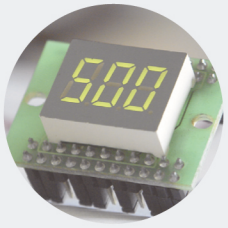
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Inteligência artificial aplicada na previsão de mercados financeiros

LUIS PEDRO PEIXOTO PEREIRA
Novembro de 2015

POLITÉCNICO
DO PORTO

INTELIGÊNCIA ARTIFICIAL APLICADA NA PREVISÃO DE MERCADOS FINANCEIROS

Luís Pedro Peixoto Pereira

Departamento de Engenharia Eletrotécnica

Mestrado em Engenharia Eletrotécnica e de Computadores

Área de Especialização em sistemas e automação

2015

Relatório elaborado para satisfação parcial dos requisitos da Unidade Curricular de Tese/Dissertação do Mestrado em Engenharia Eletrotécnica e de Computadores

Candidato: Luís Pedro Peixoto Pereira, Nº 1080501, 1080501@isep.ipp.pt

Orientação científica: José António tenreiro Machado, jtm@isep.ipp.pt

Co-Orientação científica: Filipe Azevedo, pfa@isep.ipp.pt

Departamento de Engenharia Eletrotécnica

Mestrado em Engenharia Eletrotécnica e de Computadores

Área de Especialização em Telecomunicações

2015

Agradecimentos

Aos engenheiros Filipe Azevedo e José Tenreiro Machado pela orientação, ajuda e disponibilidade que tiveram para comigo durante a realização deste trabalho.

Resumo

Neste documento, são investigados vários métodos usados na inteligência artificial, com o objetivo de obter previsões precisas da evolução dos mercados financeiros. O uso de ferramentas lineares como os modelos AR, MA, ARMA e GARCH têm muitas limitações, pois torna-se muito difícil adaptá-los às não linearidades dos fenómenos que ocorrem nos mercados. Pelas razões anteriormente referidas, os algoritmos como as redes neuronais dinâmicas (TDNN, NARX e ESN), mostram uma maior capacidade de adaptação a estas não linearidades, pois não fazem qualquer pressuposto sobre as distribuições de probabilidade que caracterizam estes mercados. O facto destas redes neuronais serem dinâmicas, faz com que estas exibam um desempenho superior em relação às redes neuronais estáticas, ou outros algoritmos que não possuem qualquer tipo de memória.

Apesar das vantagens reveladas pelas redes neuronais, estas são um sistema do tipo *black box*, o que torna muito difícil extrair informação dos pesos da rede. Isto significa que estes algoritmos devem ser usados com precaução, pois podem tornar-se instáveis.

Palavras-Chave

AR, MA, ARMA, GARCH, TDNN, NARX, ESN, inteligência artificial, redes neuronais, *black box*, pesos.

Abstract

In this document, several methods used in the field of artificial intelligence are investigated, with the objective of obtaining more precise forecasts of the financial markets. The use of linear tools such the models AR, MA, ARMA *and* GARCH *has many limitations, because it becomes very difficult to adapt them to the non linear phenomena that occur in the markets. For the reasons mentioned above, algorithms like dynamic neural networks (*TDNN, NARX *and* ESN*), show a better adaptation to these non linearities, because they don't make any assumptions of the probability distributions of these markets. The fact that these neural networks are dynamic, leads to a superior performance in relation to the static neural nets, or other algorithms that don't possess any type of memory.*

In spite of these advantages, neural networks are a black box type of system, making very difficult for the developer to extract information from the weights of a neural net. This means that these algorithms should be used with caution because they suddenly might become unstable.

Keywords

AR, MA, ARMA, GARCH, TDNN, NARX, ESN, *artificial intelligence, neural networks, black box, weights.*

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

ALM	–	Algoritmo *Levenberg–Marquardt*.
BP	–	*Back Propagation*.
BPTT	–	*Back Propagation Through Time*.
CBOE	–	*Chicago Board Options Exchange*.
CME	–	*Chicago Mercantile Exchange.*
EMA	–	*Expectation Maximization Algorithm*.
ESN	–	*Echo State Network*.
IA		Inteligência artificial.
KNN	–	*K Nearest Neighbours*.
LSTM	–	*Long Short Term Memory*.
MAPE	–	*Mean Absolute Percentage Error*.
MSE	–	*Mean Square Error*.
RND	–	Rede Neuronal Dinâmica.
NARX		*Nonlinear Autoregressive Network with Exogenous Inputs*.
NASDAQ	–	*National Association of Securities Dealers Automated Quotations*.
NYSE	–	*New York Stock Exchange*.

1. INTRODUÇÃO

Ao longo do tempo a tecnologia tem tido um impacto significativo no setor financeiro. De facto desde o uso do telégrafo, em 1889, que aumentou a velocidade com que uma negociação (compra e venda de instrumentos financeiros) era feita, até aos supercomputadores atuais, que realizam milhares de milhões de transações por segundo, verificou-se um progresso assinalável.

No entanto, não foi apenas a capacidade de transacionar instrumentos que evoluiu, os investidores/instituições financeiras também eles foram utilizando a tecnologia a seu favor. Antes da era do computador os investidores tentavam prever o comportamento dos mercados através de análises "manuais", mas após o computador se ter tornado um instrumento acessível outro tipo de técnicas automáticas começaram a ser empregues.

Estas técnicas também conhecidas como *algorithmic trading*, usam ferramentas analíticas para aplicarem um conjunto de regras (algoritmo) que automaticamente executa as negociações. Estes algoritmos, não só conseguem lidar com vastas quantidades de informação em simultâneo, mas também eliminam o fator humano, nomeadamente emoções, sentimentos, superstições que afetam negativamente o processo de avaliação dos investidores.

Apesar destes algoritmos serem muito rápidos e eficazes, têm o problema de seguirem regras muito especificas e que foram previamente programadas. Isto torna o algoritmo

pouco flexível, pois os mercados são muito dinâmicos. Assim, o desempenho destes algoritmos desce consideravelmente, face a situações novas, ao passo que o investidor humano consegue lidar muito melhor com este tipo de situações imprevistas e que colocam novos desafios.

Tendo em conta os problemas atrás referidos as instituições financeiras começaram a focar a sua atenção na área científica que integra a rapidez dos algoritmos computacionais e a capacidade do ser humano interpretar e reagir à dinâmica dos mercados. Esta área é a inteligência artificial (IA).

Muitos dos algoritmos usados pela IA já tinham sido estudados nos anos 60, mas só a partir dos anos 90 com o aumento da potência de processamento dos computadores, é que esta área começou a tornar-se popular e com aplicação mais disseminada.

Atualmente a IA é aplicada em muitas áreas da sociedade nomeadamente para fazer diagnósticos médicos, deteção de fraude, etc.

A investigação realizada neste campo vai desde a implementação de algoritmos simples de regressão ou classificação, até à mímica da forma como o cérebro humano funciona [**1**] [**2**].

1.1. CONTEXTUALIZAÇÃO

Este projeto surgiu do interesse de investigar a área da IA, pois é um campo que se encontra em forte expansão e, por consequência, vai influenciar consideravelmente toda a sociedade, nomeadamente o mercado de trabalho com particular incidência no setor de serviços onde a interação entre seres humanos é mais forte, dificultado a substituição dos humanos por máquinas.

Este documento foca-se também nos mercados financeiros devido à sua importância para toda a sociedade, pois os preços dos bens e serviços que usamos todos os dias dependem do comportamento destes mercados.

O fato destes mercados serem influenciados por muitas variáveis, faz com que se torne um tópico complexo adequado para aplicar os métodos da IA.

1.2. OBJETIVOS

Os objetivos principais deste trabalho são os definidos de seguida:

- O desenvolvimento de um algoritmo capaz de lidar com a complexidade dos mercados com uma precisão aceitável. O horizonte de previsão aponta para o período de um dia, pois valores superiores requerem algoritmos mais complexos e que ainda não se encontram consolidados a nível de investigação
- Obter um grau maior de conhecimento da ferramenta MATLAB.

1.3. ORGANIZAÇÃO DO RELATÓRIO

O capítulo 2 vai introduzir os principais instrumentos financeiros tais como: *Stocks*, *bonds*, matérias-primas, *forex*, indexes e derivativas. É também abordado o estado da arte (capítulo 3), onde são descritos os principais métodos para analisar os mercados, são referidos em maior detalhe vários modelos usados na inteligência artificial, tais como métodos de redução de dimensionalidade, técnicas de *clustering* e redes neuronais.

Com base nesta síntese, selecionam-se vários para uma implementação através da ferramenta *MATLAB* (capítulo 4).

Finalmente no capítulo 5 vai-se retirar as conclusões de todo o trabalho desenvolvido.

2. MERCADOS FINANCEIROS

Os mercados financeiros têm um lugar essencial na sociedade, pois é através deles que os bens usados pela maioria da população mundial são negociados. Como exemplos, temos casos do gás natural, petróleo, alimentos, e muitos outros. É também através destes mercados que as empresas conseguem encontrar investidores para se financiarem e, assim, se expandirem.

Estes mercados podem ser um lugar físico, onde as pessoas (*traders*) podem negociar instrumentos do tipo *stocks* de empresas (figura 1), como é o caso da bolsa de Nova York (*NYSE-New York Stock Exchange*), ou podem ser um servidor, como é o caso da *National Association of Securities Dealers Automated Quotations* (NASDAQ) onde todas as negociações são feitas eletronicamente (figura 2).

Certos instrumentos estão listados noutras bolsas, como é o caso das opções e indexes, que são negociados na CBOE (*Chicago Board Options Exchange*), ou dos contratos futuros sobre *matérias-primas* que são negociados na CME (*Chicago Mercantile Exchange*).

Por seu lado, as divisas são negociadas pelos bancos centrais dos países que consigam emitir moeda (*forex*).

Normalmente, quando um instrumento é listado numa bolsa é-lhe atribuído um nome (*ticker*) que normalmente é composto pelas iniciais da empresa. Assim, é possível

identificar facilmente a empresa, como por exemplo, *Apple computers* que se designa por AAPL, *Microsoft* que origina MSFE.

Como a estrutura dos mercados norte americanos é das mais desenvolvidas do mundo, neste documento faz-se referência às suas bolsas e instrumentos. No entanto, cada país é responsável por criar e regular os seus instrumentos. No caso dos EUA a entidade reguladora é a SEC (*Securities and Exchange Comission*) [**3**] [**4**] [**5**] [**6**].

Figura 1 NYSE [**75**]

Figura 2 NASDQ [**76**]

2.1. *STOCKS/EQUITIES/SECURITIES*

Quando uma empresa tem como objetivo financiar-se, os proprietários podem vender ao público uma parte dessa empresa (IPO *Initial Public Offer*). Essa parte é depois dividida em partes iguais que se chamam ações (*shares*). Quando um investidor compra ações duma empresa, passa a ser um dos co-proprietários da empresa e, quando os lucros são distribuídos, o investidor recebe a respetiva parte (dividendos) [**7**].

Na figura 3 pode ver-se algumas das empresas listadas na NYSE e a azul os seus respetivos *tickers*.

Figura 3 *Tickers* de algumas empresas da NASDAQ e da NYSE [**8**].

Usualmente para visualizar a progressão dos preços dos *stocks* são usados gráficos, sendo um dos mais comuns e informativos o gráfico designado por *candle sticks* (CS). Na figura 4 pode ver-se o gráfico da empresa ABB, onde é possível visualizar as CS a verde e a vermelho.

Cada CS equivale a um período que pode ir desde 1 minuto até 1ano. O topo do corpo da CS é o *open price* (OP) Ou seja se o período for diário, o OP é o preço a que o *stock* está quando a bolsa abre. Por exemplo no caso da NYSE o OP é às 9h:30. O *close price* (CP) é o preço do stock no momento que a bolsa encerra (16h). Durante a noite e os fins-de-semana não existem negociações. Se o OP for maior que o CP desse mesmo dia, então a CS é vermelha, o que significa que o stock fechou o dia em queda, Se o CP é maior que o OP então a CS fica verde.

O *High/Low* são os valores mais alto e mais baixo que o preço atingiu nesse mesmo dia (período da CS)

Outro indicador importante é o do volume. Este indicador representa a quantidade de negociações (de compra e venda) que foram feitas nesse dia. Por exemplo, grandes subidas no volume, podem significar que o preço do stock vai começar a subir, ou a descer, consideravelmente [**9**] [**10**].

Figura 4 Gráfico de candle sticks (stock da ABB) [**11**].

2.2. INDEXES

Os *indexes* são instrumentos que representam uma secção do mercado. Por exemplo, o S&P500 contém as 500 maiores empresas listadas na bolsa de Nova York e na NASDAQ. Para calcular o valor do índice soma-se o valor total dos stocks das 500 empresas e divide-se por um divisor fornecido pelo *index* .

Outro *índex* é o Dow Jones 30 que tem as 30 maiores empresas dos EUA. No caso da Inglaterra existe o FTSE 100 que tem as 100 maiores empresas da LSE (*London Stock Exchange*) [**12**] [**13**] [**14**] [**10**].

2.3. MATÉRIAS-PRIMAS

Este tipo de instrumento refere-se a recursos naturais (figura 5) que podem ser produtos tais como gás, óleo, petróleo, cereais, gado, algodão, ouro, prata, e muitos outros. Quando um investidor compra estes instrumentos (através de contratos *forward* ou futuros) deve pagar uma taxa chamada *carry*, pois o armazenamento destes recursos tem custos [**15**] [**10**].

Figura 5 Algumas matérias-primas negociadas na CME (Cobre, Algodão, Petróleo, Platina) [**16**].

2.4. DIVISAS/*FOREX* (*FOREIGN EXCHANGE*)

Quando um país (A) exporta produtos, faz com que o país que os importa (B) esteja a trocar a sua moeda (B), pela moeda do país A.

Por exemplo se a Europa exportar para os EUA, significa que os EUA estão a vender dólares no *FOREX* para comprar euros (€). A venda de grandes quantidades de USDs no mercado faz com que a oferta aumente em relação à procura. Isto tem como consequência o enfraquecimento do USD em relação ao euro cuja procura aumentou. Desta maneira, as divisas têm sempre de ser compradas aos pares como é o caso do EUR/USD, NZD/USD, AUD/USD. Um fator que influencia muito os pares de divisas é a taxa de juros dos países envolvidos. Se um investidor adquirir o par AUD/JPY, então este vai ganhar o *carry* durante todos os dias em que possuir a divisa, se a taxa de juro da Austrália for superior à do Japão [**17**] [**10**].

Por exemplo, se um investidor comprar 1 lote de AUD/JPY= 100000 unidades e a taxa de juros da Austrália for de 4.5% e do Japão 0.1%, então o investidor ganha todos os dias:

$(0.045 - 0.001)/365 \times 100000 = 12$ AUDs/lote/dia

Major Pairs		Bid	Ask			Bid	Ask
EUR/USD	↓	1.0914[8]	1.0916[5]	**EUR/GBP**	↓	0.7180[8]	0.7182[1]
USD/EUR	↑	0.9160[5]	0.9161[9]	GBP/EUR	↑	1.3923[5]	1.3926[0]
GBP/USD	↓	1.5199[5]	1.5201[3]	**EUR/CHF**	–	1.0329[7]	1.0332[2]
USD/GBP	↑	0.6578[4]	0.6579[2]	CHF/EUR	–	0.9678[5]	0.9680[8]
USD/CAD	↑	1.2538[6]	1.2540[7]	**AUD/USD**	–	0.7601[8]	0.7603[4]
CAD/USD	↑	0.7974[0]	0.7975[4]	USD/AUD	–	1.3152[0]	1.3154[8]
USD/CHF	↑	0.9462[4]	0.9465[0]	**EUR/JPY**	↓	136.29[6]	136.31[8]
CHF/USD	↓	1.0565[2]	1.0568[1]	JPY/EUR	–	0.0073[4]	0.0073[4]
USD/JPY	–	124.86[3]	124.87[7]	**GBP/JPY**	↓	189.79[1]	189.82[1]
JPY/USD	–	0.0080[1]	0.0080[1]	JPY/GBP	–	0.0052[7]	0.0052[7]

Figura 6 Taxa de intercâmbio das diferentes divisas em (01-06-2015) [**18**]

Na figura 6 pode ver-se os *ratings* dos pares de divisa mais negociados no mundo. Estes pares, por serem os mais negociados, normalmente têm muita liquidez (facilidade com que um instrumento pode ser convertido em dinheiro), o que faz baixar o *spread*. O *Ask price* (*Buy*) é o melhor preço a que um investidor pode comprar no mercado, ou seja é o preço mais baixo que estão a pedir ao investidor. O *Bid price* (*Sell*) é o preço mais alto a que o investidor pode vender a divisa. Assim se o investidor comprar o par EUR/USD e o vender no mesmo instante, então vai ter perdas, pois o *Ask* é 1.09165$ e o *Bid* é 1.09148$ ou seja uma perda de 0.00017$. Esta diferença de valores chama-se *spread*.

Agora vejamos o *spread* de uma divisa ilíquida que é o caso da lira da Turquia com o yen do Japão.

TRY/JPY	–	46.46[9]	46.52[1]
JPY/TRY	–	0.0215[0]	0.0215[2]

Podemos ver que o *spread* é 46.521 JPY – 46.469 JPY = 0.052 JPY = 0.00041$. O fato deste par não ser negociado com muita frequência, faz com que a sua liquidez seja muito baixa. Isto tem como consequência a subida do spread para mais do dobro em relação ao EUR/USD.

A liquidez tem uma influência muito grande para os investidores, não só no *forex* mas também noutros mercados, especialmente quando a quantidade de lotes/instrumentos a negociar é elevada [**10**].

2.5. BONDS

Quando um país, ou uma empresa, precisa de se financiar pode faze-lo com recurso a um empréstimo, a emitir ações (ou expandir o stock). Outra maneira de se financiar consiste em emitir *bonds* no mercado. Uma *bond* é basicamente um empréstimo e a razão para as emitir, é que, em muitas situações, a empresa/país vai conseguir-se financiar de maneira mais barata do que se recorrendo a um banco. Portanto, a entidade que emite tais instrumentos passa a ser um devedor e a entidade que compra as *bonds* passa a ser o credor.

Por exemplo se o governo dos EUA decidir emitir 10M$ em *bonds* de 1000$ cada, então vai ter de emitir 10000 *bonds*. Estes contratos têm várias características tais como:

- **Preço Nominal/Principal** – Preço total que a instituição/País vai pagar na maturidade (este preço é fixo);
- **Preço no mercado** - Quantia total a pagar (preço oferecido no mercado o qual é flutuante);
- **Cupão** – Quantia a ser paga em várias partes para compensar o credor do risco que corre. Se o cupão for pago no final, então o contrato passa-se a denominar por *Zero cupon Bond*;
- ***Yield*** - É quantia recebida num cupão a dividir pelo preço no mercado (mede o retorno);
- **Maturidade** – Data final em que o principal tem de ser pago;

Usando o exemplo acima, e supondo que o preço nominal é de 1000$, a maturidade é de dois anos e o cupão é de 50$ anual. Então o *yield* fica $50/1000 \times 100 = 5\%$. O preço no mercado e o valor dos cupões, são influenciados pela oferta/procura destes instrumentos,

pelo risco da empresa/país não cumprir com o pagamento (*default*), com as taxas de juro e a inflação.

Se houver muita procura, o preço de uma bond pode subir baixando o *yield* fazendo com que se torne menos atrativo comprar o instrumento (se o preço no mercado subir 100$ (1100$) então o *yield* fica $50/1100 \times 100 = 4.5\%$).

No caso das taxas de juros, se estas subirem faz com que o preço das *bonds* baixe, pois os investidores podem ir ao mercado e obter novas *bonds* com yields mais atrativos, ou então podem por o seu dinheiro num banco e receber dividendos.

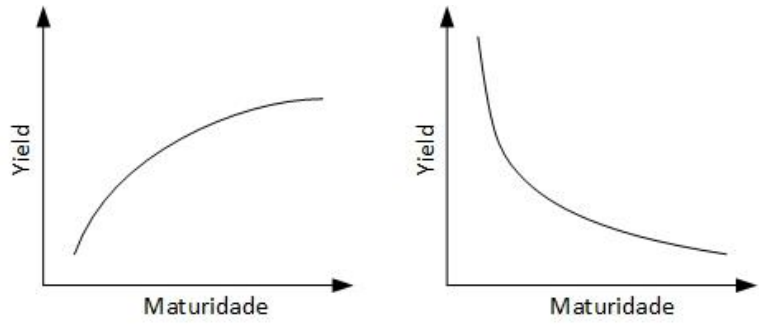


Figura 7 Curvas de yield (normal e invertida).

Na figura 7 podemos ver uma curva de *yield*. No caso de se tratar de *bonds* da tesouraria dos EUA as maturidades são de 3/6 meses e 2/3/5/10/30 anos. Estes dois gráficos podem fornecer muita informação sobre o estado da economia do país. No caso da curva normal pode ver-se que as *bonds* com maturidade baixa (empréstimos de curto prazo), têm um retorno (*yield*) muito baixo e as *bonds* de longo prazo têm um retorno alto. Isso pode significar que os investidores estão a prever que a longo prazo as taxas de juros vão subir. Se os juros subirem então os investidores vão começar a vender as *bonds*, fazendo o preço de mercado baixe, o que por sua vez faz com que o *yield* suba. No caso da curva invertida, os investidores preveem que os juros vão baixar, isso faz aumentar a procura destes contratos, fazendo assim baixar o *yield*.

As razões para as taxas de juro subirem são várias. A curto prazo elas são definidas pelos bancos centrais e podem ser vistas através dos seus indicadores como a LIBOR, EURIBOR, etc. No entanto, a longo prazo os fatores macroeconómicos definem o valor das taxas.

Uma das razões para a subida das taxas de juro é para prevenir a inflação, pois uma taxa alta evita que as pessoas adquiram empréstimos diminuindo a quantidade de moeda que entra no sistema financeiro. Como consequência muitas empresas não se podem financiar e a economia baixa o seu desenvolvimento, o que faz com que o balanceamento entre o crescimento económico e a inflação seja uma tarefa delicada. Outra razão para fazer subir as taxas é para comprar de volta as *bonds* que o próprio país emitiu. Se, por exemplo, a reserva federal dos EUA subir as taxas, então faz com que os investidores comecem a vender as suas *bonds* e, assim, estas já podem ser compradas pelos EUA.

As *bonds* que oferecem yields muito altos, são normalmente consideradas *junk bonds* pois o risco de *default* é muito grande. Para medir o risco de não cumprimento existem as entidades de *rating*. Estas classificam as empresas segundo critérios próprios.
Na figura 8 podemos ver a tabela de avaliação de três agências de *rating* (*Moody's, S&P, Fitch*). Por exemplo o *rating* AAA significa que a instituição em causa tem uma grande probabilidade de cumprir as suas obrigações perante os credores. Logo instrumentos de divida avaliados com AAA oferecem as taxas de juro mais pequenas [**20**] [**20**] [**10**].

