Susana Maria Gonçalves Tavares Rodrigues

MODELAÇÃO DO DESEMPENHO ACADÉMICO DE ESTUDANTES UNIVERSITÁRIOS UTILIZANDO REDES NEURONAIS

Lisboa
2011

Susana Maria Gonçalves Tavares Rodrigues

MODELAÇÃO DO DESEMPENHO ACADÉMICO DE ESTUDANTES UNIVERSITÁRIOS UTILIZANDO REDES NEURONAIS

Dissertação apresentada na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa para obtenção do grau de Mestre em Matemática e Aplicações - Actuariado, Estatística e Investigação Operacional.

Dissertação Orientada por:
Professor Doutor Ruy Costa

Lisboa
2011

Agradecimentos

Ao Professor Doutor Ruy Costa gostaria de expressar o meu profundo e sincero agradecimento por todo o seu empenho e dedicação durante todo o processo de elaboração desta dissertação, pelas suas sugestões, opiniões e criticas sempre construtivas. Um agradecimento muito especial pela sua paciência interminável, pela sua persistência e por ter acreditado nas minhas capacidades como aluna.

Gostaria de agradecer à Professora Margarida Cardoso e ao Professor Victor Lobo pela sua disponibilidade e pelos seus comentários construtivos e esclarecedores, que contribuíram para o desenvolvimento desta dissertação.

Gostaria também de agradecer a todos os Professores ao longo do meu progresso académico pela sua contribuição na minha formação.

À minha família, aos meus amigos e colegas um obrigado pelo seu apoio, paciência e compreensão nas minhas constantes ausências e pelos momentos menos felizes que terei certamente causado ao longo de todo o processo de desenvolvimento deste trabalho.

Ao meu namorado, melhor amigo e parceiro, em tudo o que faço, Cesário, um obrigado muito especial pela sua paciência, compreensão e apoio nos meus momentos mais difíceis durante todo o processo de elaboração deste trabalho. Sem o seu apoio, compreensão e motivação este trabalho não estaria certamente concluído.

Resumo

A modelação do desempenho académico global dos alunos universitários é da maior importância no âmbito da boa gestão universitária. Um modelo capaz de prever o desempenho académico global de alunos universitários com base no desempenho académico nos primeiros anos frequentados no ensino universitário pode ser um instrumento de maior importância na gestão universitária.

Este instrumento possibilitaria a previsão da evolução da população universitária com as óbvias implicações na previsão de receitas da instituição, da gestão do corpo docente e dos recursos materiais. Por outro lado, se utilizado em contexto de apoio pedagógico, a previsão do desempenho académico no final dos primeiros anos frequentados poderia possibilitar a sinalização de alunos com dificuldades, onde se justificaria aplicar medidas de apoio.

O objectivo desta dissertação surge neste contexto: abordar a modelação do desempenho académico global de estudantes universitários, com base no seu desempenho académico nos primeiros anos frequentados.

Nesta dissertação pretendemos criar um modelo capaz de prever o desempenho académico global de alunos de um curso de engenharia da FCT-UNL (curso de 5 anos) com base no desempenho académico no primeiro ano (ou nos dois primeiros anos). Especificamente, pretendemos criar um modelo capaz de prever duas características fundamentais do desempenho académico global: a Duração Total de Curso e a Nota Final do Curso.

A modelação levada a cabo recorre a Redes Neuronais Artificiais, que terão como variáveis de input os resultados obtidos no primeiro (ou nos dois primeiros) ano(s) frequentado(s) nas diferentes áreas cientificas e, o ano de ingresso. Como variáveis de output, considera-se a duração total do curso e a nota final do curso.

Palavras chave: *Redes Neuronais Artificiais, Retropropagação, Modelação.*

Abstract

Modelling the global academic performance of undergraduate students is a tool of good university management of utmost importance. The prediction of the global academic performance of undergraduate students based on their academic performance in the first year (or first two years) in the university would be very important to the university management.

This tool would allow the prediction of the evolution of the undergraduate population with the obvious implications for estimates of revenue of the institution, administration of the teaching staff and of material resources. On the other hand, if used in the context of pedagogical support, the prediction of poor global academic performance of an undergraduate student just after one (or two) year(s) in the university, might allow students with difficulties to be assigned support measures.

The aim of this thesis arises in this context: modelling and prediction of the global academic performance of undergraduate students based on their academic performance in the first year (or first two years) in the university.

In this thesis we intend to create a model capable of predicting the global academic performance of students of an engineering degree from FCT-UNL (a 5-years course) based on their academic performance in first year (or first two years). Specifically, we intend to create a model capable of predict two fundamental characteristics of the global academic performance for each undergraduate student: the Total Duration of the Course and Final Course Mark.

Modelling global academic performance is carried out using Artificial Neural Networks. As input variables we considered the marks in the first year (or first two years) courses and the admission year. Total duration of the engineering degree course and final course mark were the output variables.

Keywords: *Artificial Neural Networks, Back-Propagation, Modelling.*

Notações

NOTAÇÃO	*DESCRIÇÃO*
AING	Ano de ingresso no curso.
DTC	Número de anos frequentados até finalizar o curso.
FA	Frequência absoluta.
FIS	Soma da classificação das disciplinas de Física concluídas no primeiro ano frequentado.
FIS2	Soma da classificação das disciplinas de Física concluídas nos dois primeiros anos frequentados.
FR	Frequência relativa
MAT	Soma da classificação das disciplinas de Matemática concluídas no primeiro ano frequentado.
MAT2	Soma da classificação das disciplinas de Matemática concluídas nos dois primeiros anos frequentados.
MSE_{te}	MSE do conjunto de teste para um treino.
MSE_{val}	MSE do conjunto de validação para um treino.
MSE_{tr}	MSE do conjunto de validação para um treino.
$\overline{MSE}_{te}$	Valor médio do MSE do conjunto de teste de um ciclo.
$\overline{MSE}_{tr}$	Valor médio do MSE do conjunto de treino de um ciclo.
$\overline{MSE}_{val}$	Valor médio do MSE do conjunto de validação de um ciclo.
$MSE_{tr_{min}}$	MSE_{tr} mínimo de um ciclo.
$MSE_{te_{min}}$	MSE_{te} mínimo de um ciclo.
$MSE_{val_{min}}$	MSE_{val} mínimo de um ciclo.
$\overline{MSE}_{teDTC}$	Valor médio do MSE do conjunto de teste para a previsão de DTC de um ciclo.
$\overline{MSE}_{teNFC}$	Valor médio do MSE do conjunto de teste para a previsão de NFC de um ciclo.

NOTAÇÃO	*DESCRIÇÃO*
$MSE0_{E_{DTC}}$	MSE de referência para DTC com mudança de escala (MSE obtido se igualarmos a $\widehat{DTC}$ à média de DTC com mudança de escala).
$MSE0_{NE_{DTC}}$	MSE de referência para DTC real (MSE obtido se igualarmos a $\widehat{DTC}$ à média de DTC real, isto é, na escala original).
$MSE0_{E_{NFC}}$	MSE de referência para NFC com mudança de escala (MSE obtido se igualarmos a $\widehat{DTC}$ à média de NFC com mudança de escala).
$MSE0_{NE_{NFC}}$	MSE de referência para NFC real (MSE obtido se igualarmos a $\widehat{DTC}$ à média de NFC real, isto é, na escala original).
$MSE_{min_{NE_{DTC}}}$	MSE de $\widehat{DTC}$ (obtido da "rede óptima"de um ciclo) na amostra total de DTC.
$MSE_{min_{NE_{NFC}}}$	MSE de $\widehat{NFC}$ (obtido da 'rede óptima" de um ciclo) na amostra total de NFC.
NFC	Nota final de conclusão do curso.
NR	Número de repetições do treino para um ciclo de experiências.
$\widehat{DTC}$	Número de anos frequentados até finalizar o curso previsto pela "rede óptima" de um ciclo (na escala original).
$\widehat{NFC}$	Nota final de conclusão do curso prevista pela "rede óptima" de um ciclo (na escala original).
OUT	Soma da classificação de Outras disciplinas concluídas no primeiro ano frequentado.
OUT2	Soma da classificação de outras disciplinas concluídas nos dois primeiros anos frequentados.
PC	Previsões correctas.
PI	Previsões incorrectas.
QUIM	Soma da classificação das disciplinas de Química concluídas no primeiro ano frequentado.
QUIM2	soma da classificação das disciplinas de Química concluídas nos dois primeiros anos frequentados.
RNA	Rede Neuronal Artificial.
RP	Algoritmo de aprendizagem de Retropropagação.
TES	Conjunto de Teste.
TRE	Conjunto de Treino.
VAL	Conjunto de Validação.
$\sigma_{MSE_{val}}$	Desvio padrão do MSE do conjunto de validação de um ciclo.

NOTAÇÃO	*DESCRIÇÃO*
$\sigma_{MSE_{te}}$	Desvio padrão do MSE do conjunto de teste de um ciclo.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação

A modelação do desempenho académico dos alunos universitários é da maior importância no âmbito da boa gestão universitária. Um modelo matemático que consiga prever o desempenho académico no final do curso de um aluno universitário com base, no seu desempenho nos primeiros anos frequentados, poderá permitir à universidade a intervenção prévia em grupos particulares de alunos de forma a melhorar os resultados, ou mesmo criar mecanismos de alerta para estas situações.

É neste contexto que surge o principal objectivo desta dissertação, responder à seguinte questão:

> Será possível prever o Desempenho Académico Global num curso de cinco anos curriculares de um aluno com base no seu Desempenho Académico nos primeiros anos frequentados?

Para responder a esta questão temos de começar por explicar o conceito de "desempenho académico" de um aluno e de como pode ser medido.

Em geral, após a conclusão do seu curso, o aluno depara-se frequentemente, por exemplo em processos de recrutamento, com duas questões:

1. Qual a duração do seu percurso universitário?
2. Qual a nota final de curso?

Assim, parece aceitável, considerar como medidas do seu desempenho no final do curso as seguintes variáveis: Duração Total do Curso (DTC) e a sua Nota Final de Curso (NFC).

O objectivo a que nos propomos é, deste modo, criar um modelo, capaz de prever o comportamento destas duas variáveis, com base no desempenho académico do aluno nos seus primeiros anos de frequência.

A modelação do desempenho académico global (DTC e NFC) de cada aluno universitário será feita com recurso à área científica de Redes Neuronais Artificias, considerando dados relativos a um curso de engenharia da FCT-UNL (curso de 5 anos), relativos a um período de 20 anos.

Os dados utilizados foram recolhidos por Nunes [Nunes, 2007], no contexto da dissertação de mestrado em Matemática e Aplicações - Ramo Investigação Operacional, que procedeu a um primeiro tratamento estatístico dos mesmos. A amostra consiste na informação de 297 alunos de uma Licenciatura em Engenharia (pré-Bolonha) da FCT-UNL com 5 anos de duração. Nunes procurou modelar o desempenho académico global (DTC e NFC) com recurso a técnicas de simulação após tratamento estatístico.

Numa primeira fase, tomamos como *input* o desempenho do aluno no primeiro ano frequentado e procuraremos prever apenas a duração total do percurso universitário (Cenário 1).

Numa segunda fase, consideraremos adicionalmente o desempenho no segundo ano frequentado, visando melhorar a qualidade da previsão da Duração Total do Curso (DTC) (Cenário 2).

Numa terceira fase, procuraremos testar versões alternativas destes dois cenários, como por exemplo a introdução de novas variáveis de *input*.

Por último, e considerando os resultados obtidos para a previsão de DTC, procuraremos prever a NFC.

Para efeitos de gestão universitária, a previsão da Duração Total do Curso em relação à Nota Final do Curso é de maior importância. Por este motivo, a abordagem utilizada atribui implicitamente maior relevância à variável DTC. No entanto, naturalmente se prevê a existência de uma dependência entre estas duas variáveis.

1.2 Redes Neuronais Artificiais

As redes neuronais artificias (RNA), são o resultado de uma tentativa de modelar as capacidades de processamento de informação do sistema nervoso central dos seres vivos. O sistema nervoso central, em particular o do ser humano, processa a informação de uma forma completamente diferente do convencional computador. O cérebro é uma estrutura

extremamente complexa, não linear e paralela. Tem a capacidade de organizar as suas componentes estruturais, designadas por neurónios, assim como executar tarefas complexas como reconhecimento de padrões, classificação e generalização de conceitos, de uma forma muito mais rápida e eficaz que o computador digital mais rápido disponível até hoje.

Apesar dos avanços significativos que têm ocorrido no estudo do sistema nervoso central existem ainda muitas questões por esclarecer acerca do mesmo. No entanto alguns factos importantes já são bem conhecidos. Hoje sabe-se que o cérebro humano tem uma estrutura conexionista, capaz de aprender com experiências passadas, e que vai evoluindo com o tempo. Para além disso tem uma estrutura fortemente paralela flexível que lhe permite adaptar-se ao ambiente envolvente. De uma forma generalista, as rede neuronais artificiais são sistemas que procuram modelar a forma como o cérebro desempenha uma determinada tarefa ou uma função de interesse. A rede é geralmente implementada recorrendo a componentes electrónicas ou é simulada em software num computador.

Uma RNA apresenta duas semelhanças importantes com o comportamento do cérebro humano [Haykin, 1999]:

- O conhecimento é adquirido a partir de um ambiente, através de um processo de aprendizagem;
- O conhecimento é armazenado nas conexões também designadas por ligações ou sinapses, entre neurónios.

Durante o processo de aprendizagem a força ou peso associado às conexões pode ser ajustado de forma a atingir um desejado objectivo ou estado de conhecimento da rede.

O estudo de redes neuronais artificias, desde o primeiro modelo do neurónios artificial apresentado em 1943 por McCulloch e Pitts, tem vindo a evoluir bastante com modelos cada vez mais sofisticados. As RNA's têm vindo a ser utilizadas cada vez mais na modelação de sistemas, reconhecimento de padrões, em problemas de classificação, na representação de funções booleanas ou contínua e na previsão de séries temporais, optimização.

No entanto o uso de RNA's também apresenta desvantagens, nomeadamente no tempo despendido na procura do melhor tipo de rede, arquitectura e processo de aprendizagem. Uma outra dificuldade diz respeito à necessidade de, em geral, ter de se proceder ao pré-tratamento dos dados, antes de os mesmos "alimentarem" a rede.

1.3 Objectivos e Expectativas

Nesta dissertação abordaremos as Redes Neuronais Artificiais, um tema não abordado na parte escolar do Mestrado. Optamos pelo software MATLAB, para a sua implementação, módulo de RNA, que também não havíamos utilizado anteriormente. Naturalmente, desejaríamos ser capazes de criar um modelo capaz de prever o desempenho global de um aluno universitário.

Assim, os principais objectivos desta dissertação visam:

- Fazer uma introdução teórica às Redes Neuronais Artificias;
- Fazer uma introdução à implementação prática das RNA's em MATLAB;
- Encontrar um modelo com base em RNA's que permita prever a DTC e NFC com base no seu desempenho académico nos primeiros anos frequentados;
- Analisar a melhor forma de aplicar as RNA's em termos de arquitectura da rede, algoritmo de aprendizagem e processo de alimentação da rede;
- Testar, analisar e comparar os diferentes cenários para a modelação do desempenho académico global;
- Sugerir novas abordagens a ser realizadas no futuro que possam dar continuidade a este trabalho.

1.4 Estrutura da Dissertação

Neste capítulo foi descrito de uma forma geral o contexto em que este trabalho surgiu, as motivações que levaram à escolha do tema e da área científica de investigação, assim como os objectivos que nos propomos atingir.

A dissertação será ainda constituída por mais três capítulos:

Capítulo 2 - Redes Neuronais Artificiais

Este capítulo visa essencialmente fazer uma abordagem teórica ao tema das Redes Neuronais Artificiais. É feita uma introdução histórica e a descrição do que é uma RNA e as suas propriedades. Para além disso é feita uma descrição resumida das diferentes topologias de RNA's e dos variados algoritmos de aprendizagem, dando-se ênfase às redes do tipo multi-camada com aprendizagem Retropropagação (RP) ("Back-Propagation").

Finalmente, são apresentadas as vantagens e desvantagens da utilização das RNA's, introduzindo-se ainda de uma forma resumida, alguns problemas mais populares de RNA como, por exemplo, o *overfitting*.

Capítulo 3 - Caso em Estudo

Este capítulo pode ser dividido essencialmente em três partes. Numa primeira parte faz-se a descrição e análise preliminar dos dados da amostra dos 297 alunos de uma Licenciatura em Engenharia (pré-Bolonha) da FCT-UNL com cinco anos de duração. Numa segunda parte é feita a descrição de toda a abordagem e implementação de RNA's em MATLAB. São descritas todos as experiências computacionais realizadas na procura da melhor RNA para o caso em estudo, ou seja, a sua arquitectura, algoritmo de aprendizagem e alimentação dos dados. Numa terceira parte, é feita a descrição dos diferentes cenários estudados, a análise das diferentes experiências computacionais realizadas para cada um deles e é feita a comparação dos resultados obtidos, assim como algumas reflexões sobre os mesmos.

Capítulo 4 - Conclusão

No último capítulo é feita uma síntese do trabalho realizado, são apresentadas as conclusões do trabalho desenvolvido assim como apontadas as direcções para trabalhos futuros.

Capítulo 2

Redes Neuronais Artificiais

2.1 Introdução

Este capítulo tem como principal objectivo fazer uma breve introdução aos principais conceitos e temas da teoria das Redes Neuronais Artificiais (RNA), em particular às redes multi-camada com Aprendizagem Retropropagação (RP).

As RNA's multi-camada são as mais utilizadas e conhecidas na teoria das RNA e a sua popularidade advém principalmente da sua implementação, bem sucedida, na resolução dos mais diversos problemas de processamento de informação, tais como problemas de classificação de padrões, aproximação de funções e previsão.

O algoritmo de RP é o mais conhecido algoritmo de treino de RNA's e foi originalmente desenvolvido por Paul Werbos em 1974. Desde então têm surgido diversas melhorias no sentido o tornar mais rápido e eficiente. Algumas melhorias do algoritmo RP, as suas vantagens e desvantagens na aplicação a RNA multi-camada serão brevemente descritas.

2.1.1 O que é uma Rede Neuronal Artificial

De forma generalizada, as RNA's são sistemas de processamento de informação, que tiveram a sua inspiração nas redes neuronais biológicas, que procuram simular, ou modelar, as suas capacidades cognitivas, nomeadamente a capacidade de aprender através de experiências passadas e de generalizar a informação aprendida.

As RNA's são compostas por pequenas unidades de processamento simples, que processam determinadas funções matemáticas. Estas unidades, geralmente designadas por neurónios, podem ser dispostas em uma ou várias camadas e estão interligadas entre si por várias conexões. As conexões têm pesos associados e servem para armazenar o conhecimento adquirido através do meio ambiente ao longo de um processo de aprendizagem. Durante o processo de aprendizagem, normalmente referido como algoritmo de

aprendizagem, é feito o ajustamento dos pesos de forma a atingir um determinado objectivo. O ajustamento dos pesos das conexões da rede é o método tradicional de construir uma RNA. No entanto, também é possível que a rede modifique a sua própria estrutura, à semelhança do que acontece com o próprio cérebro humano, que ao longo da sua vida pode perder neurónios ou ganhar novas sinapses.

Eis a definição de RNA adaptada de Aleksander e Monton [Haykin, 1999]:

> Uma RNA é um processador fortemente paralelo e distribuído, constituído por unidades de processamento simples (neurónios) que têm uma propensão natural para armazenar conhecimento experiencial e torná-lo disponível para o uso.

2.1.2 Redes Neuronais Artificias versus Algoritmos Tradicionais

As soluções computacionais tradicionais baseiam-se em regras, ou equações explicitamente programadas num algoritmo sequencial. Apesar de este método ser eficaz em situações onde as regras são conhecidas e bem definidas, existem muitas situações cujas regras desconhecemos ou, por outro lado, apesar de conhecidas, são difíceis de implementar. Por exemplo, qualquer ser humano é capaz de executar, de forma rápida e eficaz, o reconhecimento de letras. No entanto, esta tarefa não é tão fácil de reproduzir num algoritmo sequencial capaz de a desempenhar com a mesma eficácia e rapidez que um ser humano. É a este tipo de sistemas que podemos aplicar as RNA's.

A vantagem da abordagem tradicional é que, as regras num algoritmo podem ser formalmente apresentadas e demonstradas. Acrescentar ou alterar regras num sistema é um processo relativamente fácil, tendo em conta que as conhecemos. Isto permite, que estes sistemas sejam facilmente compreendidos e actualizados. A desvantagem nestes casos está relacionada com o trabalho árduo e muitas vezes impossível, de extrair regras de um sistema humano, ou outro qualquer específico, e construir um algoritmo.

Uma RNA é um modelo generalizado baseado na experiência adquirida de um determinado conjunto de dados e consequentemente não contêm regras. É consistente no sentido em que do mesmo padrão de entrada produz sempre a mesma resposta. As RNA's são úteis em situações em que a precisão das técnicas tradicionais é, por si, um impedimento. Aplicações que requerem conhecimento de padrões ou simulação de um sistema físico demasiado complexo para se modelar com regras são perfeitas para a utilização das técnicas das RNA´s. Aplicações em que são conhecidas exactamente as regras, ou em que cada caso é único e não é possível fazer generalizações, devem ser implementadas com técnicas tradicionais.

Apesar da sua falta de rigor e precisão, as RNA's são sistemas suficientemente eficientes e potentes para permitir construir aproximações quase perfeitas, de sistemas dos quais o nosso conhecimento seja insuficiente para desenvolver um programa.

2.1.3 Fundamentos Biológicos

Como já foi referido, as RNA's surgiram numa tentativa de modelar as capacidades de processamento de informação dos sistemas nervosos, em particular do cérebro humano. Por este motivo é essencial conhecer as principais características das redes neuronais biológicas do ponto de vista do processamento de informação.

O sistema nervoso é constituído por centenas de milhões de células interligadas entre si a que chamamos células nervosas ou, mais comummente, neurónios. Cada uma destas células é um sistema extremamente complexo que lida com os sinais que lhe são transmitidos de diferentes maneiras.

Os neurónios, quando comparados com portas lógicas electrónicas, são lentos, necessitam de cerca de cinco a seis vezes mais tempo para atingir um estímulo. Ainda assim, o cérebro humano é capaz executar tarefas e resolver problemas que nenhum computador digital actual é capaz de fazer. O cérebro compensa esta lentidão dos neurónios através da sua estrutura fortemente paralela. Tudo indica que esta característica é uma condição fundamental para a emergência de comportamento consciente e complexo. No entanto, os neurologistas e biólogos concentram o seu estudo e investigação essencialmente na descoberta e compreensão das propriedades dos neurónios individualmente. Apesar de actualmente já existir um conhecimento razoável acerca do funcionamento dos neurónios como unidades individuais de processamento e da ligação entre eles, pouco se sabe acerca do seu funcionamento em grupo.

As células nervosas ou neurónios, podem executar diferentes funções, o que leva a uma grande variedade de morfologias. Se analisarmos ao microscópio o córtex humano verificamos a existência de diferentes tipos de neurónios [Rojas, 1996].

Apesar dos diferentes tipos de neurónios, podemos generalizar a sua estrutura ilustrada na figura 2.1.

A estrutura de um neurónio pode ser definida em três secções principais: axónio, dentritos e corpo celular. Os dentritos são os canais pelo qual os impulsos nervosos são transmitidos na região de contacto entre células, as sinapses. A sua função é receber a informação de outros neurónios e transmiti-los ao corpo da célula. No corpo da célula ocorrem processos químicos que geram novos impulsos. Estes são transmitidos pelo axónio, que comunica com os dentritos do neurónio seguinte.

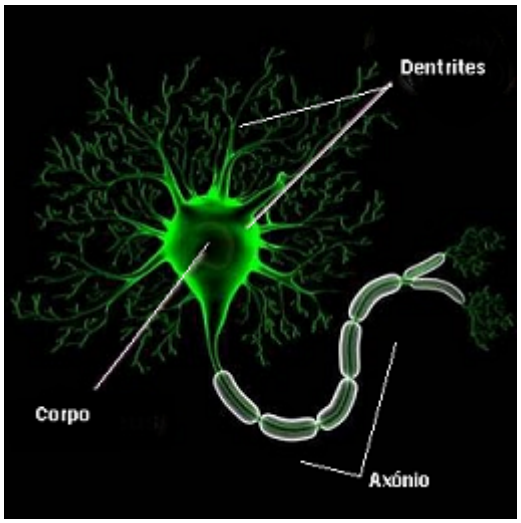


Figura 2.1: Neurónio Biológico
Adaptado de: http://sumulasepm.blogspot.com/

Os neurónios transmitem informação através de impulsos nervosos, de forma descontinuada, conhecidos por potenciais de acção. Os potenciais de acção não são mais do que acontecimentos eléctricos muito complexos. A membrana do axônio é constituída por canais iónicos que podem abrir e fechar de modo a controlar a passagem de iões com carga eléctrica, iões de sódio (Na^+) ou potássio (K^+). Quando os canais abrem, os iões (Na^+) ou (K^+) movimentam-se de acordo com gradientes eléctricos e químicos respectivamente para dentro ou para fora da célula.

A figura 2.2 demonstra a evolução do potencial de acção em função do tempo, num determinado ponto de um neurónio. Quando um estímulo é aplicado na membrana do axônio ocorre um desequilíbrio temporário entre as cargas eléctricas da membrana. Sempre que na membrana o potencial aumenta e ultrapassa um determinado valor, cerca de -50 mV, partindo de um potencial de repouso, formam-se potenciais de acção. O potencial de acção inicia-se no corpo celular com a abertura de canais de Na^+. Este acontecimento leva à entrada de iões sódio e ao rápido estabelecimento de um novo equilíbrio em poucos milissegundos. Num instante, o campo eléctrico entre os dois lados da membrana do neurónio altera em cerca de 100 mV. Muda de uma voltagem negativa no lado interior da célula (cerca de -70 mV) para um valor positivo (cerca de +30 mV), que se verifica na figura 2.2 na zona crescente do gráfico. Quase imediatamente depois abrem canais de K^+, que permitem a saída de potássio da célula, contribuindo deste modo para o restabelecimento do potencial de membrana de repouso, negativo no interior (zona decrescente do gráfico) [UNL FCT, 2000].

As fibras nervosas comportam-se como condutores e, assim, os potenciais de acção

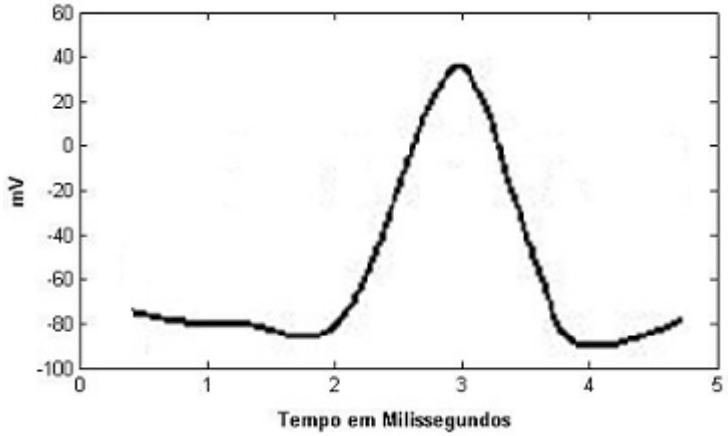


Figura 2.2: Potencial de acção de um neurónio

gerados num ponto criam gradientes de voltagem entre pontos adjacentes da membrana em estado de repouso ou em actividade. O potencial de acção é assim activamente propagado numa onda de despolarização que migra de uma ponta da fibra nervosa até ao outro extremo.

Os quatro elementos, axónio, corpo celular, dentrits e sinapses são a estrutura mínima adoptada para o modelo biológico. Os neurónios artificias são constituídos, à semelhança do neurónio biológico, por canais de entrada (*inputs*), um corpo celular e canais de saída (*output*). As sinapses são simuladas por conexões entre o corpo celular e *inputs*, ou *outputs*, e têm associadas pesos.

