e Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature* **323**(6088), 533–536.

Ruppert, D. (2015). *Statistics and Data Analysis for Financial Engineering: with R examples* (3rd ed.). New York: Springer-Verlag.

Ryan, J. A. e Ulrich, J. M. (2019). *quantmod: Quantitative Financial Modelling Framework*. R Foundation for Statistical Computing. R package version 0.4-15.

Shiryaev, A. N. (1999). *Essentials of Stochastic Finance: Facts, Models, Theory*, Volume 3. World scientific.

Shreve, S. E. (2004). *Stochastic Calculus for fFnance II: Continuous-time Models*, Volume 11. Springer Science & Business Media.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap,

T., Leach, M., Kavukcuoglu, K., Graepel, T. e Hassabis, D. (2016, January). Mastering the game of go with deep neural networks and tree search. *Nature* **529**, 484–489.

Stefanini, F. (2010). *Investment Strategies of Hedge Funds*, Volume 577. John Wiley & Sons.

Straumann, D. (2005). *Estimation in Conditionally Heteroscedastic Time Series Models*, Volume 181 of *Lecture Notes in Statistics*. Berlin: Springer-Verlag.

Ticknor, J. L. (2013). A bayesian regularized artificial neural network for stock market forecasting. *Expert Systems with Applications* **40**(14), 5501 – 5506.

Tsay, R. S. (2010). *Analysis of Financial Time Series* (3rd ed.). New York: John Wiley & Sons Inc.

Ulrich, J. (2018). *TTR: Technical Trading Rules*. R Foundation for Statistical Computing. R package version 0.23-4.

Williams, D. (1991). *Probability with Martingales*. Cambridge University Press.

Wilmott, P. (2013). *Paul Wilmott on Quantitative Finance*. John Wiley & Sons.

Yang, D. e Zhang, Q. (2000). Drift-independent volatility estimation based on high, low, open, and close prices. *The Journal of Business* **73**(3), 477–492.