Moody's		S&P		Fitch		
Long-term	Short-term	Long-term	Short-term	Long-term	Short-term	
Aaa	P-1	AAA	A-1+	AAA	F1+	Prime
Aa1		AA+		AA+		High grade
Aa2		AA		AA		
Aa3		AA-		AA-		
A1		A+	A-1	A+	F1	Upper medium grade
A2		A		A		
A3	P-2	A-	A-2	A-	F2	
Baa1		BBB+		BBB+		Lower medium grade
Baa2	P-3	BBB	A-3	BBB	F3	
Baa3		BBB-		BBB-		
Ba1	Not prime	BB+	B	BB+	B	Non-investment grade speculative
Ba2		BB		BB		
Ba3		BB-		BB-		
B1		B+		B+		Highly speculative
B2		B		B		
B3		B-		B-		
Caa1		CCC+	C	CCC	C	Substantial risks
Caa2		CCC				Extremely speculative
Caa3		CCC-				In default with little prospect for recovery
Ca		CC				
		C				
C		D	/	DDD	/	In default
/				DD		
/				D		

Figura 8 Tabelas de *rating* (Moody's S&P e Fitch) [**22**].

2.6. DERIVATIVOS

O nome destes instrumentos resulta do fato que o seu preço deriva dos outros instrumentos financeiros acima referidos. De seguida são descritos os principais tipos de derivativos.

2.6.1. OPÇÕES (AMERICANAS)

As opções Americanas são um contrato no qual a pessoa que compra, tem o direito, mas não o dever, de comprar (*Call option*)/vender (*Put Option*) uma certa quantidade do instrumento a um determinado preço (*Strike price*), dentro de um prazo estipulado (*Expire time*) . Para compensar a pessoa que escreve o contrato pelo risco que corre, o comprador paga uma taxa chamada *premium*.

Um exemplo seria um contrato que permitisse alguém comprar 100 ações da IBM a 175$ (*Strike price*), com o contrato a expirar no prazo de um mês e com *premium* 2$/ação. Se cada ação da IBM custar 171.71$, então a pessoa que compre a *call option* precisa que o preço da IBM suba mais do que 3.29$ dentro de um mês, para obter lucro. Se o investidor comprasse o contrato então ia custar-lhe $2\$ \times 100 = 200\$$. Caso no dia 20 a IBM subisse para 178$, o investidor podia exercer o contrato e a pessoa que o escreveu era obrigada a vender 100 ações por 175$, o que ia custar ao investidor 17500$. No entanto o investidor podia vender as ações no mercado por 178$ e lucrava 17800-17500=300$. Se, passados 30 dias, o preço fosse menor do que 175$, então o investidor não podia exercer a opção e perdia 200$.
Na figura 9 pode-se ver, à esquerda, o diagrama de Ganhos/Perdas da pessoa que compra o contrato e à direita o diagrama para a pessoa que escreve o contrato.

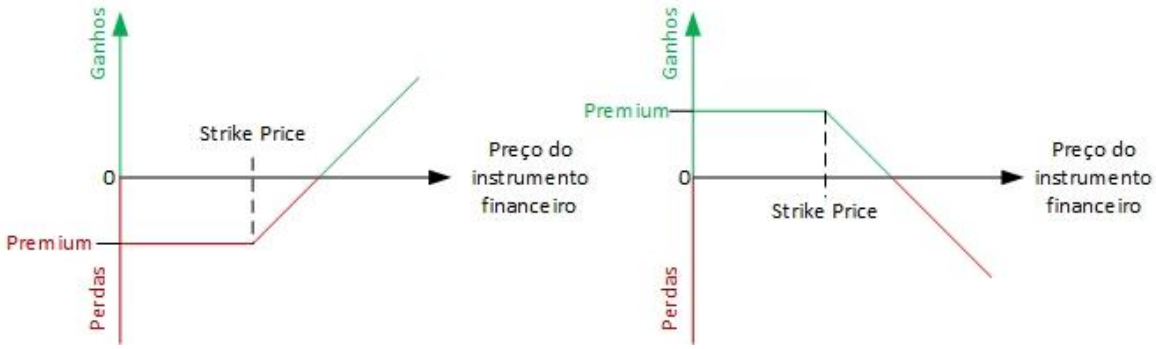


Figura 9 Gráfico dos ganhos de um comprador e de um vendedor de uma *call option* [**23**].

Como se pode ver é extremamente arriscado escrever contratos de *call options*, pois os lucros estão limitados ao *premium*, mas as perdas podem ser ilimitadas, pois o preço de uma ação pode ser teoreticamente infinito. Escrever estes contratos, sem qualquer tipo de proteção (*hedging*), pode levar à perda total do capital investido. Esta pratica também é conhecida por *writing/selling naked calls* [**24**] [**10**].

Outro aspeto importante a considerar é que estes contratos permitem à pessoa que os compra alavancar o capital investido.
Por exemplo, se o investidor tivesse comprado diretamente as ações da IBM por 171.71$ e passados 20 dias as ações estivessem a 178$, então o retorno sob o investimento era:

$$\frac{178\$-171.71\$}{171.71\$} \times 100 = 3.66\%. \quad (1)$$

No entanto, com o contrato fica o seguinte ROI:

$$\frac{178\$-171.71\$}{2\$} \times 100 = 314.5\%. \quad (2)$$

Note-se que as perdas em percentagem são mais elevadas, pois se o contrato expirar e o preço estiver abaixo de 175$, então o investidor perde 100% do capital. Todavia, em valor absoluto, a quantidade perdida é maior caso o investidor comprasse as ações, pois uma perda de 3.66% equivale a uma perda de 6.29$, enquanto que uma perda de 100% do capital numa opção equivale a uma perda de 2$.
Na figura abaixo podemos ver, à esquerda, o diagrama de Ganhos/Perdas da pessoa que compra o contrato (*Put Option*) e, à direita, o diagrama da pessoa que escreve o contrato.

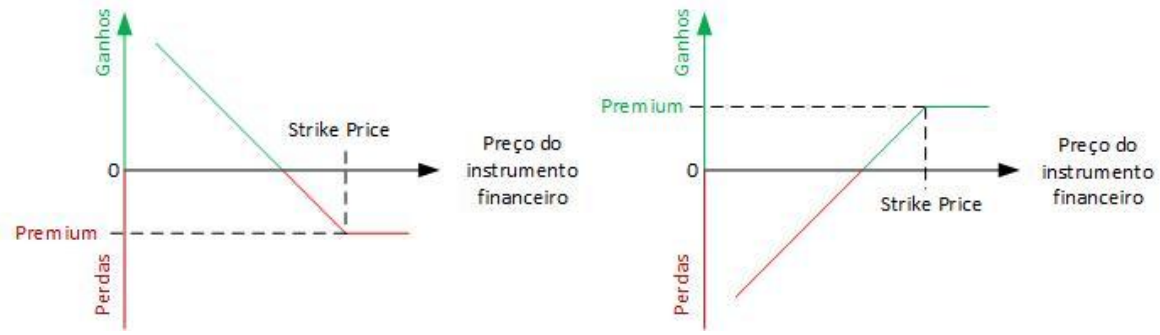


Figura 10 Gráfico dos ganhos de um comprador e de um vendedor de uma *put option* [**25**].

Neste caso a pessoa que compra o contrato espera que o preço do instrumento desça abaixo do *strike price*. Se isso acontecer, então o investidor tem o direito de vender as ações da IBM à pessoa que escreveu o contrato por 175$ quando as ações no mercado custam 171.71$.
Este contrato é importante pois permite proteger os investidores caso haja algum *crash* do preço de um instrumento [**24**] [**10**].

Outro tipo de opções são as opções europeias. A diferença entre estas e as americanas é que o contrato é exercido exatamente na altura em que expira. O fato de se chamarem opções americanas ou europeias, não está relacionando com o lugar geográfico onde são criadas [**10**].

2.6.2. CONTRATOS FUTUROS

Os futuros são contratos nos quais o investidor tem o dever pré-comprar o instrumento em causa (ações, matérias primas, divisas, etc) por um preço estabelecido e pode fazer isso dentro de um determinado prazo que também está definido no contrato [**10**].

2.6.3. CONTRATOS *FORWARD*

Os *Forwards* são contratos iguais aos futuros, mas com a diferença que o contrato tem de ser exercido exatamente na altura em que expira [**10**].

2.6.4. *SWAPS*

Existem *swaps* de vários instrumentos, sendo um dos mais usados o *swap* das taxas de juros.
Na figura 11 pode-se ver o seguinte exemplo:

A empresa A fez um empréstimo de 5M€ e o banco A oferece uma das seguintes opções:

- Pagar o empréstimo com uma taxa fixa 7%;
- Pagar uma taxa flutuante através da LIBOR.

A empresa B também fez um empréstimo de 5M€ e foram oferecidas pelo banco B as seguintes escolhas:

- Pagar o empréstimo com uma taxa fixa 10%,
- Pagar uma taxa flutuante através da LIBOR+1%.

O objetivo da empresa A é pagar uma taxa variável e o objetivo da empresa B é pagar uma taxa fixa. Então, as duas empresas pedem aos seus bancos exatamente o oposto dos seus objetivos, ou seja a empresa A vai pagar 7% fixo e a B a paga a taxa variável LIBOR + 1%. Depois, as duas empresas vão a um banco *swap* e este banco paga uma taxa fixa de 8% à empresa A e em contrapartida a empresa A paga a LIBOR ao banco *swap*, assim a empresa A pode pagar ao seu banco os 7% fixos e ainda sobra $8\% - 7\% = 1\%$. Como também tem de pagar a LIBOR ao banco *swap* faz com que o total seja $\text{LIBOR} - 1\%$ que é

menor que a LIBOR. Logo, a empresa A lucra 1% com a transação e, na realidade, paga uma taxa variável, o que é o seu objetivo. A empresa B paga 8.5% ao banco *swap* e recebe em troca a LIBOR. No total a empresa B paga uma taxa fixa de $8.5 + 1\% = 9.5\%$ que é inferior a 10%, Logo, a empresa B lucra 0.5%. Por último, o banco *swap* lucra através do spread $8.5\% - 8\% = 0.5\%$ [**10**].

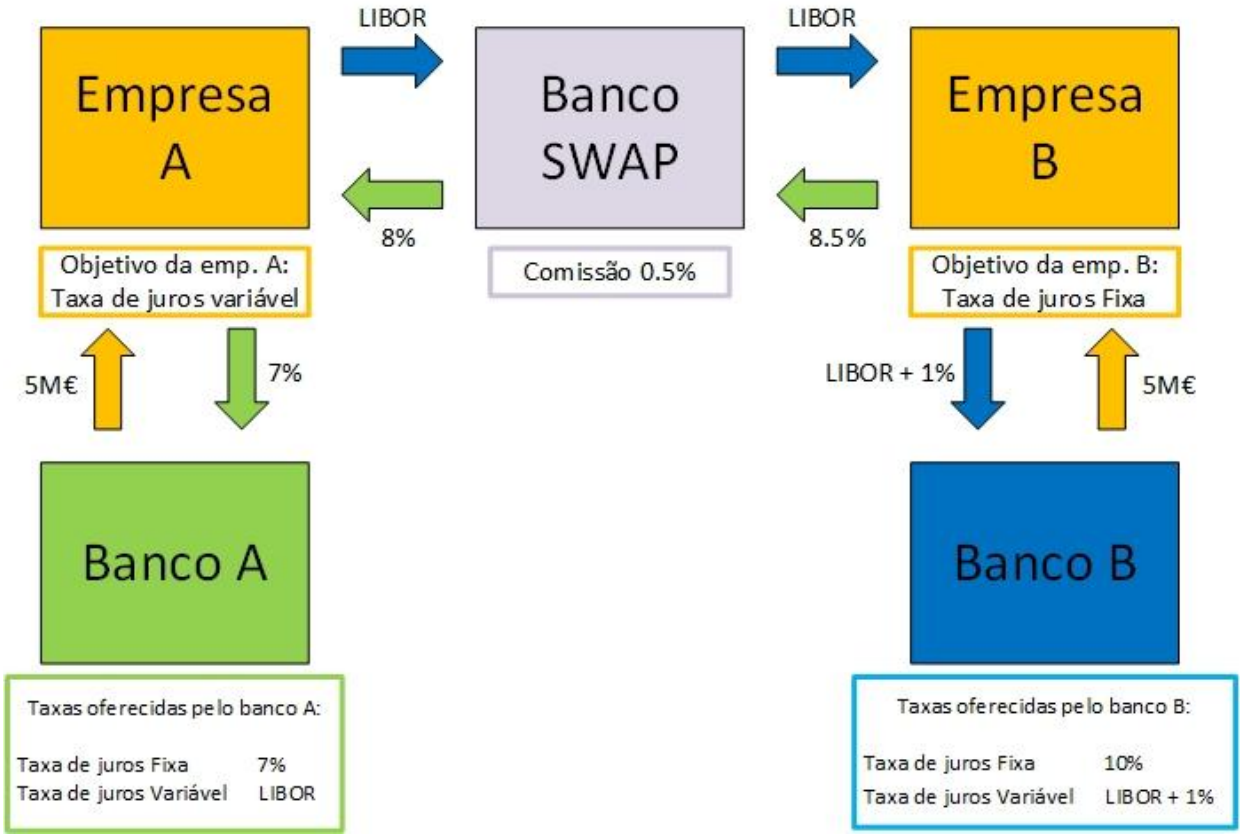


Figura 11 *Swap* de taxas de juros entre duas empresas [**26**].

3. ESTADO DA ARTE

3.1. EFICIÊNCIA DOS MERCADOS

Antes de ser abordado qualquer tipo de análise deve ser colocada a questão se os movimentos dos preços dos mercados em si são totalmente fruto do acaso (*random walk*), ou se é possível detetar padrões conseguindo assim fazer previsões. Isto leva ao estudo da hipótese dos mercados eficientes (HME). A HME pode ser dividida em três formas:

- **Eficiência dos mercados forte –** Nesta forma de ver os mercados, é deduzido que estes são tão rápidos e eficazes a responder a todos os fatores que os influenciam, que é impossível alguém fazer qualquer tipo de previsão.

  Este tipo de "eficiência" não é muito realista, pois basta um breve estudo dos mercados para perceber que na verdade existem "ineficiências" como, por exemplo, a arbitragem, caso um instrumento esteja em dois mercados mas com preços diferentes. Isto acontece pois a informação não circula instantaneamente.

Outra razão por esta maneira de ver os mercados não ser correta, é que ela implica que todos os indivíduos que participam no mercado agem racionalmente, de maneira a alocar o seu capital e bens o mais eficiente possível, o que está longe de ser a realidade. Cada indivíduo reage de maneira diferente às circunstâncias que o rodeiam e muitas das vezes cede a fatores psicológicos como o medo, otimismo excessivo, e outros.

- **Eficiência dos mercados semi-forte –** Neste tipo de eficiência os mercados são muito rápidos a reagir à informação disponível ao público. No entanto a informação privada pode gerar oportunidades de lucro. Por exemplo, as pessoas relacionadas com uma empresa podem conseguir adquirir informação mais rápidamente que o público. Deve-se ter em atenção que negociar instrumentos financeiros com informação privilegiada pode ser ilegal (*Inside Trading*).

- **Eficiência dos mercados fraca** – Aqui os mercados respondem à mesma informação (tanto pública como privada) a diferentes velocidades, gerando muitas oportunidades de lucro. Neste tipo de eficiência todos os tipos de análises podem funcionar [**27**].

Na realidade os mercados podem variar nas suas formas. Por exemplo o S&P 500 é um mercado bastante eficiente. Isto acontece porque o setor financeiro nos EUA está muito desenvolvido e a informação é disponibilizada muito facilmente, especialmente para os *hedge funds* e *mutual funds* que têm mais capital para investir em análise e pesquisa.

Em seguida vai ser descrito os vários tipos de análise (Fundamental, Técnica, Quantitativa)

3.2. ANALISE FUNDAMENTAL

Este tipo de análise envolve uma quantidade enorme de fatores, em que uma grande parte deles pode ser qualitativa e subjetiva à pessoa que os analisa. Se o objetivo for tentar analisar quais as melhores empresas para investir, normalmente usa-se a estratégia *top down*. Isto faz-se analisando indicadores macroeconómicos dos países para ver quais as melhores economias. Estes indicadores podem ser: *Gross Domestic Product* (GDP),

Employment Situation Report, *Purchaser Manager Index* (PMI), *Consumer Price Index* (CPI), etc.

Depois de se avaliar qual o ciclo económico em que os diferentes países se encontram (expansão ou recessão), vai-se descobrir quais os setores da economia que têm melhor ou pior desempenho no âmbito do ciclo económico. Em seguida analisa-se essas empresas individualmente através dos seus *cash flows*, *balance sheets* e *income statements* [**28**].

3.3. ANALISES TÉCNICAS

As análises técnicas são um método de análise de mercado com recurso a indicadores técnicos, sendo que dois dos mais usados são: o indicador de média deslizante (*moving average* MA) e o estocástico.

A média deslizante faz uma média aritmética de todos os preços contidos num determinado período de tempo.

Na figura 12 podemos ver o *stock* da *apple* (período diário), com uma média neste caso de 30 dias (MA 30). Pode-se também verificar que a MA ajuda a filtrar o ruido do *stock*. Quanto maior for a janela em causa mais ruido é filtrado. No entanto, isto faz com que o indicador se atrase excessivamente aos movimentos do *stock*. Se a janela for muito baixa então o indicador não se atrasa tanto, mas também não consegue filtrar adequadamente o ruido.

Figura 12 Gráfico do *stock* da *apple* (período diário) e o indicador MA 30 (azul).

Figura 13 Gráfico do *stock* da *apple* (período diário) e o indicador MA 5 (azul).

Na figura 12 pode ver-se uma MA 5. Esta MA não tem um desfasamento tão elevado como a MA 30. No entanto, pode ver-se que a MA 5 oscila muito mais com o movimento do preço, fazendo com que os falsos sinais de compra e venda sejam dados com mais frequência.

Uma estratégia comum é: quando o preço estiver abaixo da média e depois começar a subir cruzando-se com a mesma, gera-se um sinal de compra (seta verde figura 12); se o oposto acontecer gera-se um sinal de venda (seta vermelha figura 12). Apesar de parecer com aplicação direta os indicadores técnicos devem ser vistos no contexto dos outros métodos de análise e nunca devem ser usados isoladamente [**29**].

Figura 14 Indicador MA 5 (amarelo) e o EMA 5 (Azul).

Outra versão da MA é a média exponencial ou EMA (figura 14 a azul). A EMA põe um peso maior nos preços mais recentes pois eles têm maior influência no futuro. É possível usar uma MA normal e uma EMA para gerar sinais de compra e venda. Quando a EMA cruzar a MA (a amarelo) um sinal é gerado [**30**].

O indicador estocástico é outro indicador comum, Este indicador é composto por três variáveis que são: O período do %K, o %K, e o %D.

A formula do indicador estocástico é dada por:

$$\%K = \frac{Current\ Close - Lowest\ Low}{Highest\ High - Lowest\ Low} \times 100. \tag{3}$$

O período do *%K* é normalmente 14 (neste caso dias). Caso se queira um *%K* mais lento, normalmente é adotada uma MA 3 do *Highest High* e *Lowest Low*.
O *%D* é uma MA (3) do *%K*.

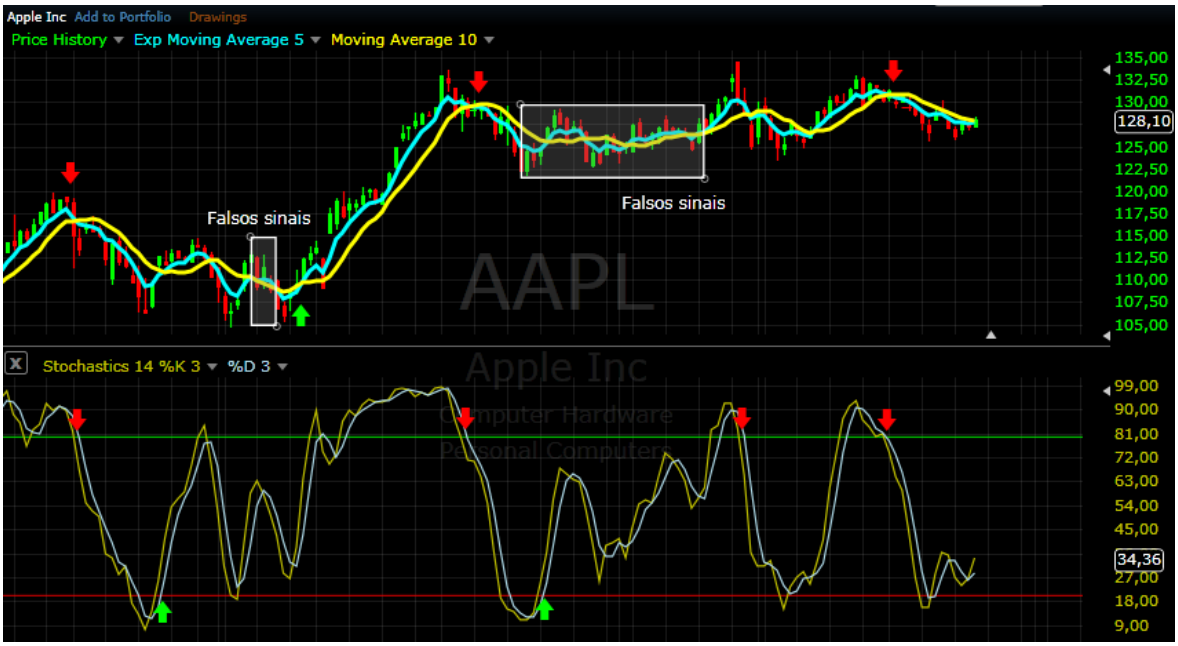


Figura 15 Indicador estocástico (parte inferior do gráfico).

Na figura 15 pode ver-se um oscilador estocástico de período 14 dias, a média dos *inputs* do %K é 3 dias e o %D uma MA dos últimos 3 %Ks.

Quando o %K for superior a 80% (linha verde), diz-se que o instrumento financeiro está sobrevalorizado e é provável que o preço desça. Um sinal de venda é gerado quando o %K cruzar o %D e estes desçam abaixo da linha verde (setas vermelhas). Caso o %K e o %D estejam abaixo de 20% e voltem a subir e cruzar a linha vermelha gera-se um sinal de compra (setas verdes).

É possível ajustar as MAs de acordo com a volatilidade do *stock*, ou até usar uma EMA em vez de uma MA. Quanto mais pequenas forem as MA mais proactivo o indicador se torna e, ao mesmo tempo, mais falsos alarmes são geradas.