2.1.4 Um pouco de História

O estudo das RNA's foi introduzido com o trabalho do neurofisiologista Warren S. McCulloch e o matemático Walter H. Pitts, publicado em 1943, intitulado "*A Logical Calculus of Ideas Immanent in Nervous Activity*". Neste trabalho McCulloch e Pitts unem os seus conhecimentos neuro-biológicos e matemáticos, respectivamente, para propor um modelo do neurónio como uma unidade de processamento binário e demonstram que um número suficiente destas unidades é capaz de executar muitas operações lógicas. Apesar de simples este modelo foi um ponto de partida para muitos estudos que se vieram a realizar na área de redes neuronais e inteligência artificial.

A figura 2.3 apresenta alguns exemplos do neurónio de McCulloch-Pitts. Cada unidade é activa, se e só se, o seu total de excitação for igual, ou superior, a zero. Por exemplo, a primeira unidade é activa, se e só se, ambas as unidades x e y são activas, isto porque só assim o total de excitação, $(+1)x + (+1)y$, ultrapassa o limite -2 estabelecido pela unidade t.

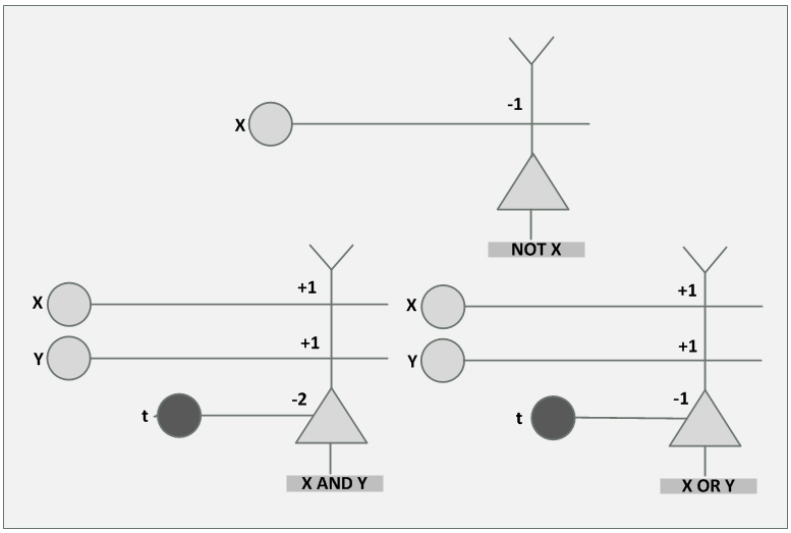


Figura 2.3: Exemplos de neurónios de McCulloch-Pitts
Adaptado de [Vemuri, 1999]

Em 1948 é publicado o livro "*Cybernetics*" de Norbert Wiener, primeiro *best-seller* na área da Inteligência Artificial, tornando-se o trabalho mais influente desenvolvido neste período. A palavra *cybernetics* foi originalmente introduzida por Wiener para descrever a importância do controle adaptativo e comunicação nos organismos vivos ou artificiais.

O próximo grande desenvolvimento em redes neuronais surge em 1949 com a publicação do livro "*The Organization of Behavior*" pelo psicólogo Donald Hebb. Hebb propõe que à medida que o cérebro vai aprendendo novas tarefas a sua conectividade vai sofrendo alterações e vão sendo criadas assembleias neuronais em consequência dessas alterações.

Em particular Hebb publica no seu livro uma das primeiras e mais famosas regras de aprendizagem para neurónios, onde afirma que a efectividade de uma sinapse é aumentada pela sua activação repetida. Segundo Hebb (1949, citado por Haykin, 1999):

> Quando um axónio de uma célula A está suficientemente perto para excitar a célula B e a dispara frequentemente e repetidamente, um processo de crescimento ou alterações metabólicas inicia-se em um ou ambas as células de forma a que a eficiência de A em disparar sobre B é aumentada.

Por outras palavras, se a célula A e B são simultaneamente activas então o peso da conexão entre ambas é aumentado.

Passados quase quinze anos da publicação do artigo de McCulloch e Pitts, Frank Rosenblatt do Laboratório Cornell Aeronáutica da Universidade de Cornell, em Ithaca, Nova Iorque, introduz uma nova abordagem para o problema de reconhecimento de padrão: o perceptrão.

A figura 2.4 ilustra um diagrama do perceptrão de Rosenblatt. O modelo clássico do perceptrão proposto por Rosenblatt é na realidade uma rede cuja topologia originalmente consistia em três níveis: as unidades sensoriais (retina), onde os estímulos ópticos são recebidos, unidades associativas, onde os pesos são fixos, e as unidades de resposta, onde ocorre o treino e é gerada uma resposta. Embora esta topologia seja constituída por três níveis, ela é originalmente conhecida por perceptrão de uma única camada, uma vez que só o nível de unidades de resposta é que possui propriedades adaptativas.

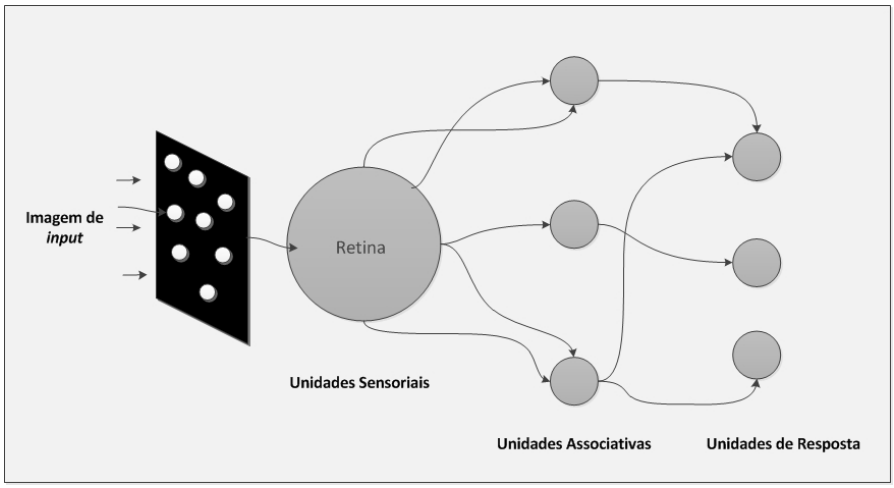


Figura 2.4: O modelo clássico do perceptrão de *Rosenblatt*
Adaptado de [Rojas, 1996]

O algoritmo de aprendizagem do perceptrão é apresentado na figura 2.5. O objectivo é determinar os pesos $W = \langle W_0, W_1, ..., W_p \rangle$ para um conjunto de padrões $E = \left\{ \langle E^1, S^1 \rangle, \langle E^2, S^2 \rangle, ..., \langle E^N, S^N \rangle \right\}$, onde E^k é um vector de dimensão p de entradas de números reais e S^k é o vector correspondente da saída desejada, que toma valores no intervalo $[-1, +1]$.

Do ponto de vista formal, a única diferença significativa entre o neurónio de McCulloch-Pitts e o perceptrão de Rosenblatt é a presença de pesos na rede.

Na mesma altura em que Rosenblatt desenvolvia o seu trabalho à volta do perceptrão, Widrow e Hoff desenvolveram uma importante variante do algoritmo de aprendizagem do perceptrão. O algoritmo Widrow-Hoff ou LMS (*Least Mean Square Algorithm*) é um algoritmo de gradiente descendente, com uma taxa ajustável de aprendizagem, rápido e conveniente para minimizar o MSE (*mean square error*). Widrow e Hoff usaram o algoritmo LMS para treinar os pesos de um modelo de célula única discreta e chamaram ao resultado ADALINE (*Adaptive Linear Element*). As vantagens do ADALINE face ao perceptrão apresentado por Rosenblatt residem essencialmente no facto de ser mais rápido, permitir que as saídas correctas sejam valores reais, em vez de estarem limitadas

0. Inicializar o vector W a 0

1. Seleccionar um padrão de treino $\left(E^K, S^K\right)$.
   A selecção dos padrões pode ser feita em ciclo (ordenado)
   ou seleccionando um aleatoriamente.

2. Se W classificar correctamente E^K, isto é, se:

   $W \cdot E^K > 0$ e $S^K = +1$ ou

   $W \cdot E^K < 0$ e $S^K = -1$

   Então:

   Não fazer nada

   Caso contrário:

   Ajustar os pesos W adicionando ou subtraindo E^K de acordo com a respectiva saída S^K seja +1 ou -1:

   $$W = W + S^K \cdot E^K$$

3. Voltar ao passo número 1.

Figura 2.5: Algoritmo de aprendizagem do *perceptrão*
Adaptado de [Rojas, 1996]

ao conjunto $[-1, +1]$, e muito importante, sendo a função do erro, utilizada no algoritmo, derivável, permite aplicação de métodos de gradiente descendente.

Durante os anos sessenta foram desenvolvidas muitas investigações sobre perceptrão que levavam a crer que as RNA's seriam capazes de realizar qualquer tarefa. A publicação do livro "*Perceptrons*" de Marvin Minsky e Seymour Papert em 1969 levou a que as RNA's perdessem a sua popularidade e a que a pesquisa nesta área permanecesse estagnada por muitos anos. Minsky e Papert usaram conceitos matemáticos modernos para demonstrar que existia limites fundamentais na aplicação de perceptrão de uma camada, demonstraram que o perceptrão apesar de ser capaz de executar operações booleanas AND ou OR não eram capazes de realizar operações simples como a função lógica XOR (OU-Exclusivo). Numa breve secção sobre perceptrão multi-camadas Minsky e Papert afirmam que não existe nenhuma razão lógica plausível para acreditar que as limitações encontradas no perceptrão de uma camada seriam ultrapassadas nos perceptrões multi-camada.

Apenas na década de oitenta começaram novamente a surgir grandes contributos para a teoria das redes neuronais. Em 1982 John J. Hopfield propõe um modelo associativo de redes neuronais. Este modelo, baseado em pesos fixos e activação adaptativa, causa grande impacto na comunidade da Inteligência Artificial. Hopfield usou a ideia de uma

função de energia para formular uma nova forma de compreender o processamento de uma rede recorrente com ligações sinápticas simétricas. Este tipo de redes recorrentes com *feedback* atraiu uma grande atenção nos anos oitenta e, com o decorrer do tempo, ficaram conhecidas como redes de Hopfield. O trabalho de Hopfield atraiu principalmente matemáticos e engenheiros para a pesquisa nesta área e as redes de Hopfield foram estudadas como sistemas de memória distribuídos e também utilizados como ferramentas na solução de problemas de optimização restrita.

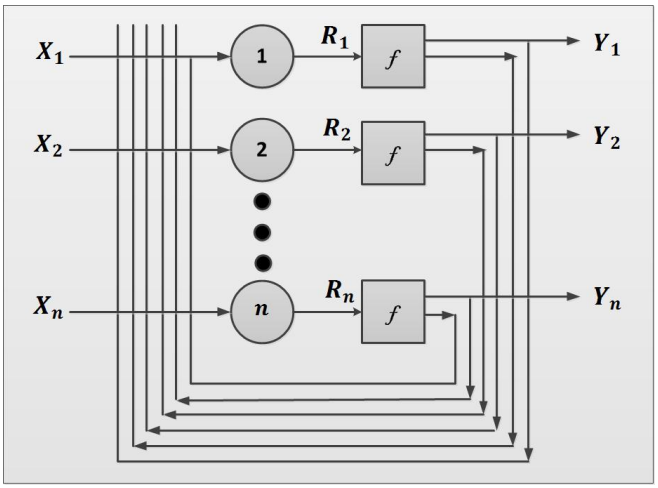


Figura 2.6: Paradigma de Hopfield
Adaptado de [Kartalopoulos, 1996]

Em 1986 Rumelhart, Hinton e Williams divulgam o desenvolvimento do algoritmo Retro-Propagação, que surgiu como o mais popular algoritmo de aprendizagem. Nesse mesmo ano é publicado o livro "*Parallel Distributed Processing: Exploration in Microstrutures of Cognition*" de Rumelhart e McClelland que tem sido uma grande influência no uso do algoritmo de Retro-Propagação em perceptrões de multi-camada. Na realidade este algoritmo foi descoberto simultaneamente em dois locais diferentes por Parker (1985) e Lecun (1985) no entanto este já tinha sido proposto anteriormente por Werbos na sua dissertação de doutoramento na universidade de Harvard em Agosto de 1974. A dissertação de Werbos foi o primeiro documento escrito com a descrição da eficiência do método de retropropagação do gradiente aplicado a modelos de redes generalizados onde as redes neuronais artificiais surgem como um caso particular. Apesar de ter sido originalmente descrito por Werbos muito do crédito do algoritmo de Retro-Propagação é devido a Rumelhart, Hinton e William (1986) por terem proposto o seu uso em *machine learning* e terem demonstrado como poderia funcionar.

A partir de 1986 são vários e diversificados os contributos de investigadores talentosos na área de RNA's. Em 1988, Broomhead e Lowe apresentam uma alternativa às redes

de perceptrão multi-camada, as funções de base radial.

No inicio dos anos noventa, Vapnike e os seus colegas de trabalho inventaram uma classe de redes com aprendizagem supervisionada muito potentes designadas por *Suport Vector Machines* para a resolução de problemas de reconhecimento de padrões, regressão e estimativa de densidade.

As Redes Neuronais Artificias percorreram um grande caminho desde as primeiras publicações de McCullock e Pitts e, apesar de a sua origem estar muito ligada à área de neurobiologia, ao longo dos anos as RNA's tem-se estabelecido como interdisciplinares nas áreas de psicologia, matemática, física e engenharia.

Embora teoricamente as RNA's possam ser utilizadas em qualquer sistema, na prática têm sido utilizadas em situações particulares onde a dimensão da amostra é elevada e onde os métodos convencionais não são possíveis aplicar por se tratar de sistemas demasiado complexos para serem reproduzidos por meio de regras ou algoritmos. Nos dias de hoje procuram-se, não só, redes mais eficientes como também algoritmos mais rápidos e que exijam menos recursos computacionais. A aplicação da teoria de RNA's é cada vez mais diversificada e generalizada em áreas como a Economia, Robótica, Sistemas Autónomos e Estatística.

2.2 Modelo de um Neurónio

2.2.1 Modelo do Neurónio Artificial

As redes neuronais artificiais são constituídas por pequenas unidades processadoras, normalmente conhecidas por neurónios. O neurónio ou neurónio artificial (como por vezes é referido evidenciando as sua diferenças face ao neurónio biológico) é o elemento chave de uma RNA.

O primeiro modelo de um neurónio artificial foi proposto por McCulloch e Pitts em 1943 e surgiu como uma tentativa de reproduzir, de uma forma simplificada, o funcionamento do neurónio biológico. No entanto, as diferenças entre o neurónio biológico e o neurónio artificial são significativas, estando este último longe de representar o real funcionamento do primeiro. Ao longo do tempo este modelo tem sofrido sucessivas melhorias dando origem a novos modelos, mais sofisticados e eficientes.

O modelo do neurónio generalizado consiste num conjunto de n entradas (*inputs*), $X_1, X_2, ..., X_n$, que representam as sinapses, e uma saída Y (*output*). Cada *input* X_i é ponderado, antes de chegar ao corpo principal do elemento de processamento (neurónio), por um factor de peso W_i. À semelhança do neurónio biológico, este peso W_i, pode ser

positivo ou negativo, dependendo das suas capacidades inibidoras ou excitantes.

Um neurónio biológico reproduz um sinal quando a soma dos seus impulsos ultrapassa um determinado potencial de activação. No neurónio artificial o corpo do neurónio é reproduzido por um mecanismo simples que corresponde à soma das entradas associadas aos respectivos pesos, ou seja, a soma dos valores $X_i \times W_i$ (combinação linear). Esta soma é comparada com um limite θ (*threshold*) que quando ultrapassado produz um sinal R. A este sinal R é aplicado numa função F designada por função de activação, reproduzindo o *output* Y. No modelo original de McCulloch e Pitts a activação do neurónio é obtida pela aplicação de uma função de activação limiar, ou seja, se $\sum_i^n X_i \times W_i > \theta$ o neurónio produz um sinal de saída igual a um, caso contrário, a saída é igual a zero (figura 2.7).

No modelo do neurónio artificial generalizado os valores de saída podem ser diferentes de zero ou um e ter funções de activação diferentes da limiar. A função de activação é também conhecida por *squashing function* uma vez que limita (*squashes*) a amplitude do intervalo de valores permitidos para a saída. Normalmente este intervalo de amplitude é $[0, 1]$ ou em alternativa $[-1, +1]$.

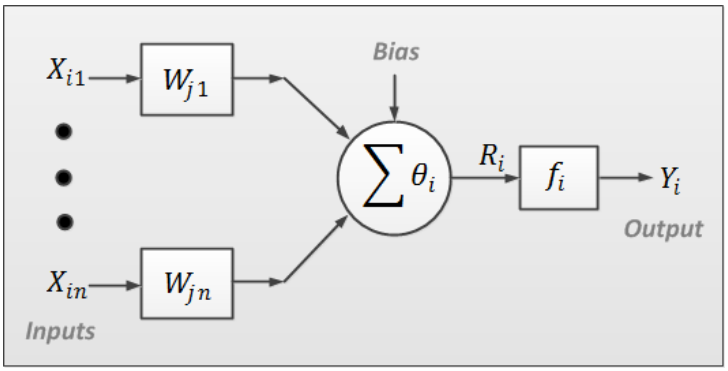


Figura 2.7: Neurónio Artificial.
Adaptado de [Kartalopoulos, 1996]

O modelo referido na figura 2.7 inclui também o *bias* B. O *bias* pode ser visto como um peso que tem entrada constante igual a um e que nos permite introduzir pequenas transformações nos dados de entrada, aumentando ou diminuindo o efeito causado pelos mesmos.

Em termos matemáticos podemos descrever um neurónio i de uma rede neuronal artificial da seguinte forma [Kartalopoulos, 1996]:

$$Y_i = F_i\left(\sum_k^n X_{ik} \times W_{jk}\right) \qquad \text{com} \quad \sum_k^n X_{ik} \times W_{jk} \geq \theta_i$$

onde $X_{i1}, X_{i2}, ..., X_{in}$ são as entradas (*inputs*), $W_{j1}, W_{j2}, ..., W_{jn}$ são os pesos associados às respectivas entradas, $B_1, B_2, ..., B_n$ o bias e F_i a função activação. Y_i é a saída (*output*) do neurónio i. A figura 2.8 exemplifica o funcionamento de um neurónio artificial.

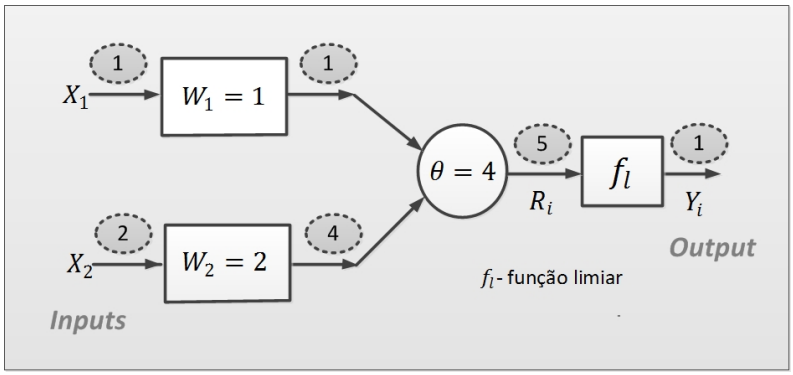


Figura 2.8: Exemplificação do funcionamento de um neurónio artificial.
Adaptado de [Kartalopoulos, 1996]

2.2.2 Função de Activação

O objectivo da função de activação $F(x)$ é assegurar que a resposta do neurónio seja limitada, isto é, que a resposta do neurónio como resultado de um pequeno ou grande estímulo seja condicionada, ou amortecida, e portanto controlável.

No mundo biológico, o condicionamento dos estímulos é feito continuamente por todas as entradas sensoriais. Por exemplo, para atingir um som duas vezes mais alto é necessário um aumento real na amplitude do som de cerca de dez vezes, esta é a resposta algorítmica do ouvido. Os neurónios artificias condicionam as suas saídas de uma maneira semelhante, o que torna este conceito consistente com o neurónio biológico, no entanto a função de activação não linear está muito longe de ser uma réplica da do neurónio biológico, sendo muitas vezes usada por conveniência matemática [Kartalopoulos, 1996].

Existem diferentes funções de activação não lineares que são usadas dependendo do paradigma e do algoritmo usado. Geralmente a função de activação é uma função monótona não decrescente. As funções de activação mais usuais são: Função Linear, Função Rampa, Função Degrau e Função Sigmóide.

Função Linear

A função de activação Linear (*linear function*) é definida pela seguinte equação:

$$F(x) = \alpha x, \text{ onde } \alpha \text{ é um valor real.}$$

A função linear é muito utilizada na camada de saída das redes multi-camada.

Função Rampa

A função linear pode ser restringida para reproduzir valores constantes em determinados intervalos, passando a designar-se Função Rampa ou Linear por partes (*piecewise-linear function*):

$$F(x) = \begin{cases} +\epsilon & se \quad x \leq \epsilon \\ x & se \quad x = \epsilon \\ -\epsilon & se \quad x \geq -\epsilon \end{cases}$$

Função Degrau

A função degrau (*step function*) é semelhante à função rampa:

$$F(x) = \begin{cases} +\epsilon & se \quad x > 0 \\ -\epsilon & se \quad x \leq 0 \end{cases}$$

É uma função não monótona (tem uma descontinuidade na origem) e assim não é facilmente diferenciável, no entanto é linear entre os seus limites, superior e inferior. Portanto, enquanto o neurónio operar dentro dos seus limites a sua derivada é constante: $F'(x) = k$.

Função Sigmóide

Uma das funções de activação mais populares é a função sigmóide (*sigmoid function*), também conhecida por função *S-Shape*. A função sigmoide é uma função real $\mathbb{R} \rightarrow (0, 1)$ definida pela seguinte expressão:

$$F(x) = \frac{1}{1 + e^{-bx}}$$

A constante b pode ser seleccionada arbitrariamente e o seu inverso $1/b$ é normalmente conhecido como parâmetro de temperatura em redes neuronais estocásticas. O formato da função sigmóide altera em função do valor de b, como podemos observar na figura 2.9. Valores mais elevados de b aproximam a forma da sigmóide à da função degrau.

A função sigmóide é muito popular essencialmente por ser uma função monótona (estritamente crescente), limitada e por ter uma derivada simples, o que facilita a sua

aplicação em algoritmos de aprendizagem, como é o caso do algoritmo retropropagação.

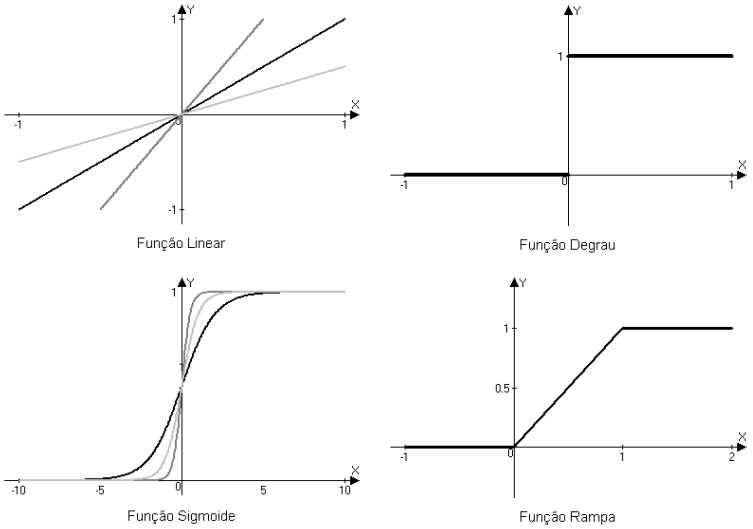


Figura 2.9: Funções de Activação

2.3 Arquitectura da RNA

A estrutura da RNA, ou seja a forma como os neurónios estão interligados, designa-se por arquitectura ou topologia da rede.

Existem vários tipos de arquitectura de RNA que estão classificadas geralmente em três categorias principais [Haykin, 1999]: Redes *Feedforward* de uma camada, Redes *Feedforward* Multi-Camada e Redes Recorrentes.

2.3.1 Redes *Feedforward* de uma camada

Numa rede de camadas os neurónios estão organizados por camadas. Numa rede com apenas uma camada existe uma camada de neurónios de entrada (*input layer*), cuja função é receber os dados de entrada externos, e por uma camada de neurónios de saída (*output layer*), que devolve a resposta final da rede.

Numa rede *feedforward* o fluxo de informação é unidireccional, ou seja, a informação de cada camada não constitui informação de entrada de nenhuma camada anterior (alimentação directa), não existindo ciclos.

A camada de entrada não é contabilizada uma vez que não se efectuam nenhuns cálculos nesta camada.

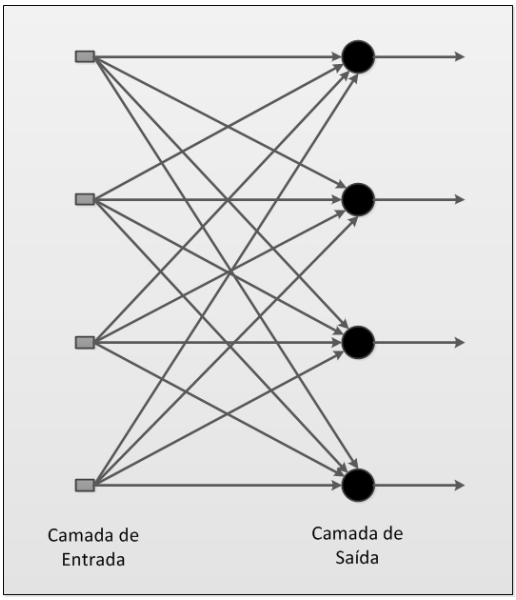


Figura 2.10: Rede *Feedforward* com uma camada de neurónios.
Adaptado de [Haykin, 1999]

2.3.2 Redes *Feedforward* Multi-camada

As redes *feedforward* multi-camada são caracterizadas pelo facto de serem constituídas por uma, ou várias, camadas intermédias, ou escondidas (*hidden layer*), entre as camadas de entrada e saída. A função das camadas escondidas é extrair recursos úteis dos padrões de entrada fornecidos à rede e que serão utilizados para prever os valores da camada de saída.

Ao introduzir camadas escondidas na rede, está-se a aumentar a sua capacidade de simular funções com maior complexidade, o que é particularmente útil quando o número de neurónios da camada de entrada é elevado. Por outro lado o aumento do número de camada aumenta exponencialmente o tempo de aprendizagem da rede.

O valor gerado por cada um dos neurónios é normalmente referido como a sua activação e mede o grau com que influencia os neurónios seguintes. A activação dos neurónios de entrada é despoletada pelos padrões de entrada fornecidos à rede e a activação dos neurónios de saída é descodificada para fornecer a resposta final.

Os neurónios da camada de entrada da rede fornecem os respectivos padrões de activação que constituíram as entradas da segunda camada, ou seja, da primeira camada escondida. Os valores de saída da segunda camada são utilizados como entradas para a terceira camada (segunda camada escondida) e assim sucessivamente ao longo da rede. Antes de passar para os neurónios da camada seguinte as entradas/saídas são "pesadas", somados e aplicadas numa função activação, que limita o intervalo de valores

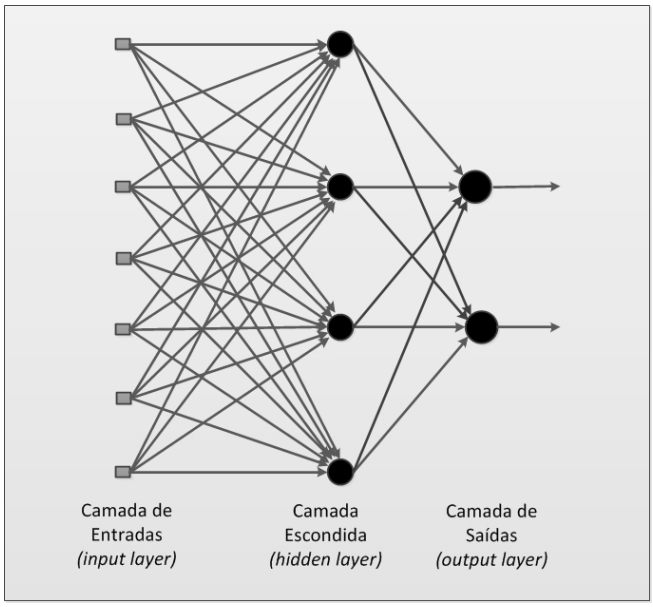


Figura 2.11: Rede *Feedforward* com uma camada escondida de neurónios.
Adaptado de [Haykin, 1999]

de saída. O conjunto de elementos de saída da rede na camada de saídas final constitui a resposta da rede para o conjunto de padrões de activação fornecidos pelos neurónios de entrada da camada de *input*.