No caso de a MA dos inputs do %K for 1 dia o indicador chama-se *fast stochastic* se for maior que 1 chama-se *slow stochastic* no caso contrário [**30**] [**31**].

Estes indicadores podem ser incluídos como estratégia na abordagem *topdown* acima descrita. Por exemplo, o investidor analisa os indicadores macroeconómicos e verifica em que parte do ciclo económico cada país se situa. Consequentemente analisa se deve investir mais em *stocks*, *bonds* ou outros instrumentos (normalmente é uma mistura de instrumentos para diversificar o portefólio). De seguida vê as empresas que mais se destacam nesse ciclo económico e analisa a estrutura dessas empresas. Por último, pode usar os indicadores técnicos para encontrar o *timing* certo para entrar no mercado.

Outro fator que também é importante considerar são os média. No caso de ocorrerem notícias negativas ou positivas sobre as empresas, ou sobre a economia, o preço dos instrumentos em causa podem ser afetados. Algumas empresas são mais sensíveis do que outras às notícias que os média transmitem. Por exemplo, uma má noticia para um banco pode ser muito grave pois pode desencadear o que é conhecido como *run on the bank*. Isto acontece quando os clientes perdem confiança nesse banco e começam a retirar os seus depósitos, todos ao mesmo tempo, deixando o banco sem liquidez.

Outros setores como os de distribuição de bens essenciais (distribuição de água, energia, saúde, etc) são menos afetados por notícias negativas do que o setor financeiro ou tecnológico [**32**].

3.4. ANALISE QUANTITATIVA

A análise quantitativa tem como objetivo fazer previsões, mas apenas com o recurso de ferramentas matemáticas normalmente usadas para estudar séries temporais, como é o caso dos métodos de regressão (AR, MA, ARMA, ARCH, etc) e dos métodos não paramétricos úteis para funções não lineares (regressão de *Kernel*, AR aditiva, etc).

Na análise de séries temporais aplicadas aos mercados financeiros nomeadamente *stocks*, é comum descrever o desempenho de um *stock* através do seu retorno em vez do seu preço. Os retornos têm certas vantagens na análise estatística e para os analistas são uma medida independente do desempenho de um investimento.

Os retornos podem ser calculados da seguinte maneira.

Se considerarmos o retorno de 1 período tem-se:

$$R_t = \frac{P_t}{P_{t-1}} - 1. \tag{4}$$

Para *k* períodos tem-se a seguinte fórmula:

$$R_t[k] = \frac{P_t - P_{t-k}}{P_{t-k}}, \tag{5}$$

onde *R* é o retorno e *P* é o preço do *stock*.

Caso o instrumento em causa pague dividendos, estes devem ser contabilizados, onde *D* é o dividendo pago desde o período *t-1* ou *lag (1)* até *t*.

$$R_t = \frac{P_t + D_t}{P_{t-1}} - 1. \tag{6}$$

Outra característica importante ao analisar o *stock* através do seu retorno, é que ao derivar os preços (retornos) a série fica estacionária no tempo. Isto só é verdade se for comprovado que o valor esperado dos retornos é constante em todos os instantes ($E(R_t) =$ ⍰) e que a sua covariância só é dependente do valor t e *t-k,* ou seja $COV(R_t, R_{t-k}) = Y_k$(também conhecido como autocovariancia). Isto implica que se torna muito mais fácil fazer previsões para uma série estacionária. Na figura 16 podemos ver a diferença de duas séries, onde uma é estacionária no tempo (em cima) e a outra é não-estacionária (em baixo), pois o gráfico não retorna ao mesmo valor e a sua variância não é constante [**33**].

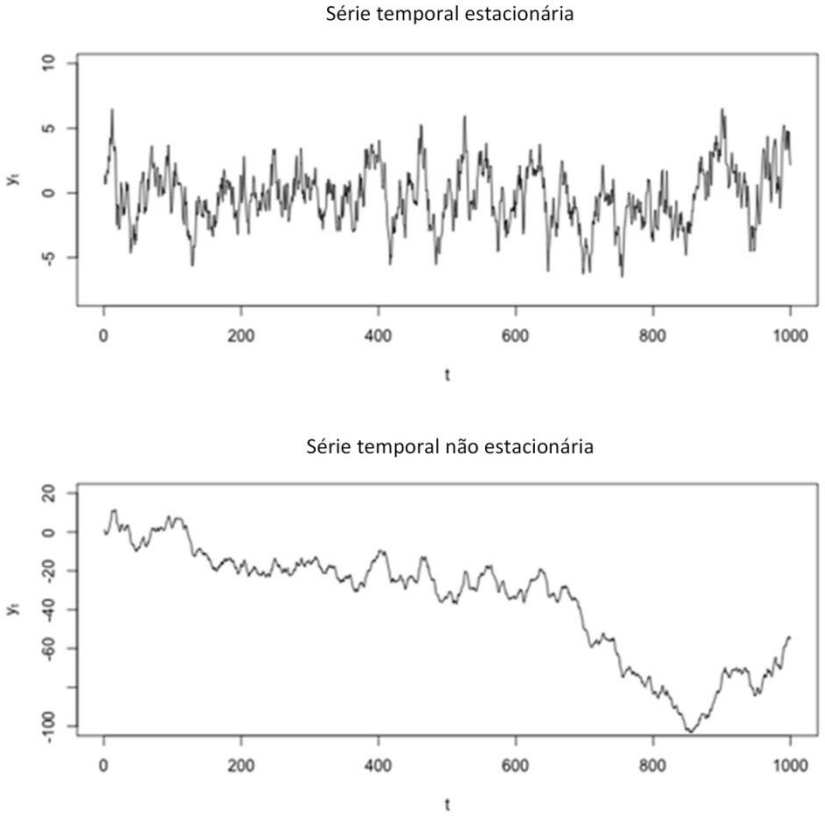


Figura 16 Série temporal estacionária (gráfico superior) e não estacionária (gráfico inferior) [**34**].

Através da covariância é possível achar a auto-correlação, onde VAR é a variância:

$$\rho_k = \frac{COV(R_t, R_{t-k})}{VAR(R_t)}, -1 \leq \rho_k \leq 1. \quad (7)$$

O resultado $\rho_k = 0$ significa que o valor em *k* não tem qualquer influência no valor *t*, o que não é muito útil para fazer previsões. O resultado $\rho_k < 0$ significa que o valor em *k* está inversamente correlacionado com *t*, ou seja, quando um aumenta o outro tende a diminuir. Se $\rho_k > 0$ significa que o valor em *t* tende a aumentar quando o valor no instante *k* aumenta.

Para achar a auto-correlação considerando vários períodos (*lags*) deve-se fazer um teste de hipóteses conhecido como teste de *Ljung-box*. Isto é útil para achar a auto-correlação de um instrumento. A hipótese nula neste teste significa que as primeiras *h* auto-correlações são nulas.

O teste é feito através da seguinte fórmula:

$$Q = n(n+2)\sum_{k=1}^{h}\frac{\hat{\rho}_k^2}{n-k}, \quad (8)$$

$$H_0: \hat{\rho}_k = 0,$$

$$H_1: \hat{\rho}_k \neq 0,$$

onde *n*- É o tamanho da amostra de ρ,.$\hat{\rho}_k$ é a amostra atual, *h* é o número de testes a ser feito.

Se $Q > \chi^2_{1-\alpha,h}$ então a hipótese nula deve ser rejeitada, onde $\chi^2_{1-\alpha,h}$ é o valor critico de rejeição de H_0 com um grau de significância α e *h* graus de liberdade.

Na figura 17 pode-se ver um gráfico com os retornos do *S&P500* e a sua respetiva auto-correlação.

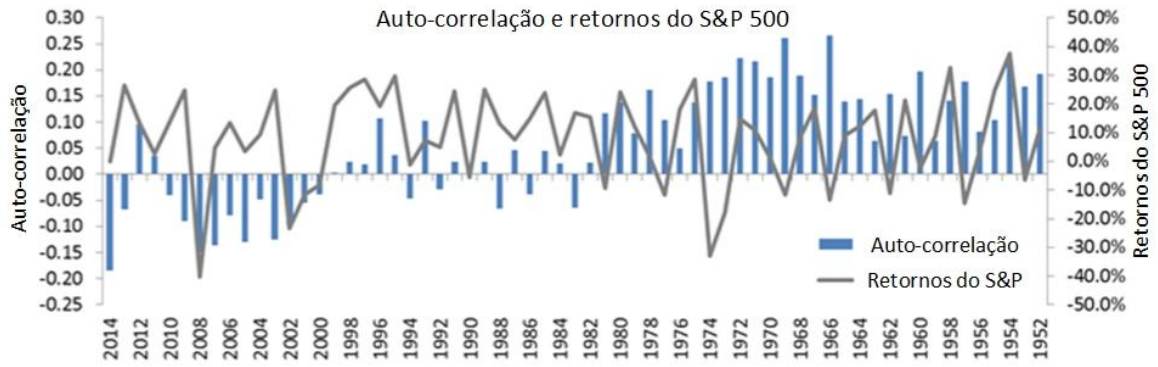


Figura 17 Auto-correlação e os retornos do *S&P500* [**35**].

Séries lineares

Uma série é linear se poder ser escrita da seguinte forma:

$$R_t = \mu + \sum_{k}^{\infty} \psi_k a_{t-k}, \tag{9}$$

onde R_t são os retornos no instante *t*, μ é o valor esperado de *R*, a_{t-k} é um conjunto de variáveis aleatórias independentes cujo valor medio é nulo (ruído), ψ_k são os coeficientes lineares de a_{t-k},

Se R_t for estacionário, então o valor esperado e a variância podem ser escritos da seguinte forma:

$$E(R_t) = \mu, \tag{10}$$

$$VAR(R_t) = \sigma_a^2 + \sum_{k=0}^{\infty} \psi_k^2, \tag{11}$$

onde o somatório da variância tem de convergir, pois a variância não pode tomar valores no infinito.

Modelo Auto Regressivo (*AR(k)*)

O modelo autorregressivo é um modelo que descreve uma variável através da dependência linear dos seus valores passados. Para que o *AR* seja usado, a série temporal tem de ser estacionária (no tempo) e deve haver correlação entre os *lags* em causa [**33**].

Na formula seguinte pode-se ver a representação de um *AR(1)*, ou seja *k=1*:

$$R_t = \psi_0 + \psi_1 R_{t-k} + a_t, \tag{12}$$

onde a_t é um conjunto de variáveis aleatórias independentes (erros ou ruído), ψ_k são coeficientes lineares e o seu valor está relacionado com a função de auto-correlação $\rho_k = \psi_k^k$.

Caso os erros tenham algum tipo de correlações, ou seja, não sejam independentes, pode-se usar a MA (*moving average*). Apesar de o nome ser igual ao indicador técnico, as funções são diferentes. O indicador tenta suavizar o gráfico para descobrir tendências.

O *MA (j)* pode ser descrito da seguinte fórmula:

$$R_t = \theta_0 + a_t + \theta_j a_{t-j}\ , \qquad j = 1, \tag{13}$$

onde θ_j são coeficientes lineares e o seu valor está relacionado com a função de auto-correlação $\rho_j = \theta_j^j$.

Notar que os coeficientes ψ_1 θ_1, no *AR* e no *MA*, não devem ser iguais a 1, pois isto significa que a série é um *random walk*.

Se juntarmos estes dois modelos temos o modelo *ARMA(1,1)*:

$$R_t = \psi_0 + \psi_1 R_{t-1} + a_t + \theta_1 a_{t-1}. \tag{14}$$

Outros modelos como o *GARCH* ou o *ARCH*, são usados para modelar a volatilidade da série [**33**].

O problema em usar estes modelos, ou outros idênticos, é que, para eles funcionarem adequadamente é preciso assumir certos parâmetros que podem não coincidir com a realidade. Por exemplo, assumir que a série se comporta de forma linear, que as suas distribuições são normais, etc.

Os mercados financeiros são influenciados por muitas variáveis, as quais muito provavelmente vão inserir não linearidades nos seus instrumentos e, por consequência, alterar também a suas funções de distribuição.
Durante o *crash* de 2008-2009 vários portefólios ficaram expostos a riscos elevados. Muitos dos modelos usados implicavam funções de distribuição normal, quando na realidade as distribuições dos instrumentos tinham o que é conhecido como *fat tails* (figura 18), ou seja, a probabilidade de eventos indesejados é maior do que o esperado (pela distribuição normal).

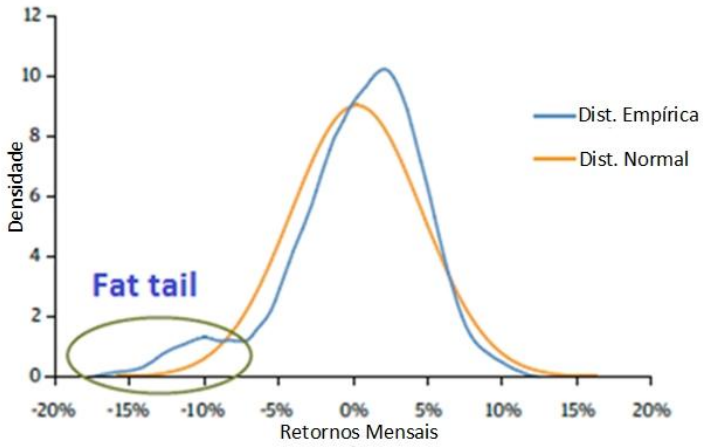


Figura 18 Distribuição dos retornos Empírica vs. Normal [**36**].

Outro fator que pode aumentar o risco é a correlação das variáveis destes modelos ter comportamento inesperado. Por exemplo, dois *stocks* num portefólio sem qualquer correlação, podem ficar correlacionados devido a uma situação inesperada, duplicando assim, o risco do portefólio [**36**].

Tendo em vista este problema, este documento irá estudar métodos de análise não linear, que ultimamente têm sido investigados. Alguns destes métodos adotam ferramentas usadas em IA. Estas ferramentas são muito úteis, pois podem modelar séries temporais de maneira empírica sem considerar a priori nada sobre as propriedades das suas distribuições ou correlações.

3.5. INTELIGÊNCIA ARTIFICIAL

Normalmente para desenhar um algoritmo capaz de resolver um determinado problema, o programador tem de saber quais os *inputs* que esse algoritmo admite e, tem de programar o processo. Ou seja, é necessário saber o que o algoritmo vai fazer a esses *inputs* para os converter nos *outputs* desejados.

O problema surge quando os *inputs* estão disponíveis, mas o processo não é conhecido ou então é muito complexo para o descrever num código. Por exemplo, na deteção de *spam*, onde os *emails spam* estão disponíveis, mas o processo de como separação dos *emails* (legítimos dos ilegítimos) é desconhecido.

A falta de conhecimento acerca do processo pode ser compensada através da quantidade disponível de *inputs*. Ou seja, através de uma grande quantidade de dados o algoritmo poderá encontrar uma aproximação ao processo real. O algoritmo pode não descobrir o processo exato, mas, se a aproximação for razoavelmente precisa, será possível fazer previsões úteis.

Pelo que foi descrito acima pode-se concluir que estes algoritmos podem ser muito úteis para fazer previsões dos mercados financeiros, pois os *inputs* são conhecidos, mas o processo é muito difícil de modelar e caraterizar. Além disso, existem grandes quantidades de dados históricos disponíveis ao público [**37**].

Existem várias técnicas ou algoritmos usados em IA, tais como:

- Métodos paramétricos: Onde a densidade de probabilidade de todas as classes é pré-definida. Alguns dos algoritmos são: Estimação *Bayesiana*, modelo oculto de *Markov*, modelos gráficos, etc.

- Métodos semi-paramétricos: Por, exemplo o algoritmo *K-means-clustering*, Este algoritmo recebe como informação prévia o número de *clusters* que existe e, depois, tenta agrupar os seus *inputs* no número de *clusters* que foi estipulado.

- Métodos não paramétricos: Onde a densidade de probabilidade das classes é definida empiricamente, como é o caso da estimação através do histograma, estimação *kernel* e o algoritmo *K nearest neighbour*.

- Outro tipo de algoritmos que não usam nenhum tipo de predefinição sobre as distribuições dos dados são as redes neuronais, que tentam imitar o cérebro para processar informação.

Existem vários métodos e algoritmos propostos no âmbito em IA, mas o problema a ser resolvido normalmente é de dois tipos: classificação ou/e regressão [**37**].

3.5.1. CLASSIFICAÇÃO

Num problema de classificação, o algoritmo tenta aglomerar os *inputs* que partilhem as mesmas características no espaço euclidiano. Depois, tenta inferir uma regra geral que permita fazer previsões.

Um exemplo comum no setor bancário é a atribuição de crédito, onde o banco necessita de fazer uma previsão do risco do cliente entrar em incumprimento.

Ao analisar milhares de clientes do banco, o algoritmo "aprende" a classificação apresentada na figura 19. A partir desta classificação o algoritmo analisa os futuros clientes com a seguinte função discriminante:

IF Salário do cliente > θ_1 ***AND*** Dinheiro disponível > θ_2

THEN

Cliente de baixo risco,

ELSE

Cliente de alto risco.

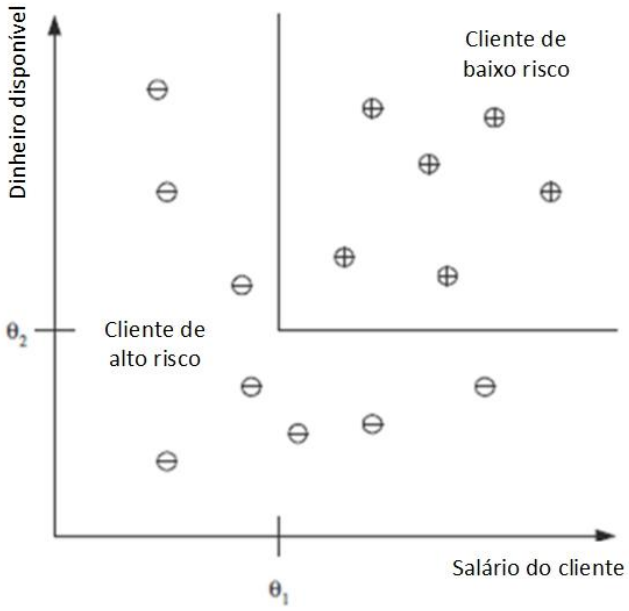


Figura 19 Classificação do risco de incumprimento dos clientes de um banco [**37**].

Este problema também pode ser analisado em termos de probabilidade.

Por exemplo, sabendo a probabilidade de $P(X = x)$ tem-se $P(Y = Baixo\ risco|X = x) = 0.8$. Neste exemplo imaginou-se que a probabilidade do cliente ser de baixo risco era de 80%. Assim, o banco podia quantificar o risco associado a esse cliente.

Outra propriedade a ter em conta é a capacidade de extrair informação do processo (*Knowledge Extraction*), Por exemplo, nas redes neuronais é muito difícil de extrair

informação nomeadamente da influência que os pesos estão a ter nos *outputs*. No caso do algoritmo do tipo árvores de decisão, é muito mais fácil extrair informação do processo.

3.5.2. REGRESSÃO

A regressão é normalmente usada quando os *outputs* são contínuos, como, por exemplo, o preço de um carro em função da quilometragem feita (figura 20):

$$y = wx + w_0 \tag{15}$$

Onde w e $w_0$são os coeficientes lineares da regressão (pesos), x é a quilometragem do carro e y é o preço do carro tendo em conta a quilometragem.

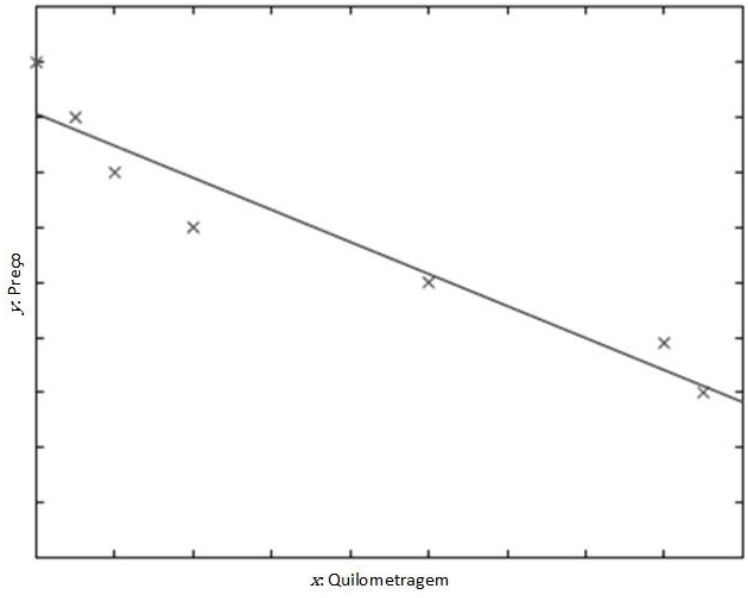


Figura 20 Regressão linear [**37**].

O objetivo do algoritmo consiste em otimizar os valores dos pesos *w*, de maneira que a função se adapte aos *inputs* usados para treinar o algoritmo (eq. 15). Caso os *inputs* tenham uma forma mais complexa pode-se usar polinómios de ordem superior como é o caso de um polinómio de segundo grau (eq. 16):

$$y = w_0 x^2 + w_1 x + w_0. \quad (16)$$

3.5.3. TIPOS DE APRENDIZAGEM

Basicamente existem três maneiras de treinar um algoritmo:

1. **Aprendizagem não supervisionada**

Neste tipo de aprendizagem o algoritmo não tem como analisar se os outputs que gera apresentam erros consideráveis.

O objetivo de um algoritmo não supervisionado é fazer estimações de densidades de probabilidade. Uma maneira de fazer isto é com recurso a técnicas de *clustering*. Como já foi anteriormente referido o algoritmo tenta usar os *inputs* para gerar grupos. Um exemplo em que estes algoritmos podem ser usados é na segmentação de clientes, onde uma empresa que tenha uma enorme quantidade de informação sobre clientes tenta achar grupos de clientes que partilhem as mesmas características [**37**].

2. **Aprendizagem reforçada**

Na aprendizagem reforçada o algoritmo executa uma série de ações dentro de um determinado ambiente e é posteriormente recompensado se essas ações, como um todo, atingirem um determinado objetivo, ou é penalizado no caso contrário. Não se pode definir uma ação em particular como "boa" ou "má", é o conjunto de ações que determina se o algoritmo atinge o objetivo. Um exemplo onde este tipo de aprendizagem pode ser usado, é no desenvolvimento de um algoritmo para jogar xadrez. Neste caso, o algoritmo não pode ser supervisionado, pois seria muito dispendioso ter um jogador humano a dar *feedback* sobre muitos jogos.

Num jogo de xadrez uma jogada em si não é "boa" ou "má". Assim o algoritmo tenta executar uma série de jogadas que o façam ganhar o jogo e só no fim é recompensado (caso ganhe) ou penalizado (caso perca) [**37**].

3. Aprendizagem supervisionada

Neste tipo de aprendizagem o algoritmo é treinado com um conjunto de vetores de *inputs* e *outputs*. Ou seja, o algoritmo gera os seus próprios *outputs* e, de seguida, estes são comparados com os *outputs* de treino (*targets*), que são previamente fornecidos. Na fase final o algoritmo tenta corrigir o erro resultante da comparação entre os *outputs* e os *targets*.

Por exemplo, no caso de conseguir classificar um carro (familiar ou outro tipo) de acordo com o seu preço e cilindrada do motor. No gráfico da figura 21 podemos ver:

Os *inputs* $Z = [x_1; x_2]$ (preço e potência),

Os *targets* $r = \begin{cases} 1\ se\ o\ carro\ for\ familiar\ (+) \\ 0\ se\ o\ carro\ não\ for\ familiar\ (-) \end{cases}$'

onde $X = \{Z^i; r^i\}_{i=1}^N$, onde o *i* é um índice e *N* é o tamanho dos vetores de treino.

Enquanto está a treinar o algoritmo faz o seguinte:

1. Recebe os *inputs* X como vetores de treino;
2. De acordo com os *inputs* de treino o algoritmo gera uma hipótese, ou seja, gera um retângulo onde os *inputs* que entram dentro dele (representados por um "+" na figura 21) são considerados como um carro familiar;
3. O algoritmo compara se os outputs que gerou são parecidos com os *r*;
4. O algoritmo tenta minimizar o erro empírico (eq. 17):

$$E(h|X) = \sum_{i=1}^{N} 1\ se\ (h(Z^i) \neq r^i) + 0\ se\ \left(h\left(Z^i\right) = r^i\right), \tag{17}$$

onde:

$$h(Z) = \begin{cases} 1\ se\ a\ hipotese\ classifica\ Z\ \ como + \\ 0\ se\ a\ hipotese\ classifica\ Z\ \ como - \end{cases}. \tag{18}$$

O objetivo do algoritmo é achar a hipótese que mais se assemelhe ao retângulo C (figura 21), que é desconhecido.

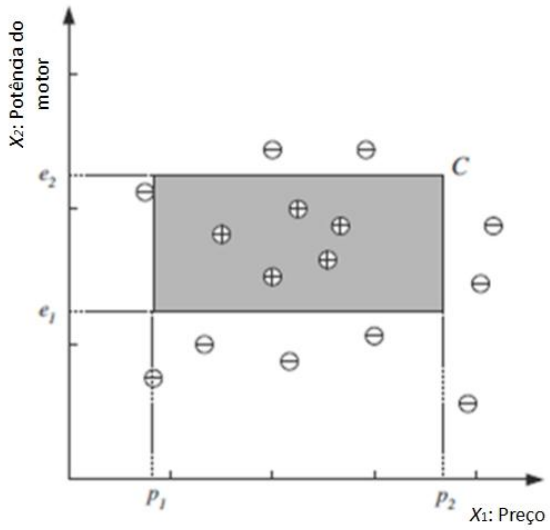


Figura 21 Hipótese que melhor classifica os *inputs* (C).

Se o algoritmo conseguir achar uma hipótese semelhante a C, então pode-se dizer que este consegue generalizar, ou seja que conseguiu encontrar um processo e que os seus *outputs* têm um grau de confiança elevado.

No caso de uma regressão o objetivo não é encontrar uma classe, mas sim um número real $r^i \in \Re$ onde $r^i = f(Z^i) + \epsilon$, o objetivo é encontrar a função $f(Z^i)$ [**37**].

O símbolo ϵ representa o ruído e está associado a variáveis desconhecidas.

O erro do algoritmo pode ser calculado da seguinte forma:

$$E(g|X) = \frac{1}{N}\sum_{i=1}^{N}[r^i - g(Z^i)]^2, \tag{19}$$

onde $g(Z)$ são os outputs gerados pelo algoritmo e r^i são os *targets*.

Para minimizar a função de erro tem-se que encontrar a função:

$$g(Z) = \sum_{j=1}^{d} w_j Z_j + w_0. \tag{20}$$

No caso da eq (19) e eq. (20) fica:

$$g(Z) = w_1 Z + w_0, \tag{21}$$

$$E(w_1, w_0 | X) = \frac{1}{N} \sum_{i=1}^{N} [r^i - (w_1 Z^i + w_0)]^2. \tag{22}$$

Para achar o mínimo calculam-se as derivadas parciais em ordem a w_0, w_1 e iguala-se a zero.

$$w_1 = \frac{\sum_i Z^i r^i - \overline{Zr} N}{\sum_i (Z^i)^2 - N(\bar{Z})^2} \text{ e } w_0 = \bar{r} - w_1 \bar{Z}, \tag{23}$$

onde :

$$\bar{Z} = \frac{\sum_i Z^i}{N}, \tag{24}$$

$$\bar{r} = \frac{\sum_i r^i}{N}. \tag{25}$$

3.5.4. *OVERFITTING*

Quando um algoritmo tenta generalizar através de funções muito complexas, o risco de *overfitting* é elevado. Isto é, o algoritmo começa a memorizar os outputs de treino (figura 22), ou, caso haja ruído, o algoritmo tenta generalizar o ruído. Por exemplo, é como se um estudante em vez de estudar os exercícios (de maneira a conseguir generalizar e conseguir resolver o exame), memorizasse as soluções. Normalmente, ao aumentar o espaço de treino *N* o risco de *overfitting* diminui [**37**] [**38**].

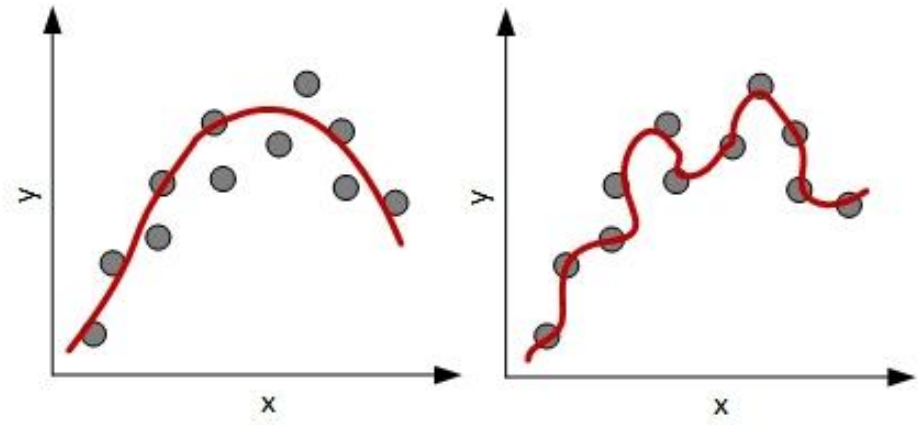


Figura 22 Comparação entre duas regressões uma sem *overfitting* (esquerda) e outra com *overfitting* (direita) [**38**].

Para saber quando um algoritmo deve parar o treino e, ao mesmo tempo minimizar, o erro, é necessário dividir os *inputs* de teste em três partes: treino, validação e teste.

O treino desenvolve-se da seguinte maneira:

1. O algoritmo usa os *inputs* de treino e os *targets* para tentar minimizar o erro dos *outputs* gerados;

2. O algoritmo pára de usar os *inputs* de treino e começa a usar os *inputs* de validação.

3. O algoritmo pára de usar os *inputs* de validação e começa a usar os *inputs* de teste.

   Após a totalidade do vetor dos *inputs* ter entrado todo no algoritmo diz-se que foi completado um *epoch*;

4. Calcula-se o erro entre os *outputs* e os *targets* para o treino, validação e teste;

5. Se o erro de validação descer em relação ao erro de treino (figura 23), então volta-se a calcular mais um *epoch*.

6. Se o erro de validação começar a subir em relação ao erro de treino, o algoritmo calcula mais um determinado número de *epochs*, para se certificar que não ficou preso num mínimo local, e depois pára o treino.

O vetor de teste é uma confirmação do vetor de validação. No caso dos erros entre a validação e o teste forem muito diferentes, isso pode significar que o vetor de *input* não foi bem dividido.

O uso dos *inputs* de validação e treino é conhecido como *cross-validation*. Normalmente a quantidade de treino/validação /teste que se usa é na proporção 50/25/25 se o *N* for elevado, caso contrário adota-se 60/20/20 [**38**].

Na figura 23 pode-se ver o momento no qual se deve parar o treino do algoritmo.

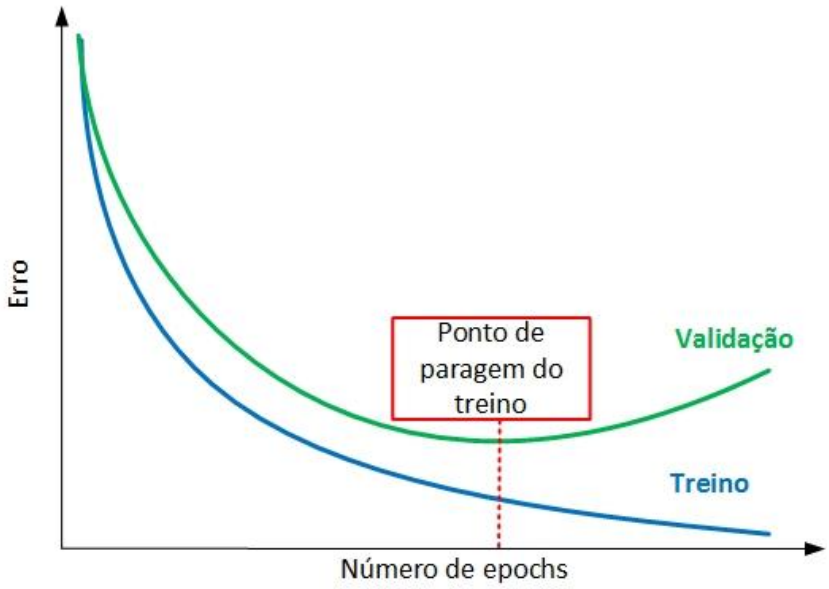


Figura 23 *Cross-validation* e o momento em que o algoritmo deve parar do treino [**38**]

3.5.5. *UNDERFITTING*

O *underfitting* é comum acontecer quando o espaço de treino *N* é muito reduzido e o algoritmo não tem dados suficientes para generalizar o problema. Usando outra vez a analogia do estudante, seria como se o estudante não estudasse o suficiente para o exame.

***Bias* e variância**.

Se X for uma amostra de uma população especificada por um parâmetro θ e supondo que $\hat{\theta}$ é um estimador de θ, então para verificar a qualidade do estimador deve-se calcular o erro $(\hat{\theta} - \theta)^2$. Todavia, mas como $\hat{\theta}$ depende da amostra X deve-se achar o valor médio do erro (MSE):

$$MSE(\theta; \hat{\theta}) = E_\theta[\hat{\theta} - \theta]^2, \tag{25}$$

onde o *bias* pode ser calculado da seguinte maneira:

$$b_\theta(\hat{\theta}) = E_\theta[\hat{\theta}] - \theta. \tag{26}$$

À medida que a amostra X aumenta, o valor esperado vai aproximando-se ao da função de distribuição da população, de maneira que $E_\theta[\hat{\theta}] = \theta$. Neste caso o estimador não tem *bias*.

A variância do estimador ou seja o valor máximo do erro do parâmetro é calculada da seguinte forma:

$$VAR(\theta) = E[(\hat{\theta} - E_\theta[\hat{\theta}])^2] \tag{27}$$

Caso a amostra X aumente a variância diminui.

Pode-se então verificar que o valor médio do erro é:

$$VAR(\theta) = E[(\hat{\theta} - E_\theta[\hat{\theta}])^2], \tag{27}$$

$$MSE(\theta;\hat{\theta}) = E_\theta[\hat{\theta} - \theta]^2 = E[(\hat{\theta} - E_\theta[\hat{\theta}])^2] + (E_\theta[\hat{\theta}] - \theta)^2 \quad (28)$$
$$= VAR(\theta) + (b_\theta(\hat{\theta}))^2.$$

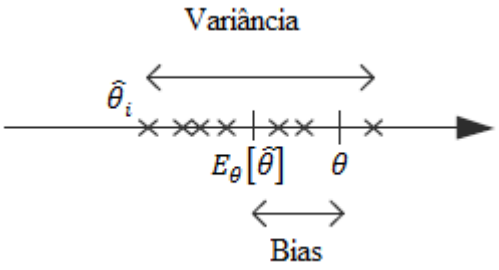


Figura 24 Bias e variância do parâmetro θ.

Na figura 24 é possível ver que θ é o parâmetro a ser estimado, os pontos 'x' são várias estimações retiradas da amostra X e o *bias* é a distância entre o valor médio dos estimadores e o parâmetro (que é desconhecido). A variância é todo o espaço onde os estimadores se podem dispersar [**37**].

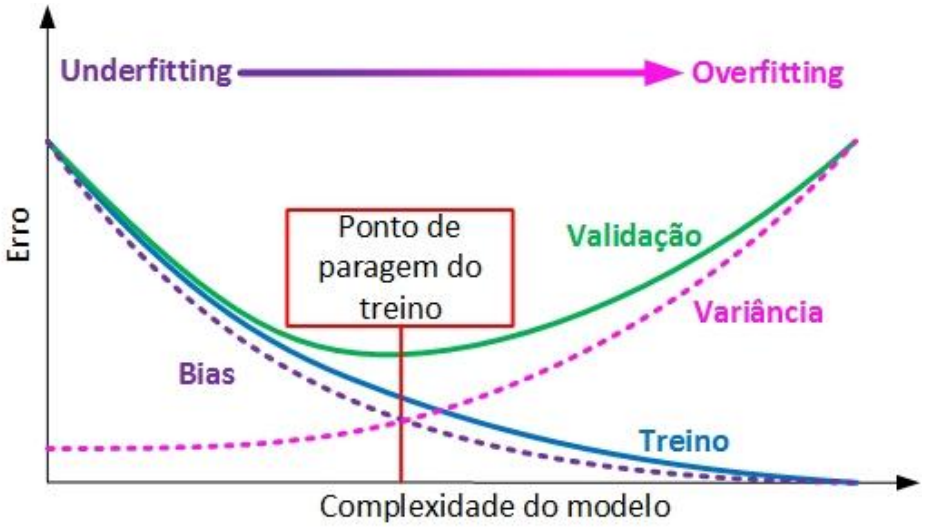


Figura 25 Relação entre o *bias*, variância e a complexidade do modelo em relação ao problema a resolver [**39**].

Na figura 25 pode ver-se a relação entre os erros, a variância, o *bias* e a complexidade do modelo que o algoritmo está a tentar implementar. Se o modelo for muito complexo (por exemplo se usar uma função polinomial com uma ordem elevada e tentar adaptar a uma linha), então este vai acabar por memorizar ou aprender o ruído e as suas estimações vão-se afastar muito do parâmetro real θ. Se o oposto acontecer e a complexidade for muito baixa (como por exemplo tentar adaptar uma função linear a uma polinomial), o algoritmo não vai conseguir aprender a função e o valor medio dos estimadores vai ficar muito distante de θ.

Na prática não é possível calcular o *bias* e a variância, pelo que a complexidade do sistema é ajustada através do treino de validação.

3.5.6. REDUÇÃO DIMENSIONAL.

Para que um algoritmo consiga fazer uma classificação ou uma regressão com um erro de validação baixo, este deve ter a máxima informação sobre o processo em questão. O problema é que ao inserir uma grande quantidade de variáveis de *input* (dimensão) no sistema, a complexidade do mesmo aumenta. Isso não é uma boa prática, pois para além dos problemas referidos anteriormente existe também o problema do aumento do tempo de computação. Para isso existem formas de eliminar os *inputs* que não contribuem significativamente para a redução do erro de validação, reduzindo assim a dimensão do problema.

As formas de redução dimensional são:

- **Feature Selection**

Este tipo de redução pode ser dividida em duas partes nomeadamente:

Forward Selection - Onde se começa o algoritmo com zero dimensões e se vai adicionando dimensão a dimensão até o erro de validação parar de descer;

Backward Selection – Aqui o algoritmo começa com todas as variáveis e vai retirando uma a uma até que o erro de validação páre de descer.

O problema destes processos de redução é que podem falhar no caso em que as variáveis em conjunto diminuem significativamente o erro, mas em que isoladamente não têm grande impacto [**37**].

- ***Feature Extraction***

Nestes tipos de algoritmos o objetivo é encontrar combinações de variáveis que influenciem o erro. Para isso temos métodos de projeção linear tais como:

1. *Principal components analysis* (PCA) – O PCA é um método onde se usa uma transformação ortogonal para converter um grupo de variáveis que estão possivelmente correlacionadas, num grupo de variáveis linearmente não correlacionadas. Este último grupo chama-se componente principal.

Basicamente este algoritmo tenta eliminar todos os componentes que tenham uma variância reduzida, pois provavelmente não vão ter um impacto no erro de validação.

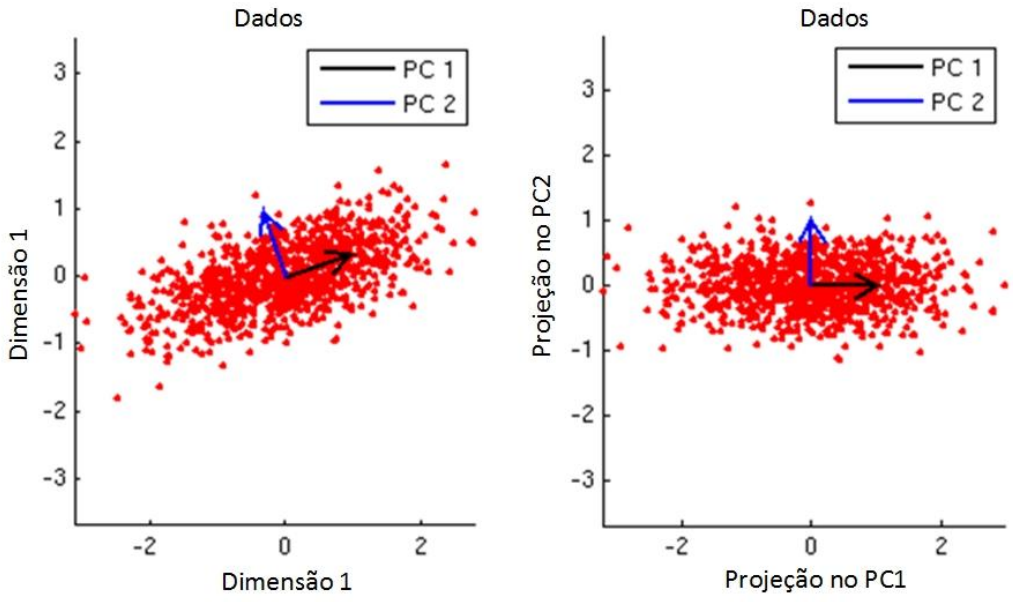


Figura 26 Projeção dos componentes principais [**40**].

Na figura 26 foi feita uma rotação dos *inputs* de maneira a verificar quais os componentes principais com menor variância. Neste caso o PC2 é o que tem menor variância, pelo que a dimensão dois será candidata a ser eliminada, reduzindo, a dimensão de dois para um.

A desvantagem deste algoritmo é que existe a possibilidade de uma dimensão com variância baixa ter uma influência grande no erro de validação [**37**].

2. *Factor Analysis* (FA) – O FA baseia-se no fato que um conjunto de variáveis observáveis e correlacionadas poderem ser explicados por um grupo de fatores não observáveis, tal que este grupo de fatores é menor que o grupo de variáveis observáveis. Se isto acontecer então existe uma diminuição da dimensão.

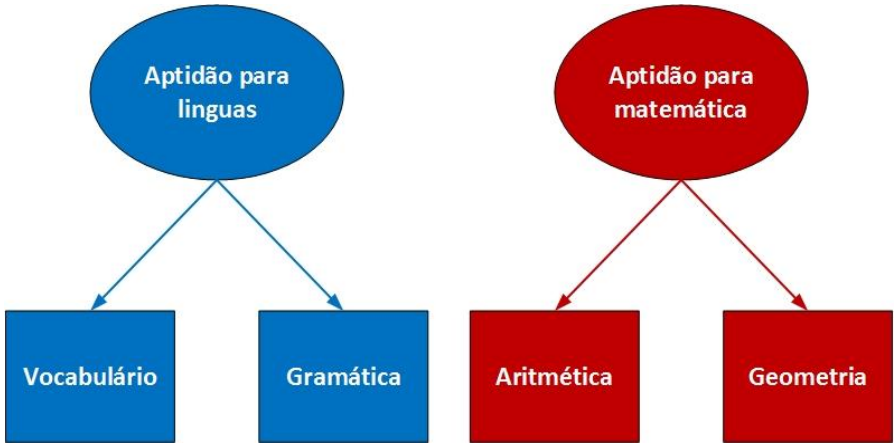


Figura 27 Redução de quatro dimensões para duas através do FA [**41**].

Na figura 27 pode ver-se que as variáveis vocabulário e gramática estão correlacionados por um único fator, que é a aptidão de um individuo para línguas. Por seu lado, a aritmética e a geometria estão correlacionados com a habilidade de um individuo para matemática. Como quatro variáveis podem ser explicadas através de somente duas pode concluir-se que o número de dimensões diminuiu.

A vantagem que o FA tem em relação ao PCA é que é possível fazer uma melhor extração de conhecimento do processo [**37**].

3. *Multidimensional Scaling* (MDS) - Esta técnica consiste em achar a distância que os *outputs* têm uns dos outros. Depois o objetivo é colocar os outputs em dimensões menores de maneira que a distância euclidiana dos novos *outputs* se aproxime à distância original [**37**].

4. ISOMAP - Usa-se esta técnica quando o subespaço onde os *outputs* se encontram não é linear. Neste caso usa-se o MDS com a diferença que em vez de se usar a distância euclidiana usa-se a distância geodésica (figura de baixo) [**37**].

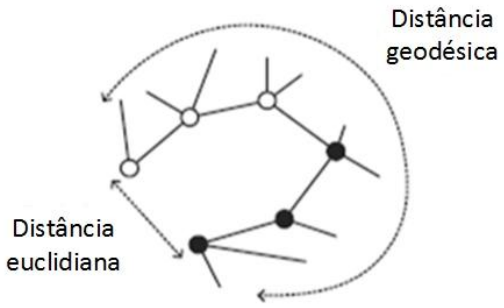


Figura 28 Distância geodésica [**37**].

5. *Locally linear embedding* (LLE) - Esta técnica consiste em calcular cada ponto de um plano não linear através de uma soma linear ponderada dos pontos vizinhos (linhas a tracejado na figura 21). O passo seguinte é tentar colocar esses pontos num novo plano (com menor dimensão), mas mantendo fixos os pesos *W* da soma linear (figura de baixo) [**37**].