De referir que a definição de uma topologia ideal, ou seja do numero ideal de neurónios em cada camada escondida bem como o número de camadas escondidas, é muitas vezes uma tarefa complicada, obtida muitas vezes através de métodos empíricos que podem levar a um tempo de dimensionamento alto.

2.3.3 Redes Recorrentes

As redes recorrentes distinguem-se das redes *feedforward* por terem pelo menos um caso de retro-alimentação (*feedback loop*), ou seja, um caso em que o *output* de um neurónio influência de alguma forma o *input* desse mesmo neurónio (*self-feedback loop*) ou a entrada de um neurónio de uma camada não consecutiva, criando conexões cíclicas na rede. A sua estrutura pode ou não ser por camadas.

A figura 2.12 ilustra um exemplo de uma rede neuronal artificial recorrente em que os *outputs* gerados por uma rede com apenas uma camada voltam para trás como *inputs* dessa mesma camada. Neste exemplo não existem situações de *self-feedback loop*.

A presença de *feedback loop's* tem um efeito profundo na capacidade de aprendi-

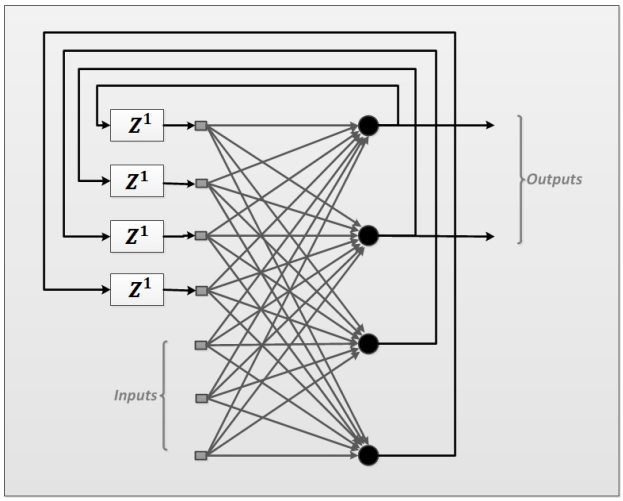


Figura 2.12: Rede Recorrente.
Adaptado de [Haykin, 1999]

zagem de rede e na sua performance [Haykin, 1999]. Nas redes recorrentes os *output* não são funções exclusivas das conexões entre neurónios mas também de uma dimensão temporal, isto é, estamos na presença de um cálculo recursivo, que obedecerá a um critério de paragem com a última iteração a ser dada como *output* para o neurónio [Domany et al., 1996].

2.4 Aprendizagem

Importa saber como se processa a aprendizagem numa RNA. Aprendizagem, num sentido estatístico, refere-se a qualquer processo que a partir de um conjunto de dados procura ajustar um conjunto de parâmetros que descrevem um modelo estatístico desses dados. O processo de aprendizagem não implica quaisquer qualidades humanas como compreensão, consciência ou inteligência associadas às nossas capacidades de aprendizagem.

Como já foi mencionado, a principal vantagem das RNA's é a sua capacidade de aprendizagem e de melhorar o seu desempenho ao longo da aprendizagem. A aprendizagem nas RNA's consiste num processo iterativo de ajuste dos pesos associados às conexões da rede aprendendo assim a partir do seu meio ambiente. Idealmente a rede torna-se mais experiente a cada iteração do processo de aprendizagem. Existem muitas formas diferentes para determinar o conjunto de pesos apropriado a cada situação e geralmente existe mais do que um conjunto de pesos apropriado.

De acordo com [Haykin, 1999], a definição do processo de aprendizagem implica a seguinte sequência de eventos:

1. A RNA é estimulada pelo ambiente envolvente;
2. A RNA sofre alterações nos pesos em consequência desse estímulo;
3. A RNA responde de uma nova forma ao ambiente em consequência às alterações que terão ocorrido na sua estrutura interna.

O ajuste de pesos é feito através de um conjunto de regras bem definidas a que se chama algoritmo de aprendizagem. Existe uma grande variedade de algoritmos de aprendizagem que diferem entre si essencialmente no método de ajuste dos pesos.

Numa RNA são importantes dois conceitos:

- Paradigmas de Aprendizagem: modelos do ambiente em que a rede opera;
- Regras de Aprendizagem: método de ajuste dos pesos, de forma a resolver determinado problema.

2.4.1 Paradigmas de Aprendizagem

O paradigma de aprendizagem pode ser essencialmente de dois tipos: aprendizagem supervisionada e aprendizagem não supervisionada. Outro paradigma bastante importante é o da aprendizagem por reforço, que pode ser considerado como um caso particular da aprendizagem supervisionada, assim como o da aprendizagem por competição pode ser considerado um caso particular da aprendizagem não supervisionada [Rojas, 1996].

Aprendizagem Supervisionada

A aprendizagem supervisionada, também conhecida como aprendizagem com “professor” (*learning with a teacher*), é a mais comum no treino de RNA’s. Designa-se por supervisionada precisamente porque neste método é fornecida à rede um conjunto de respostas correctas, ou desejadas, através de um supervisor (ou professor). Cada padrão fornecido à rede é constituído por um par de vector de entrada e vector de saída desejada (*target*). O objectivo é ajustar os pesos da rede de forma a encontrar uma relação entre os pares de entrada-saída fornecidos. A aprendizagem é feita comparando o valor actual de saída da rede com o valor da saída desejada.

A figura 2.13 ilustra um diagrama da aprendizagem supervisionada. O supervisor ou professor fornece o valor correcto da saída à rede, com o objectivo de direccionar o processo de treino da rede. A rede produz uma saída que é comparada com a saída desejada resultando desta comparação um erro. Os pesos das conexões da rede são então ajustados de forma a minimizar este erro. Este processo de ajuste dos pesos é realizado de uma forma iterativa com o objectivo de, eventualmente, tornar a rede capaz de simular o "professor", ou seja, de dar respostas correctas para novos vectores de entrada.

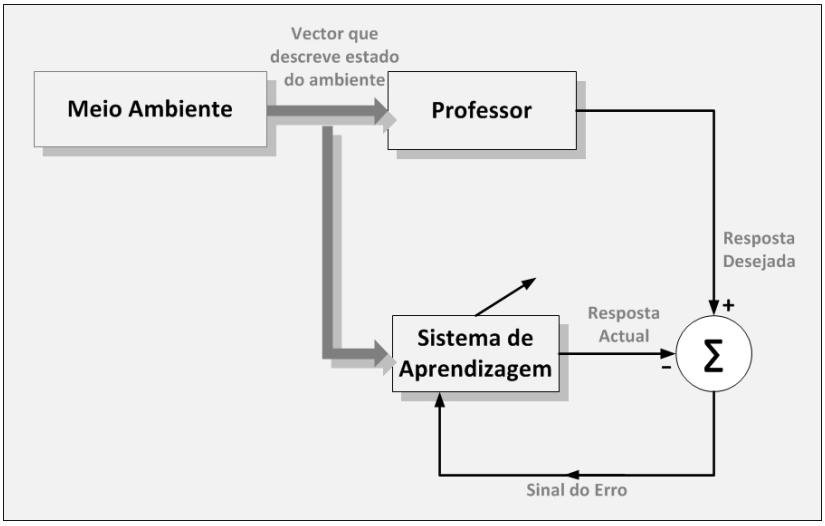


Figura 2.13: Diagrama da Aprendizagem Supervisionada.
Adaptado de [Haykin, 1999]

O método de aprendizagem supervisionada que acabamos de descrever é o método de aprendizagem por Correcção do Erro. Podemos utilizar várias medidas de desempenho da rede como por exemplo o Erro Quadrático Médio (*mean-square error*), ou a soma dos Erros Quadráticos, definidos como uma função dos pesos a ajustar (função erro ou custo da rede). Esta função pode ser interpretada como uma superfície de desempenho do erro multidimensional com os pesos a ajustar como coordenadas. Toda a operação devolvida pela rede sob supervisão do professor, é representada como um ponto sobre esta superfície. Para o sistema melhorar o seu desempenho ao longo do tempo e consequentemente aprender com o "professor" o ponto resultante de cada operação da rede tem de descer sucessivamente em direcção ao mínimo local ou global. Uma rede com aprendizagem supervisionada é capaz de fazer isto com a informação útil que obtém sobre o gradiente da superfície de erro correspondente ao comportamento actual do sistema. O gradiente de uma superfície de erro em qualquer ponto é um vector que aponta na direcção da maior descida.

Uma das desvantagens da aprendizagem supervisionada reside no facto de na ausência de respostas correctas (professor) a rede não consegue aprender novas estratégias para situações que não estejam contempladas nos padrões fornecido à rede para o seu treino.

São vários os algoritmos de aprendizagem supervisionada, como por exemplo a Regra de Delta de Widrow e Hoff (1960), ou a sua generalização para redes multi-camadas, o algoritmo de Retropropagação, que será aprofundado na secção 2.5.

Aprendizagem por Reforço

A aprendizagem por reforço pode ser considerada como um caso particular da aprendizagem supervisionada. A principal diferença entre a aprendizagem supervisionada e por reforço reside no facto de que, na primeira, a medida de desempenho da rede é baseada num conjunto de respostas correctas, ou desejadas, usando um critério de erro conhecido, enquanto na segunda, o desempenho é baseado em qualquer medida que possa ser fornecida à rede. Na aprendizagem por Reforço à semelhança da aprendizagem supervisionada também se admite a presença de um "professor" no entanto não é fornecida a resposta correcta à rede, apenas a informação se uma determinada saída gerada pela rede está correcta ou não. Assim, o erro gerado durante o processo de aprendizagem é binário.

A figura 2.14 ilustra a aprendizagem por reforço. O método consiste na aprendizagem por tentativa de erro, de modo a optimizar um índice de performance chamado sinal de reforço, ou seja, é dado um prémio aos pesos que dão a resposta desejada e uma penalização aos que dão a resposta errada. Durante o processo de aprendizagem não há qualquer indicação sobre se a resposta da rede está a mover-se na direcção correcta ou a que distância se encontra da mesma.

Alguns dos parâmetros que devem ser observados durante este método de aprendizagem são: o tempo por iteração e o numero de iterações por padrão para atingir o valor de saída pretendido durante o processo de treino, se o neurónio atinge um mínimo local ou global e, quando atinge um mínimo local, se consegue ou não sair desse mínimo.

Quando se utiliza este tipo de aprendizagem devem ser estabelecidas algumas medidas de paragem suplementares, para que a fase de treino não dure tempo infinito.

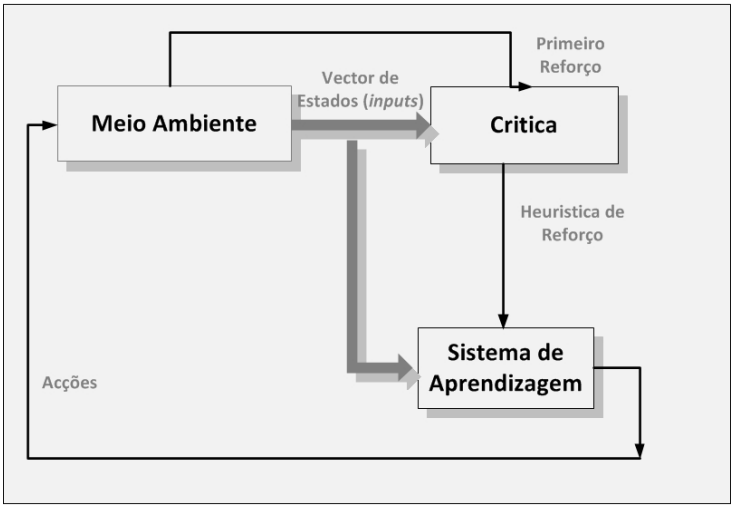


Figura 2.14: Diagrama de aprendizagem por reforço.
Adaptado de [Haykin, 1999]

Aprendizagem Não Supervisionada

Em contraste com aprendizagem supervisionada, a aprendizagem não supervisionada não requer a presença de um "professor" externo, ou seja, não existe qualquer resposta correcta ou desejada fornecida à rede. A aprendizagem é feita a partir de uma medida da qualidade da representação da tarefa que a rede deve aprender e os pesos são ajustados em função da optimização dessa medida. Uma vez a rede adaptada às características estatísticas dos dados de entrada e às suas irregularidades, desenvolve a capacidade de organizar os dados em categorias automaticamente. Cada vez que é apresentado à rede um padrão de entrada a rede produz uma resposta de saída indicando a que categoria pertence essa entrada. Sempre que a rede não é capaz de atribuir uma categoria já existente cria uma nova.

A figura 2.15 ilustra o método da aprendizagem não supervisionada.

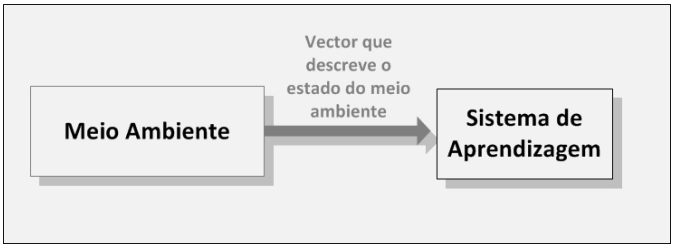


Figura 2.15: Diagrama de aprendizagem não supervisionada.
Adaptado de [Haykin, 1999]

Apesar de na aprendizagem não supervisionada não existir um "professor" é necessário dar à rede algumas orientações para determinar como irá formar os grupos. O agru-

pamento pode ser baseado em características dos dados de entrada, como por exemplo, a sua cor, tamanho ou peso, mas se não forem dadas nenhumas orientações sobre que tipo de recursos devem ser usados para criar as categorias, a classificação dos dados de entrada pode ser ou não bem sucedida. Em algumas experiências o critério de selecção é incluído na estrutura da rede neuronal.

Uma das desvantagens no uso da aprendizagem não supervisionada reside no facto da sua aplicação só ser possível quando existe uma quantidade considerável de dados, caso contrário, torna-se difícil, ou mesmo impossível identificar quaisquer características nos dados de entrada.

Aprendizagem por Competição

Um caso particular de redes com aprendizagem não supervisionada é a de aprendizagem por competição. Neste tipo de aprendizagem, o objectivo é dividir os dados de entrada em categorias, para que, informações similares sejam classificadas na mesma categoria e por conseguinte activem a mesma saída. Ou seja, quando um determinado estímulo de entrada é introduzido na rede, os neurónios da rede competem entre si, de acordo com uma regra de aprendizagem, para produzir a saída mais próxima da saída correcta e terem os seus pesos actualizados. O neurónio vencedor (*winner-take-all*), o mais forte, torna-se dominante deixando todos os outros neurónios inactivos. Para outro estímulo, outra saída torna-se dominante e por aí adiante. Assim cada saída é treinada para responder a diferentes estímulos de entrada.

O facto de neste tipo de aprendizagem haver apenas um neurónio “vencedor” ou dominante torna-a ideal para o reconhecimento de padrões. A cada neurónio associamos uma possível classe e identificamos o neurónio “vencedor”, com um determinado conjunto de entradas, como a classe reconhecida pela rede para aquele conjunto de entradas.

As redes SOM (*Self Organizing Maps*), dos mapas de Kohonen de Grossberg têm como base a aprendizagem por competição [Horst, 1996].

2.4.2 Regras de Aprendizagem

De acordo com [Haykin, 1999], as regras de aprendizagem dividem-se em cinco tipos básicos, que descrevemos adiante.

Aprendizagem de Hebbian

O postulado de aprendizagem de Hebb, já mencionado anteriormente, é dos mais antigos e famosos de todas as regras de aprendizagem. Segundo Hebb citado por [Haykin, 1999]:

> Quando um axónio de uma célula A está suficientemente perto para excitar a célula B e a dispara frequente e repetidamente, um processo de crescimento ou alterações metabólicas inicia-se, em uma ou ambas as células, de forma a que a eficiência de A em disparar sobre B é aumentada.

Este postulado, que surgiu num contexto neurobiológico foi expandido e repartido em duas regras:

- Se dois neurónios em cada lado de uma sinapse (conexão) são activados simultaneamente então a força dessa sinapse é aumentada;
- Se dois neurónios em cada lado de uma sinapse são activados de forma não sincronizada então essa sinapse é progressivamente enfraquecida ou eliminada.

Apesar de serem fornecidos à rede os pares entrada-saída (à semelhança do que acontece na aprendizagem supervisionada), a Regra de Hebb é classificada como aprendizagem não supervisionada uma vez que não existe um professor externo para verificar a qualidade da resposta da rede e orientar no ajuste dos pesos. Neste caso, a aprendizagem da rede é feita independentemente da reposta actual através de um mecanismo local à sinapse.

Aprendizagem Baseada na Memória

Na aprendizagem baseada na memória a totalidade, ou a maioria, da experiência passada é explicitamente armazenada numa grande memória de exemplos de pares de entrada-saída correctamente classificados. Sempre que surge um novo vector X_i de entradas nunca antes apresentado à rede, a rede responde procurando um vector na sua região vizinha. Todos os algoritmos de aprendizagem baseados na memória têm duas componentes essenciais:

- Um critério usado para a definição da vizinhança local do vector X_i;
- Uma regra de aprendizagem aplicada aos exemplos de treino na vizinhança local do vector X_i.

Os algoritmos diferem entre si na forma em como são definidas estas duas componentes. Um exemplo deste tipo de aprendizagem são as redes com funções de base radial [Haykin, 1999].

Regra Gradiente Descendente

Como já foi referido anteriormente na aprendizagem supervisionada, o objectivo desta regra é reduzir o erro ϵ resultante da comparação entre a saída da rede e a saída esperada. O erro funciona como medida de performance da rede, sendo que o ajuste dos pesos vão idealmente originando em cada iteração melhores reposta. O objectivo é minimizar uma função de custo (ou função erro) ξ definida em termos do sinal do erro ϵ . Neste processo de minimização de uma função custo, originalmente referido por Windrow e Hoff em 1960, conhecido por Regra Delta ou Windrow-Hoff, o ajuste dos pesos das conexões da rede é definido pela seguinte equação:

$$\Delta\omega = \gamma\nabla\xi, \quad \mu \in \mathbb{R}^+, \tag{2.1}$$

onde γ é uma constante positiva designada por taxa de aprendizagem, que determina o tamanho do "passo" de avançado a cada iteração do processo de aprendizagem e $\nabla\xi$ o gradiente da função de custo. A generalização desta regra a redes multi-camadas é designado por regras de aprendizagem por Retropropagação e será aprofundado na secção 2.5.

Regra de Boltzmann

A Regra de Boltzmann, nomeada em nome de Ludwing Boltzmann, é um algoritmo de aprendizagem estocástica derivado de ideias enraizadas em máquinas estatísticas. Uma rede baseada na Regra de Boltzmann é designada por Máquina de Boltzmann (*Boltzmann Machine*). Numa máquina de Boltzmann os neurónios constituem uma estrutura recorrente e operam numa forma binária, isto é, alternam entre os estados *on* e *off*, representado por +1 e por -1, respectivamente, seguido por um ajuste progressivo dos pesos. Neste método os estados dos neurónios são determinados por uma distribuição probabilística.

Existem outras máquinas estocásticas, como por exemplo, o Arrefecimento Simulado (*Simulated Annealing*) aplicado a máquinas de Cauchy [Rojas, 1996].

2.5 Redes Multi-Camada com Aprendizagem Retropropagação

O algoritmo de Retropropagação (*Back-Propagation*) foi desenvolvido por Paul Werbos em 1974 e independentemente redescoberto por Rumelhart e Parker. Desde a sua redescoberta este algoritmo tem sido bastante utilizado como um algoritmo de aprendizagem em redes *feedforward* multi-camadas. O algoritmo de RP é aplicado em RNA's *feedforward* com uma ou mais camadas escondidas (*hidden layers*).

O que torna este algoritmo diferente dos outros é a forma como actualiza os pesos durante o processo de aprendizagem. Em geral, a dificuldade da utilização de redes multi-camada reside na determinação dos pesos das camadas escondidas de uma forma eficiente que resulte no menor erro das saídas. Para actualizar os pesos é necessário calcular o erro. Na camada de saída este erro é facilmente calculado pela diferença entre a saída actual da rede e a saída desejada. No entanto, nas camadas escondidas não existe nenhuma observação directa do erro, assim é necessário utilizar outras técnicas para calcular o erro nas camadas escondidas que correspondam à minimização do erro da saída, uma vez que este é o objectivo final.

2.5.1 Aprendizagem com Algoritmo Retropropagação

O algoritmo de Retropropagação procura o mínimo da função erro no espaço dos pesos, utilizando para isso o método do gradiente descendente. A combinação dos pesos que minimiza a função de erro é considerada a solução do problema de aprendizagem.

O problema de Aprendizagem

Uma rede *feed-forward* é um gráfico computacional cujos neurónios são unidades de processamento, ligados entre si, que transmitem informação numérica uns para os outros. Cada neurónio processa uma função de activação dos seus *inputs* e gera *outputs*. Na realidade uma RNA é uma cadeia de composições de funções que transformam *inputs* em *outputs* $(x \rightarrow y)$, a que se chama função da rede. O problema de aprendizagem consiste em determinar a combinação óptima dos pesos das conexões de maneira a que a função da rede se aproxime o máximo possível da função $f(x)$ desejada. No entanto, a função $f(x)$ não é conhecida de forma explícita, apenas implicitamente pelo conhecimento de alguns exemplos de pares de *input-output*.

Consideremos uma rede *feed-forward* constituída por L camadas de N_l neurónios em cada camada l e um conjunto de padrões de treino $\{(X_1, Y_1), (X_2, Y_2), ..., (X_p, Y_p)\}$, constituído por p pares ordenados de vectores de dimensão n e m, respectivamente. Suponhamos que a função de activação φ em cada um dos neurónios é contínua e diferenciável e, por motivos de simplificação de notação, igual para todas as camadas. Cada conexão entre o neurónio i da camada $l-1$, e o neurónio j da camada l, tem o peso W_{ij}^{l-1} associado. Os pesos W_{ij}^{l} são números reais iniciados aleatoriamente com média igual a zero.

O algoritmo de Retropropagação utiliza dois passos distintos de computação. O primeiro passo é geralmente referido como o "passo em frente" e o segundo com o "passo de retropropagação".

Passo em frente

No passo em frente, os pesos da rede permanecem inalterados e o sinal é propagado de neurónio em neurónio, de uma camada para outra, no sentido da camada de *outputs*. O processo inicializa-se fornecendo um padrão de entrada X_k à camada de neurónios de *inputs*. É então gerado um sinal nos neurónios da primeira camada, que por sua vez, faz com que seja gerado um novo sinal nos neurónios da camada seguinte, e assim sucessivamente, até que é obtida uma resposta da rede P_k na camada de saída, L.

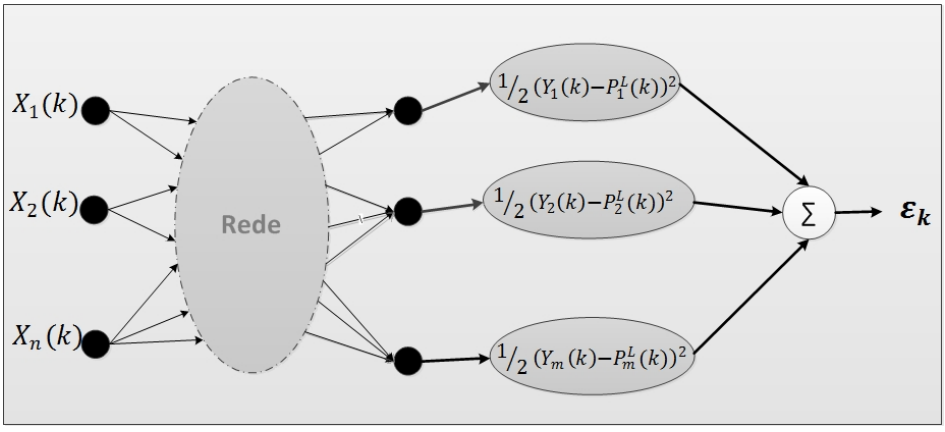


Figura 2.16: Expansão da rede para o cálculo da função erro
Adaptado de [Kartalopoulos, 1996]

Cada neurónio j na camada l recebe a resposta do neurónio i da camada $l-1$ através dos pesos W_{ij}^{l-1}. O sinal do neurónio j na camada l é expresso da seguinte forma:

$$P_j^l(K) = \varphi\left(\sum_{i=1}^{N_{l-1}} W_{ij}^{l-1} P_i^{l-1}(k)\right) = \varphi\left(\nu_j^l(k)\right) \tag{2.2}$$

Se o neurónio j pertencer à camada de *output* o sinal de saída $P_j^l(k)$ corresponde à componente j do vector de *output* da rede, P_k. O erro do sinal do neurónio j para o padrão de treino k é então definido pela seguinte expressão:

$$\epsilon_j(k) = \frac{1}{2}\left(Y_j(k) - P_j^L(k)\right)^2 = \frac{1}{2}\left(e_j(k)\right)^2 \tag{2.3}$$

onde $Y_j(k)$ representa a componente j do vector de *output* desejado, Y_k. O erro total para o padrão k é obtido somando $(1/2)e_j(k)^2$ para todos os neurónios da camada de *outputs*. Estes são os únicos neurónios "visíveis" para os quais o erro pode ser calculado directamente:

$$\epsilon_k = \frac{1}{2}\sum_{j=1}^{N_L=m}\left(Y_j(k) - P_j^L(k)\right)^2 = \frac{1}{2}\sum_{j=1}^{N_L=m} e_j(k)^2. \tag{2.4}$$

O erro total para o conjunto dos p padrões de treino é obtido pela soma de todos os erros quadráticos ϵ_k:

$$\epsilon = \sum_{k}^{p} \epsilon_k. \tag{2.5}$$

A função erro ϵ da rede é função de parâmetros livres da rede, ou seja, dos *bias* e pesos da rede. Para um conjunto de padrões de treino ϵ representa a função de custo que funciona como uma medida de performance da rede.

Retropropagação

O "passo de retropropagação", ao contrário do "passo em frente", começa na camada de *outputs* e propaga o erro ao longo da rede, camada por camada, em direcção à camada de *inputs*. Os pesos vão sendo actualizados padrão a padrão até terminar uma época, ou seja, quando terminar uma apresentação completa à rede de todos os padrões do conjunto de treino (ajuste dos pesos *on-line*). O objectivo é evoluir para um conjunto de pesos W_{ij}^l em todas as camadas da rede que minimizem a função custo ϵ.

A actualização dos pesos pode ainda ser feita através de outra abordagem, designada por actualização dos pesos em *batch*, que consiste em ajustar os pesos apenas ao fim de cada época através da minimização da acumulação dos erros ϵ_k de todos os padrões de aprendizagem.

Embora a resposta seja propagada para a frente, os pesos são calculados num movimento para trás, daí o nome Retropropagação.

A regra de aprendizagem é específica por estabelecer a actualização dos pesos proporcional à negativa da derivada do erro em relação aos pesos. Cada peso é actualizado da seguinte forma:

$$\Delta W_{ij}^{l} = -\gamma \frac{\partial \epsilon_k}{\partial W_{ij}^{l}} \tag{2.6}$$

onde γ representa a constante de aprendizagem (ou taxa de aprendizagem), um parâmetro de proporcionalidade que define o "comprimento do passo" de cada iteração na direcção negativa do gradiente.

Consideremos o neurónio j localizado na camada de *outputs* L da rede (figura 2.17).