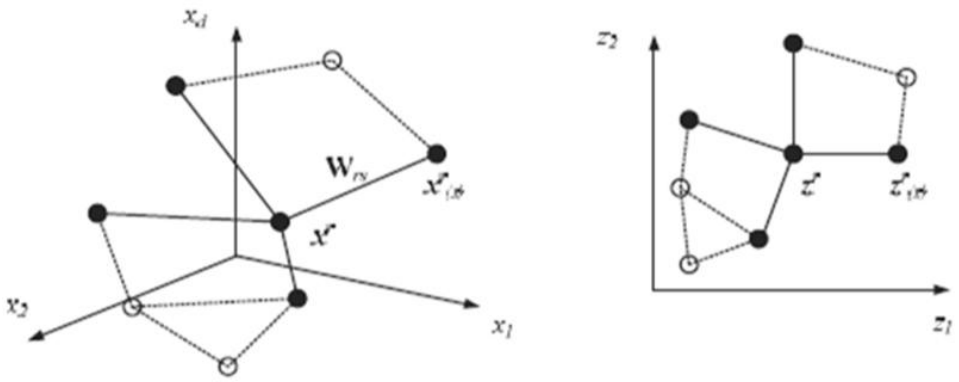


Figura 29 Projeção LLE de um plano de três dimensões em duas dimensões [**37**].

3.5.7. *CLUSTERING*

Quando são usados os métodos paramétricos a densidade de probabilidade de todas as dimensões é especificada previamente como, por exemplo, uma densidade Gaussiana. Isto simplifica muito o processo de aprendizagem do algoritmo, pois a sua complexidade baixa. O problema deste método é que ao reduzir a complexidade do processo, corre-se o risco de induzir uma quantidade excessiva de *bias*, Por outro lado, ao termos um processo governado por uma densidade de probabilidade, assumimos que as instâncias de uma classe formam um único grupo o que pode não ser verdade, pois, em muitas aplicações, existem vários grupos na mesma classe.

No caso das técnicas de *clustering* a densidade de probabilidade de um grupo pode ser definida por um conjunto das densidades de probabilidade dos *inputs*, o que dá origem à designação de métodos semi paramétricos.

Um algoritmo usado para ordenar *inputs* em grupos é o *K means cluster*. De seguida apresenta-se o exemplo de um algoritmo com o objetivo de dividir os objetos em três grupos (verde, azul e vermelho). Como o número de grupos é três o *K=3* (este valor tem de ser previamente declarado), de seguida são escolhidos aleatoriamente três objetos que vão ser as centróides (figura 30).

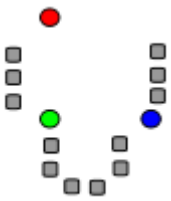


Figura 30 Centróides iniciais [**42**].

De seguida é calculado a distância de cada elemento a cada centróide. Os objetos que ficarem mais perto do centróide verde do que do azul ou do vermelho ficam no grupo verde. O mesmo acontece para cada um dos outros grupos (figura 31).

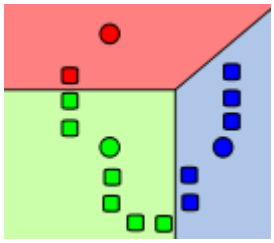


Figura 31 Grupos relativos a cada centróide [**43**].

Uma vez definidos os grupos, o algoritmo faz uma média aritmética das coordenadas de cada elemento do grupo. A coordenada resultante passa a ser o novo centróide (figura 31).

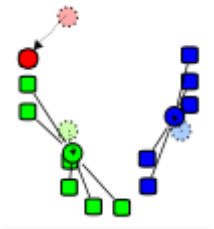


Figura 32 Novas centróides [**44**].

Uma vez definido os novos centróides voltam-se a calcular as distâncias e a dividir os elementos em novos grupos, o processo repete-se até nenhum elemento conseguir mudar de grupo (figura 32) [**37**].

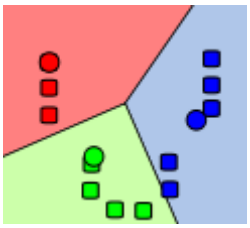


Figura 33 Grupos finais [**45**].

Existem também algoritmos onde o utilizador não precisa de especificar o número de *clusters*. O DBSCAN (*Density-based spatial clustering of applications with noise*) consegue lidar melhor com o ruído e consegue achar densidades com formas variadas (figura 44), algo que o *K mean clustering* e o EM (*Expectation Maximization Algorithm*) não conseguem.

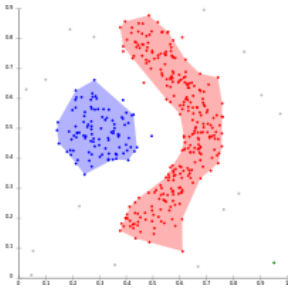


Figura 34 Algoritmo DBSCAN [**46**].

Outro algoritmo é o OPTICS (*Ordering points to identify the clustering structure*) (figura 36), que corrige um dos pontos fracos do DBSCAN, nomeadamente conseguindo agrupar *clusters* com uma densidade variável (figura 35 a cinzento). No entanto, se as distribuições forem Gaussianas, e o algoritmo e os clusters tiverem uma densidade constante, o EM tem quase sempre um desempenho melhor (figura 37).

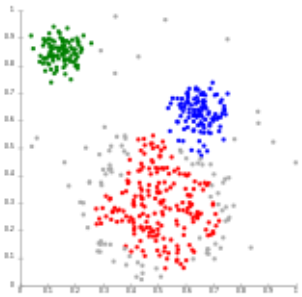


Figura 35 Algoritmo DBSCAN e *clusters* com densidade variada [**77**].

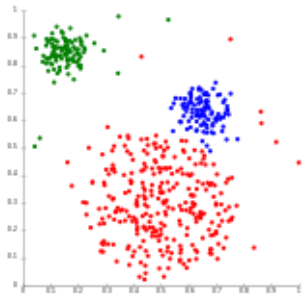


Figura 36 Algoritmo OPTICS [**78**].

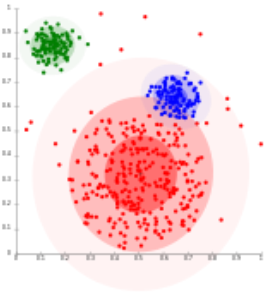


Figura 37 Algoritmo EM [**47**].

3.5.8. MÉTODOS NÃO PARAMÉTRICOS

No caso da distribuição de probabilidade ser desconhecida, pode-se calcular esta distribuição empiricamente. Uma maneira de fazer isso consiste em usar uma estimação através de um histograma (figura 38).

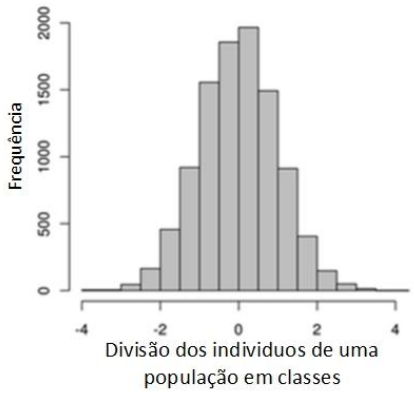


Figura 38 Histograma [**48**].

Outra maneira é usar o estimador *Kernel* (neste caso o estimador é contínuo). O estimador usa a soma de várias funções (funções *kernel* a vermelho na figura 39) para se tentar moldar aos *inputs*.

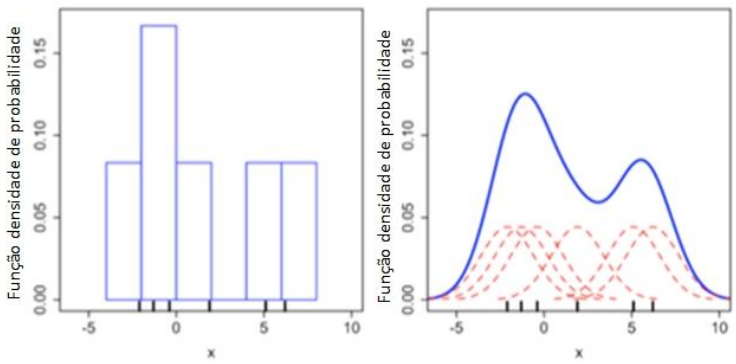


Figura 39 Comparação entre um histograma (esquerda) e um estimador *kernel* (direita) [**49**].

Existem várias funções *kernel* a mais usada é a função Gaussiana:

$$K(u) = \frac{1}{\sqrt{2\pi}} . e^{[-\frac{u^2}{2}]}. \tag{29}$$

A função densidade (KDE) calcula-se da seguinte maneira:

$$f(x) = \frac{1}{h.N} \sum_{i=1}^{N} K(\frac{x - x_i}{h}), \tag{30}$$

onde h é uma constante chamada largura de banda e serve para suavizar a curva da distribuição [**37**].

Na figura 40 pode ver-se uma função Gaussiana normal a cinzento, uma KDE com *h=0.05* a vermelho, *h=0.337* a preto e *h=2* a verde.

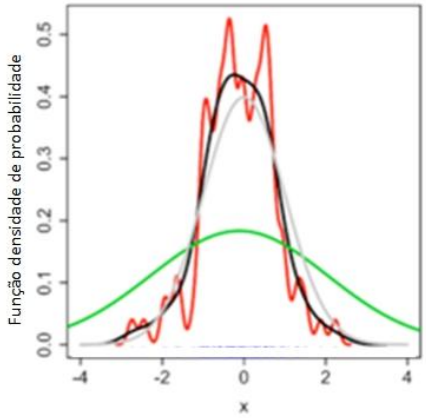


Figura 40 Função normal e KDEs com *h=0.05* (vermelho), *h=0.337* (preto), *h=2* (verde) [**50**].

3.5.9. *K NEAREST NEIGHBOURS (KNN)*

O algoritmo KNN calcula a distância do *input* inserido até aos *inputs* de treino (vizinhos) que estão mais perto. Se houver vários vizinhos idênticos à beira do *input* isso significa que o *input* deve pertencer ao mesmo tipo de classe que esses vizinhos.

Considere-se o exemplo (figura 41) se os *inputs* de treino forem de dois tipos (quadrados, triângulos) e o *input* for o ponto verde. Se *K* for 3 (circulo mais pequeno), então o KNN calcula a distância dos três vizinhos mais próximos. Neste caso o *input* passava a ser um triângulo, pois existem dois triângulos e um quadrado. Caso *K* seja 5 (circulo maior) o *input* passa a ser um quadrado pois dos cinco vizinhos mais perto três são quadrados.

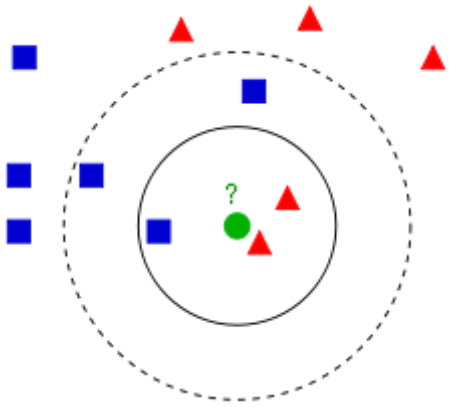


Figura 41 Algoritmo KNN (a verde o *input* a ser avaliado) [**51**].

No exemplo anterior todos os vizinhos tinham o mesmo peso. No entanto, é comum atribuir o peso *1/d* aos vizinhos onde *d* é a distância. Assim, os vizinhos que se encontram mais longe não têm tanta influência. A escolha do parâmetro *K* tem influência na complexidade do algoritmo. Se este for muito baixo, então o algoritmo adapta-se demasiado aos *inputs* de treino e, como os exemplos vizinhos a ser calculados são poucos, vai ser introduzido muito ruído (variância). Caso *K* seja muito alto, o algoritmo perde mais flexibilidade e o *bias* começa a crescer [**37**] [**38**].

3.5.10. REDES NEURONAIS

Este algoritmo tem como base o cérebro humano para conseguir aprender um determinado processo. O cérebro é um órgão extremamente poderoso e que consegue generalizar processos muito complexos muitas vezes com ruído. Apesar do cérebro em si ser muito complexo, os blocos fundamentais que o constituem (neurónios) são muito simples e as suas propriedades podem ser simuladas através de algoritmos simples.

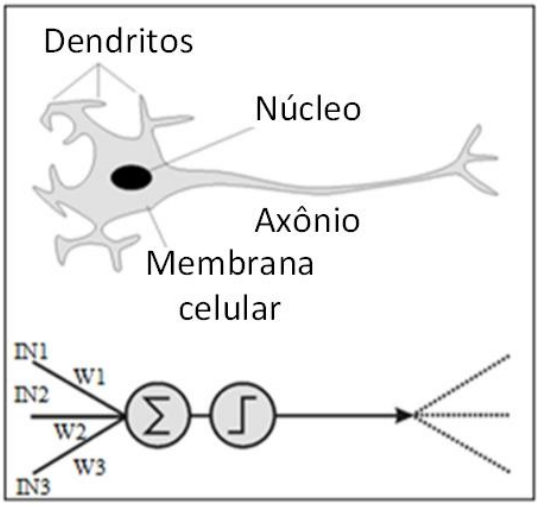


Figura 42 Comparação de um neurónio biológico com um neurónio digital [**52**].

Na figura 42 pode-se ver uma comparação entre um neurónio biológico e um neurónio digital. No neurónio biológico os sinais *input* são introduzidos nas dendrites. Estes *inputs* por sua vez vêm dos *outputs* (*axons*) de outros neurónios formando, assim, uma rede neuronal.

Com o decorrer da aprendizagem os neurónios vão produzir reações químicas. Estas reações vão fazer com que o sinal elétrico se propague com mais ou menos intensidade. No caso do neurónio digital, as reações químicas que abrem, ou fecham, o canal por onde o sinal é transmitido chamam-se pesos *W*.

Num neurónio digital os pesos são multiplicados pelos respetivos *inputs*, desseguida são somados e transformados através de uma função não linear. As funções mais usadas são a função sigmoide e a tangente hiperbólica.

Estas funções são responsáveis por fazer um neurónio ser ativado ou desativado. Por exemplo, no caso de uma função binária (figura 43) o neurónio é ativado se o resultado do somatório *a* ultrapassar um determinado limiar. Caso não ultrapasse o limiar o output *y* é zero.

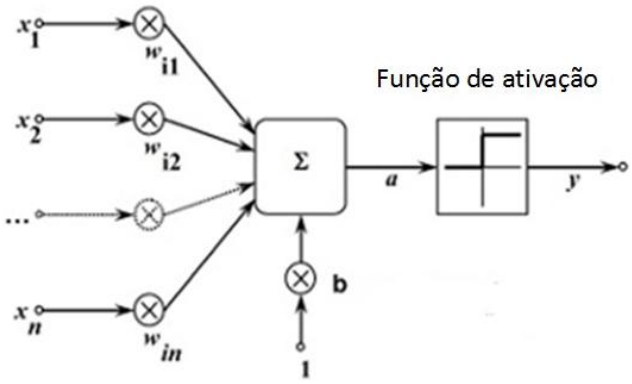


Figura 43 Estrutura de um neurónio digital [**38**].

Normalmente não se usam funções descontínuas, pois para treinar uma rede neuronal, é preciso usar um algoritmo que vai calcular os gradientes dos erros em cada neurónio. Logo, é necessário que as funções de ativação tenham derivada, o que não acontece no caso da função degrau unitário. No entanto, é possível obter bons resultados com uma sigmoide, ou uma tangente hiperbólica, que têm derivada.

Na figura 43 pode ver-se que existe um peso especial *b*: o *input* para o peso *b* chama-se *bias* e é comum ter o valor unitário. Este *input* é usado pois consegue regular o limiar da função de ativação.

Por exemplo, caso dois neurónios sejam controlados pelo mesmo *input*, pode não ser possível conseguir que, o resultados dos dois neurónios difiram um do outro. Este objetivo não pode ser atingido se o *input* for nulo, pois sem o *bias* a função de ativação nunca irá ativar, mesmo que o peso seja muito elevado o somatório irá ser sempre nulo. No entanto, ao ser inserido o *bias*, o somatório *a* já não é nulo e o primeiro neurónio pode ser ativado [**38**].

Rede Feedforward

Anteriormente foi descrito o funcionamento de um neurónio, Todavia, no caso de ser preciso representar uma função mais complexa é necessário combinar vários neurónios, obtendo-se assim uma rede neuronal.

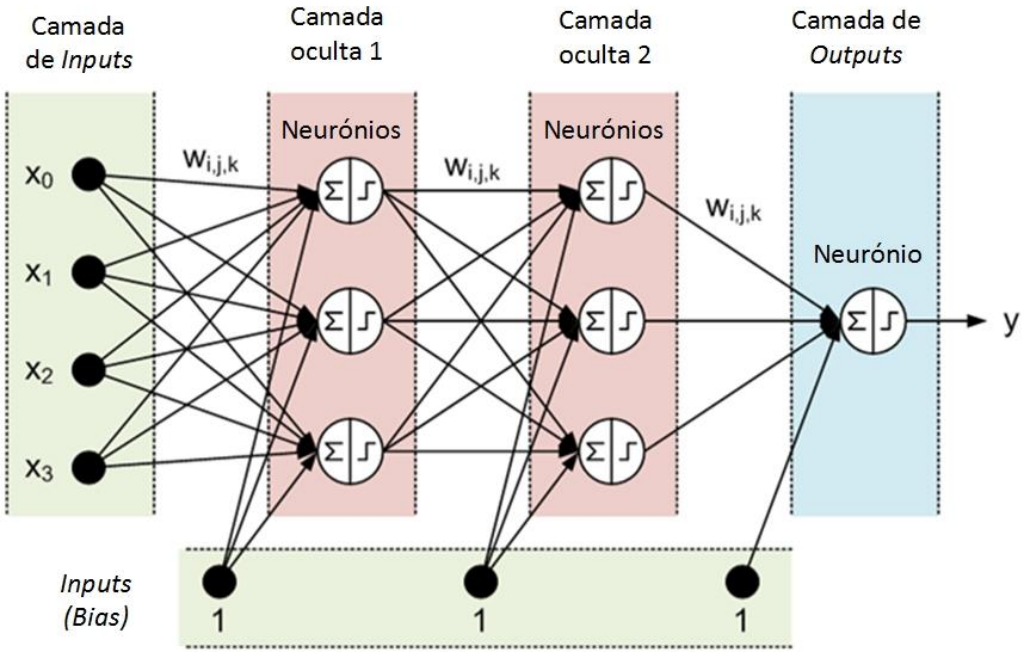


Figura 44 Rede neuronal *feed forward* com duas camadas ocultas [**53**].

Na figura 44 pode-se ver uma rede neuronal com 4 *inputs*, 3 *bias*, duas camadas ocultas com 3 neurónios cada e um *output*. Se considerarmos *n* como sendo o número de *inputs*,

k^1 o número de neurónios na primeira camada oculta (*hidden layer*), k^2 o número de neurónios na segunda camada oculta e M como sendo o número de *outputs*, então pode-se calcular o número de pesos que é preciso treinar:

$$Nº\ de\ W = (n+1) \times k^1 + (k^1+1) \times k^2 + (k^2+1) \times m. \quad (31)$$

Tendo em conta a rede de cima será preciso treinar:

$$(4+1) \times 3 + (3+1) \times 3 + (3+1) \times 1 = 31\ pesos.$$

Na maior parte dos casos uma rede *feedforward* é composta apenas por uma camada oculta, uma camada de *inputs* e outra de *outputs*, pois segundo o teorema de aproximação universal, uma camada oculta é o suficiente para aproximar qualquer função. No entanto, para obter o mesmo resultado será preciso uma camada oculta maior. O tamanho da camada oculta tem de ser determinado empiricamente, pois não existe nenhum teorema para achar o número ideal de neurónios na camada.

Como regra geral o tamanho de cada vetor de *input* deve ser 10 vezes superior ao número de pesos da rede. No caso de uma rede com 2000 pesos o vetor de *input* deve ter um tamanho mínimo de 20000 unidades [**38**].

Treino de uma rede neuronal

A aprendizagem de uma rede neuronal está associada à definição dos pesos pois, são estes que vão fazer ativar, ou não, um conjunto de neurónios, de forma que o *output* tenha o mínimo de erro possível. Uma maneira de calcular os pesos poderia ser através de uma estratégia de força bruta, ou seja, tentar todas as combinações possíveis de pesos. O problema é que esse método é quase impossível pois em termos práticos o número de combinações, é muito elevado.

Por exemplo, no caso anterior com 31 pesos, suponha-se que um peso pode ter valores compreendidos entre e 0 e 1000 com incrementos de 0.01. Assim cada peso tem 100000 valores possíveis. Se combinarmos os 31 pesos resultam 100000^{31} combinações, o que levaria muitos anos a calcular mesmo com um supercomputador.

Para ultrapassar o problema anterior foram criados vários algoritmos de otimização, como os algoritmos evolucionários, *backpropagation*, etc. Os algoritmos evolucionários têm a vantagem de ser mais difícil a solução ficar retida num mínimo local. Apesar da vantagem referida, estes algoritmos demoram muito tempo a convergir e por isso costumam ser usados apenas para achar quais as melhores topologias de uma rede neuronal.

O algoritmo *backpropagation through gradient descent* (BP) é o mais usado na literatura, pois é simples de implementar e converge muito mais rápidamente que os outros algoritmos. Apesar do risco de a sua solução convergir para num mínimo local ser maior, este algoritmo continua a ser o mais usado pois a rapidez de convergência compensa as suas desvantagens minimizando, assim, os custos de computação.

De seguida será descrito como o BP é aplicado a uma rede neuronal.

Backpropagation through gradient descent

O algoritmo BP basicamente vai propagar o erro desde o output até ao início da rede, este erro é calculado através das derivadas das funções de ativação. O resultado do BP não são os valores dos pesos totais mas sim as variações que se devem aplicar aos pesos.

A fórmula para calcular os pesos é a seguinte:

$$\nabla E = \frac{\partial E}{W_{1,1}} + \cdots + \frac{\partial E}{W_{i,j}}, \tag{32}$$

$$\Delta W_{i,j} = -\alpha . \frac{\partial E}{W_{i,j}}, \tag{33}$$

$$W_{i,j} = W_{i,j} + \Delta W_{i,j}, \tag{34}$$

onde o ∇E é o gradiente do erro a minimizar, $\Delta W_{i,j}$ é a variação dos pesos e α é o *learning rate*.

Quando é iniciado o treino da rede os *inputs* são todos calculados. Uma passagem de todos os *inputs* pela rede é chamado *epoch*. No fim de cada *epoch* são calculadas as derivadas parciais e a variação dos pesos é achada (eq. 33). Caso esta variação seja aplicada aos pesos (eq. 34) no final de cada *epoch*, diz-se que algoritmo está a ter uma atualização *online* (*online learning*). Caso as variações sejam aplicadas ao fim de vários *epochs*, diz-se que a atualização dos pesos é feita *offline*, também conhecida como *batch learning*.

Como se pode ver na figura 45 a variação dos pesos segue o gradiente negativo. Se este gradiente se aproxima de zero, então é altura de parar o algoritmo.

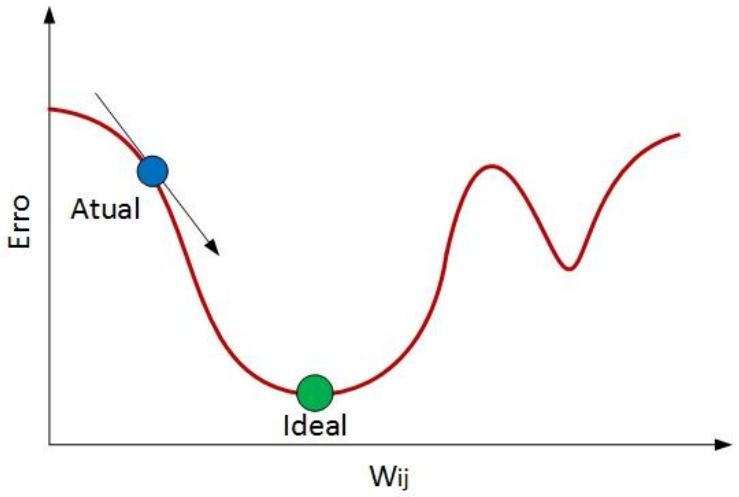


Figura 45 Gráfico Erro vs. Pesos [**54**].

A cadência com que o gradiente se desloca é caracterizada pela constante α que é chamada de *learning rate* e os seus valores situam-se na gama $0.1 < \alpha < 0.4$. Se o valor de α for pequeno, então o algoritmo converge mais devagar. Todavia consegue localizar melhor os mínimos pois a sua precisão aumenta (figura 46, esquerda). Se α tiver um valor elevado, então o algoritmo tem mais dificuldade em convergir para um mínimo pois as variações dos pesos são muito altas. No entanto, ao tornar o algoritmo mais instável este fica menos vulnerável a uma convergência para um mínimo local (figura 46, direita).

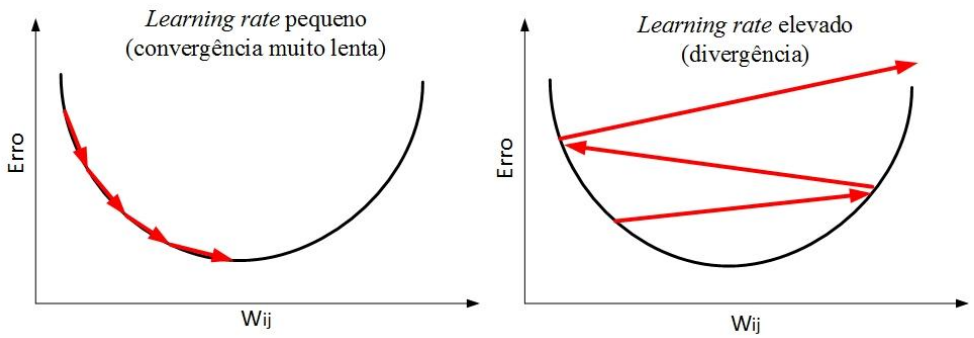


Figura 46 *Learning rate* pequeno (esquerda) e *Learning rate* elevado (direita) [**55**]

Considere-se a seguinte rede neuronal de uma camada oculta com $(n+1)$ *inputs*, $(k+1)$ neurónios na camada oculta e m *outputs*.

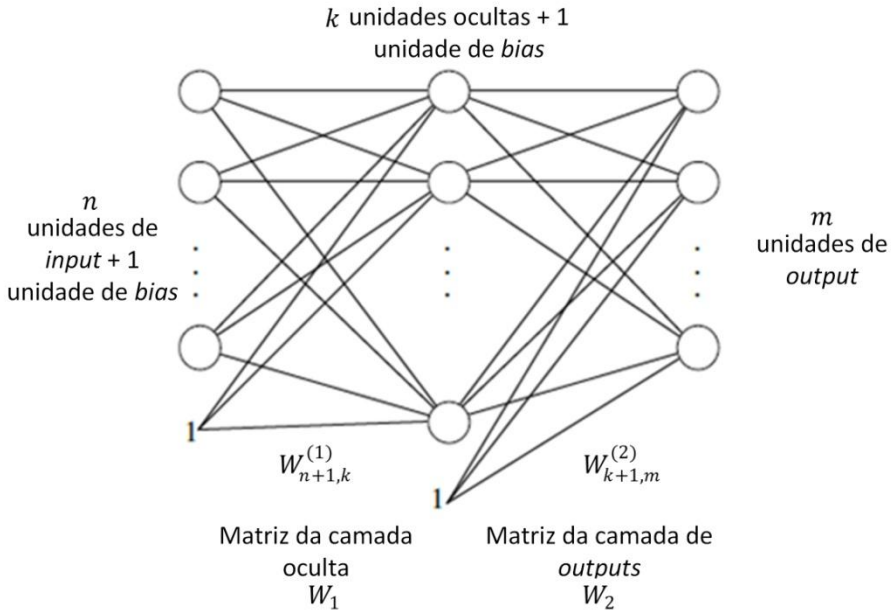


Figura 47 Rede *feedforward* com uma camada oculta [**56**].

A matriz dos pesos da camada oculta é a representada por W_1 e a sua dimensão é $n \times k$, $\overline{W_1}$ é a matriz W_1com os pesos dos *bias*, logo a sua dimensão é $(n+1) \times k$. A matriz dos pesos da camada de *outputs* é a matriz W_2 e a sua dimensão é $k \times m$, $\overline{W_2}$ é a matriz W_2 com os pesos dos *bias* logo a sua dimensão é $(k+1) \times m$.

A matriz dos *inputs* da camada oculta é constituída por $o^0 = [o_1^0, \ldots., o_n^0]$ e a matriz dos *inputs* com o *bias* é $\hat{o}^0 = [o_1^0, \ldots., o_n^0, 1]$.

Logo, os *outputs* da camada oculta podem ser calculados da seguinte forma:

$$o^1 = s(\hat{o}^0 . \overline{W_1}), \tag{35}$$

onde s representa a função de ativação sigmoide.

A matriz dos *outputs* da camada oculta mais a *bias* é dada por:

$$\hat{o}^1 = [o_1^1, \ldots, o_k^1, 1].$$

O *output* da rede é calculado através de $\overline{W_2}$ e $\hat{o}^1$:

$$o^2 = s(\hat{o}^1 . \overline{W_2}). \tag{36}$$

Uma vez completada a passagem de cada *input* pela rede (*forward pass*), calcula-se as derivadas da camada de *output* da rede (D_2) e da camada oculta (D_1):

$$D_2 = \begin{bmatrix} o_1^2(1-o_1^2) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & o_m^2(1-o_m^2) \end{bmatrix}, \tag{37}$$

$$D_1 = \begin{bmatrix} o_1^1(1-o_1^1) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & o_k^1(1-o_k^1) \end{bmatrix}. \tag{38}$$

Em seguida calcula-se a matriz dos erros da rede:

$$e = \begin{bmatrix} o_1^2 - t_1 \\ \vdots \\ o_m^2 - t_m \end{bmatrix}. \tag{39}$$

Agora é possível calcular a matriz com os erros de cada neurónio da camada de *output* e da camada oculta:

$$\delta^2 = D_2 . e, \tag{40}$$

$$\delta^1 = D_1 . W_2 . \delta^2. \tag{41}$$

Por último, calcula-se a variação dos pesos:

$$\Delta \overline{W_2}^T = -\alpha . \delta^2 . \hat{o}^1, \tag{43}$$

$$\Delta \overline{W_1}^T = -\alpha . \delta^1 . \hat{o}^0. \tag{44}$$

Generalizando para uma rede com *L* camadas ocultas [**56**], em que a camada de *input* é *L=0*, tem-se:

$$\delta^L = D_L . e, \tag{45}$$

$$\delta^i = D_i . W_{i+1} . \delta^{i+1}, \quad i = 1, \dots, L-1, \tag{46}$$

$$\Delta \overline{W_i}^T = -\alpha . \delta^i . \hat{o}^{i-1}, \quad i = 1, \dots, L-1. \tag{47}$$

Uma maneira de melhorar a velocidade de convergência deste algoritmo consiste em usar a informação das derivadas de segunda ordem dos pesos, ou seja, em calcular a matriz *Hessiana*.

No entanto, calcular a matriz *Hessiana* é uma tarefa computacionalmente intensiva. Uma maneira de resolver este problema é através do algoritmo *Levenberg–Marquardt (ALM)* .Este algoritmo usa a matriz jacobiana para calcular uma aproximação da matriz *Hessiana* e depois, através da manipulação do coeficiente de combinação μ, este algoritmo regula o seu próprio *learning rate*. Na verdade o ALM é a junção do algoritmo BP (estável, mas a sua convergência é mais lenta) e do algoritmo *Gauss-Newton* (convergência mais rápida,

mas com alguma tendência para ser instável). Se o coeficiente de combinação μ se aproximar de zero, então o algoritmo é o *Gauss-Newton* se μ for muito elevado então o ALM comporta-se como o BP.

O coeficiente de combinação μ é o inverso da constante *learning rate* α

$$\mu = \frac{1}{\alpha}. \tag{47}$$

O ALM calcula-se da seguinte maneira:

$$\Delta W_i = -H_i^{-1}.J_i.e, \tag{48}$$

$$H_i = J_i^T.J_i + \mu.I, \tag{50}$$

$$J_i = \delta^i.\hat{o}^{i-1}, \tag{51}$$

onde H_i é a aproximação à matriz *Hessiana*, J_i é a matriz jacobian e I é a matriz identidade.

Quando está a treinar a rede o ALM calcula o erro. Se este erro descer, então o μ é dividido por um fator, por ex., 10. Caso o erro suba, o μ é multiplicado por 10. Ou seja, quando o erro desce o algoritmo comporta-se como o *Gauss-Newton*, quando o erro para de descer o ALM comporta-se como um BP, ficando assim mais estável [**57**].

Redes Neuronais dinâmicas (RND)

As redes neuronais até agora descritas são úteis para tentar encontrar processos estáticos no tempo. Isto significa que a informação dos *inputs* do passado não influência como a rede vai reagir no presente ou seja a rede não tem memória. Em muitas situações, como é o caso das séries de temporais que descrevem os mercados financeiros, os seus comportamentos estão dependentes do passado. Para resolver este problema é necessário usar uma rede dinâmica que não é mais que uma rede neuronal com memória.

De seguida serão descritas as principais arquiteturas de RND.

***Time Delay Neural Network* (TDNN)**

Esta rede é das redes mais simples que existe, basicamente é uma rede *feedforward* cujos *inputs* são os *time lags* da série temporal. Quanto maior o número de *inputs*, tanto maior será a memória da rede e, consequentemente, a sua complexidade.

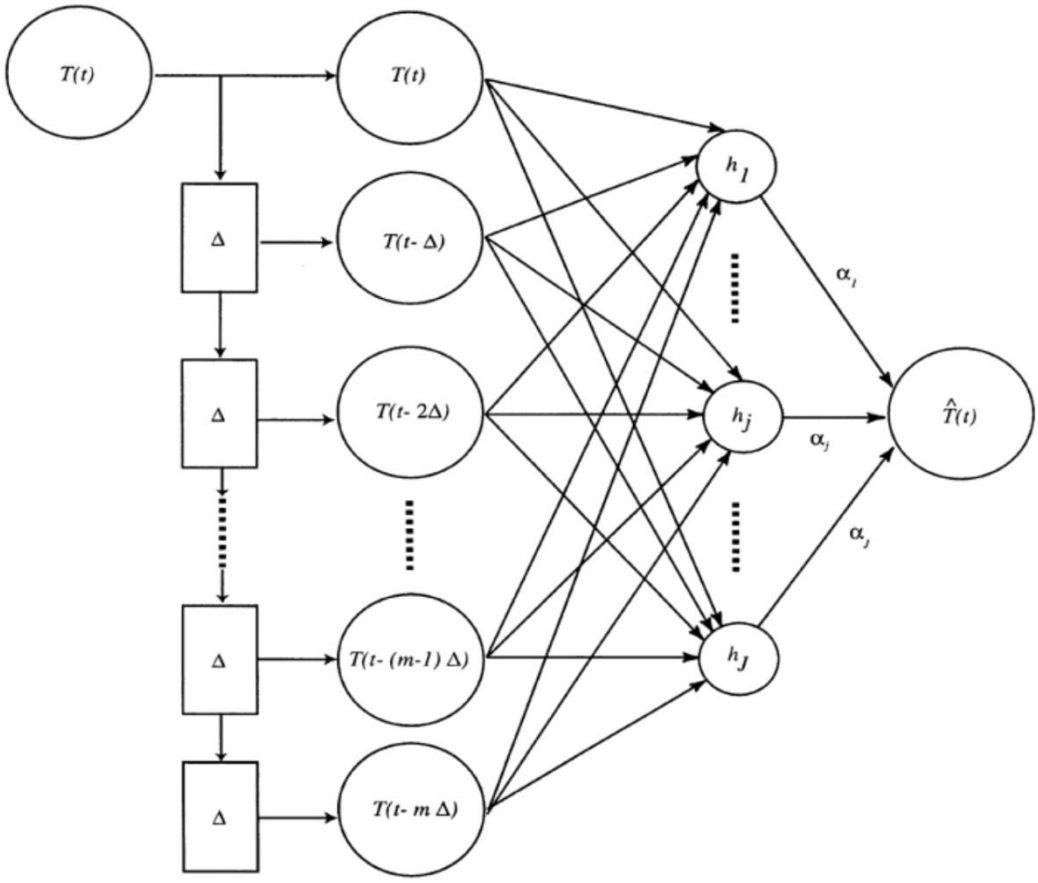


Figura 48 *Time delay neural network* com uma camada oculta [**58**].

Uma TDNN (figura 48) pode ser treinada com o algoritmo BP tal como uma rede *feedforward*.

Rede Elman

Esta rede dinâmica é igual a uma rede *feedforward, mas* com a diferença dos neurónios de contexto (figura 49). Estes neurónios recebem como *inputs* os *outputs* da camada oculta e, por sua vez, a camada de contexto vai servir de *input* à camada oculta criando um *loop*.

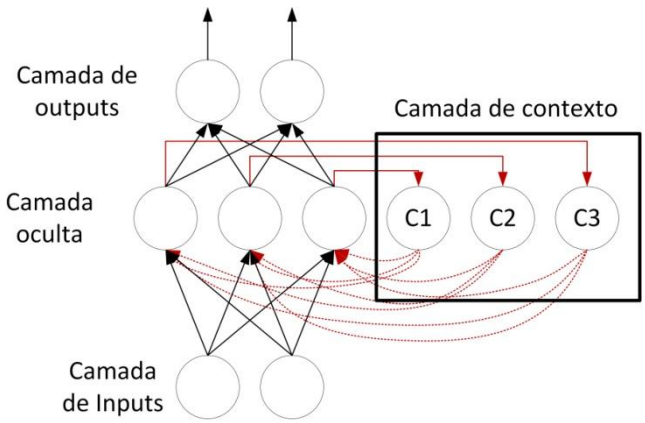


Figura 49 Rede *Elman* com 3 camadas de contexto [**59**].

Uma das desvantagens da *Elman* é que é preciso um neurónio da camada oculta para cada neurónio de contexto. Isto significa que, se for preciso aumentar a camada de contexto obrigatoriamente, então é necessário aumentar a camada oculta [**60**].

Para treinar esta rede é necessário usar o algoritmo *back propagation through time* (BPTT). Este algoritmo consiste em expandir a rede uma certa quantidade de passos no passado. Na figura 50 pode ver-se a interação entre um neurónio da camada oculta e um neurónio da camada de contexto (esquerda). Do lado direito pode ver-se a sua expansão no tempo, que, neste caso, são 3 iterações no passado (a variável $x(4)$ é o input presente e o a variável $x(1)$ é o terceiro input no passado).

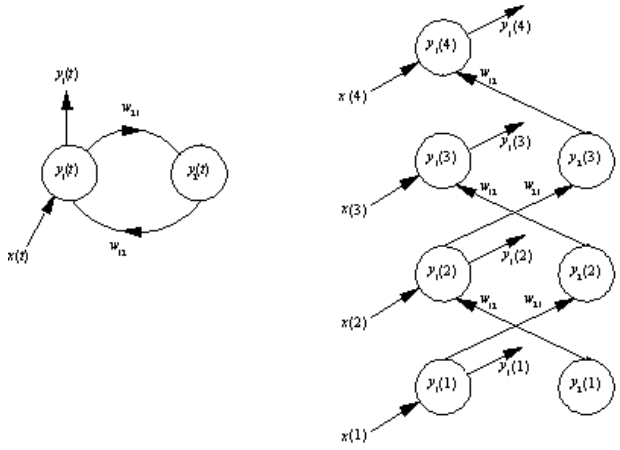


Figura 50 *Back propagation through time* (BPTT) [**61**].

Após a rede ter sido expandida no tempo, o algoritmo BP pode ser empregue como se uma rede *feedforward* se tratasse. Como se pode verificar, à medida que se expande a rede no tempo o seu tamanho aumenta, sendo necessário mais recursos computacionais.

Outra desvantagem deste algoritmo é a explosão/anulamento dos gradientes. Este problema ocorre se uma rede *feedforward* tiver muitas camadas ocultas (*deep neural network*). Se os gradientes forem menores do que 1, então estes podem fazer com que o erro das camadas anteriores se anulem. Se o valor for alto, então o erro pode convergir para valores muito altos. Isto acontece porque os gradientes de cada camada multiplicam-se pelos gradientes da camada posterior [**62**] [**63**].

Rede *Jordan*

Esta rede é semelhante à *Elman*, apenas diferindo do fato da camada de contexto receber *feedback* da camada de *output* em vez da camada oculta (figura 51).

A rede *Jordan* tem as mesmas desvantagens que a *Elman*, com a agravante da camada de contexto ter de ser igual à camada de *outputs*, ou seja, a dependência do número de *outputs* faz com que a rede deixe de ser aplicada em muitas situações [**64**].

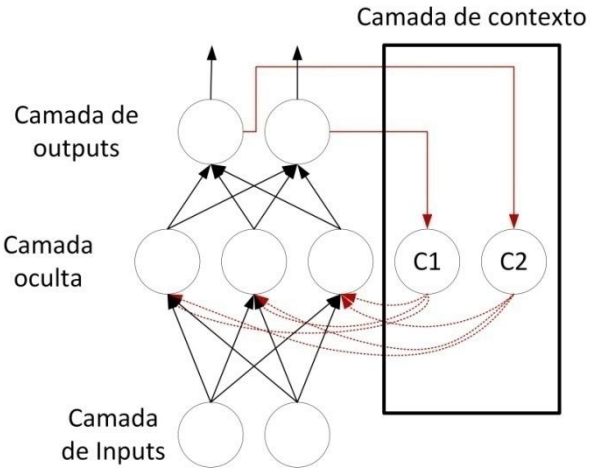


Figura 51 Rede *Jordan* com 2 camadas de contexto [**65**].

***Echo state network* (ESN)**

Uma ESN é um tipo de rede mais avançado, que tem como base um reservatório com neurónios onde estes vão armazenar os estados do processo. O objetivo é que a rede entre num estado de eco. Assim o reservatório vai funcionar como um oscilador onde a informação vai circular (ecoar) dentro do mesmo. Uma ESN não é treinada com o BP. Os pesos são inicializados aleatoriamente e apenas os pesos dos *outputs* é que são treinados através de um treino forçado.

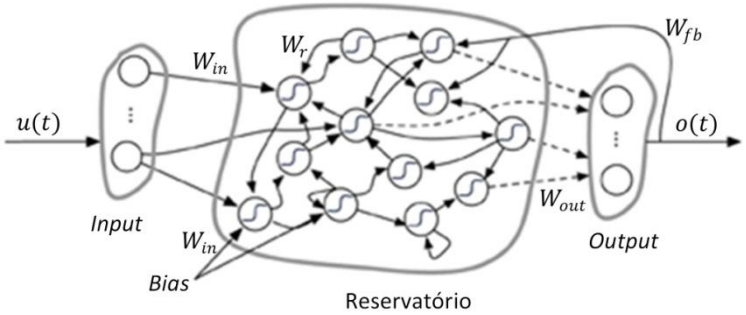


Figura 52 Estrutura de uma ESN [**66**].

Como se pode ver na figura 52 a ESN é composta por uma camada de *inputs*, um reservatório e uma camada de *outputs* (o *feedback* dos *outputs* para o reservatório é opcional pois em algumas situações pode tornar a rede instável). No início os pesos W_{in}, W_r, W_{out}, W_{fb} são gerados aleatoriamente. Normalmente W_{in} e W_r têm apenas 5% dos elementos diferentes de zero. A dimensão de W_{in} é igual a $N_x.(N_{in}+1)$ onde N_{in} é o número de inputs e N_x é o número de estados. A dimensão de W_r é igual a $N_x.N_x$. A dimensão de W_{out} é igual a $N_y.(N_x+N_{in}+1)$ onde N_y é o número de outputs. Finalmente, a dimensão de W_{fb} é igual a $N_y.N_x$.

De seguida é calculado o raio espectral da matriz W_r:.

$$W_r = \alpha \frac{1}{|\lambda_{max}|} W_r, \tag{51}$$

onde λ_{max} é o raio espectral e é calculado através do valor eigen maior da matriz W_r,

α é um escalar e deve ter valores entre 0 e 1, O raio espectral deve ser menor que 1 para garantir o estado de eco da rede. Também é possível atingir o estado de eco se os *inputs* da rede e os *outputs* (no caso de haver *feedback* para o reservatório) não forem nulos.

Os estados do reservatório são calculados da seguinte maneira:

$$\tilde{x}[n] = \tanh\big(W_{in}.\big[1;u[n]\big] + W_r.x[n-1] + W_{fb}.y[n-1]\big), \tag{52}$$

$$x = (1-\gamma).x[n-1] + \gamma.\tilde{x}[n], \tag{53}$$

onde γ é a constante *leaking rate* e serve para ajustar a dinâmica dos estados do reservatório. Se o *leaking rate* for alto, então os estados são atualizados com mais frequência, o que é útil para sinais de alta frequência. Isto significa que os estados antigos são apagados com mais frequência. Os valores típicos situam-se entre $0 < \gamma \leq 1$.