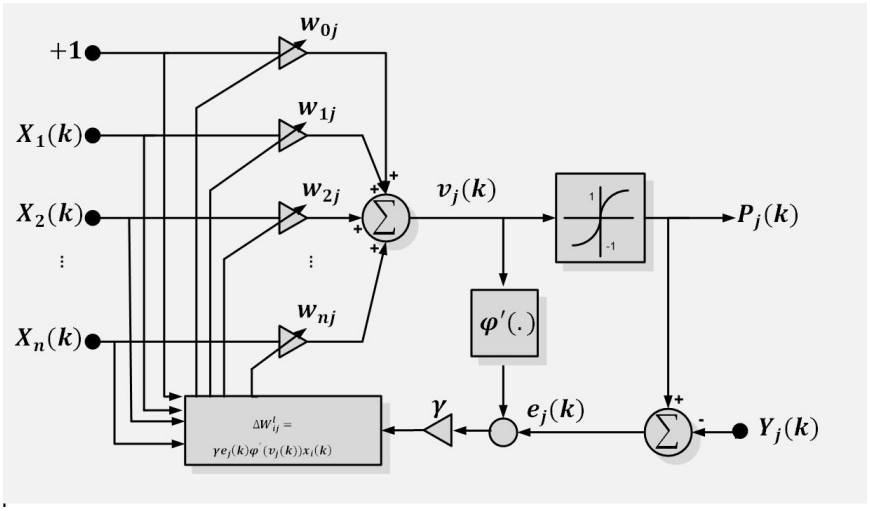


Figura 2.17: Implementação do algoritmo de RP para um neurónio da camada de *output*. Adaptado de [Cichocki and Unbehauen, 1993]

Neste caso, a derivada $\frac{\partial \epsilon_k}{\partial W_{ij}}$ pode ser escrita da seguinte forma:

$$\begin{aligned} \frac{\partial \epsilon_k}{\partial W_{ij}} &= \frac{\partial \epsilon_k}{\partial e_j(k)} \frac{\partial e_j(k)}{\partial P_j(k)} \frac{\partial P_j(k)}{\partial \nu_j(k)} \frac{\partial \nu_j(k)}{\partial W_{ij}(k)} = \\ &= e_j(k) \cdot (-1) \cdot \varphi'(\nu_j(k)) \cdot P_i(k) = \\ &= -e_j(k) \varphi'(\nu_j(k)) \cdot P_i(k) \end{aligned} \tag{2.7}$$

Assim o procedimento de ajuste dos pesos para a camada de saída é:

$$\Delta W_{ij} = -\gamma \frac{\partial \epsilon_k}{\partial W_{ij}} = \gamma \delta_j(k) P_i(k), \tag{2.8}$$

onde $\delta_j(k)$ representa o gradiente local e é definido por:

$$\delta_j(k) = -\frac{\partial \epsilon_k}{\partial \nu_j(k)} = e_j(k)\varphi'\left(\nu_j(k)\right), \tag{2.9}$$

O gradiente local indica direcções necessárias para a alteração dos pesos. De acordo com a equação (2.9), o gradiente local do neurónio j, $\delta_j(k)$, é igual ao produto do correspondente erro para o neurónio j, $e_j(k)$, e a derivada da função de activação respectiva, $\varphi'\left(\nu_j(k)\right)$. O elemento chave para a actualização dos pesos é o erro para o neurónio de *output* j. No caso particular dos neurónios de saída este erro é fácil de calcular, uma vez que a cada um dos neurónios da camada de *outputs* é fornecida a respectiva resposta desejada $Y_j(k)$.

No caso em que o neurónio j pertence a uma camada escondida, apesar de não estarem directamente associados ao erro, contribuem indirectamente para ele. A questão aqui é como ajustar os pesos associados ao neurónio j da camada escondida de forma a que não contribuam para o aumento do erro.

Consideremos um neurónio j de uma camada escondida l diferente da camada de *outputs* L. Neste caso o gradiente local define-se da seguinte maneira:

$$\delta_j(k) = -\frac{\partial \epsilon_k}{\partial P_j^l(k)} \frac{\partial P_j^l(k)}{\partial \nu_j^l(k)} = -\frac{\partial \epsilon_k}{\partial P_j^l(k)} \varphi'\left(\nu_j^l(k)\right). \tag{2.10}$$

Para calcular a derivada parcial $\frac{\partial \epsilon_k}{\partial P_j^l(k)}$ temos pela equação (2.4):

$$\epsilon_k = \frac{1}{2} \sum_{n=1}^{N_L=m} e_n(k)^2, \tag{2.11}$$

onde n é um neurónio da camada de *outputs*. Suponhamos, sem perda de generalidade, que j é um neurónio da camada $L-1$, isto é, da camada imediatamente anterior à camada de *output*:

$$\frac{\partial \epsilon_k}{\partial P_j^l(k)} = \sum_{n=1}^{N_L} e_n(k) \frac{\partial e_n(k)}{\partial P_j^l(k)} = \sum_{n=1}^{N_L} e_n(k) \frac{\partial e_n(k)}{\partial \nu_n^L(k)} \frac{\partial \nu_n^L(k)}{\partial P_j^l(k)}. \tag{2.12}$$

Sendo

$$e_n(k) = Y_n(k) - P_n^L(k) = Y_n(k) - \varphi'\left(\nu_n^L(k)\right) \tag{2.13}$$

e

$$\nu_n^L(k) = \sum_{i=1}^{N_l} W_{in}^l P_i^l(k) \tag{2.14}$$

então vem, que para $i = j$:

$$\frac{\partial e_n(k)}{\partial \nu_n^L(k)} \frac{\partial \nu_n^L(k)}{\partial P_j^l(k)} = -\varphi'\left(\nu_n^L(k)\right) W_{jn}^l. \tag{2.15}$$

Resulta então que:

$$\frac{\partial \epsilon_k}{\partial P_j^l(k)} = -\sum_{n=1}^{N_L} e_n(k)\varphi'\left(\nu_n^L(k)\right) W_{jn}^l = -\sum_{n=1}^{N_L} \delta_n(k) W_{jn}^l, \tag{2.16}$$

onde $\delta(k)$ é definido pela equação (2.9).

A partir da equação (2.10) concluímos que, o gradiente local para o neurónio j de uma camada escondida é definido pela seguinte expressão:

$$\delta_j(k) = \varphi'\left(\nu_j^l(k)\right) \sum_{n=1}^{N_L} \delta_n(k) W_{jn}^l. \tag{2.17}$$

O factor $\varphi'\left(\nu_j^l(k)\right)$ da equação (2.17) depende unicamente da função de activação do neurónio j. O segundo factor depende de dois conjuntos de termos. O primeiro conjunto de termos, $\delta_n(k)$, requer o conhecimento dos sinais dos erros $e_n(k)$, para todos os neurónios da camada imediatamente à direita da camada do neurónio j e que estão directamente ligados ao neurónio j. O segundo conjunto de termos, W_{jn}, consiste nos pesos associados a essas ligações.

Resumindo

$$\Delta W_{ij}^l = \gamma[\delta_j^l] P_i^{l-1}, \tag{2.18}$$

onde para os pesos da camada de saída L,

$$\delta_j^L = e_j(k)\varphi'\left(\nu_j^L(k)\right) \tag{2.19}$$

e para os pesos das camadas escondidas,

$$\delta_j^l = \varphi'\left(\nu_j^l(k)\right) \sum_{n=1}^{N_L} \delta_n(k) W_{jn}^{l-1}. \tag{2.20}$$

O processo de cálculo do gradiente e consequente ajuste dos pesos é repetido até ser encontrado um erro mínimo. Na prática é introduzido no algoritmo um critério de paragem para que este processo iterativo não continue para sempre.

Parece óbvio que, para os neurónios da camada l, o calculo do δ_i^l depende do erro calculado para a camada $l+1$, ou seja, o calculo das diferenças é realizado para trás.

0. Inicialização dos pesos W_{ij}^l.
   (Geralmente valores aleatórios com média igual a 0)

1. Apresentação à rede de um padrão de entrada X_k e calcula os valores actuais da rede P_k utilizando os valores presentes dos pesos.

2. Especificar o valor de *output* desejado Y_k e calcular o erro associado ϵ_k.

3. Ajustar os pesos de acordo com a seguinte fórmula:

   $$\Delta W_{ij}^l = -\gamma \frac{\partial \epsilon_k}{\partial W_{ij}^l}.$$

4. Apresentar outro padrão de entrada e voltar ao passo 2.

Figura 2.18: Método de aprendizagem com retropropagação num neurónio
Adaptado de [Kartalopoulos, 1996]

2.5.2 Limitações Inerentes ao Algoritmo Retropropagação

O algoritmo de Retropropagação emergiu como o mais popular algoritmo de aprendizagem para o treino supervisionado de redes multi-camada. Basicamente é uma técnica de gradiente (derivação) e não uma técnica de optimização. O algoritmo de RP tem duas propriedades distintas:

1. É simples de calcular localmente;
2. Executa o gradiente descendente de forma estocástica no espaço dos pesos (padrão a padrão actualizando os pesos).

Estas duas propriedades do algoritmo de RP no contexto das redes multi-camada são também responsáveis pelas suas limitações.

Problema do Mínimo Local da Função Erro

Para cada conjunto de treino (padrão) pode ser definida uma superfície de erro, que resulta da representação gráfica do erro em função dos parâmetros do vector de pesos da rede a serem ajustados. Uma particularidade da superfície de erro que influencia o desempenho do algoritmo de RP é a presença de mínimos locais, para além dos mínimos globais. Um mínimo local é um valor mínimo numa determinada vizinhança, neste caso em concreto, um "vale" isolado na superfície do erro.

O algoritmo de RP pode ser interpretado como um processo físico onde uma pequena bola percorre a superfície da função erro até alcançar o local mais profundo. Com a existência de mínimos locais na superfície de erro ("vales" isolados) corre-se o risco da bola ficar presa num desses "vales", onde a função erro é maior do que seria se alcança-se o mínimo global. É claramente indesejável que o processo de aprendizagem termine num mínimo local, principalmente se este se encontra próximo do mínimo global.

A figura 2.19 mostra um exemplo onde existem mínimos locais com um erro superior a mínimo noutras regiões. Existe um "vale" na superfície da função erro e caso o gradiente descendente fosse inicializado num neste "vale" o algoritmo não convergiria para o mínimo global.

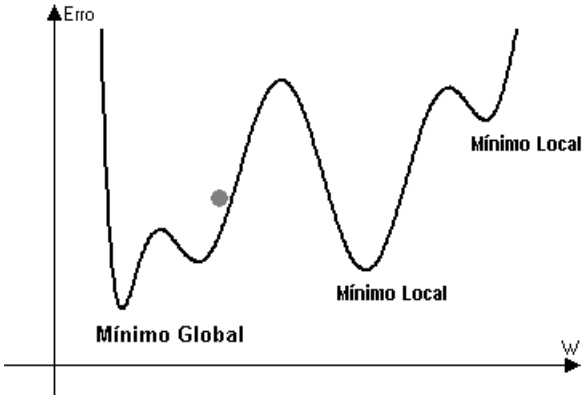


Figura 2.19: Exemplo de Mínimo Local de uma função de erro
Adaptado de [Alves, 2002]

O problema do mínimo local na aprendizagem com o algoritmo de RP foi levantado no epílogo da edição ampliada do livro clássico de Minsky e Papert, onde a atenção está focada na discussão do livro de dois volumes *Parallel Distributed Processing*, de Rumelhart e McClelland. No capitulo 8 deste último livro é afirmado que ficar preso num mínimo local é na prática um problema raro na aprendizagem com o algoritmo de RP. Minsky e Papert contrariaram esta ideia salientando que toda a história de reconhecimento de padrões

mostra o contrário.

Problema de *Overfitting* e Generalização

As RNA's utilizam um conjunto de dados de treino (padrão de treino) para ajustar os pesos da rede. Uma vez treinada, ou seja ajustados os pesos para a minimização de uma função de custo, a rede deve ser capaz de na presença de novos *inputs*, nunca antes apresentados à rede, produzir valores de *output* previstos. A rede deve ser capaz de reconhecer se os *inputs* apresentados se assemelham a algum dos padrões aprendidos e produzir uma resposta similar.

Diz-se que ocorre sobre-ajustamento (*overfitting*) numa RNA quando esta apresenta um desempenho quase perfeito nos dados de treino mas um desempenho fraco na presença de novos *inputs*, ou seja, sempre que a rede apresenta uma fraca capacidade de generalização. Neste caso a rede cria um modelo que descreve o padrão de treino em vez de um modelo generalizado da tarefa que se pretende modelar.

O *overfitting* pode ser devido ao facto de existir demasiado "ruído" no conjunto de treino ou por existir uma quantidade insuficiente de dados. Um outro motivo para a fraca capacidade de generalização de uma rede, pode estar relacionada com a sua complexidade, ou seja, quando a rede é constituída por demasiadas unidades de processamento (neurónios) e, consequentemente, tem muitos parâmetros (pesos) a ajustar. Por outro lado, se o número de neurónios for insuficiente a rede pode não ser capaz de se ajustar ao conjunto de treino. Por este motivo a determinação do número de neurónios a pertencer a cada camada escondida deve ser feita de forma empírica e normalmente depende da distribuição do conjunto de treino da rede.

A figura 2.20 ilustra um exemplo de *overfitting*. A função de aprendizagem oscila excessivamente de forma a se ajustar a todos os pontos do conjunto de treino no entanto para o conjunto de pontos novos não se consegue adaptar [Alves, 2002].

A solução para este tipo de problema, que é considerado critico nas RNA's, pode passar por usar uma quantidade elevada de dados no padrão de treino, ou ainda usar um critério de paragem antecipado que termina o treino quando o erro obtido no conjunto de validação (exemplo de *inputs* não usados no treino) sobe, ou ainda pelo uso de métodos de regularização que penalizem modelos demasiados complexos.

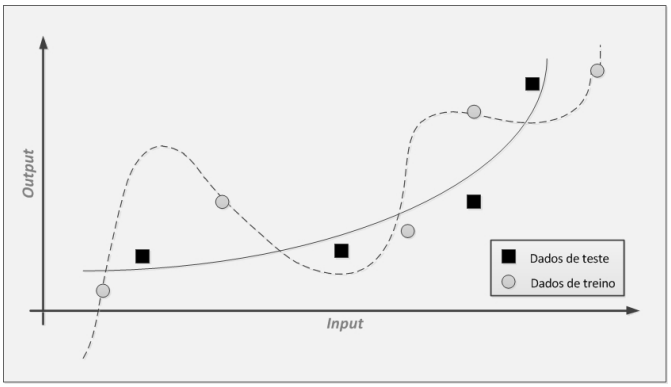


Figura 2.20: Exemplo de *overfitting*
Adaptado de [Alves, 2002]

Problema da Convergência Lenta

O algoritmo de RP utiliza uma "estimativa instantânea" do gradiente da superfície de erro no espaço dos pesos. Portanto o algoritmo de RP é estocástico por natureza, isto é, tem a tendência de fazer o caminho em ziguezague em torno da verdadeira direcção para o mínimo da função erro. Consequentemente, a velocidade de convergência no algoritmo de RP tende a ser relativamente lenta. Podemos identificar duas causas fundamentais para este facto:

1. A superfície de erro é relativamente plana ao longo da dimensão dos pesos, o que significa que a derivada da superfície de erro em relação aos pesos é de amplitude pequena. Nesta circunstância, o ajuste a ser aplicado aos pesos é pequeno e consequentemente, serão necessárias várias iterações do algoritmo para produzir reduções significativas no erro global da rede. Alternativamente, a superfície do erro pode ter muitas curvas e, neste caso, a derivada da superfície de erro em relação ao pesos é grande em magnitude, o que implica que o ajuste dos pesos também seja grande, o que pode originar que o algoritmo ultrapasse o erro mínimo (passe por cima);

2. A direcção negativa do gradiente (isto é, a negativa da derivada da função erro em relação aos pesos) pode apontar para longe do mínimo da superfície de erro, uma vez que o ajuste aplicado aos pesos pode induzir o algoritmo para mover na direcção errada.

Existem casos em que a velocidade da aprendizagem é uma limitação em aplicações práticas de RNA's. Existem vários métodos para acelerar a convergência do algoritmo RP, alguns abordados mais à frente.

Problema da "Caixa Negra"

Um dos problemas no treino de redes multi-camada com o uso do algoritmo de RP está relacionado com a definição dos seus parâmetros. A selecção dos seus parâmetros é um processo pouco conhecido não existindo regras claras para a sua definição. Apesar de ao longo do tempo terem vindo a ser propostas algumas soluções sofisticadas como os algoritmos de corte, construtivos e genéricos, a escolha dos parâmetros depende muito do caso pratico em causa pelo que é muitas vezes feito por tentativa e erro. Pequenas diferenças nos parâmetros podem provocar grandes divergências tanto no tempo de aprendizagem como na capacidade de generalização da rede.

Muitas vezes as RNA's são referidas como "caixa-negra", das quais pouco se sabe, a razão pela qual devolve determinado resultado em determinadas circunstâncias. A falta de demonstração nas resposta obtidas numa RNA é muitas vezes um motivo de cepticismo e consequente impedimento na sua aplicação pratica. Por este motivo, muitas pesquisas têm vindo a ser desenvolvidas no sentido de criar procedimentos explicativos onde se procurar compreender e explicar o comportamento da rede em determinadas situações.

2.5.3 Melhorias no Algoritmo de Retropropagação

A apresentação do algoritmo de RP alterou por completo o cenário da investigação em redes multi-camadas e desde então têm surgido uma grande variedade de novos algoritmos de aprendizagem. Esta exploração deveu-se essencialmente a duas razões: à convergência lenta do algoritmo de RP e ao facto de se basear no gradiente descendente, o que permitiu que todas as técnicas de optimização não linear do gradiente pudessem ser aplicadas.

Nesta secção são referidas algumas melhorias simples que podem ser aplicada ao algoritmo de RP com o objectivo de melhorar a sua performance e convergência lenta.

A Função de Activação

O método utilizado no algoritmo de RP requer o cálculo do gradiente da função de erro em cada iteração, pelo que, a função de erro tem de ser contínua e diferenciável, consequentemente, teremos de usar uma função de activação diferente da função degrau utilizada no perceptrão, uma vez que a função composta produzida pelos perceptrões interligados é descontínua e consequentemente a função de erro também. São várias as funções

de activação utilizadas no algoritmo de RP, no entanto as mais comuns são as do tipo sigmóide. Uma função de activação diferenciável faz com que a função de erro da RNA a ser minimizada durante a aprendizagem também seja diferenciável, ou seja, assumindo que a função de integração em cada neurónio é a soma dos *inputs*, que é aplicada na função de activação, que por sua vez gera o *output*, e sendo a função do erro função da diferença entre *outputs* desejados e *outputs* da rede, então também é diferenciável.

A figura 2.21 mostra o nivelamento produzido pela função sigmóide num "passo" da função erro. Uma vez que o objectivo é seguir a direcção do gradiente para encontrar o mínimo da função erro, é importante que não existam zonas onde a função erro seja plana. Como a função sigmóide tem sempre derivada positiva, a inclinação da função erro fornece a direcção de uma maior ou menor descida a ser seguida.

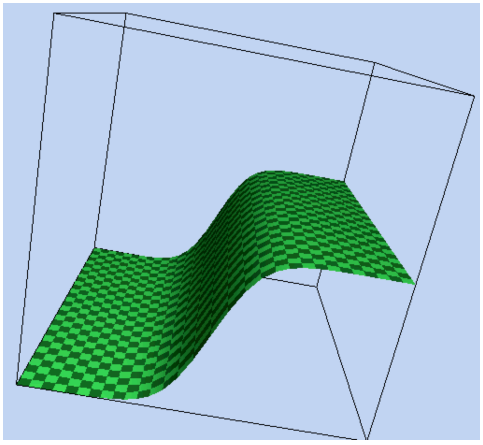


Figura 2.21: Um "passo" da função erro
Adaptado de [Rojas, 1996]

Retropropagação com Taxa de *Momentum*

Quando o mínimo da função do erro atinge um mínimo local a seguir a direcção do gradiente pode levar a grandes oscilações no processo de busca. A figura 2.22 mostra exemplo do ajuste de uma rede com dois pesos. A melhor abordagem neste caso é orientar a busca atravessando o centro do vale mas a função de erro é de tal forma que o gradiente não orienta neste sentido.

Existem várias técnicas para acelerar o processo de aprendizagem e simultaneamente evitar os mínimos locais. Uma solução simples é introduzir um elemento *momentum*. O gradiente da função erro é calculado para cada combinação de pesos, mas em vez de seguir apenas a direcção negativa do gradiente, é calculada uma média ponderada do gradiente corrente e a anterior correcção da direcção. Teoricamente, o *momentum* deve

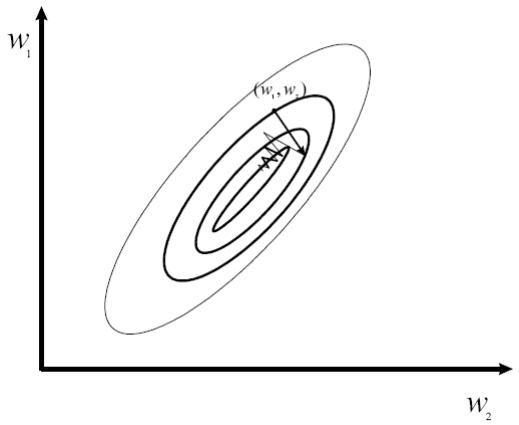


Figura 2.22: Oscilações na convergência do método do gradiente
Adaptado de [Moreira, 1997]

acelerar o treino em regiões muito planas da superfície de erro e ajudar e evitar oscilações excessivas em "vales" profundos na superfície de erro.

A correcção do erro w_k na iteração i, para uma rede com n pesos diferentes $w_1, ..., w_l$, passa a ser:

$$\Delta w_k(i) = -\gamma \frac{\partial \epsilon}{\partial w_k} + \alpha \Delta w_k(i-1),$$

onde γ e α são os parâmetros de aprendizagem e *momentum*, respectivamente. A taxa de *momentum* é tal que $0 < |\alpha| < 1$, podendo ser um valor negativo ou positivo, apesar de não ser muito provável o uso de valores negativos. Quando $\alpha = 0$ o algoritmo de RP funciona sem taxa de *momentum*. Quando $\partial\epsilon/\partial w_k$ tem o mesmo sinal algébrico em iterações consecutivas, $\Delta w_k(i)$ cresce em magnitude e o peso $w_k(i)$ é ajustado por um valor elevado, verificando-se uma tendência em acelerar a descida na direcção do declive. Quando $\partial\epsilon/\partial w_k$ tem o sinal algébrico contrário em iterações consecutivas, $\Delta w_k(i)$ diminui em magnitude e o peso $w_k(i)$ é ajustado por um valor baixo, verificando-se um efeito estabilizador na direcção em que o sinal oscila.

Taxa de Aprendizagem

O algoritmo de aprendizagem de RP é conhecido por providenciar uma "aproximação" à trajectória, no espaço dos pesos, calculada pelo método da "descida mais rápida". Quanto mais pequena for a taxa de aprendizagem γ, menor serão as alterações nos pesos da rede de uma iteração para outra, de modo a que a procura do mínimo global será favorecida por uma trajectória mais suave, no entanto também mais lenta. Por outro lado, quanto

mais elevados forem os valores da taxa de aprendizagem, maior o risco de ocorrerem "saltos" muito elevados nas mudanças dos pesos que podem provocar instabilidade no treino.

O ajuste de ambos os parâmetros, taxa de aprendizagem e *momentum*, para obter uma possível convergência, é geralmente feito por tentativa erro, ou ainda por algum tipo de procura aleatória. Uma vez que os parâmetros óptimos estão fortemente dependentes da tarefa de aprendizagem que se procura modular, não existe nenhuma estratégia geral desenvolvida para lidar com este problema. Existem no entanto algoritmos de aprendizagem, variantes do algoritmo de RP, que fazem os ajuste destes parâmetros automaticamente através de algoritmos ao longo do processo de treino da rede.

Critério de Paragem

O processo de minimização da função erro da rede, em geral, não tem uma convergência para o mínimo global garantida e o algoritmo de RP não tem definido em que momento se deve parar a aprendizagem. Por este motivo, é legitimo que surja naturalmente a dúvida de que em que momento devemos parar o treino da rede.

Existem vários métodos para determinar o momento de paragem do algoritmo, a esses métodos chamamos critérios de paragem [Silva, 1998].

Um critério fácil de implementar é definir previamente um número fixo de épocas. Apesar de simples, este critério, não é muito recomendável uma vez que ignora por completo o estado do processo iterativo de treino.

Existem critérios de paragem que têm em conta alguma informação relativa ao estado do processo iterativo.

Para formular estes critérios de paragem consideremos a existência de mínimos locais. Seja w_0^* o vector de pesos associado a um mínimo, local ou global. Uma condição necessária para que um qualquer vector de pesos w^* seja mínimo é a de que $\nabla\epsilon$, da superfície de erro em relação ao vector de pesos, seja 0 em w^*. Com base nesta propriedade do mínimo, podemos formular o seguinte critério de paragem: terminar o treino quando o $\|\nabla\epsilon\|$ atinge um valor suficientemente pequeno. O problema deste critério é que para ser bem sucedido, o tempo de treino é muito longo e requer o cálculo da norma do gradiente.

Outra propriedade do mínimo que pode ser utilizada para formular um critério de pa-

ragem é a de que a função erro é estacionária no ponto w^*. Podemos então definir o seguinte critério: terminar o treino quando a variação da função erro ϵ de uma época para outra atingir um valor suficientemente pequeno ou alternativamente, quando atinge um valor inferior a um limite θ, $\epsilon \leq \theta$, onde θ deve ser suficientemente pequeno.

Se o critério de paragem for um valor mínimo para o erro ϵ então não podemos garantir que o algoritmo atinja o valor desejado. Por outro lado se considerarmos o critério de paragem um valor mínimo para $\|\nabla\epsilon\|$ devemos estar conscientes de que o mais provável é que o algoritmo devolva o mínimo local mais próximo da condição inicial.

A figura 2.23 ilustra precisamente um exemplo onde, se o critério de paragem fosse $\epsilon \leq \theta$ então, o método não seria capaz de devolver o mínimo global, porque o mínimo da superfície de erro é superior do que o valor de θ estabelecido. Todos os "vales" da curva constituem mínimos locais onde $\|\nabla\epsilon\| = 0$. Utilizando o critério $\|\nabla\epsilon\| \leq \theta$, o algoritmo não será capaz de encontrar um conjunto de pesos que satisfaçam o critério, independente dos mínimos locais.

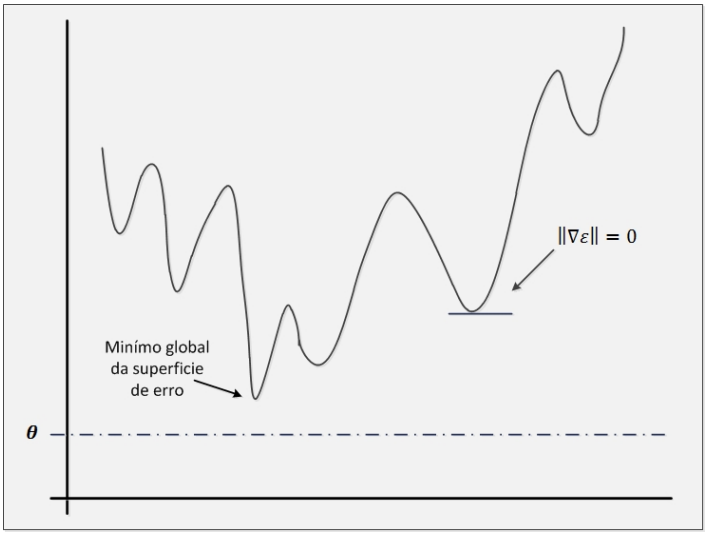


Figura 2.23: Superfície de erro com vários mínimos
Adaptado de [Silva, 1998]

Existem outros critérios de paragem bastante úteis, nomeadamente os critérios baseados na capacidade de generalização da rede, após cada época de treino, em que o processo de treino termina quando uma determinada percentagem de dados do conjunto de validação obtém uma resposta da rede correcta.

Em geral e na prática, não se utiliza apenas um critério de paragem mas a combinação

de vários.

O Método de Treino (sequencial ou em lote)

Em aplicações práticas do algoritmo de RP a aprendizagem é resultado de repetidas apresentações do padrão de treino à rede. A cada apresentação de todo o padrão de treino, ou seja, de todos os exemplos do conjunto de treino à rede chama-se época (*epoch*). O processo é repetido época após época até que o conjunto de pesos estabilize e a função de erro da função convirja para um valor mínimo, ou um critério de paragem seja atingido.

A ordem de apresentação dos dados do conjunto de treino à rede deve ser feita aleatoriamente de uma época para a outra, para que não exista tendências para valorizar certos padrões no conjunto de treino. Esta apresentação pode ser feita de duas formas diferentes: no modo sequencial (*on-line*) ou em lote (*batch*).

No modo sequencial calcula-se o ajuste dos pesos logo após a escolha de um exemplo de treino. Consideremos um conjunto de padrões de treino, $\{(x_1, y_1), (x_2, y_2), ..., (x_p, y_p)\}$, constituído por p pares *input-output*. O primeiro padrão de treino, (x_1, y_1), da época é apresentado à rede e são processados os dois passos do algoritmo de RP, resultando num ajuste dos pesos. De seguida, o segundo padrão, (x_2, y_2), é apresentado à rede resultando um segundo ajuste dos pesos da rede. Este processo é repetido até ao ultimo padrão, (x_p, y_p). Este método também é conhecido por actualização dos pesos por padrão.

Na apresentação dos padrões do conjunto de treino em modo de lote ou por época (*batch*), o ajuste dos pesos só é feito após a apresentação de todos os padrões no conjunto de treino. O ajuste relativo a cada apresentação de um padrão é acumulado.

Esta técnica é mais estável que a do modo sequencial, no entanto, também pode ser mais demorada principalmente se o conjunto dos padrões de treino é grande e redundante. Esta abordagem apresenta uma estimativa mais precisa do gradiente decrescente ao custo da necessidade de mais memória local. É também um processo mais fácil de paralelizar.

A escolha da abordagem a ser utilizada depende da aplicação e da distribuição estatística dos dados.

Inicialização dos Pesos

Em geral os pesos são inicializados de forma aleatória, no entanto, existem outros métodos mais sofisticados, muito utilizados na prática.

Uma boa escolha dos pesos pode acelerar bastante o processo de ajuste dos pesos. Interessa saber o que se entende por uma boa escolha dos pesos iniciais.

Valores elevados podem levar a um problema de saturação da rede atrasando o processo de aprendizagem, por outro lado, valores pequenos podem levar a que o algoritmo de RP opere numa sistema demasiado plano por volta da origem da superfície de erro. Deste modo, quer valores muito pequenos ou muito grandes devem ser evitados.

Uma boa abordagem é escolher valores aleatórios com média igual a zero que dependem do número de conexões de um neurónio.

Pré-Tratamento dos Dados

O pré-tratamento dos dados antes de os fornecer à rede para o treino é por vezes extremamente útil e importante. A ideia é, por exemplo, eliminar situações de *outliers*, ou ajustar os dados para que a rede os possa tratar de forma eficiente

Este tema será mais extensamente apresentado no próximo capítulo, onde será apresentado o caso em estudo e o pré-processamento dos dados.

2.6 Aplicações das RNA's

As primeiras divulgações do uso prático de RNA's surgem no prognóstico de mercados financeiros. Alguns grupos de investimento conhecidos utilizaram RNA's para analisar os mercados financeiros bolsistas e ajudar nas tomadas de decisão. Ao longo do tempo, o uso de RNA's tem vindo a ser cada vez mais significativo nas diversas tarefas de aprendizagem desempenhadas por RNA [Neves, 2011], [Alves, 2002].

Diagnóstico/Classificação

Uma das tarefas mais comuns desempenhada por RNA's é a tarefa de diagnóstico, muito utilizada em áreas como medicina, engenharia ou produção. Consiste essencialmente numa tarefa de classificação, ou seja, o objectivo é associar a um conjunto de entradas, que representam indicadores de um determinado estado (sintomas ou comportamentos anormais), uma saída que corresponde ao diagnóstico (doença ou falha no

sistema).

Alguns exemplos são:

- Diagnósticos de falhas em linhas de montagem;
- Diagnóstico de doença, como por exemplo na classificação de células cancerígenas.

Reconhecimento de Padrões

As RNA's são também muito eficazes no desempenho da tarefa de reconhecimento de padrões. O reconhecimento de padrões consiste essencialmente na atribuição de uma categoria (saída da RNA) a um sinal/padrão recebido pela rede (entrada).

Alguns exemplos são:

- Processamento de imagem para o reconhecimento automático de indivíduos;
- Validação de assinaturas automáticas;
- Reconhecimento de voz;
- Detecção de fraudes de cartões de crédito através da análise de padrões de compras no uso de cartões de crédito para a identificação de situações de fraude;
- Classificação de clientes para a atribuição de créditos bancários.

Controlo

Esta tarefa envolve o controlo de um processo ou uma parte crítica de um sistema que tem de ser mantido numa situação controlada. O objectivo principal do controlador é providenciar *inputs* x apropriados ao sistema, a ser mantido sobre controlo, para que o *output* y acompanhe um sinal de referência.

Alguns exemplos são:

- Controlo do processo de fabrico;
- Controlo de veículos de condução automática.

Optimização

O objectivo neste tipo de tarefa é reduzir o espaço de procura de uma solução óptima de forma a atingir uma solução aceitável.

Alguns exemplos são:

- Optimização de recursos militares e detecção de alvos;
- Robôs com movimento inteligente, ou seja, capazes de optimizar uma trajectória.

Regressão/Previsão

As RNA's são também muito utilizadas para modelar funções. O objectivo é que a RNA seja capaz de modelar uma função desconhecida $f(x)$ que se aproxime de uma função $F(x)$ dada por um conjunto de pares *input-output* $(x \rightarrow y)$ de forma a minimizar a distância euclidiana entre $f(x)$ e $F(x)$ para todas as entradas. A previsão, caso particular da regressão, tem como objectivo "adivinhar" valores futuros de uma função desconhecida.

Alguns exemplos são:

- Previsão de solvências ao analisar determinadas características das empresas;
- Previsão de vendas e marketing;
- Previsão do comportamento dos mercados financeiros para compra e venda de acções.

Podemos concluir que as RNA's, originalmente inspiradas nas redes neuronais biológicas, podem ser interpretadas como unidades de processamento de informação que procuram "aprender" através de experiência passada. O elemento principal de uma RNA surgiu como uma tentativa de reproduzir o funcionamento das células nervosas do sistema nervoso, o neurónio. Uma RNA tem uma estrutura extremamente interligada destes elementos (neurónios) que transmitem sinais de uns para os outros. Os sinais são transmitidos através de conexões que têm pesos associados. Cada neurónio tem uma função de activação associada que é aplicada à sua entrada (*input*), reproduzindo assim uma saída (*output*) que transmite para o neurónio seguinte. Durante o processo de aprendizagem da rede, dado por um algoritmo de aprendizagem ou treino, os pesos associados às

conexões são ajustados para que a rede reproduza a saída desejada.

As RNA's têm vindo ao longo dos anos a ganhar cada vez mais popularidade, em particular devido à sua aplicação bem sucedida nas mais diversas áreas, como medicina, em tarefas associadas a diagnóstico, na economia, na previsão de séries temporais, na engenharia, e muitas outras. No entanto, a utilização das RNA's tem limitações, quer pela dificuldade de demonstrar as suas respostas ou decisões, quer pelo elevado tempo despendido na procura da melhor arquitectura e algoritmo de aprendizagem. Ainda assim, a aplicação das RNA apresenta-se bastante vantajosa em situações em que os métodos convencionais são de difícil aplicação por se desconhecer as regras inerentes ao sistema ou tarefa que se pretende modelar.

O trabalho desenvolvido nesta dissertação é um exemplo de utilização de RNA's para a tarefa de previsão. A ideia principal é conceder uma RNA capaz de modelar/prever o desempenho académico global de um aluno de uma engenharia de cinco anos, com base no seu desempenho no primeiro (ou primeiros) ano (s) de ingresso.

Capítulo 3

Caso em Estudo

3.1 Introdução

As RNA's têm vindo a ser aplicadas de forma bem sucedida em diversas áreas para diferentes tarefas, nomeadamente modelação de uma função, classificação de dados ou reconhecimento de padrões.

No caso em estudo, o objectivo é a modelação do desempenho académico dos alunos de uma Licenciatura em Engenharia (pré-Bolonha) da FCT-UNL com cinco anos de duração. Em particular, o objectivo deste trabalho é treinar uma RNA para que seja capaz de prever a duração total do curso (DTC) de um aluno e/ou a respectiva nota final (NFC), com base no seu desempenho académico no primeiro (ou dois primeiros) ano(s) frequentado(s) na FCT-UNL.

Este capítulo tem duas secções principais. Numa primeira secção principal é feita a descrição relativa aos dados, o seu tratamento, sendo também definidos os principais parâmetros inerentes à RNA utilizada. Na segunda secção, mais relevante, são descritas todas as experiências computacionais levadas a cabo, no contexto dos diferentes cenários testados, e analisados os respectivos resultados. Nesta secção são também apresentadas breves conclusões.

3.2 Pré-Processamento dos Dados

Antes de serem alimentados à rede, os dados devem ser pré-processados. O pré-processamento dos dados consiste em efectuar alterações aos mesmos, antes de serem fornecidos à rede, de maneira a que a rede os possa tratar de forma mais eficiente e rápida. O pré-

processamento dos dados é essencial para o sucesso ou insucesso de uma RNA e consiste em diversas operações, que dependem do caso em estudo, das quais destacamos:

- Validação dos dados – consiste na validação dos dados para verificar a existência possíveis erros de alguma espécie;
- Representação dos Dados – consiste na conversão de dados, em particular na codificação dos dados não numéricos, na definição das variáveis, que podem representar dados agrupados, ou por outro lado, quando pretendemos que uma variável seja representada por um ou mais neurónios;
- Mudança de Escala – consiste na mudança de escala dos dados envolvidos no processo de aprendizagem com o intuito de acelerar a aprendizagem e melhorar a performance da rede;
- Filtragem dos dados – consiste na aplicação de técnicas de filtragem aos dados para a eliminação de ruído e consequente suavização da função de aprendizagem.

3.2.1 Tratamento dos Dados

Os dados utilizados foram recolhidos por Nunes no contexto da sua dissertação de Mestrado em Investigação Operacional [Nunes, 2007]. A amostra consiste na informação de 297 alunos de uma Licenciatura em Engenharia (pré-Bolonha) da FCT-UNL com cinco anos de duração, relativos a um período de vinte anos.

Para cada aluno dispomos da classificação e do respectivo número de inscrições de todas as disciplinas concluídas dos dois primeiros anos frequentados. Dispomos também do ano de ingresso do aluno, a duração total do curso (DTC) e a respectiva nota final do curso (NFC).

	A	B	C	D	E	F	G	H	I	J	K	L	M
1					Disciplinas do 1º Ano Lectivo concluidas no 1º ano frequentado								
2	Aluno	DTC	NFC		AMI	ALGA	DGA	ICP	Q1	AM2	F1	Q2	TO
3					*Nota*	*Nota*	*Nota*	*Nota*	*Nota*	*Nota*	*Nota*	*Nota*	*Nota*
4	966	5	15		11	11	12	18	13	11	14	10	12
5	972	7	13		13	12	11	15	12	10	16	10	12
6	986	5	14		10	12	14	18	16	12	11	10	13
7	988	7	15		13	14	10	20	15	14	14	11	13
8	1117	6	14		10	11	14	11	13		15		15

Data 1º Ano Frequentado | Data 2º Ano Frequentado

Figura 3.1: Dados do primeiro ano frequentado

Numa primeira fase, procuraremos prever apenas DTC e tomaremos como *input* da RNA o desempenho do aluno no primeiro ano frequentado (Cenário 1). Numa segunda

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1					Disciplinas do 1º Ano Lectivo concluidas no 1º ou 2º ano frequentado									Disciplinas do 2º Ano Lectivo concluidas no 2º ano frequentado									
2	Aluno	DTC	NFC		AMI	ALGA	DGA	ICP	Q1	AM2	F1	Q2	TO	MA 1	CM 1	F 2	CN	AM 3	AM 4	PEC	F 3	DF	GI
3					*Nota*	*Nota*	*Nota*	*Nota*	*Nota*	*Nota*	*Nota*	*Nota*	*Nota*	*Nota*	*Nota*	*Nota*	*Nota*	*Nota*	*Nota*	*Nota*	*Nota*	*Nota*	*Nota*
4	966	5	15		11	11	12	18	13	11	14	10	12	12	0	14	11	12	11	14	11	12	16
5	972	7	13		13	12	11	15	12	10	16	10	12	0	15	13	10	12	0	0	0	13	14
6	986	5	14		10	12	14	18	16	12	11	10	13	13	12	15	13	14	13	16	13	15	16
7	988	7	15		13	14	10	20	15	14	14	11	13	13	10	0	12	12	0	10	13	10	13
8	1117	6	14		10	11	14	11	13	13	15	11	15	10	17	14	10	12	12	0	0	0	16

Data 1º Ano Frequentado | Data 2º Ano Frequentado

Figura 3.2: Dados dos dois primeiros anos frequentados

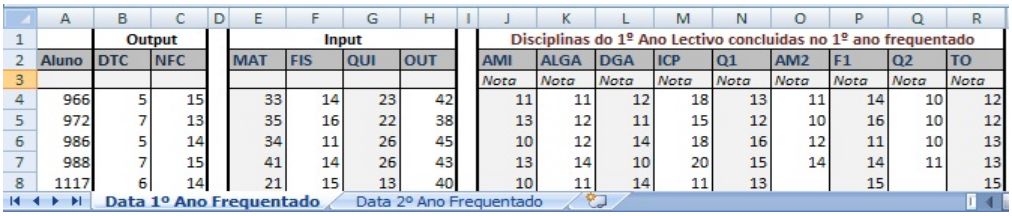
fase, consideraremos adicionalmente o desempenho no segundo ano frequentado, visando melhorar a qualidade da previsão de DTC (Cenário 2). Numa terceira fase, procuraremos testar versões alternativas destes dois cenários, com a introdução de uma nova variável de *input* o ano de ingresso do aluno (Cenário 1A e Cenário 2A). Na última fase, consideraremos o melhor dos cenários testados para prever DTC e procuraremos prever NFC (Cenário 3). Nesta fase, procuraremos ainda testar dois cenários alternativos considerando como *input* adicional o ano de ingresso ou a DTC (Cenário 3A e 4).

Sendo o objectivo prever DTC e NFC com base no desempenho académico do aluno nos primeiros anos de frequência na FCT-UNL e tratando-se de um curso de cinco anos não fará sentido prever DTC e NFC com base no seu desempenho para além do segundo ano frequentado.

Para representar o desempenho académico do aluno, no primeiro e nos dois primeiros anos agruparam-se as disciplinas da mesma natureza científica por uma questão de simplificação do modelo e diminuição do número de variáveis.

Temos, assim, as seguintes variáveis que caracterizaram o desempenho académico no primeiro ano de frequentado:

- MAT – soma da classificação das disciplinas de Matemática concluídas no primeiro ano frequentado;
- FIS – soma da classificação das disciplinas de Física concluídas no primeiro ano frequentado;
- QUIM – soma da classificação das disciplinas de Química concluídas no primeiro ano frequentado;
- OUT – soma da classificação de Outras disciplinas concluídas no primeiro ano frequentado.



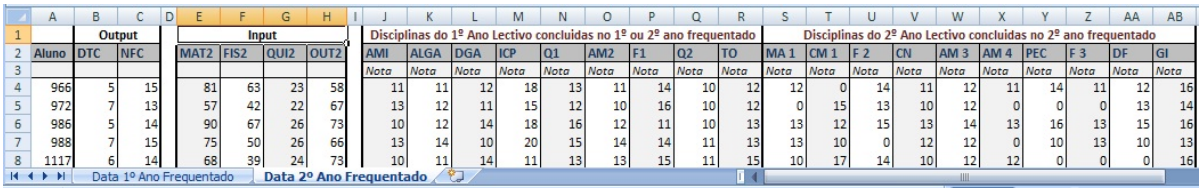
	Output		Input				Disciplinas do 1º Ano Lectivo concluidas no 1º ano frequentado								
Aluno	DTC	NFC	MAT	FIS	QUI	OUT	AMI	ALGA	DGA	ICP	Q1	AM2	F1	Q2	TO
							Nota	Nota	Nota	Nota	Nota	Nota	Nota	Nota	Nota
966	5	15	33	14	23	42	11	11	12	18	13	11	14	10	12
972	7	13	35	16	22	38	13	12	11	15	12	10	16	10	12
986	5	14	34	11	26	45	10	12	14	18	16	12	11	10	13
988	7	15	41	14	26	43	13	14	10	20	15	14	14	11	13
1117	6	14	21	15	13	40	10	11	14	11	13		15		15

Data 1º Ano Frequentado | Data 2º Ano Frequentado

Figura 3.3: Dados do primeiro ano frequentado

As seguintes variáveis caracterizaram o desempenho académico nos dois primeiros anos de frequentados:

- MAT2 – soma da classificação das disciplinas de Matemática concluídas nos dois primeiros anos frequentados;
- FIS2 – soma da classificação das disciplinas de Física concluídas nos dois primeiros anos frequentados;
- QUIM2 – soma da classificação das disciplinas de Química concluídas nos dois primeiros anos frequentados;
- OUT2 – soma da classificação de Outras disciplinas concluídas nos dois primeiros anos frequentados.



	Output		Input				Disciplinas do 1º Ano Lectivo concluidas no 1º ou 2º ano frequentado									Disciplinas do 2º Ano Lectivo concluidas no 2º ano frequentado								
Aluno	DTC	NFC	MAT2	FIS2	QUI2	OUT2	AMI	ALGA	DGA	ICP	Q1	AM2	F1	Q2	TO	MA 1	CM 1	F 2	CN	AM 3	AM 4	PEC	F 3	DF	GI
							Nota	Nota	Nota	Nota	Nota	Nota	Nota	Nota	Nota	Nota	Nota	Nota	Nota	Nota	Nota	Nota	Nota	Nota	Nota
966	5	15	81	63	23	58	11	11	12	18	13	11	14	10	12	12	0	14	11	12	11	14	11	12	16
972	7	13	57	42	22	67	13	12	11	15	12	10	16	10	12	0	15	13	10	12	0	0	0	13	14
986	5	14	90	67	26	73	10	12	14	18	16	12	11	10	13	13	12	15	13	14	13	16	13	15	16
988	7	15	75	50	26	66	13	14	10	20	15	14	14	11	13	13	10	0	12	12	0	10	13	10	13
1117	6	14	68	39	24	73	10	11	14	11	13	13	15	11	15	10	17	14	10	12	12	0	0	0	16

Data 1º Ano Frequentado | Data 2º Ano Frequentado

Figura 3.4: Dados dos dois primeiros anos frequentados

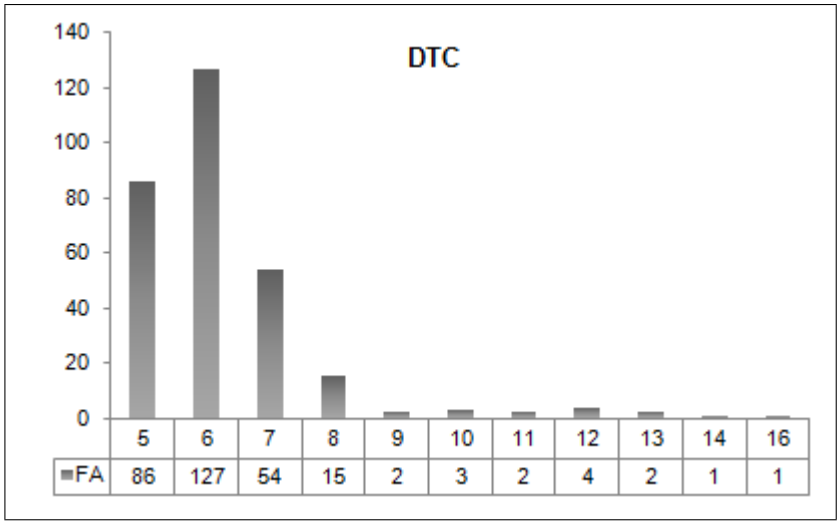
3.2.2 Estatística Descritiva e Ajuste de *Outliers*

Para cada umas das variáveis procedeu-se à análise da sua estatística descritiva e ao ajustamento dos *outliers*. O objectivo deste procedimento é validar os dados e detectar possíveis erros na recolha e tratamento dos mesmo, mas também transformar os dados de modo a acelerar e melhorar o processo de aprendizagem na rede.

Variáveis de Saída (*Output*)

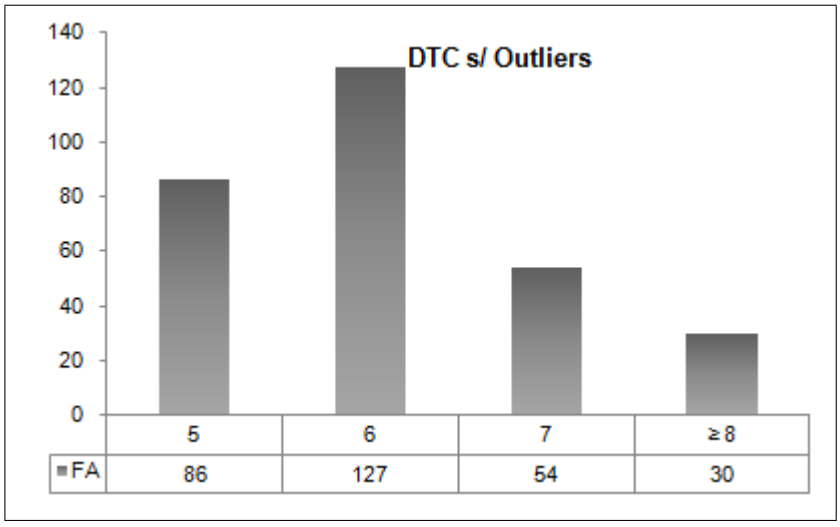
DURAÇÃO TOTAL DO CURSO

A duração total do curso (DTC), representa o número de anos que o aluno frequentou até finalizar o curso. Sendo o curso de uma duração mínima de cinco anos, esta variável toma valores no intervalo $[5, 16]$ anos. A figura 3.5 representa o histograma da DTC e as respectivas estatísticas descritivas.



Estatística Descritiva	
Média	6,276
Desvio Padrão	1,537
Variância	2,363
Mínimo	5
Máximo	16
Amplitude	11
Coef. Simetria	2,866
Coef. Kurtosis	11,137

Figura 3.5: Histograma e Estatística Descritiva de DTC



Estatística Descritiva	
Média	6,094
Variância	0,869
Desvio Padrão	0,932
Mínimo	5
Máximo	8
Amplitude	3
Coef. Simetria	0,566
Coef. Kurtosis	-0,498

Figura 3.6: Histograma e Estatística Descritiva de DTC (sem *outliers*)

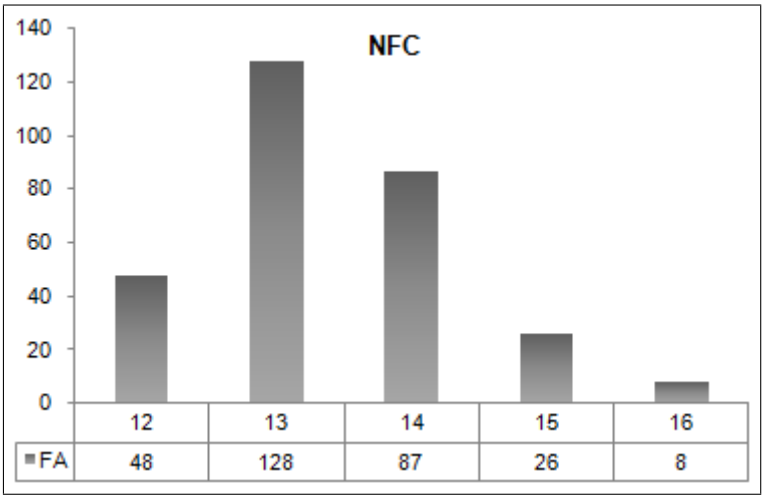
Como se pode verificar no histograma da figura 3.5 as classes acima de oito (anos) têm baixa frequência absoluta (FA) quando comparadas com as restantes classes, repre-

sentando menos de 5% do total da população. Por este motivo, faz sentido eliminar as classes com DTC superior a oito agrupando-as à classe oito.

A figura 3.6 representa o histograma e respectivas estatísticas básicas da DTC depois do ajustamento dos *outliers*.

NOTA FINAL DO CURSO

Os dos últimos cenários testados (Cenário 3, 3A e 4) tem como *output* a variável Nota Final de Curso (NFC). A NFC representa a nota final de conclusão do curso do aluno, de acordo com os critérios definidos à data de conclusão do mesmo. A figura 3.7 representa o histograma e respectivas estatísticas descritivas da NFC.



Estatística Descritiva	
Média	13,387
Variância	0,9
Desvio Padrão	0,949
Mínimo	12
Máximo	16
Amplitude	4
Coef. Simetria	0,542
Coef. Kurtosis	0,124

Figura 3.7: Histograma e Estatística Descritiva de NFC

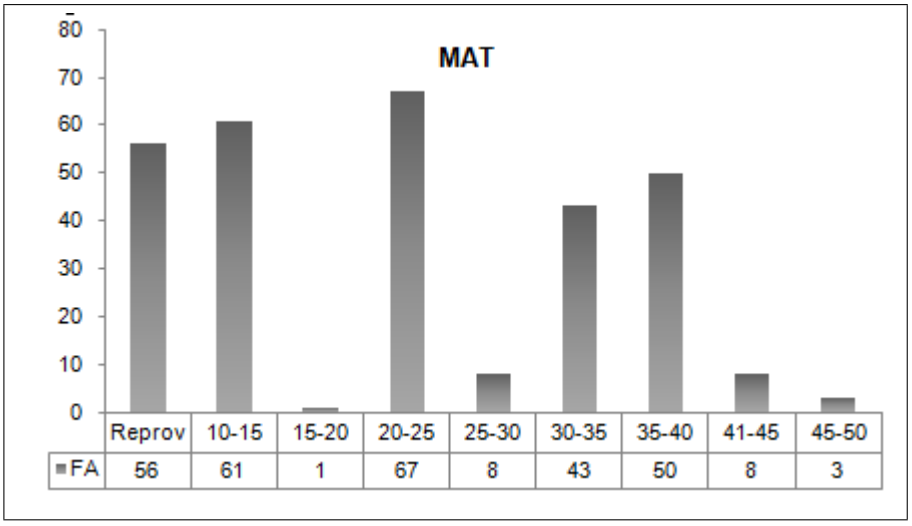
Analisando o histograma verificamos que a classe dezasseis, com menos de 3% da população total de alunos, poderia ser eventualmente considerada como *outlier* e agrupa-la à classe quinze. Por não se considerar essencial não optamos por esta abordagem, tendo sido mantida a amostra original para a NFC.

Variáveis de Entrada (*Input*)

MAT E MAT2

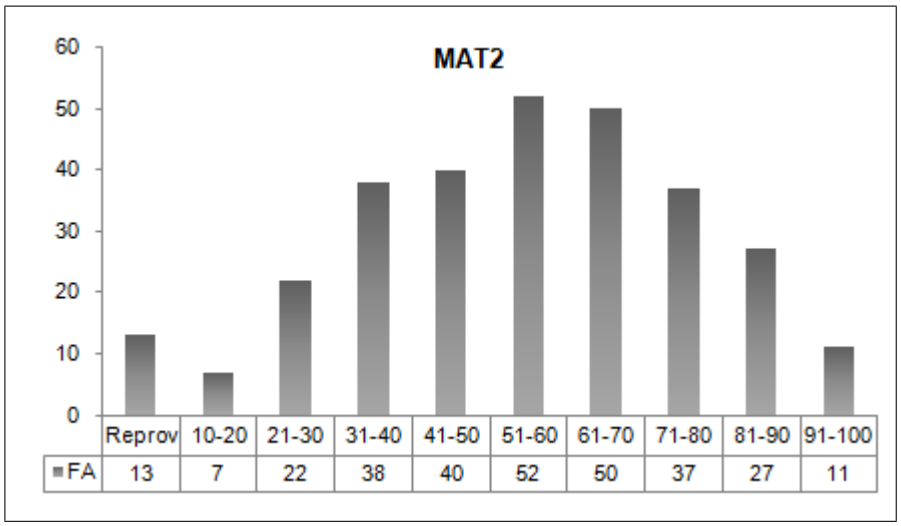
Como já descrito anteriormente, as variáveis MAT e MAT2 representam a soma das classificações obtidas nas disciplinas de Matemática concluídas no primeiro ano frequentado e nos dois primeiros anos frequentados, respectivamente.