O *output* é:

$$y[n] = W_{out}\big[1;u[n];x[n]\big]. \tag{54}$$

Para treinar a rede usa-se a regressão rígida:

$$Y^{target} = W_{out} X, \tag{55}$$

onde Y^{target} é a matriz dos targets e o seu tamanho é $N_y \cdot T$ em que T é o comprimento dos vetores de input e output $n, \dots\dots, T$, $.X$ é a matriz de estados do sistema e o seu tamanho é $T(N_x + N_{in} + 1)$.

Para realizar uma aprendizagem forçada resolve-se a equação em ordem a W_{out} :

$$W_{out} = Y^{target} X^T (XX^T + \beta I)^{-1}, \tag{56}$$

onde β é o coeficiente de regularização e é usado para penalizar os pesos com valores muito elevados (serve para evitar *overfitting*).

Depois do treino a rede pode gerar o seu próprio *output* para avaliação do erro de treino:

$$Y = W_{out} X, \tag{57}$$

$$MSE = \sum_{n=1}^{T} (Y^{target}[n] - Y[n])^2. \tag{58}$$

A ESN tem a grande vantagem do treino necessitar de poucos recursos computacionais. No entanto, se o problema em questão utilizar muitos *inputs*, então a matriz de estados fica muito grande. Como regra para obter uma boa generalização do problema deve-se respeitar a inequação $N_{in} T < (N_x + N_{in} + 1)$.

Por exemplo, considere-se uma rede com 5 *inputs*, cada um com *T*=10000. Isto faz com que $N_x \approx 50000$ então, a matriz W_r tem uma dimensão de $50000 \times 50000 = 25 \times 10^8$. Se cada elemento da matriz for um *float* de 4 *bytes*, então a matriz vai ocupar cerca de 9 *Tbytes* de memória. O que leva a concluir que esta rede não é usada para resolver problemas muito complexos [**66**] [**67**].

Rede *Nonlinear AutoRegressive with eXogenous inputs* (NARX)

A rede NARX ou *Nonlinear AutoRegressive with eXogenous inputs* é um sistema hibrido que utiliza o modelo autorregressivo e a TDNN.

O modelo matemático e a sua estrutura são os seguintes:

$$y(t) = f\big(y(t-1), y(t-2), \dots, y(t-n_y), u(t-1), u(t-2), \dots, u(t-n_u)\big), \qquad (59)$$

onde u são os outputs, n_u é a ordem de memória dos *inputs* y são os *outputs* e n_y é a ordem de memória dos *outputs*.

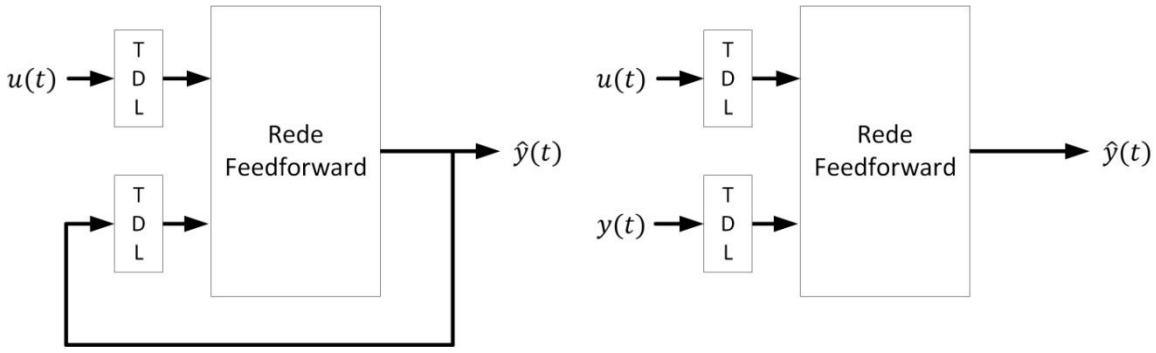


Figura 53 NARX Arquitetura série-paralela (esquerda) e arquitetura paralela (direita) [**68**].

Como se pode ver na figura 53 existem duas topologias da rede NARX, a arquitetura paralela (lado direito), onde a rede é realimentada pelos valores que ela própria gera. Esta arquitetura é utilizada para treinar a rede. Para o treino, validação e teste usa-se a arquitetura série-paralela (lado esquerdo), onde os *targets* são inseridos na realimentação e comparados aos *outputs* da rede, para, depois, se calcular o MSE. Assim, durante o treino obtém-se é uma rede TDNN, que é treinada como uma simples rede *feedforward*, Esta estratégia simplifica muito o treino pois evita usar o algoritmo BPTT.

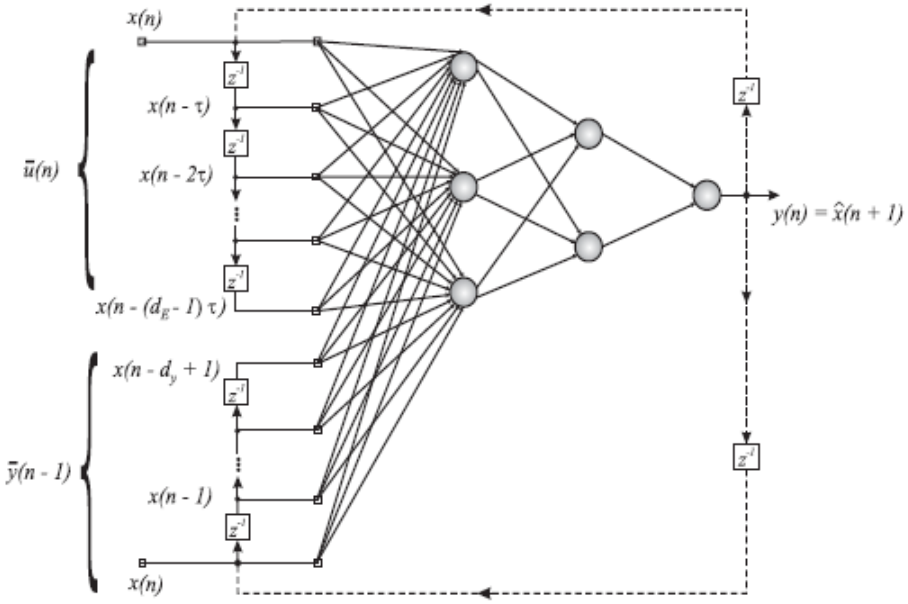


Figura 54 Rede NARX com duas realimentações [**69**].

Na figura 54 pode-se ver a realimentação a tracejado que só se usa depois de treinar a rede. No caso de se pretender fazer previsões de um período no futuro, pode-se usar a rede sem quaisquer realimentações (arquitetura paralela), ou, então, pode-se optar pela realimentação inferior. Para fazer previsões de múltiplos períodos no futuro (previsão *multistep*) usam-se as duas realimentações, ou seja, os outputs são usados recursivamente como *inputs*.

Como se pode ver a NARX consegue usar a simplicidade da TDNN para ser treinada. No entanto, depois do treino é possível usar a informação dos *outputs* passados aumentando o dinamismo da rede. Basicamente esta rede consegue resolver o problema apresentado pelo modelo autorregressivo estudado anteriormente, nomeadamente o de não conseguir lidar com não-linearidades.

Apesar das suas vantagens esta rede requer de muitos *inputs* caso a ordem das memórias seja elevada. O número elevado de *inputs* faz com que a rede aumente significativamente o seu tamanho [**70**] [**69**] [**71**].

4. IMPLEMENTAÇÃO DO PROJETO

No capítulo anterior estudaram-se vários métodos para a previsão de mercados financeiros. Alguns desses métodos estão mais relacionados com fatores económicos, como é o caso das análises fundamentais e técnicas. No entanto, no âmbito deste trabalho o objetivo será implementar um algoritmo que lide eficazmente com as não-linearidades dos mercados e consiga "automatizar", o mais possível, o processo de previsão. Para isso, será aplicado um dos algoritmos relacionados com a área da IA estudados anteriormente. Apesar das análises fundamentais e técnicas estarem essencialmente relacionadas com fatores económicos, o seu estudo é importante, pois, dependendo dos horizontes de previsão, estes métodos podem ser usados como *inputs* nos algoritmos de IA.

Tendo em conta os algoritmos estudados, os mais eficazes para analisar séries temporais são as redes dinâmicas e, dentro das redes dinâmicas, temos as TDNN, *Elman*, *Jordan*, ESN e NARX.

As redes *Elmen* e *Jordan* não serão escolhidas, pois têm as desvantagens das camadas de contexto estarem relacionadas com o número de neurónios da camada oculta e de output. Outro motivo para estas redes não serem escolhidas é devido ao seu método de treino ser o BPTT, que sofre do problema da explosão/anulamento dos gradientes.

A ESN foi escolhida pois a sua arquitetura é simples e não requer que seja predefinido o número de *lags* dos *inputs*/*outputs* a memorizar.

Por último, a NARX é a rede que tem uma melhor relação entre as suas vantagens e desvantagens. Como foi observado no capítulo anterior a NARX é simples de treinar, pois é treinada como uma TDNN, mas ao contrário da TDNN a NARX pode usar *feedback*. Uma desvantagem da NARX é ser preciso especificar o número das memórias dos *inputs* e *outputs*. Outra desvantagem é que caso se já for preciso usar memórias com períodos longos, ou caso a série temporal tenha dependências de longa duração, pode levar a que os *inputs* da rede cresçam aumentando assim a complexidade da rede.

Pelas razões especificadas anteriormente vão ser testadas duas arquiteturas (ESN e NARX) e o seu desempenho vai ser testado num *Intel dual core* 1.6GHz, 2Gb RAM, o sistema operativo é o Windows 7.

O *software* utilizado para construir as redes consiste no *Matlab R15*, pois este já tem incorporado várias *toolboxes* que são úteis neste projeto. Outra vantagem deste *software* é já ter incorporado funções que conseguem aceder à API da *Yahoo finance*.

Os dados a serem processados têm todos o *time frame* de um dia. Ou seja, os dados dos *stocks* são compostos pelos seguintes campos: data, *open*, *high*, *low*, *close* e o volume referentes ao período de um dia.

Antes de prosseguir com a implementação das redes é necessário ver se se existe correlações/auto-correlações entre os *inputs* e os *targets*. No caso das correlações o objetivo é identificar *lags* negativas, pois significa que a variável pode ser usada para prever os *target*.

Nos gráficos seguintes (figura 55), podemos ver que o *close price* que é um *input* e ao mesmo tempo um *target*, a sua auto-correlação é positiva e estende-se, no mínimo, até 1000 dias no passado.

O *High*, *Low* e *Open price* também eles têm fortes correlações positivas com o target.

O Volume tem uma correlação muito forte, mas esta é negativa, o que indica que quando o volume sobe o preço do *stock* desce. No entanto, o volume pode ter no futuro uma correlação positiva. Caso o *stock* esteja muito desvalorizado, uma subida de volume pode indicar uma subida do preço.

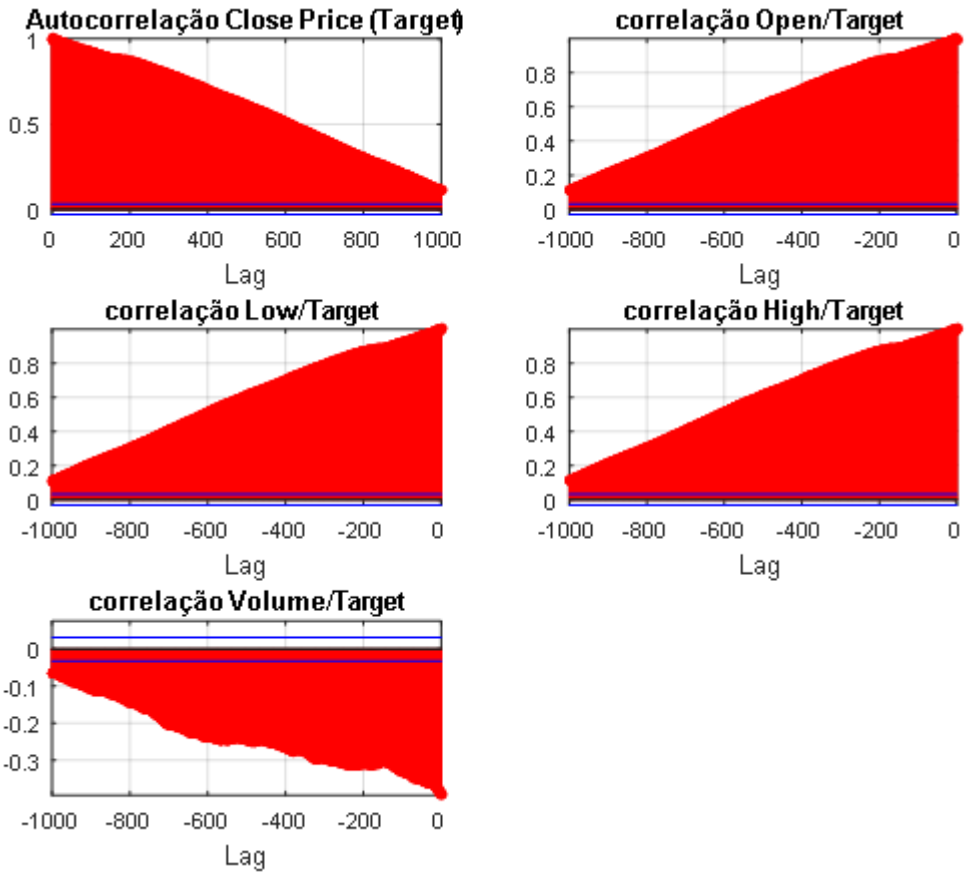


Figura 55 Auto-correlação do *close price* e as correlações dos outros *inputs* restantes com o *target*.

Apesar de se terem estudado métodos de redução de dimensionalidade, neste caso não se justifica o seu uso pois o número de dimensões é muito pequeno.

4.1. IMPLEMENTAÇÃO DA *ECHO STATE NETWORK*

Para a implementação da ESN foram testadas quatro versões, onde alguns parâmetros da rede como: *feedback*, número de *inputs* e tamanho do reservatório foram modificados para tentar melhorar o desempenho.

A primeira versão a ser testada é uma ESN sem *feedback*, apenas com um *input* tal que seu reservatório tem 1000 estados. Esse *input* consiste no *close price* do *stock* da IBM desde 1 de Janeiro de 2000, até 22 de Setembro de 2015, sendo que o *target* da rede também é o *close price*.

Como se pode observar na figura 56, o treino e o teste da rede parecem indicar uma boa generalização da mesma.

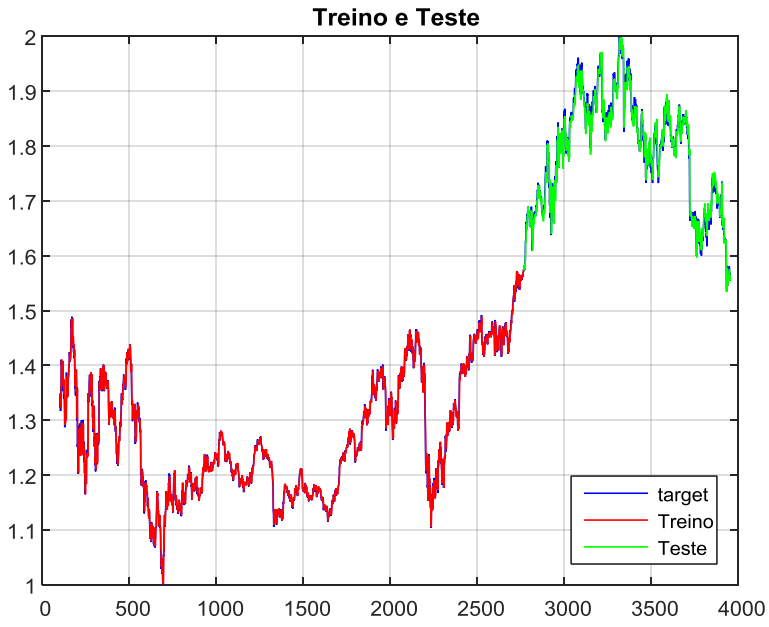


Figura 56 Treino e teste da ESN (1 *input*, sem *feedback*).

Outro aspeto importante são os estados do reservatório *x(n)*. Como se pode ver na figura 57 os estados parecem oscilar de maneira estável, característica necessária para o estado de eco.

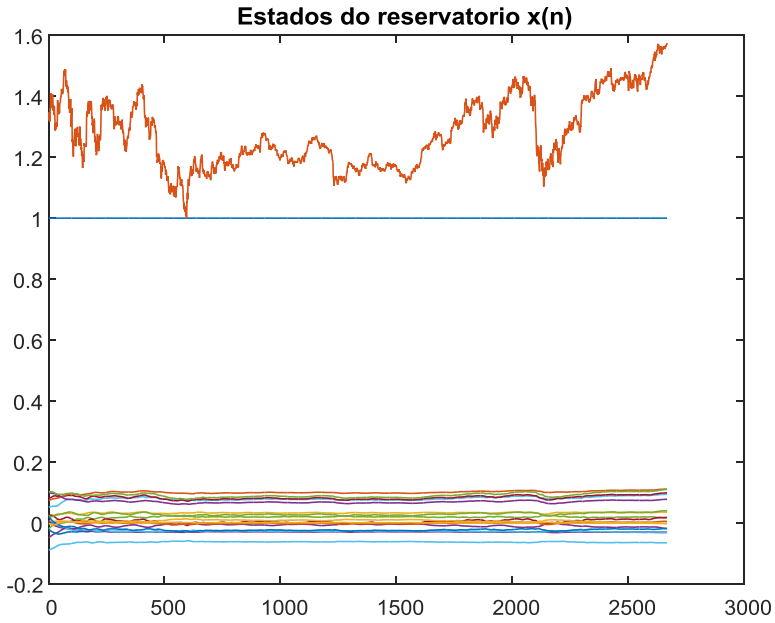


Figura 57 Estados do reservatório. (1 *input*, sem *feedback*).

A segunda versão é identica à primeira, exceto que se ativou o *feedback* da rede. Os gráficos não são mostrados pois são muito semelhantes aos dois anteriores. Como se poderá verificar mais à frente, na tabela de resultados, o *feedback* introduz mais instabilidade à rede aumentando o seu erro.

A terceira versão da rede tem dois *inputs* (*Close price* e Volume), o *feedback* está desativado e o reservatório foi aumentado para 1500 estados. Não são adicionados mais *inputs* pois isto implica um aumento do reservatório proibitivo para o *hardware*. O Volume foi escolhido pois é um indicador muito importante para prever mudanças de direção nos preços.

Como se pode ver na figura 58, a rede também tem um bom desempenho.

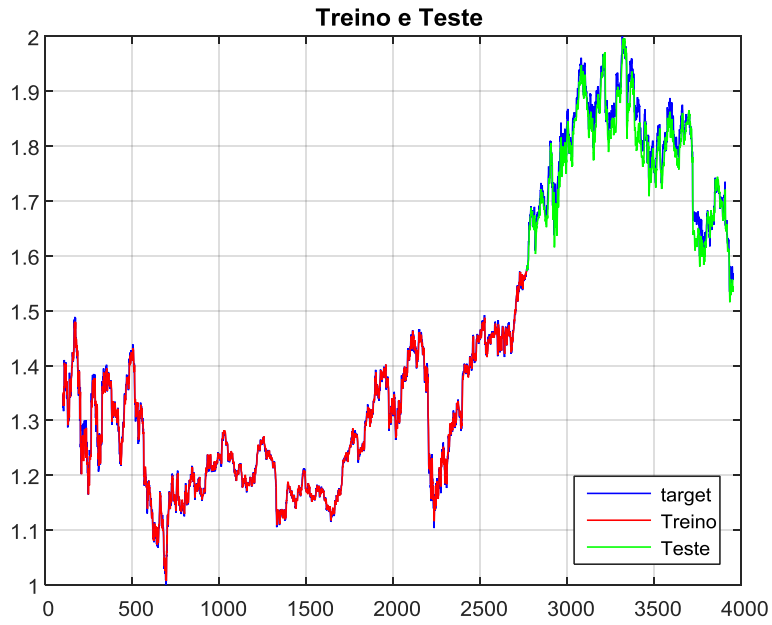


Figura 58 Treino e teste da ESN (2 *input*, sem *feedback*)

Com o aumento de dimensões e do reservatório pode verificar-se um aumento de estados (figura 59).

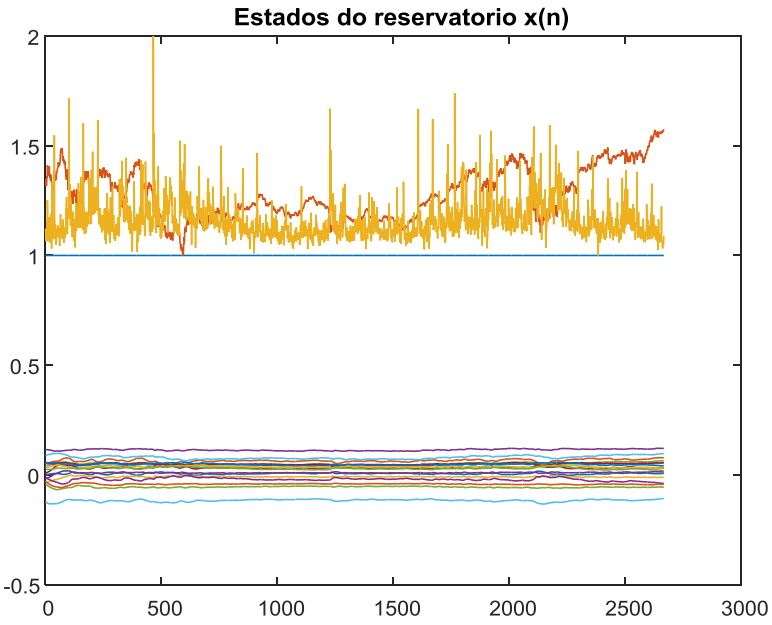


Figura 59 Estados do reservatório (2 *input*, sem *feedback*).

A quarta versão é igual à terceira, mas com o feedback ativado. Tal como aconteceu com um *input*, o feedback aumentou o erro da rede.

Na tabela seguinte estão representadas as características e os resultados numéricos das quatro versões da rede.

Versão	1	2	3	4
Feedback	Não ativado	Ativado	Não Ativado	Ativado
Número de *Inputs*	1	1	2	2
Tamanho do reservatório	1000	1000	1500	1500
Leak rate	0.3	0.3	0.3	0.3
W_{in} fator de escala	0.1	0.1	0.1	0.1
W_r fator de escala	0.1	0.1	0.1	0.1
Tamanho total dos vetores de *input*	3955	3955	3955	3955
Tamanho de treino (*washout*)	100	100	100	100
Tamanho de treino real	70% (2768-100) = 2668	70% (2768-100) = 2668	70% (2768-100) = 2668	70% (2768-100) = 2668
Tamanho do teste	60% 1186	60% 1186	60% 1186	60% 1186

Raio espectral	0.41841	0.4159	0.5097	0.51791
MSE Treino	0.00011858	0.0001208	0.00011462	0.00011531
MAPE Treino	0.88285%	0.88745%	0.96963%	0.98687%
MSE Teste	0.00023955	0.00077836	0.00041274	0.00061578
MAPE Teste	1.602%	2.2553%	2.1867%	2.1175%

Tabela 1 Resultados das 4 versões da ESN.