As figuras 3.8 e 3.9 apresentam os histogramas e estatísticas descritivas das variáveis de *input* MAT e MAT2, respectivamente.



Estatística Descritiva	
Média	20,118
Variância	183,03
Desvio Padrão	13,529
Mínimo	0
Máximo	49
Amplitude	49
Coef. Simetria	-0,088
Coef. Kurtosis	-1,2

Figura 3.8: Histograma e Estatística Descritiva de MAT



Estatística Descritiva	
Média	53,7
Variância	529,967
Desvio Padrão	23,021
Mínimo	0
Máximo	100
Amplitude	100
Coef. Simetria	-0,391
Coef. Kurtosis	-0,31

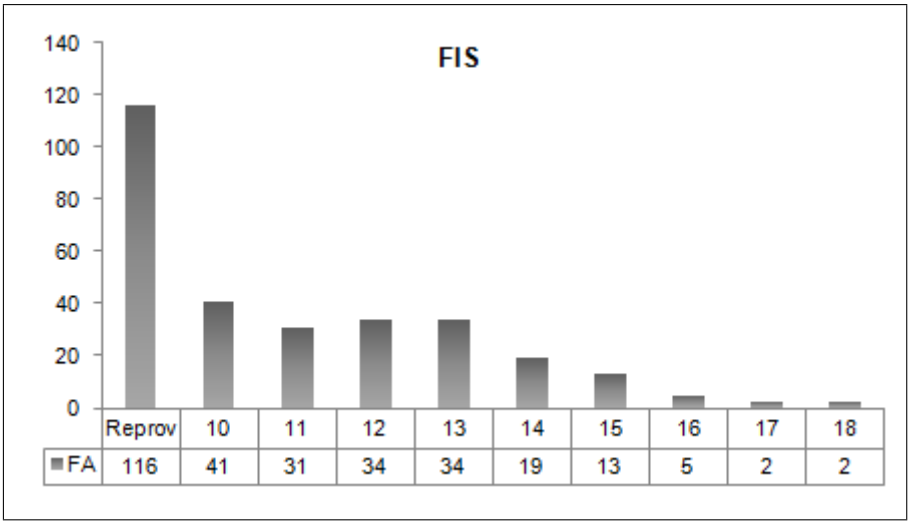
Figura 3.9: Histograma e Estatística Descritiva de MAT2

Não foram efectuados quaisquer ajustes às variáveis MAT e MAT2, por não se verificar essa necessidade.

FIS E FIS2

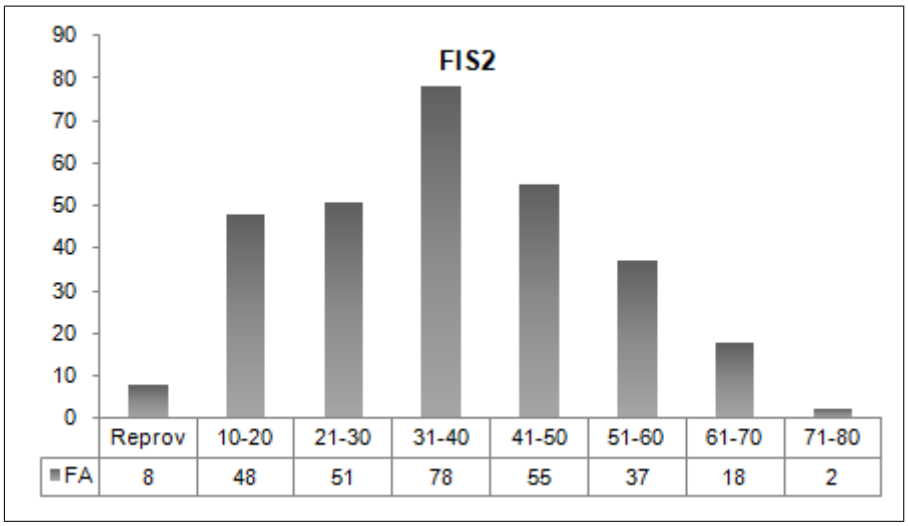
As variáveis FIS e FIS2, como já foi descrito anteriormente, são a soma das classificações obtidas nas disciplinas de Física concluídas no primeiro ano frequentado e nos dois primeiros anos frequentados, respectivamente.

As figuras 3.10 e 3.11 apresentam os histogramas e estatísticas descritivas das variáveis de *input* FIS e FIS2, respectivamente.



Estatística Descritiva	
Média	7,448
Variância	37,748
Desvio Padrão	6,144
Mínimo	0
Máximo	18
Amplitude	18
Coef. Simetria	-0,274
Coef. Kurtosis	-1,685

Figura 3.10: Histograma e Estatística Descritiva da variável FIS



Estatística Descritiva	
Média	34,906
Variância	268,194
Desvio Padrão	16,377
Mínimo	0
Máximo	78
Amplitude	78
Coef. Simetria	-0,04
Coef. Kurtosis	-0,615

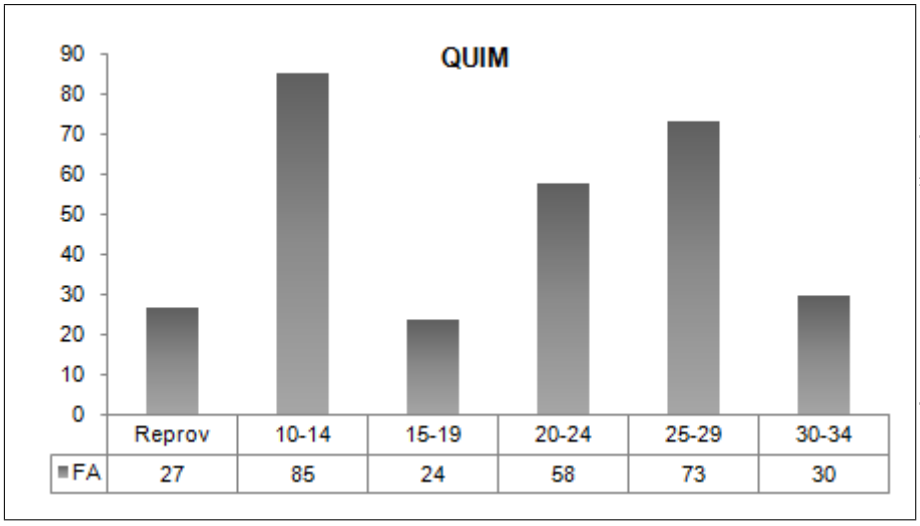
Figura 3.11: Histograma e Estatística Descritiva de FIS2

Para nenhuma destas variáveis se considerou a existência de *outliers* pelo que não houve necessidade de ajuste das mesmas.

QUIM E QUIM2

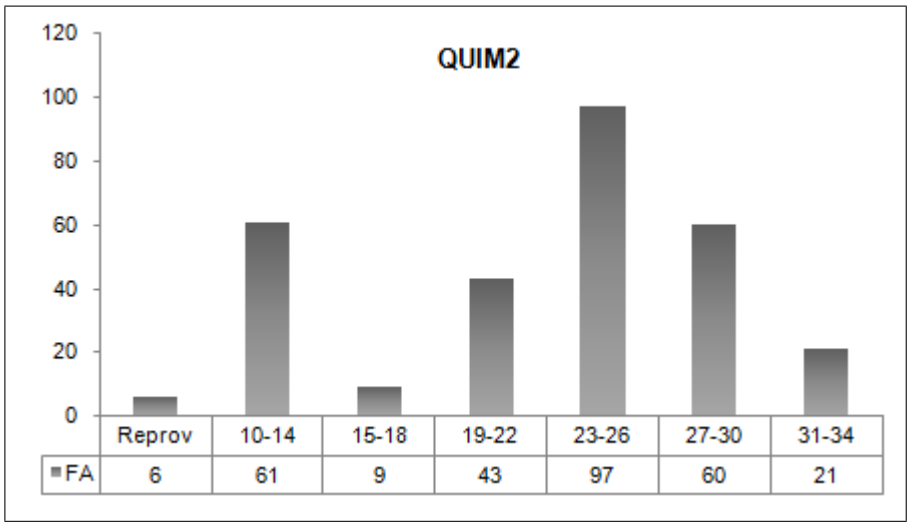
As variáveis QUIM e QUIM2, à semelhança das variáveis anteriores, resultam da soma das classificações obtidas nas disciplinas de Química concluídas no primeiro ano frequentado e nos dois primeiros anos frequentados, respectivamente.

As figuras 3.12 e 3.13 representam os histogramas e estatísticas descritivas das variáveis de *input* QUIM e QUIM2, respectivamente.



Estatística Descritiva	
Média	18,761
Variância	79,419
Desvio Padrão	8,912
Mínimo	0
Máximo	34
Amplitude	34
Coef. Simetria	-0,527
Coef. Kurtosis	-0,515

Figura 3.12: Histograma e Estatística Descritiva da variável QUIM



Estatística Descritiva	
Média	21,976
Variância	50,942
Desvio Padrão	7,137
Mínimo	0
Máximo	34
Amplitude	34
Coef. Simetria	-0,866
Coef. Kurtosis	0,272

Figura 3.13: Histograma e Estatística Descritiva de QUIM2

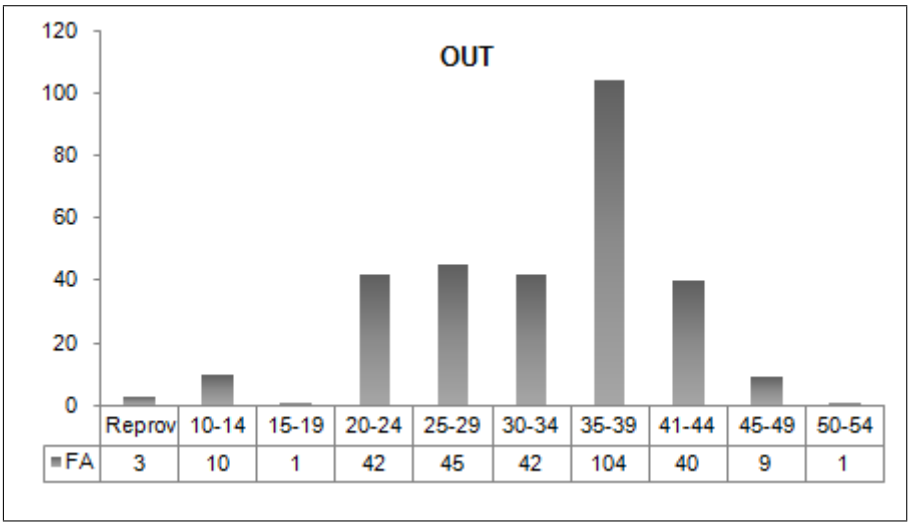
Analisando os histogramas e as respectivas estatísticas descritivas não se verificou a necessidade de efectuar qualquer tipo de ajuste nas variáveis QUIM e QUIM2.

OUT E OUT2

As variáveis OUT e OUT2 representam a soma das classificações obtidas em outras disciplinas que não tenham sido consideradas como disciplinas de Matemática, Física ou Química, concluídas no primeiro ano frequentado e nos dois primeiros anos frequentados,

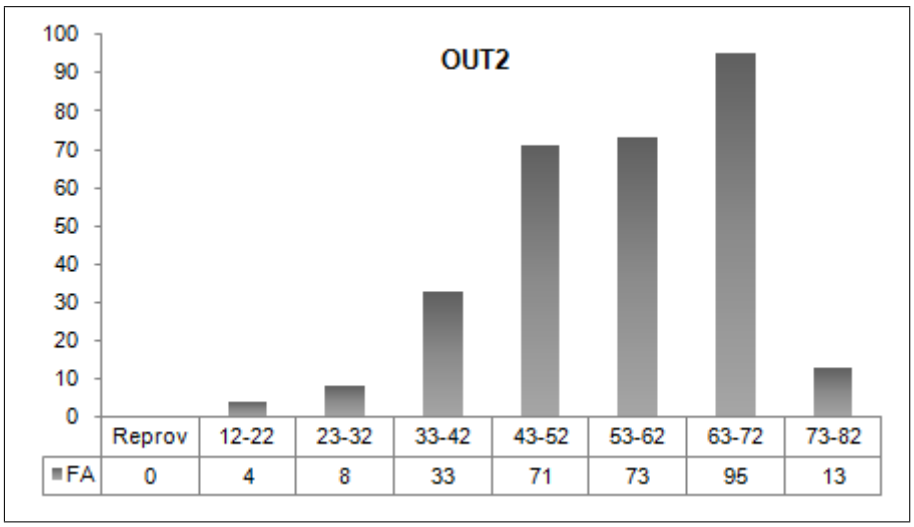
respectivamente.

As figuras 3.14 e 3.15 representam os histogramas e estatísticas descritivas das variáveis de *input* OUT e OUT2, respectivamente.



Estatística Descritiva	
Média	32,33
Variância	73,499
Desvio Padrão	8,573
Mínimo	0
Máximo	52
Amplitude	52
Coef. Simetria	-0,946
Coef. Kurtosis	1,249

Figura 3.14: Histograma e Estatística Descritiva de OUT



Estatística Descritiva	
Média	55,599
Variância	152,977
Desvio Padrão	12,368
Mínimo	12
Máximo	82
Amplitude	70
Coef. Simetria	-0,657
Coef. Kurtosis	0,056

Figura 3.15: Histograma e Estatística Descritiva de OUT2

AING

A variável AING representa o ano de ingresso do aluno no curso em estudo. O ano de ingresso do conjunto dos 297 alunos considerados, cuja licenciatura foi concluída no período dos 20 anos considerados, varia entre 1986 e 2000.

A figura 3.16 representa o histograma da variável de *input* AING.

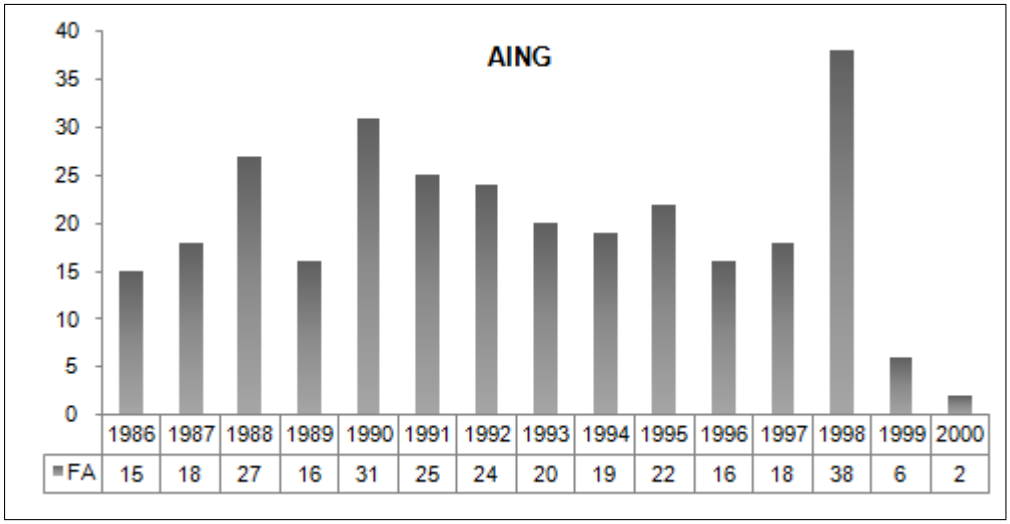


Figura 3.16: Histograma da variável AING

3.2.3 Correlação das variáveis

Sendo o objectivo do caso em estudo prever o comportamento das variáveis DTC e NFC, com base no comportamento das variáveis MAT, FIS, QUIM e OUT (ou MAT2, FIS2, QUIM2 e OUT2) é importante a análise da correlação, não só entre as variáveis de *input* e *output*, mas também entre si. A existência de uma forte correlação entre as variáveis de *input* e *output* da RNA que se pretende treinar pode ser uma boa motivação para o caso em estudo, uma vez que, na presença de uma maior dependência entre estas variáveis será espectável melhores resultados na previsão de DTC e/ou NFC. Além disso, o estudo da correlação das variáveis de *input*, e *output*, entre si, contribuí para a definição dos diferentes cenários, na medida em que, na presença de forte correlações podemos optar por eliminar variáveis que não acrescentem contributo para o modelo em causa.

O conceito de correlação está associado à relação entre duas variáveis e a sua intensidade, podendo ser avaliado através de coeficiente de correlação ou graficamente através de gráficos de dispersão, para variáveis métricas, ou através de outra técnicas de representação gráfica, como por exemplo, histogramas da FA e/ou da FR da distribuição de uma variável condicionada à outra. O coeficiente de *Pearson*, é a medida de correlação linear mais usual e avalia a intensidade e direcção (negativa ou positiva) das relações lineares entre variáveis.

O coeficiente de *Pearson* varia entre -1 (correlação linear perfeita negativa) e 1 (correlação linear perfeita positiva), onde 0 significa ausência de correlação linear (podem existir relações não lineares) entre as variáveis [Murteira et al., 2010].

A tabela 3.1 apresenta os coeficientes de correlação linear das variáveis de *input* e de *output* no contexto do Cenário 1, em que se considera apenas a informação relativa ao primeiro ano frequentado.

	DTC	**NFC**	**AING**	**MAT**	**FIS**	**QUIN**	**OUT**
DTC	1						
NFC	-0,31	1					
AING	-0,056	-0,276	1				
MAT	-0,404	0,44	-0,206	1			
FIS	-0,05	0,204	-0,179	0,317	1		
QUIM	-0,236	0,34	-0,34	0,444	0,238	1	
OUT	-0,089	0,149	0,017	0,213	0,31	0,168	1

Tabela 3.1: Matriz de Correlação de *inputs* e *outputs* (Cenário 1)

Observando a tabela 3.1 podemos verificar que entre as variáveis de *input* MAT e QUIM se verifica uma correlação positiva mais elevada do que para os restantes pares de variáveis de *input*. MAT e QUIM são também as variáveis de *input* que apresentam uma maior correlação negativa com DTC, o que significa que valores elevados de DTC estão mais frequentemente associados a valores baixos de MAT e QUIM (e vice-versa).

Como já seria de esperar, uma vez que DTC e NFC têm uma correlação moderada negativa, NFC apresenta também uma correlação moderada positiva com MAT e QUIM. Isto significa que valores elevados de NFC estão mais frequentemente associados a valores elevados de MAT e QUIM (e vice-versa).

A moderada correlação de NFC e DTC com a variável MAT pode ser explicado pelo facto da licenciatura em causa ser constituída por uma forte componente de Matemática, não só pelo número de disciplinas nesta área elevado (três em nove no primeiro ano curricular) mas também devido ao elevado grau de dificuldade a que estas disciplinas estão geralmente associadas face à preparação dos alunos. Em geral, alunos com um bom desempenho nas componentes de matemática e química terão uma maior capacidade e naturalmente, apresentam um melhor desempenho global.

As variáveis FIS e OUT apresentam uma correlação fraca com NFC e DTC, sendo esta mais evidente com DTC.

Analisando a matriz de correlações relativamente à variável de *input* AING, verificamos uma correlação fraca com a variável de *output* DTC e moderada com NFC.

Observando as figuras 3.17 e 3.18, podemos facilmente observar que, à excepção dos anos 1999 e 2000, que devido à sua reduzida amostra não são relevantes, existe

uma clara tendência de aumento das classificações mais baixas associadas a uma diminuição das mais altas em anos mais recentes, o que pode indicar uma menor preparação desses alunos quando comparados com os alunos dos anos anteriores.

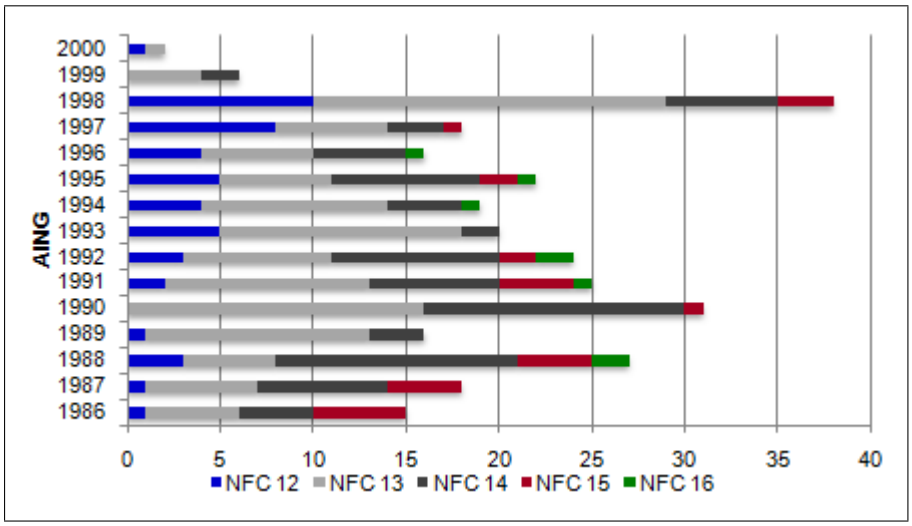


Figura 3.17: Frequência Absoluta de NFC condicionada por AING

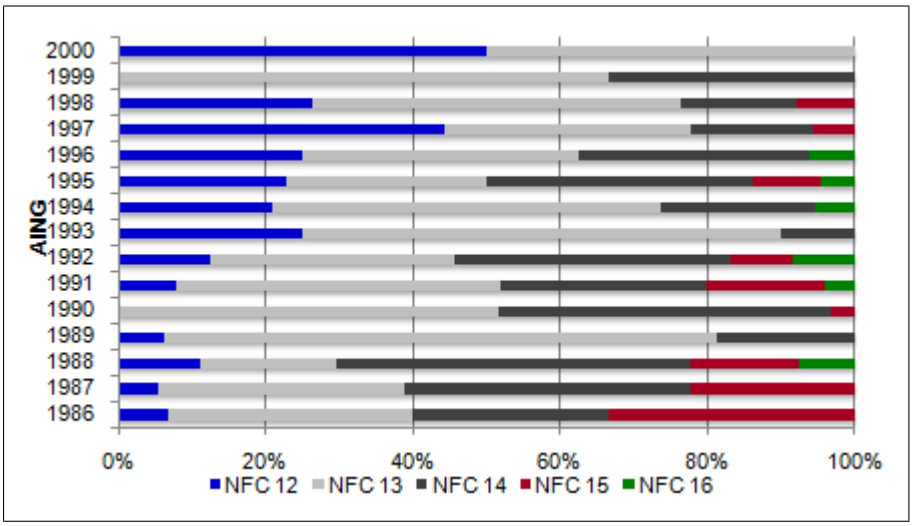


Figura 3.18: Frequência Relativa de NFC condicionada por AING

Na figura 3.19, onde se representa a NFC média por AING, podemos confirmar que existe uma tendência decrescente de NFC ao longo do período de vinte anos considerado. Este facto parece indicar que pode ser relevante a consideração da variável AING como variável de *input* no contexto do Cenário 3.

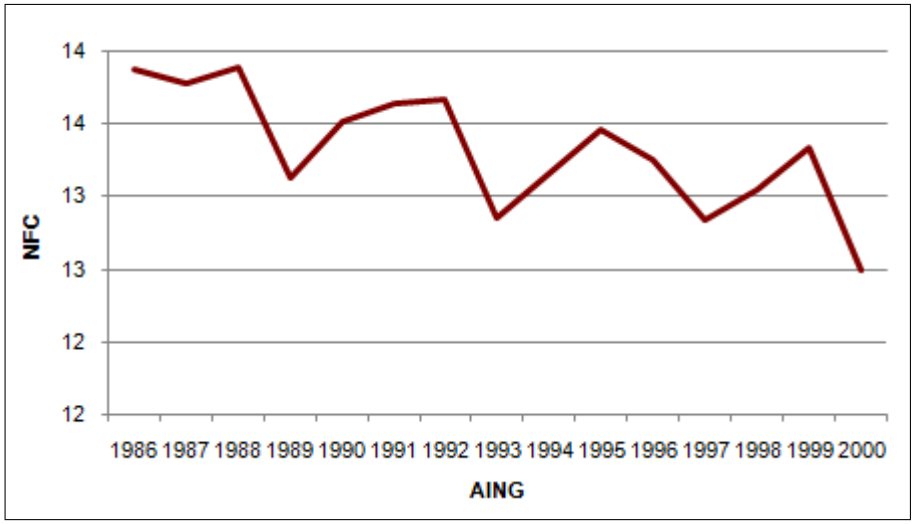


Figura 3.19: Evolução da NFC média por AING

Analisando as figuras 3.20 e 3.21 com FA e FR de DTC condicionada por AING, respectivamente, confirmamos a não existência, pelo menos de uma forma evidente, de alguma tendência na variação de DTC ao longo do período dos vinte anos.

Ainda na tabela 3.1 podemos constar que para as variáveis de *output*, DTC e NFC, se observa uma correlação negativa moderada de $-0,310$ entre estas duas variáveis. Isto significa que valores elevados de DTC têm maior probabilidade de estar associados a valores baixos de NFC. O gráfico da figura 3.22 apresenta a FA de NFC condicionada por DTC. Como podemos confirmar existe uma tendência de crescimento das notas baixas e diminuição das notas altas, com o aumento de DTC. Este facto pode ser relevante no contexto do Cenário 4, onde se considera a variável DTC como variável de *input* para a previsão de NFC.