O tamanho do reservatório aumenta apenas 500 unidades devido a limitações de memória. O tamanho de treino (*washout*) é uma parte do treino que é usada para limpar os estados iniciais do reservatório. O *washout* não é usado para fazer o treino forçado da rede. O *leak rate* toma o valor de 0.3. Este valor foi achado por tentativa erro e está dentro do intervalo normalmente usado. Os fatores de escala dos pesos foram escolhidos de maneira a que o valor inicial dos pesos não causasse instabilidades na rede. Estes valores também foram escolhidos por tentativa erro.

Outro ponto importante é o fato do raio espetral ser sempre inferior a 1. Isso garante o estado de *echo* da rede (como se verificou nos gráficos das figuras 57 e 59), pois os estados não ficam instáveis.

Na tabela 2 pode-se também verificar os erros MSE (*Mean Square Error*) discutido anteriormente. Outra maneira (muito usada) de calcular o erro de uma rede, consiste no MAPE (*Mean Absolute Percentage Error*) Este erro é calculado da seguinte maneira:

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{target - output}{target}\right| \times 100, \tag{60}$$

onde *n* é o tamanho do treino real ou o tamanho do teste.

O MAPE serve para calcular o erro em termos de percentagem. No entanto, tem algumas desvantagens. Por exemplo, se houver um *target* nulo, então o cálculo do MAPE é uma divisão por zero. Outra desvantagem é que se o *output* tiver um valor muito alto em relação ao *target*, então o MAPE pode ter valores superiores a 100%. Para evitar o problema da divisão por zero os *targets* e os *outputs* são normalizados para ficarem compreendidos em valores entre 1 e 2.

Como se pode ver nos resultados, a primeira versão teve um melhor desempenho, o que indica que o reservatório não é suficientemente grande para receber um segundo *input*. Outro aspeto importante que pode ser observado é que, quando a rede é realimentada, o seu erro aumenta ligeiramente.

Mesmo com os problemas referidos anteriormente a rede conseguiu generalizar muito bem, pois com apenas um *input* conseguiu obter uma precisão de $(1 - 0.01602) \times 100 = 98.4\%$, o que é muito próximo de exemplos analisados em vários trabalhos [**72**] [**73**] [**74**].

4.2. IMPLEMENTAÇÃO DA NARX

Na implementação da NARX foram testadas quatro versões da rede em *openloop*:

- 1ª Versão tem 1 *input* (*Close Price*).
- 2ª Versão tem 2 *inputs* (*Close Price* + *Volume*).
- 3ª Versão tem 3 *inputs* (*Close Price* + *High Price* + *Low Price*).
- 4ª Versão tem 5 *inputs* (*Close Price* + *High Price* + *Low Price* + *Volume*).

Os gráficos desta rede são muito semelhantes por isso só é mostrado a segunda versão.

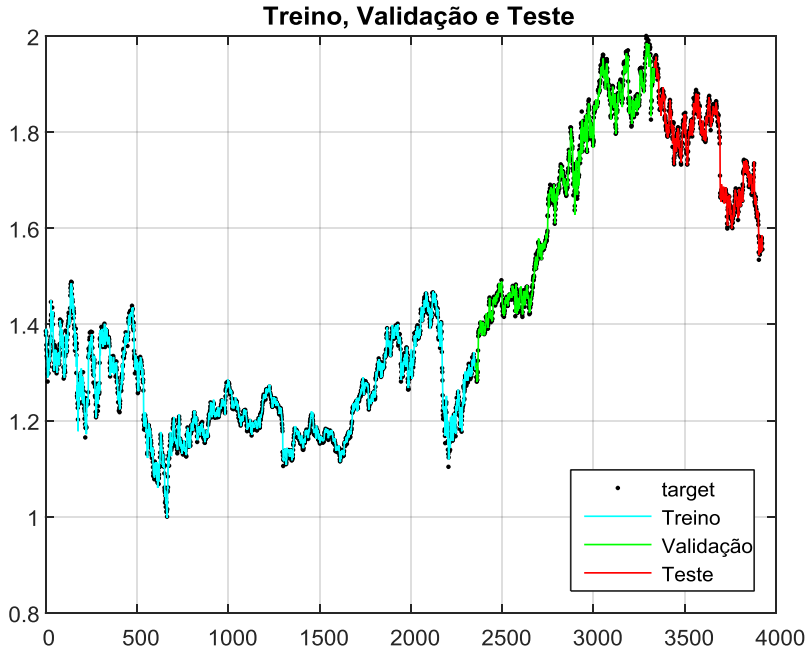


Figura 60 Treino, validação e teste da NARX (2 *inputs*, 30 *delays* e 15 camadas ocultas).

Na figura 60 pode-se ver os vetores de treino, validação e teste. Como é possível verificar a rede está a generalizar muito bem.

Na tabela 3 estão os resultados da rede:

Versão da rede	1	2	3	4
Nº de *Inputs*	1	2	3	5
Epochs calculados até à convergência	6	7	7	5
Delays dos *Inputs*	30	30	30	30
Delays do *Output*	30	30	30	30

Tamanho da camada Oculta	15	15	15	15
Tamanho do vetor de treino	60% 2373	60% 2373	60% 2373	60% 2373
Tamanho do vetor de Validação	25% 988	25% 988	25% 988	25% 988
Tamanho do vetor de teste	15% 593	15% 593	15% 593	15% 593
MSE Treino	0.00017355	0.00012435	0.00012198	0.00017592
MSE Validação	0.00019204	0.00014636	0.0001600	0.00022337
MSE Teste	0.00019475	0.00015565	0.00018435	0.00030442
MAPE Treino	0.0002439%	5.0577e-06%	5.4611e-06%	0.00024045%
MAPE Validação	0.00025873%	3.1207e-05%	0.0001768%	0.00053837%
MAPE Teste	0.00057805%	0.00013654%	0.00052844%	0.0011709%

Tabela 2 Resultados das 4 versões da NARX

Neste caso foram escolhidos 30 *delays*. Este número normalmente é encontrado através do número de *lags* onde existe auto-correlação e correlação entre os *inputs*. Todavia, como foi visto anteriormente, o número de *lags* é muito elevado e, por isso, foram escolhidos 30 *delays*, que é o número máximo que o *hardware* consegue suportar.

Como se pode ver na tabela 3 o melhor resultado foi obtido pela segunda versão (dois inputs). Neste caso pode-se ver que a rede tem um desempenho superior à ESN, sendo que a precisão da NARX é $(1 - 0.0000013654) \times 100 = 99.9998\%$.

Caso seja necessário aumentar o horizonte de previsão, basta testar a rede com os *feedbacks* e obtém-se o gráfico da figura 61.

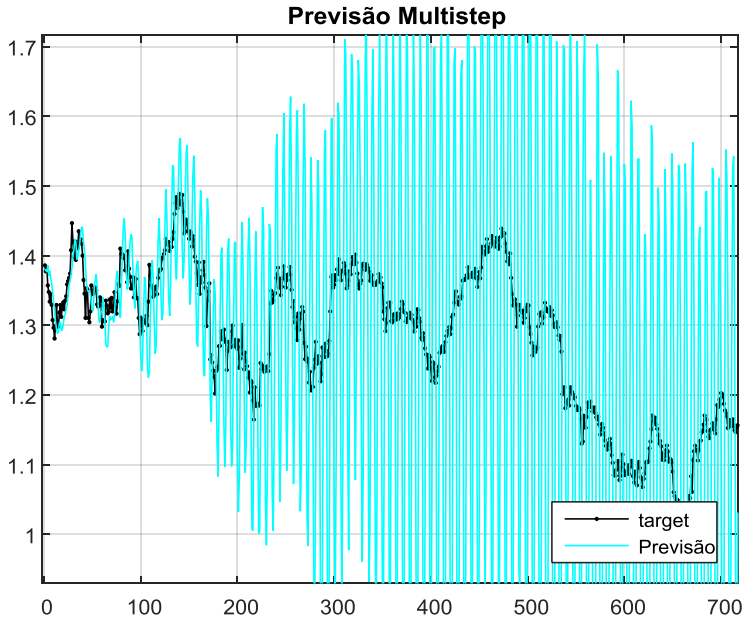


Figura 61 NARX em *close loop* (previsão multistep).

No gráfico da figura 61 pode-se ver que a variância da previsão começa a crescer significativamente à medida que o horizonte aumenta. Isto acontece sempre que se usam os *outputs* da rede recursivamente, pois os erros desses *outputs* vão sendo somados, também eles recursivamente, aumentando assim a variância.

5. CONCLUSÕES

Ao longo deste documento foram apresentados alguns dos principais instrumentos financeiros que existem, tais como *stocks*, divisas (*forex*), matérias-primas e *bonds*. Também foram descritos alguns instrumentos mais "exóticos", como os derivativos, sendo que as principais são: contratos *forward*, futuros, opções e *swaps*. Estas derivativas permitem aos investidores protegerem os seus investimentos (*hedging*) de maneira a minimizar os riscos.

Para estes instrumentos serem usados com eficácia, eles devem ser cuidadosamente analisados para desta maneira conseguir fazer previsões com o máximo de precisão.

Algumas das análises efetuadas nesta tese focaram fatores económicos, como é o caso das análises fundamentais (e.g., análise *top down*). Outras usam ferramentas matemáticas, como é o caso da análises técnicas (indicador estocástico e média deslizante) e quantitativas (AR, MA e ARMA).

O tipo de análise que este documento focou foi no uso de ferramentas do campo da IA, tais como *K mean clustering*, *K nearest neighbours* e redes neuronais. Das redes neuronais

concluiu-se que as redes que melhor se ajustam às séries temporais são as redes dinâmicas nomeadamente a ESN e a NARX.

A implementação da ESN e da NARX permitiu verificar que, mesmo com uma rede muito básica, é possível obter precisões muito altas (para horizontes de previsão curtos), No entanto, deve-se estar consciente que os mercados são muito dinâmicos e que as redes podem perder o seu poder de generalização com o decorrer do tempo. Isto quer dizer que quanto maior for o horizonte de previsão, maior o risco de a rede se tornar instável e de deixar de generalizar, pois podem acontecer eventos que influenciem a série temporal em questão.

Outro ponto importante a referir, tem a ver com constatação que a NARX teve um desempenho superior à ESN para o mesmo poder computacional. Isto verifica-se porque a ESN não lida muito bem com o aumento de dimensionalidade, o que faz com que seja impossível usar mais *inputs* na rede.

Apesar dos resultados a nível de precisão serem satisfatórios é ainda possível fazer melhorias significativas, nomeadamente a nível de robustez dos resultados.

As melhorias que se podem implementar são:

- Usar uma potência computacional superior.
- Recorrer a *data feeds* para obter mais *inputs* e em tempo real;
- Pesquisar outras arquiteturas de redes neuronais que eventualmente possam revelar um melhor desempenho.

Uma arquitetura que valerá a pena analisar futuramente é a *Long Short Term Memory* (LSTM). Esta arquitetura só tem sido usada recentemente e, por isso, ainda não está tão divulgada como as demais redes. No entanto, a LSTM tem tido resultados superiores, nomeadamente em ler texto e reconhecer o contexto das sequências de palavras que constituem o texto, o que indica que pode ser útil para séries temporais financeiras.

Na realidade algumas instituições financeiras (como, por exemplo, os *hedge funds*), usam estas técnicas para analisar milhares de instrumentos e descobrir quais onde se deve investir, para maximizar o retorno e diminuir o risco dos seus portefólios de instrumentos.

Tendo em conta o estudo desenvolvido neste documento perspetiva-se o desenvolvimento de novos algoritmos no futuro. A aquisição de novos conhecimentos no campo da IA e do setor financeiro, pode também servir de base para futuramente expandir esta área de investigação.

Referências Documentais

[1] (2015, Oct.) [Online]. https://is.vsfs.cz/repo/3061/135_12_vsfs_acta_1-2012_hilo-konz.pdf

[2] (2015, Oct.) [Online]. http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-825-techniques-in-artificial-intelligence-sma-5504-fall-2002/lecture-notes/Lecture1Final.pdf

[3] (2015, Oct.) [Online]. http://www.cmegroup.com/education/options.html

[4] (2015, Oct.) [Online]. http://www.cmegroup.com/trading/

[5] (2015, Oct.) [Online]. https://www.nyse.com/products/equities

[6] (2015, Oct.) [Online]. http://www.cboe.com/aboutcboe/default.aspx

[7] (2015, Oct.) [Online]. http://www.cnbc.com/id/47099278

[8] (2015, Oct.) [Online]. https://www.trade.education/wp-content/uploads/stock-ticker-symbols.jpg

[9] (2015, Oct.) [Online]. http://finviz.com/help/technical-analysis/charts-patterns.ashx

[10] Paul Wilmott, *Quantitative Finance*., 2006, vol. 1,2,3.

[11] (2015, Oct.) [Online]. http://finviz.com/help/technical-analysis/charts-patterns.ashx

[12] (2015, Oct.) [Online]. http://www.spindices.com/documents/methodologies/methodology-index-math.pdf

[13] (2015, Oct.) [Online]. http://www.ftse.com/products/indices/our-business

[14] (2015, Oct.) [Online]. http://new.dowjones.com/products/djx/

[15] (2015, Oct.) [Online]. http://ieor.columbia.edu/files/seasdepts/industrial-engineering-operations-research/pdf-files/Eliezer_David.pdf

[16] (2015, Oct.) [Online]. http://i.telegraph.co.uk/multimedia/archive/01870/basket_1870914b.jpg

[17] (2015, Oct.) [Online]. http://pages.stern.nyu.edu/~lpederse/papers/CarryTradesCurrencyCrashes.pdf

[18] (2015, June) [Online]. http://www.oanda.com/currency/live-exchange-rates/

[19] (2015, Oct.) [Online]. https://www.moodys.com/researchdocumentcontentpage.aspx?docid=PBC_79004

[20] (2015, Oct.) [Online]. http://www.sec.gov/investor/alerts/ib_corporatebonds.pdf

[21] (2015, Oct.) [Online]. https://farm7.static.flickr.com/6225/6323015485_3175b224d6_b.jpg

[22] (2015, Oct.) [Online]. http://www.option-dojo.com/en/le/call.html

[23] (2015, Oct.) [Online]. http://www.nasdaq.com/article/options-coach-selling-naked-calls-cm25448

[24] (2015, Oct.) [Online]. http://www.option-dojo.com/en/le/put.html

[25] (2015, Oct.) [Online]. http://www.slideshare.net/himanshujaiswal/derivatives-42674278

[26] (2015, Oct.) [Online]. https://www.princeton.edu/ceps/workingpapers/91malkiel.pdf

[27] Bernard Baumohl, *The Secrets of Economic Indicators*, Third Edition ed., 2013.

[28] (2015, Oct.) [Online]. https://www.credit-

suisse.com/pwp/pb/pb_research/technical_tutorial_de.pdf

[29] (2015, Oct.) [Online]. http://www.mrao.cam.ac.uk/~mph/Technical_Analysis.pdf

[30] (2015, Oct.) [Online]. http://www.metastock.com/customer/resources/taaz/?c=3&p=106

[31] (2015, Oct.) [Online]. http://www.zora.uzh.ch/49785/4/Dzielinski_New_sensivity_and_the_cross-section_of_stock_returns-V.pdf

[32] Ruey S. Tsay, *Analysis of Financial Time Series*, 2nd ed., 2005.

[33] (2015, Oct.) [Online]. https://upload.wikimedia.org/wikipedia/en/thumb/e/e1/Stationarycomparison.png/220px-Stationarycomparison.png

[34] (2015, Oct.) [Online]. http://static.cdn-seekingalpha.com/uploads/2014/4/21505361_13969659221773_rId7.png

[35] (2015, Oct.) [Online]. https://www.jpmorgan.com/cm/BlobServer/AM_Non-normality_of_market_returns.pdf?blobcol=urldata&blobtable=MungoBlobs&blobkey=id&blobwhere=1158543264199&blobheader=application%2Fpd.

[36] Ethem Alpaydın, *Introduction to machine learning*, 2nd ed., 2010.

[37] Stephen Marsland, *Machine Learning An algorithmic perspective*, 2nd ed., 2015.

[38] (2015, Oct.) [Online]. http://www.brnt.eu/phd/bias-variance.png

[39] (2015, Oct.) [Online]. https://s16-us2.ixquick.com/cgi-bin/serveimage?url=http%3A%2F%2F4.bp.blogspot.com%2F-wduUats3Qrg%2FTxEs6SuI41I%2FAAAAAAAAAA7Y%2FiYA5qcv_GTI%2Fs1600%2F018-pca-001.png&sp=5855aba077a4f780289bde752c872c64

[40] (2015, Oct.) [Online].

https://assessingpsyche.files.wordpress.com/2014/01/independentfactors1.png?w=696&h=522

[41] (2015, Oct.) [Online]. https://upload.wikimedia.org/wikipedia/commons/thumb/5/5e/K_Means_Example_Step_1.svg/124px-K_Means_Example_Step_1.svg.png

[42] (2015, Oct.) [Online]. https://upload.wikimedia.org/wikipedia/commons/thumb/a/a5/K_Means_Example_Step_2.svg/139px-K_Means_Example_Step_2.svg.png

[43] (2015, Oct.) [Online]. https://upload.wikimedia.org/wikipedia/commons/thumb/3/3e/K_Means_Example_Step_3.svg/139px-K_Means_Example_Step_3.svg.png

[44] (2015, Oct.) [Online]. https://upload.wikimedia.org/wikipedia/commons/thumb/d/d2/K_Means_Example_Step_4.svg/139px-K_Means_Example_Step_4.svg.png

[45] (2015, Oct.) [Online]. https://upload.wikimedia.org/wikipedia/commons/thumb/0/05/DBSCAN-density-data.svg/220px-DBSCAN-density-data.svg.png

[46] (2015, Oct.) [Online]. https://upload.wikimedia.org/wikipedia/commons/thumb/d/d8/EM-Gaussian-data.svg/186px-EM-Gaussian-data.svg.png

[47] (2015, Oct.) [Online]. https://upload.wikimedia.org/wikipedia/commons/thumb/5/53/Cumulative_vs_normal_histogram.svg/2000px-Cumulative_vs_normal_histogram.svg.png

[48] (2015, Oct.) [Online]. https://upload.wikimedia.org/wikipedia/en/thumb/4/41/Comparison_of_1D_histogram_and_KDE.png/500px-Comparison_of_1D_histogram_and_KDE.png

[49] (2015, Oct.) [Online]. https://upload.wikimedia.org/wikipedia/en/thumb/e/e5/Comparison_of_1D_bandwidth_selectors.png/220px-Comparison_of_1D_bandwidth_selectors.png

[50] (2015, Oct.) [Online]. https://en.wikipedia.org/wiki/File:KnnClassification.svg

[51] (2015, Oct.) [Online]. http://www.webpages.ttu.edu/dleverin/neural_network/fig_3p6_neurons_NEURON4.jpg

[52] (2015, Oct.) [Online]. https://c.mql5.com/18/20/NN1__1.gif

[53] (2015, Oct.) [Online]. https://camo.githubusercontent.com/e6a0e02bd080acc585a622d2c03ca6e44a9e9adc/687474703a2f2f692e696d6775722e636f6d2f36565a6542706e2e706e67

[54] (2015, Oct.) [Online]. http://www.willamette.edu/~gorr/classes/cs449/figs/jen/musmall.gif

[55] (2015, Oct.) [Online]. http://page.mi.fu-berlin.de/rojas/neural/chapter/K7.pdf

[56] (2015, Oct.) [Online]. http://www.eng.auburn.edu/~wilambm/pap/2011/K10149_C012.pdf

[57] (2015, Oct.) [Online]. http://www.palgrave-journals.com/jors/journal/v54/n3/images/2601523f6.gif

[58] (2015, Oct.) [Online]. http://mnemstudio.org/ai/nn/images/Elman1.gif

[59] (2015, Oct.) [Online]. http://mnemstudio.org/neural-networks-elman.htm

[60] (2015, Oct.) [Online]. http://neuron.eng.wayne.edu/tarek/MITbook/chap5/img00188.gif

[61] (2015, Oct.) [Online]. http://ir.hit.edu.cn/~jguo/docs/notes/bptt.pdf

[62] (2015, Oct.) [Online]. http://arxiv.org/pdf/1211.5063.pdf

[63] (2015, Oct.) [Online]. http://www.ice.ge.cnr.it/~muselli/papers/jh08.pdf

[64] (2015, Oct.) [Online]. http://pubs.sciepub.com/ajie/2/1/3/image/fig3.png

[65] (2015, Oct.) [Online]. http://minds.jacobs-university.de/sites/default/files/uploads/papers/ESNTutorialRev.pdf

[66] (2015, Oct.) [Online]. http://minds.jacobs-university.de/sites/default/files/uploads/papers/PracticalESN.pdf

[67] (2015, Oct.) [Online]. http://www.mathworks.com/help/releases/R2015b/nnet/ug/dynamic_narx_ser_par.gif

[68] (2015, Oct.) [Online]. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.69.9329&rep=rep1&type=pdf

[69] (2015, Oct.) [Online]. http://www.mathworks.com/help/nnet/ug/design-time-series-narx-feedback-neural-networks.html

[70] (2015, Oct.) [Online]. http://www.mafy.lut.fi/timeseries/ESTSP/PDF/09.pdf

[71] (2015, Oct.) [Online]. http://www.cs.berkeley.edu/~akar/IITK_website/EE671/report_stock.pdf

[72] (2015, Oct.) [Online]. http://people.cs.pitt.edu/~hashemi/papers/CISIM2010_HBHashemi.pdf

[73] (2015, Oct.) [Online]. http://www.ijcaonline.org/volume22/number2/pxc3873497.pdf

[74] (2015, Oct.) [Online]. http://hyxep.vns.me/berliner-pilarmonike/ne-ork-stock-

excange-tours/

[75] (2015, Oct.) [Online].
http://en.wikipedia.org/wiki/Stock_market#/media/File:NASDAQ.JPG

[76] (2015, Oct.) [Online].
http://www.slaughterinvest.com/assets/img/commentaries/yield-curves.jpg

[77] (2015, Oct.) [Online].
https://upload.wikimedia.org/wikipedia/commons/thumb/2/28/DBSCAN-Gaussian-data.svg/186px-DBSCAN-Gaussian-data.svg.png

[78] (2015, Oct.) [Online].
https://upload.wikimedia.org/wikipedia/commons/thumb/8/8a/OPTICS-Gaussian-data.svg/186px-OPTICS-Gaussian-data.svg.png