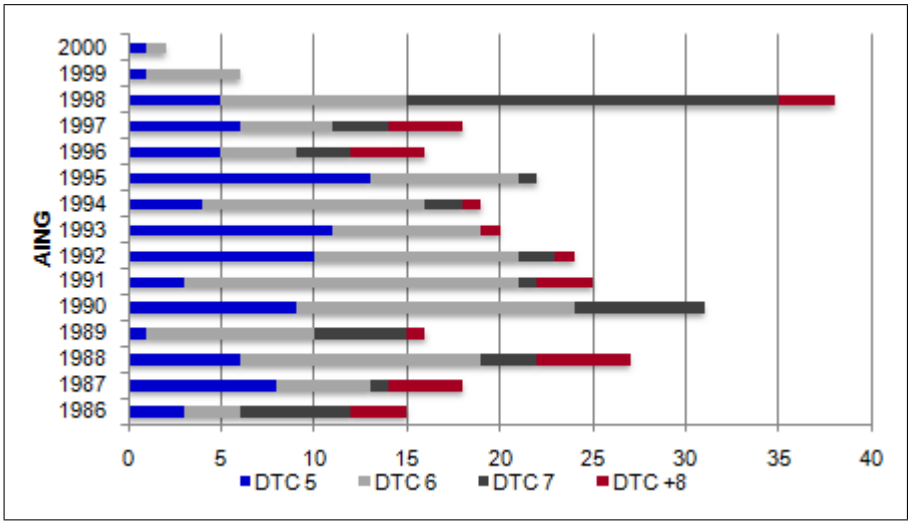


Figura 3.20: Frequência Absoluta de DTC condicionada por AING

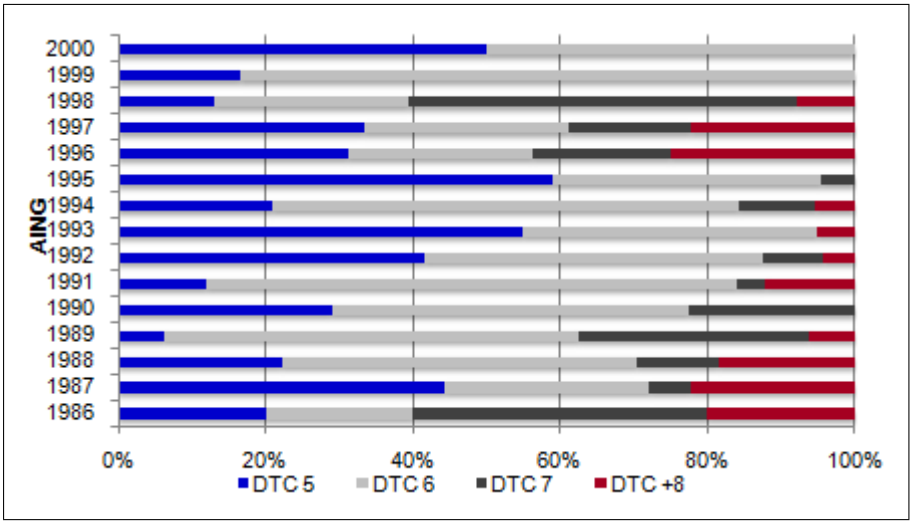


Figura 3.21: Frequência Relativa de DTC condicionada por AING

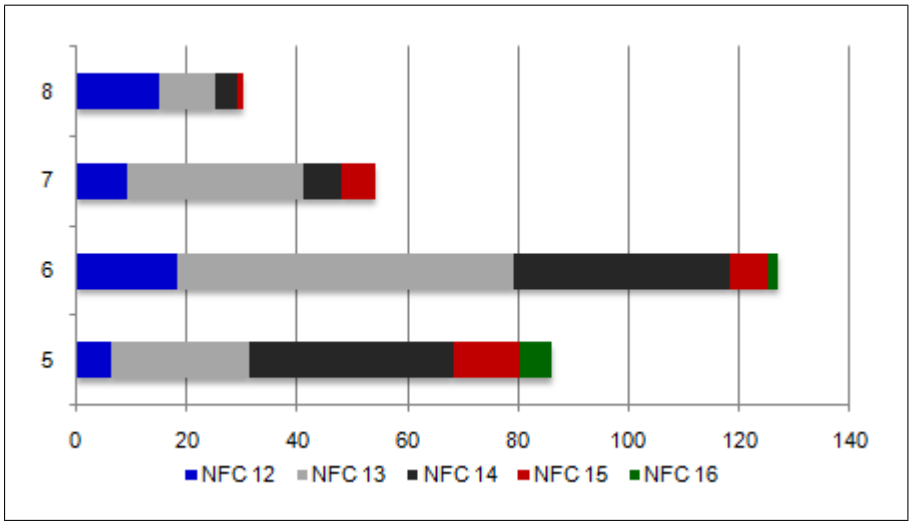


Figura 3.22: Frequência Absoluta de NFC condicionada por DTC

A tabela 3.2 apresenta os coeficientes de correlação linear das variáveis de *input* e de *output* no contexto dos cenários que consideram a informação relativa aos dois primeiros anos frequentados.

Analisando a tabela 3.2, constatamos que as características verificadas para os coeficientes de correlação no contexto do Cenário 1 também se verificam, mas neste caso de forma mais expressiva. MAT2 e QUIM2 apresentam também uma moderada correlação positiva entre si. No entanto, neste caso a correlação positiva entre MAT2 e FIS2 é a mais evidente. Este facto pode ser explicado no curso considerado pela forte componente de Física, superior à de Química, do segundo ano curricular com quatro disciplinas de Física e nenhuma de Química.

	DTC	NFC	AING	MAT2	FIS2	QUIM2	OUT2
DTC	1						
NFC	-0,31	1					
AING	-0,056	-0,276	1				
MAT2	-0,534	0,439	-0,123	1			
FIS2	-0,398	0,448	-0,236	0,597	1		
QUIM2	-0,281	0,393	-0,321	0,478	0,516	1	
OUT2	-0,304	0,374	-0,286	0,488	0,548	0,431	1

Tabela 3.2: Matriz de Correlação de *inputs* e *output* (Cenário 2)

DTC e NFC apresentam uma correlação moderada negativa com as variáveis de *input*, em particular com MAT2 e FIS2. Quando se compara os coeficientes de correlação linear apresentados na tabela 3.1, pode-se constatar que os valores correspondentes da tabela 3.2, em geral, aumentaram, o que se justifica pelo aumento da amostra, agora correspondente aos dois primeiros anos frequentados.

Podemos concluir da análise da correlação entre as variáveis de *input* e *output*, em ambos os cenários, a existência de uma correlação suficiente para justificar o caso em estudo. Em particular, verificou-se que será mais relevante considerar o cenário alternativo, de inclusão de AING como *input*, para o Cenário 3. Conclui-se também que nenhuma correlação entre variáveis de *input* é suficientemente forte para justificar que algumas destas variáveis deva ser eliminado dos cenários a testar. Por este motivo optou-se por não eliminar nenhuma das variáveis apresentadas.

3.2.4 Mudança de Escala

O objectivo na mudança de escala dos dados é transforma-los de modo a acelerar e melhorar a performance do processo de aprendizagem da rede. Em geral, a mudança de escala depende do tipo de dados e do algoritmo de aprendizagem [Swingler, 1996].

No caso particular, de algoritmos de gradiente descendente, como é o caso do algoritmo de RP, para cada variável de *input* deve ser efectuada a mudança de escala para que a sua média no conjunto de casos de treino esteja próxima de zero, pelo que a mudança de escala aconselhada é a para o intervalo $[-1, 1]$.

Existem diversas formas de efectuar a mudança de escala dos dados [Swingler, 1996] [Alves, 2002], das quais referimos as duas principais:

- Normalização

  Uma variável pode ser normalizada para a média igual a zero e desvio padrão igual a um aplicando a seguinte função para cada elemento da amostra:

$$x' = \frac{x - \overline{X}}{\sigma}, \tag{3.1}$$

  onde $\overline{X}$ é a média de X e σ é o desvio padrão de X.

- Valor Mínimo -1 e Valor Máximo 1

  Para efectuar uma mudança de escala para o intervalo $[-1, 1]$ aplica-se a seguinte função a cada elemento da amostra:

$$x' = \frac{x - \frac{\max + \min}{2}}{\frac{\max - \min}{2}}, \tag{3.2}$$

  onde $\min$ e $\max$ são o valor mínimo e máximo da variável x, respectivamente.

A mudança de escala das variáveis de *output* também é muito importante para a eficácia no processo de aprendizagem. Em primeiro lugar, no caso em que existe mais do que uma variável de *output*, e se a função de erro da rede é sensível à escala, como é o caso dos algoritmos de gradiente descendente, a diferença de escalas das variáveis de *output* podem influenciar a performance da aprendizagem da rede. Se as escalas das variáveis de *output* forem muito diferentes (por exemplo uma de 1 a 100 e outra de 1 a 10000) o algoritmo irá utilizar a maior parte do esforço na aprendizagem da variável com maior escala. Assim, saídas com igual importância devem ser alteradas para a mesma escala utilizando, por exemplo a equação 3.1.

Em segundo lugar, pode-se querer ajustar as variáveis de *output* aos valores do contradomínio da função activação, em particular ao da função logística $[0, 1]$. Neste caso podemos utilizar a seguinte função:

$$y = \frac{(x - \min)(B - A)}{\max - \min} + A, \tag{3.3}$$

para uma mudança de escala de domínio $[A, B]$, com *min* e *max* o limite máximo e mínimo, respectivamente, da variável. Para mudança de escala da equação 3.3 é essencial o conhecimento dos limites máximos e mínimos da variável.

No caso em estudo, para todas as variáveis de *input* e *output*, utilizou-se a mudança de escala utilizando a equação 3.2. O facto de não serem conhecidos os limites de todas

as variáveis, em particular da variável DTC, e de ser um procedimento simples, que nos permite efectuar a distribuição dos dados por todo o intervalo $[-1, 1]$ contribuiu para esta opção.

3.3 Definição do Modelo de RNA

3.3.1 Conjunto de Treino, Validação e Teste

A RNA que se pretende treinar deve ter como característica uma boa capacidade de generalização, como se referiu no Capítulo 2. Existem diversos métodos para avaliar a capacidade de generalização de uma RNA.

O método mais popular para a estimação do erro de generalização de uma RNA consiste em dividir os dados em três conjuntos: conjunto de treino (TRE), conjunto de validação (VAL) e conjunto de teste (TES). O conjunto de dados de treino é constituído pelos dados que são efectivamente alimentados à rede para que esta "aprenda", ou seja, ajuste os pesos e *bias*. O erro do conjunto de validação não é utilizado para ajustar os pesos e *bias*, mas é monitorizado durante o processo de treino. Quando o erro do conjunto de validação aumenta sucessivamente mais do que um número fixo de vezes, designado por "número máximo de falhas" que é tipicamente igual a seis, o treino pára. O conjunto de teste não contribui directa nem indirectamente para a aprendizagem da RNA, mas é utilizado para comparar diferentes modelo, ou cenários do caso em estudo.

A divisão dos dados pode ainda ser realizada de forma aleatória (*rand*) ou em bloco (*block*). Isto é, se considerarmos, por exemplo, uma divisão de 60%, 25% e 15% para TRE, VAL, TES, respectivamente, na divisão *rand* os dados da amostra que pertencem a cada um dos conjuntos são seleccionados aleatoriamente, enquanto na divisão *block*, são seleccionados os primeiros 60% da amostra para TRE, os 25% seguintes para VAL e os restantes 15% para TES.

Por defeito os dados do problema são divididos aleatoriamente com 60% para TRE, 20% para VAL e 20% para TES.

Para determinar qual a divisão dos dados mais apropriada para o nosso caso em estudo, foram realizados dois grupos de testes: um com divisão *rand* e outro com divisão *block*. Para cada grupo, realizaram-se sete ciclos de experiências com diferentes divisões de dados, de dez repetições do treino de uma RNA *feed-forward*, com as seguintes características:

- *Inputs*: MAT, FIS, QUIM e OUT;
- *Output*: DTC;
- Número de camadas escondidas: 1;
- Número de neurónios na camada escondida: 5;
- Algoritmo de Aprendizagem: RP com Taxa de Aprendizagem Adaptativa com *Momentum*;
- Taxa de *Momentum*: 0,9;
- Função de Activação: sigmóide (*tansig*) para a camada escondida e linear (*purelin*) para a camada de *output*;
- Treino em lote;
- Medida De erro: MSE;
- Inicialização dos pesos: Algoritmo de Inicialização de Nguyen-Widrow;
- Número de repetições: 10.

A figura 3.23 apresenta a separação dos dados originais em conjunto de treino, teste e validação para os sete ciclos de experiências realizados.

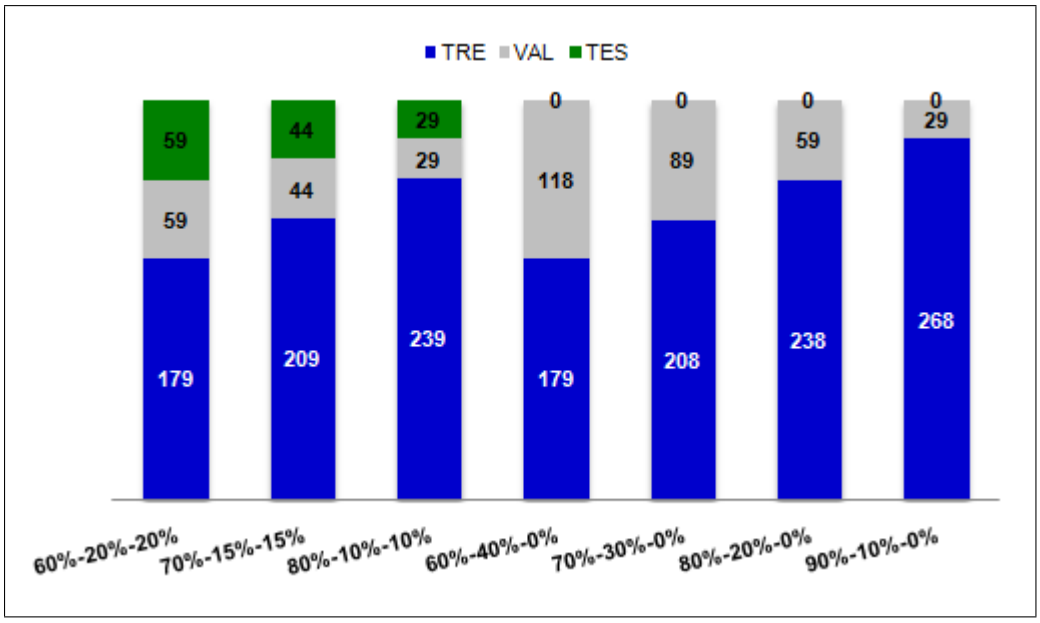


Figura 3.23: Divisão dos dados dos sete ciclos

A reduzida dimensão da amostra levou-nos a contemplar a hipótese de suprimir o conjunto de teste, uma vez que este não contribui para o treino da rede, e utilizar o conjunto de validação para comparar diferentes cenários. Nestas circunstancias, estaríamos a assumir que a RNA treinada se comportaria com novos dados de forma semelhante à do comportamento com o conjunto de validação. A RNA deve ser avaliada pela sua capacidade de resposta na presença de novos dados que não tenham sido apresentados durante o seu treino, ou seja, no conjunto de teste. No entanto, na presença de uma amostra reduzida podemos suprimir o conjunto de teste.

Para cada ciclo (dez repetições de treino da RNA) das sete experiências, dos dois grupos (*rand* e *block*), registou-se o MSE obtido para o conjunto de TRE (MSE_{tr}), de VAL (MSE_{val}) e TES (MSE_{te}).

Para consulta dos resultados integrais dos teste realizados para a divisão dos dados consultar as tabelas A.1 e A.2 em apêndice.

O código utilizado na realização das experiências para a divisão dos dados em MATLAB pode ser consultado nas figuras B.1 e B.2 em apêndice.

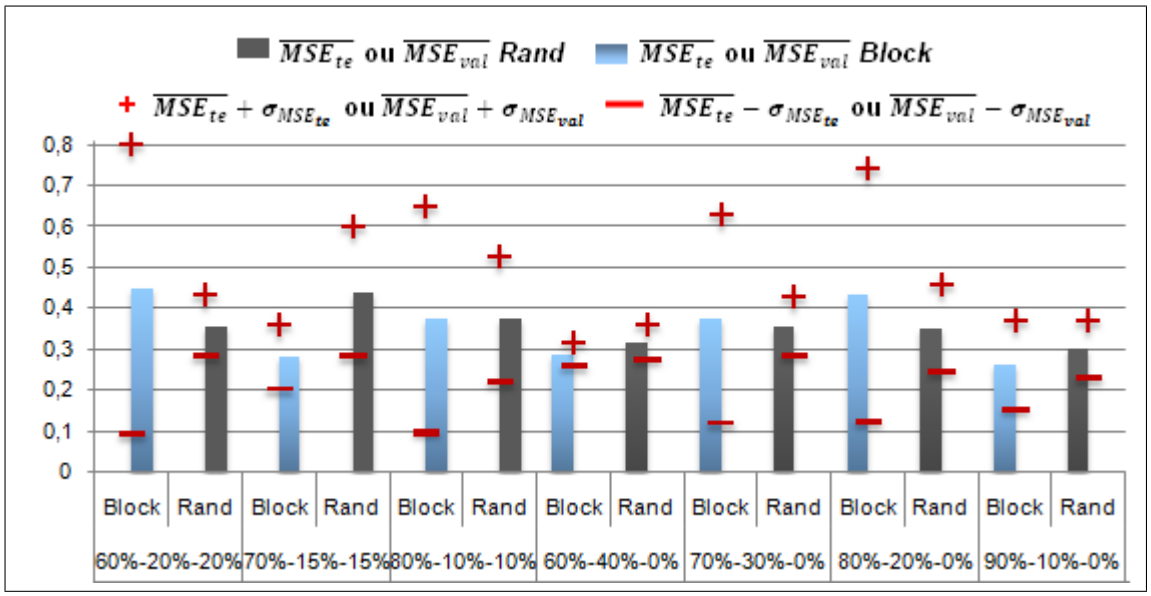


Figura 3.24: Resultados das Experiências para Divisão de Dados

A figura 3.24 apresenta os resultados para os sete ciclos de experiências realizadas com diferentes divisões de dados, para os dois grupos. As barras do gráfico são o valor médio de MSE_{te}, $(\overline{MSE}_{te})$, nos casos em que o conjunto de teste foi considerado, ou de MSE_{val} $(\overline{MSE}_{val})$, para os casos em que não existe conjunto de teste. Os símbolos + e - representam o valor de $\overline{MSE}_{te} + \sigma_{MSE_{te}}$ e $\overline{MSE}_{te} - \sigma_{MSE_{te}}$, respectivamente, ou no caso de ausência de conjunto de teste, o valor de $\overline{MSE}_{val} + \sigma_{MSE_{val}}$ e $\overline{MSE}_{val} - \sigma_{MSE_{val}}$, respectivamente.

Analisando a figura 3.24 podemos observar que em quatro dos sete ciclos de experiências realizadas, a divisão *block* se traduz em piores resultados do que a divisão *rand*. Adicionalmente constata-se que, em geral, a divisão *block* está associada a uma maior variabilidade dos resultados.

Nas figuras 3.25 e 3.26 apresentam os *boxplots* (caixa de bigodes) do MSE_{te} dos sete ciclos de experiências para os quais se considerou a existência de conjunto de teste, para o grupo de divisão *rand* e *block*, respectivamente.

Nas figuras 3.27 e 3.28 apresentam os *boxplots* (caixa de bigodes) do MSE_{val} dos sete ciclos de experiências para o grupo de divisão *rand* e *block*, respectivamente.

Comparando os *boxplots*, entre a divisão de dados *rand* e *block*, não se verifica grandes diferenças. No contexto do Cenário 1A e 2A, onde se contempla a hipótese de inclusão da variável AING como input, a comparação da divisão dos dados *block* ou *rand* torna-se relevante. Estando a amostra dos 297 dados ordenada cronologicamente, isto é, ordenada por ano de ingresso, caso se verificasse efectivamente uma tendência de variação do tipo de alunos ao longo dos anos, então seria de esperar que houvesse maior dispersão dos resultados, ou seja *boxplots* mais "extensos", na divisão de dados *block*, uma vez que neste tipo de divisão os dados são atribuídos a TRE, VAL e TES, sequencialmente de acordo com a sua ordenação. No entanto, também já se tinha verificado que esta seria uma hipótese remota, visto existir fraca correlação entre as variáveis DTC e AING e a ausência de alguma tendência na DTC ao longo dos vinte anos (figuras 3.20 e 3.21).

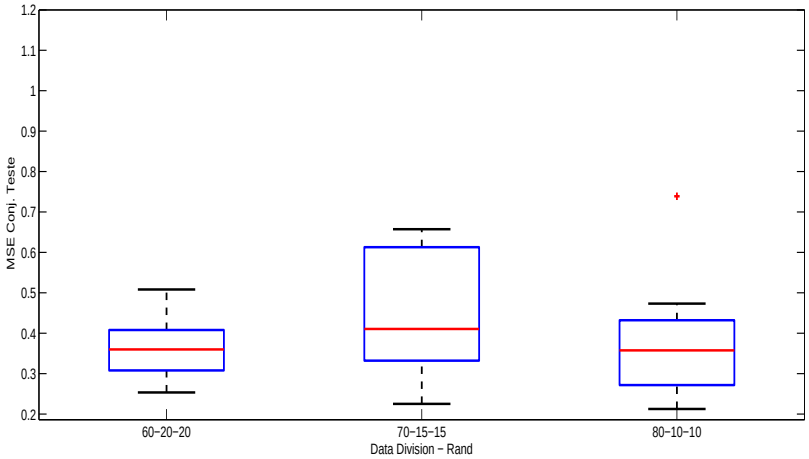


Figura 3.25: Divisão de Dados *rand*: MSE_{te}

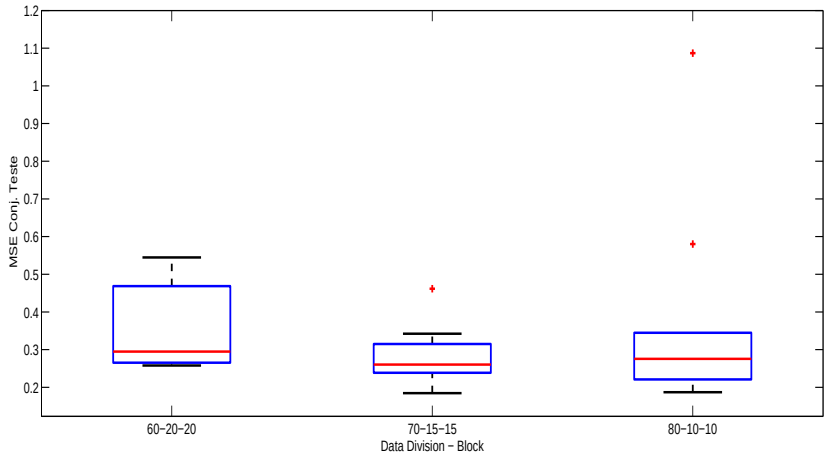


Figura 3.26: Divisão de Dados *block*: MSE_{te}

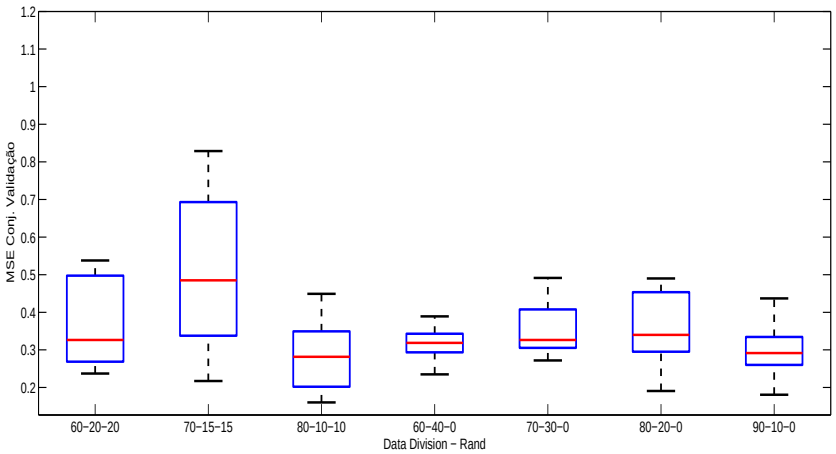


Figura 3.27: Divisão de Dados *rand*: MSE_{val}

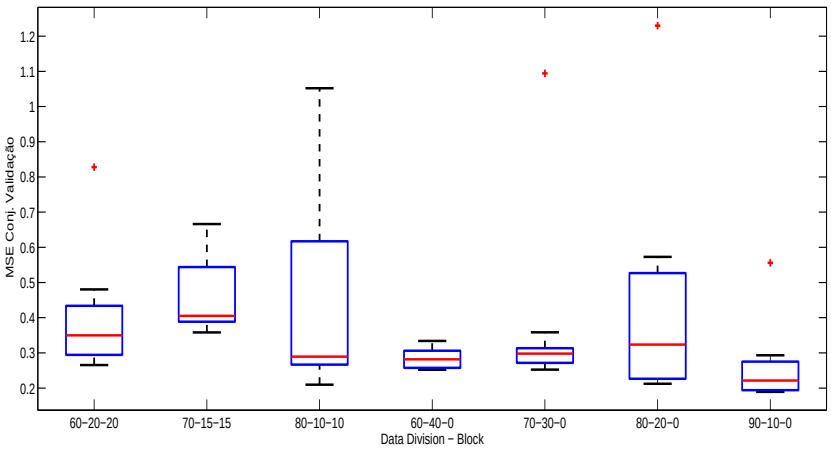


Figura 3.28: Divisão de Dados *block*: MSE_{val}

Tendo-se verificado uma melhoria, pouco significativa, na utilização da divisão de dados *rand*, mas não existindo evidência quanto à divisão da amostra para TRE, VAL e TES, e em consequência da reduzida dimensão da nossa amostra, optou-se por uma divisão do tipo *rand* com 80% para o TRE, 10% para VAL e 10% para TES.

3.3.2 Topologia da RNA

Número de Camadas Escondidas

A arquitectura (número de camadas, neurónios e conexões) de uma RNA depende principalmente do caso particular em estudo. No caso particular das RNA *feed-forward* multi-camada, normalmente opta-se apenas por uma camada intermédia, de forma a reduzir o número de parâmetros a ajustar, reduzindo deste modo a probabilidade de *overfitting*. Por este motivo, e tendo em conta o tamanho reduzida da amostra do caso em estudo, optamos por utilizar também apenas uma camada intermédia.

Número de Neurónios

Utilizadores de RNA têm sugerido que o número de parâmetros a estimar de uma rede não deve exceder uma décima parte da amostra de treino. Desta forma, sendo a nossa amostra constituída por 297 alunos,considerando 80% para TRE, o número máximo de parâmetros a estimar, ou seja de pesos associados aos neurónios, não deve exceder os vinte e quatro.

Se considerarmos apenas as quatro variáveis de *input* (Cenário 1, 2 ou 3), sem incluirmos *bias*, o número de parâmetros a estimar será igual a $4n + n = 5n$, onde n designa o número de neurónios da camada escondida. Assim, n poderá variar de um a cinco (arredondado às unidades) para os Cenários 1, 2 ou 3. No caso do Cenário 1A, 2A, 3A e 4, com mais uma variável de *input* (AING ou DTC) o número de neurónios na camada escondida não deve ultrapassar os quatro (arredondado às unidades). Considerando o *bias*, o número de parâmetros a estimar, no caso dos Cenário 1, 2 ou 3, altera para $6n$, ou seja, n poderá variar de um a quatro. No caso dos cenários 1A, 2A e 3A, ao considerar o *bias*, o número máximo de neurónios na camada escondida não deveria ultrapassar os três.

Por uma questão de uniformização das experiências e pelos motivos apresentados, foram efectuados testes para RNA *feed-forward* com uma camada escondida, com o número de neurónios da camada escondida a variar entre um e cinco.

Algoritmo de Aprendizagem

O algoritmo utilizado para treino das redes foi o algoritmo de aprendizagem com supervisão por Retropropagação do Erro com Taxa de Aprendizagem adaptativa com *Momentum*, utilizando o software Matllab com a função *traingdx* [Howard et al., 1992].

Taxa de Aprendizagem

No algoritmo de RP original a taxa de aprendizagem é mantida constante ao longo de todo o treino da rede. Como já se referiu no Capítulo 2, o desempenho do algoritmo de RP é muito sensível ao valor adequado da taxa de aprendizagem, podendo tornar-se instável para valores muito elevados, ou muito lento para valores muito baixos. Não é prático determinar o valor ideal da taxa de aprendizagem antes do treino. Na realidade, o próprio valor ideal vai-se alterando ao longo do processo de treino, à medida que o algoritmo percorre a superfície de erro.

A *performance* do algoritmo de RP pode ser melhorada se for permitido alterar a taxa de aprendizagem ao longo do processo de aprendizagem. Na função do MATLAB, *traingdx*, a taxa de aprendizagem adaptativa procura manter o tamanho "passo" de aprendizagem do algoritmo tão largo quanto possível, mantendo simultaneamente a rede estável.

O Algoritmo de RP com taxa de aprendizagem adaptativa com *momentum* é obtido do algoritmo de RP standard com algumas alterações [Howard et al., 1992]:

1. São calculados os *outputs* e *bias* iniciais, de acordo com a função de inicialização dos pesos;
2. Em cada época, os pesos e *bias* são ajustados com a taxa de aprendizagem e de *momentum* corrente;
3. São calculados novos *outputs* e *bias*;
4. No caso do erro corrente exceder o erro antigo por mais de um rácio pré-definido (tipicamente $1,04$), os novos pesos e *bias* são ignorados e a taxa de aprendizagem, com um valor inicial por defeito igual a $0,01$, é reduzida (tipicamente de 30%). Caso contrário, os novos pesos e *bias* são mantidos. Se o novo erro for menor do que o erro antigo, a taxa de aprendizado é aumentada (normalmente, de 5%).

O procedimento de actualização da taxa de aprendizagem permite aumentar a taxa de aprendizagem, mas apenas na medida possível que permita à rede aprender sem grande aumento do erro. Desde modo, é obtida uma taxa de aprendizagem próxima do ideal. Quando uma maior taxa de aprendizagem resulta numa aprendizagem mais estável, a taxa é aumentada. Quando a taxa de aprendizagem é demasiado elevada para garantir que erro diminua, é reduzida até que se obtenha novamente uma aprendizagem estável.

Taxa de *Momentum*

Como já referido no Capítulo anterior, tal como para taxa de aprendizagem, a determinação do valor adequado para a taxa de *momentum* não é uma tarefa fácil, e, na maioria das vezes, dependente do caso de estudo e é determinado empiricamente.

No algoritmo de RP com taxa de aprendizagem adaptativa com *momentum* o valor constante para a Taxa de *Momentum* é, por defeito, $0,9$. Este valor de taxa de *momentum* é um valor em geral sugerido pela maioria dos autores. [Rojas, 1996] aconselham a sua alteração apenas em situações em que seja óbvio a melhoria da *performance* da rede com outros valores.

Sabemos que um valor elevado de taxa de *momentum* irá reduzir o risco da pesquisa do mínimo do erro fique "presa" num mínimo local mas por outro lado, aumenta o risco de "ultrapassar" o valor mínimo da função que procuramos. Para verificar a necessidade de outro valor para a taxa de *momentum*, que não 0,9 foram realizados testes com diferentes valores de *momentum* para uma mesma RNA *feed-forward*, com as seguintes características:

- *Inputs*: MAT, FIS, QUIM e OUT;
- *Output*: DTC;
- Número de camadas escondidas: 1;
- Número de neurónios na camada escondida: 5;
- Algoritmo de Aprendizagem: RP com Taxa de Aprendizagem adaptativa com *Momentum*;
- Função de Activação: sigmóide (*tansig*) para a camada escondida e linear (*purelin*) para a camada de *output*;

- TRE, VAL, TES: 60%-20%-20%;
- Divisão dos Dados: *rand*;
- Treino em lote;
- Medida de erro: MSE;
- Inicialização dos pesos: Algoritmo de Inicialização de Nguyen-Widrow;
- Número de repetições: 10.

Foram realizados nove ciclos de experiências de dez repetições, com diferentes valores de taxa de *momentum* (α) onde se treinou uma RNA com as características acima mencionadas, e registou-se MSE_{tr}, MSE_{val} e MSE_{te}, obtido para cada treino.

O código utilizado na realização das experiências para a Taxa de *Momentum* em MATLAB pode ser consultado na figura B.3 em apêndice.

A figura 3.29 apresenta os resultados para os nove ciclos de experiências realizadas com diferentes valores de taxa de momentum. As barras do gráfico representam $\overline{MSE}_{te}$, os símbolos + e - representam o valor de $\overline{MSE}_{te} + \sigma_{MSE_{te}}$ e $\overline{MSE}_{te} - \sigma_{MSE_{te}}$, respectivamente.

Os resultados estatísticos integrais das experiências realizadas para a Taxa de Momentum podem ser consultados na tabela A.3 em apêndice.

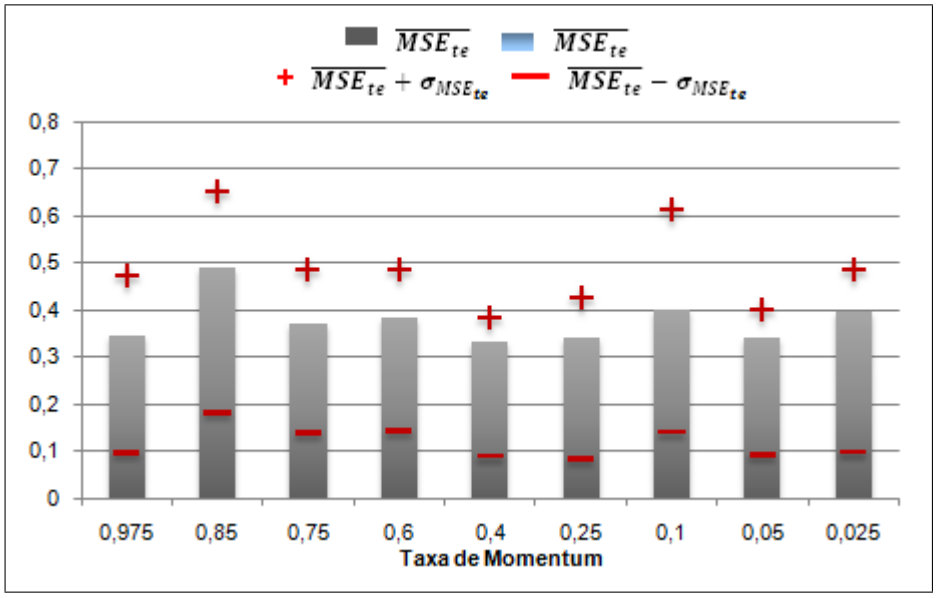


Figura 3.29: Resultados das Experiências para Taxa de *Momentum*

Observando a figura 3.29, o melhor resultado (em termos de $\overline{MSE}_{te}$ foi obtido para uma taxa de *momentum* de $0,4$ e $0,25$. Analisando este gráfico não se verifica um grande

variação do $\overline{MSE}_{te}$, à excepção da taxa de *momentum* $0,85$ que obteve um valor mais elevado, nem do desvio padrão $(\sigma_{MSE_{te}})$.

A figura 3.30 apresenta os *boxplots* do MSE_{te} dos nove ciclos de experiências.

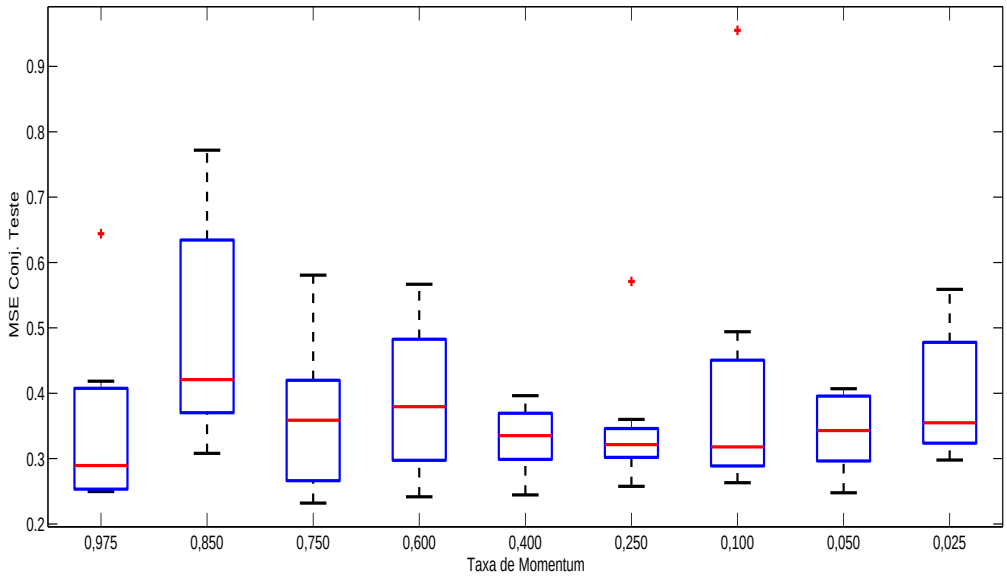


Figura 3.30: Boxplots das Experiências realizadas para Taxa de *Momentum*

Observando os *boxplots* dos nove ciclos testados parece existir uma maior dispersão para valores "extremos" da Taxa de *Momentum*. Isto é facilmente compreensível uma vez que sabemos que, por um lado, valores muito elevados da taxa de *momentum* aumentam a probabilidade de "ultrapassar" o mínimo, o que se pode verificar por exemplo para a taxa de *momentum* $0,975$ e $0,85$, em que os valores mínimos atingidos são dos mais elevados dos nove ciclos. Por outro lado valores muito baixos, aumentam a probabilidade de ficarmos presos em mínimos locais, e por esse motivo mais frequentemente têm mínimos elevados ou a presença de *outliers*, representados na figura 3.30 por um **+**, como se verifica por exemplo para a taxa de *momentum* $0,25$ e $0,1$.

Apesar de algumas diferenças no conjunto dos nove ciclos de experiências não se verificou um variação significativa do $\overline{MSE}_{te}$ (considerando que a ordem de grandeza da variável de *output* DTC escalonada para o intervalo $[-1,1]$ é das décimas) que justifique utilizar outro valor que não $0,9$, sugerido pelos autores.

Função de Activação

A função de activação das camadas da RNA utilizada nas experiências realizadas foi

a aconselhada para o algoritmo de RP, isto é, do tipo sigmóide para a camada escondida e lineares para a camada de saída. Em particular, foram utilizadas as seguintes funções de activação, definidas por defeito:

$$Tansig(x) = \frac{2}{(1 + e^{(} - 2x) - 1)}$$

para camada escondida, e

$$Purelin(x) = x,$$

para camada de (*outputs*).

Modo de Treino: Em lote ou *On-line*

Como já se referiu no Capítulo anterior, a RNA pode ser treinada em modo de treino "em lote" (*bacth training*) ou "on-line" (*incremental training*). O treino foi efectuado "em lote", ou seja, os pesos e *bias* só são actualizados após a apresentação de todos os dados de treino à rede (uma época). O motivo da escolha deste método de treino esteve relacionado com a utilização da função "*train*" em MATLAB, por ter acesso a algoritmo de treino mais eficientes, como por exemplo o utilizado no caso em estudo (*traingdx*), o que limitou a opção do modo de treino ao método "em lote", sendo também este o método mais comummente utilizado para o algoritmo de RP com taxa de aprendizagem adaptativa com *momentum*.

Inicialização dos Pesos

Antes de inicializar o treino de uma RNA *feed-forward* os pesos associados às ligações têm de ser inicializados.

No nosso caso em estudo os pesos foram inicializados utilizando o algoritmo, definido por defeito, de Inicialização de Nguyen-Widrow [Howard et al., 1992].

Este algoritmo selecciona os valores iniciais dos pesos de maneira a que as regiões activas das camadas de neurónios sejam distribuídas aproximadamente uniformes sobre o espaço das camadas de *input*. Os valores dos pesos atribuídos contêm um certo grau de aleatoriedade, o que faz com que sempre que esta função seja invocada, os valores dos pesos sejam diferentes. Uma das vantagens da utilização deste método de inicialização dos pesos em relação ao método de inicialização aleatório, é a de o treino ser mais rápido [Howard et al., 1992].

Medida de Erro

A medida de *performance* utilizada para avaliar o desempenho da RNA foi o Erro Médio Quadrático (MSE), definido por:

$$MSE = \frac{1}{N}\sum_{k}^{N}\epsilon_k^2$$

onde N é a dimensão da amostra.

Critério de Paragem

Os critérios de paragem utilizados foram os estabelecidos pela função de treino utilizada no software utilizado, MATLAB (*traingdx* [Howard et al., 1992]). O treino ocorre até ser atingido um dos seguintes critério:

- Número máximo de épocas (repetições) para treino: 1000 (*default value*);
- Objectivo de *performance* (*MSE*): 0 (*default value*);
- O gradiente mínimo da *performance* (*MSE*): $1e^{-10}$ (*default value*);
- Tempo Máximo para o treino da rede: infinito (*default value*);
- Número Máximo de falhas no conjunto de validação: 6 (*default value*).

3.3.3 Ciclo de Experiências (Número de Repetições)

O treino de uma RNA envolve processos de inicialização, como por exemplo o processo de inicialização dos pesos, que fazem com que os resultados de várias simulações do mesmo algoritmo com condições iniciais distintas sejam sempre diferentes, uma vez que há uma dependência em relação à condição inicial. Para apresentar resultados que representem o desempenho da RNA que pretendemos treinar, devem ser feitas várias simulações e apresentadas estatísticas da distribuição. É aconselhado que este número de repetições deva ser uma potência de dez.

Para determinar o número de repetições a utilizar foram efectuadas experiências com diferente número de repetições, potências de dez, para uma mesma RNA *feed-forward*, com as seguintes características:

- *Inputs*: MAT, FIS, QUIM e OUT;

- *Output*: DTC;
- Número de camadas escondidas: 1;
- Número de neurónios na camada escondida: 5;
- Taxa de *Momentum*: 0,9;
- Algoritmo de Aprendizagem: RP com Taxa de Aprendizagem adaptativa com *Momentum*;
- Função de Activação: sigmóide (*tansig*) para a camada escondida e linear (*purelin*) para a camada de *output*;
- TRE, VAL, TES: 60%-20%-20%;
- Divisão dos Dados: *rand*;
- Treino em lote;
- Medida de erro: MSE;
- Inicialização dos pesos: Algoritmo de Inicialização de Nguyen-Widrow.

Foram realizados quatro ciclos de experiências onde se treinou uma RNA, com as características acima mencionadas, uma, dez, cem e mil vezes (número de repetições de treino). Para cada ciclo registou-se MSE_{tr}, MSE_{val} e MSE_{te}, obtido em cada treino.

O código utilizado na realização das experiências para a divisão dos dados em MATLAB pode ser consultado na figura B.4 em apêndice.

NR	**Conjunto**	$\overline{MSE}_{te}$	$\sigma_{MSE_{te}}$	***T***
1	Teste	0,308	Na	00:00:02
10	Teste	0,361	0,092	00:00:18
100	Teste	0,358	0,098	00:04:34
1000	Teste	0,356	0,107	00:16:16

Tabela 3.3: Estatística dos resultados para testar NR

A tabela 3.3 apresenta a média do MSE do conjunto de teste $(\overline{MSE}_{te})$, o respectivo desvio padrão $(\sigma_{MSE_{te}})$ e o tempo de duração (T) para os quatro ciclos de experiências realizados para uma, dez, cem e mil repetições (NR). Na figura 3.31 estão representados os respectivos *boxplots*.

Para consulta dos resultados integrais dos teste realizados para a divisão dos dados consultar a tabela A.4 em apêndice.

Observando a tabela 3.3 podemos verificar que, apesar de existir um decréscimo do $\overline{MSE}_{te}$ com o aumento de número de repetições, com a excepção do número de repetições igual a um, não existe um variação significativa do $(\overline{MSE}_{te})$ que justifique a utilização de um valor para o número de repetições superior a dez. Valores superiores a cem verificam um tempo computacional consideravelmente alto, não apresentando por isso melhorias significativas na *performance* da rede.

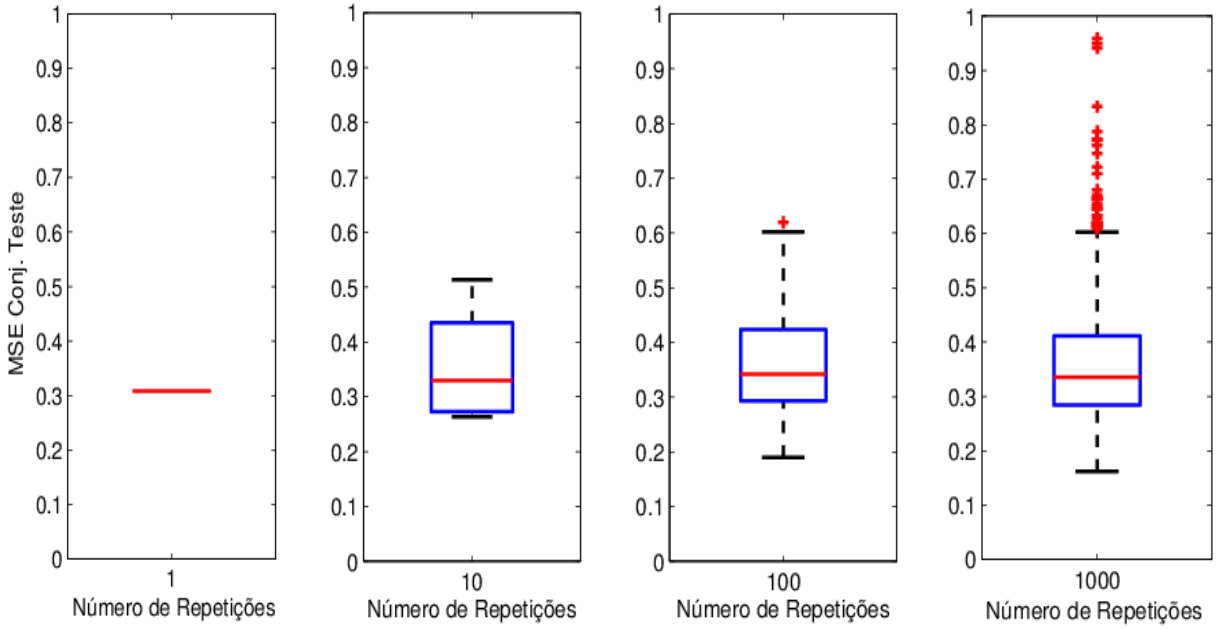


Figura 3.31: *Boxplot* das 4 experiências realizadas para testar o NR

Analisando os *boxplots* dos quatro ciclos de experiência podemos verificar que o aumento do número de repetições para além das dez, aumenta a probabilidade de obter valores muito elevados que se distanciam dos restantes resultados, como podemos verificar com a existência de *outliers* para o número de repetições mil. Em geral o aumento exponencial do número de repetições, não se demonstra benéfico para além de que aumenta a dispersão dos resultados para valores elevados do $\overline{MSE}_{te}$.

Os *boxplots* permitem constatar que a gama de valores observados entre o primeiro e o terceiro quartil praticamente não sofrem alterações e, por este motivo, considerou-se, em todas as experiências realizadas para os diferentes cenários, um número de repetições igual a dez.

3.4 Cenários

Na secção anterior foram definidos os parâmetros mais relevantes associados ao treino de uma RNA. Para que essa caracterização fique completa falta determinar quais as variáveis que devem ser consideradas como *input*, as de *output* e ainda o número de neurónios da camada escondida. A cada uma destas estruturas testadas designamos por cenário (*inputs*, *outputs* e *hidden layer*). Nesta secção descrevemos os diferentes cenários, apresentando os respectivos resultados obtidos para cada um.

Para o conjunto de experiências descritas nesta secção foi utilizado uma RNA com as seguintes características, definidas na secção anterior:

- Número de camadas escondidas: 1;
- Número de neurónios na camada escondida: variável de 1 a 5;
- Algoritmo de Aprendizagem: RP com Taxa de Aprendizagem adaptativa com *Momentum*;
- Taxa de *Momentum*: 0,9;
- Função de Activação: sigmóide (*tansig*) para a camada escondida e linear (*purelin*) para a camada de *output*;
- TRE, VAL, TES: 80%-10%-10%;
- Divisão dos Dados: *rand*;
- Treino em lote;
- Medida de erro: MSE;
- Inicialização dos pesos: Algoritmo de Inicialização de Nguyen-Widrow;
- Número de Repetições: 10.

O código utilizado na realização das experiências no contexto dos diferentes cenários, em MATLAB, pode ser consultado no apêndice B.

3.4.1 Cenário 1

Na primeira fase dos testes realizados na procura do melhor cenário, tomamos como *input* o desempenho académico dos alunos no primeiro ano frequentado e procuramos prever a respectiva duração total do percurso universitário, a que nos referimos como Cenário 1.

A figura 3.32 apresenta um diagrama com a estrutura da RNA no contexto do Cenário 1. As variáveis de *input* são, as já descritas na secção anterior, MAT, FIS, QUIM e OUT, a de *output* DTC e o número de neurónios da camada escondida n variável, entre um e cinco (pelos motivos já apresentados na secção anterior).

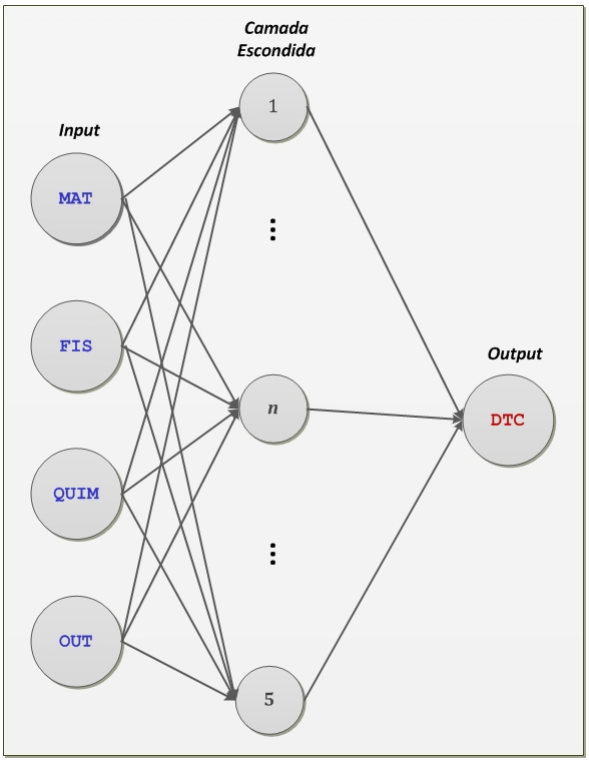


Figura 3.32: Diagrama da RNA para o Cenário 1

Foram realizados cinco ciclos de experiências, cada um com um número n de neurónios na camada escondida que varia entre um e cinco. Para cada ciclo de experiências, isto é, para cada RNA com as características descritas anteriormente e arquitectura $4{-}n{-}1$ (quatro variáveis de input, n neurónios da camada escondida e um *output*), repetimos dez vezes o treino, e guardamos, para cada repetição, o MSE_{tr}, MSE_{val} e MSE_{te}. Adicionalmente, guardamos a rede com o menor MSE_{te} $(MSE_{te_{min}})$ para a realização de possíveis simulações e comparação com os dados reais. A rede com $MSE_{te_{min}}$ designaremos por "rede óptima" da estrutura $4{-}n{-}1$.

A figura 3.33 representa um resumo dos resultados obtidos para os cinco ciclos de experiências. Para cada ciclo de experiências está representado $(\overline{MSE_{te_{DTC}}})$ nas bar-

ras cinzentas ($\overline{MSE_{te}}$ para a previsão de DTC), na linha vermelha o MSE de referência para DTC com mudança de escala $(MSE0_{E_{DTC}})$, ou seja, o MSE obtido se igualarmos as previsões de DTC para cada aluno à média de DTC com mudança de escala, e a variação obtida do $\overline{MSE_{te_{DTC}}}$ em relação ao $MSE0_{E_{DTC}}$, $\left(\Delta\overline{MSE_{te}}_{DTC}/MSE0_{E_{DTC}}\right)$ nos triângulos vermelhos.

Os resultados estatísticos integrais do Cenário 1 podem ser consultados na tabela A.5 em apêndice.

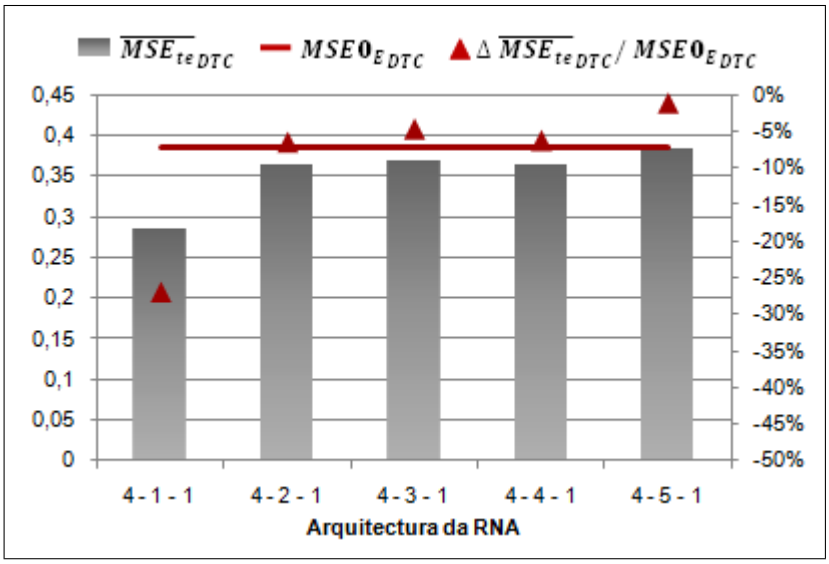


Figura 3.33: Resultados Cenário 1

Analisando a figura 3.33 verificamos que, os melhores resultados foram obtidos para a arquitectura mais simples, com apenas um neurónio na camada escondida $4-1-1$. No entanto, o melhor resultado obtido representa apenas uma redução do $\overline{MSE_{te}}_{DTC}$ face ao $MSE0_{E_{DTC}}$, de aproximadamente 27%.

Em geral, o aumento do número de neurónios na camada escondida representa uma diminuição na capacidade preditiva, ou seja, um aumento do $\overline{MSE_{te}}_{DTC}$, o que poderá ser explicado pelo facto do aumento de neurónios na camada escondida implicar um aumento do número de parâmetros a ajustar (pesos), que é desaconselhado face à amostra relativamente pequena.

Com cada rede óptima, de cada ciclo de experiências, simulámos os valores de DTC, invertemos a mudança de escala destas previsões e arredondamos às unidades, obtendo desta forma um valor previsto $\left(\widehat{DTC}\right)$ de DTC real (na escala original).

A figura 3.34 apresenta a distribuição dos erros absolutos, para cada uma das RNA

óptimas. Por erro absoluto, entende-se o valor absoluto da diferença entre DTC real e $\widehat{DTC}$. O erro absoluto toma valor zero, para previsões correctas (PC), e, para previsões incorrectas (PI), $\pm 1, \pm 2$ ou $\pm\ 3$ anos. Adicionalmente, apresentamos o MSE obtido de $\widehat{DTC}$ $\left(MSE_{min_{NE_{DTC}}}\right)$ no conjunto das 297 observações de DTC, isto é, o MSE entre DTC real, na escala original, e $\widehat{DTC}$. Para termos de comparação, temos o valor de referência de MSE na escala original $(MSE0_{NE_{DTC}})$, isto é, o MSE de referência obtido de forma idêntica ao $MSE0_{E_{DTC}}$ mas utilizando a média de DTC real.

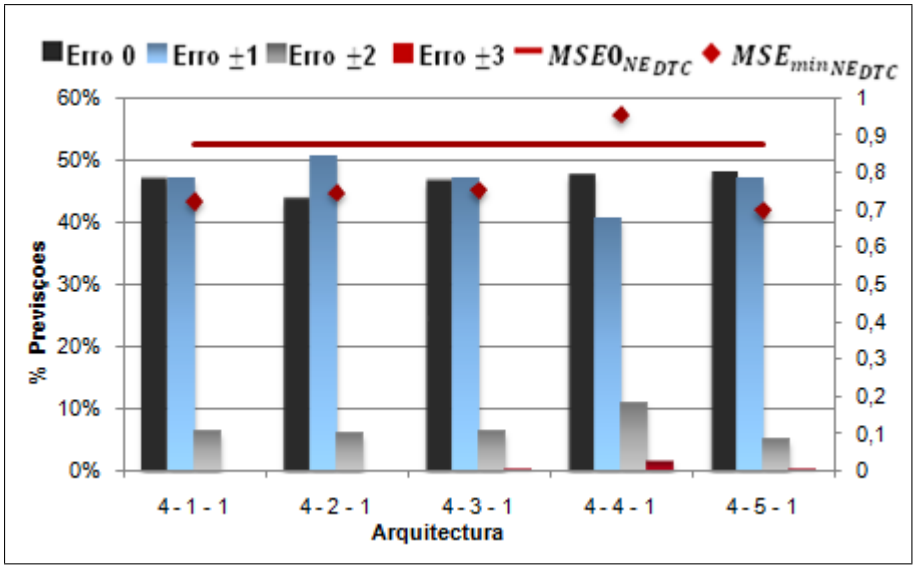


Figura 3.34: Resultados Cenário 1 para "redes óptimas"

Na figura 3.34 verificamos que a "rede óptima" com o menor $MSE_{min_{NE_{DTC}}}$ é a da arquitectura $4-5-1$, representando uma redução em relação ao $MSE0_{NE_{DTC}}$ de 20%. É também a arquitectura com maior percentagem de PC (Erro 0), aproximadamente 48%.Comparando as diferentes arquitecturas verificamos que a percentagem de PC é sempre superior a 43% e difere pouco de uma arquitectura para outra.

Igualmente se verifica que nas cinco arquitecturas, a maioria dos $\widehat{DTC}$ incorrectos diferem de um ano de DTC real. Em todas as arquitecturas, mais de 40% das previsões erram por um ano, sendo que menos de 6% erram por dois anos e menos de 2% por três anos. Isto significa que se quisermos prever DTC com uma margem de erro máxima de um ano, teremos "previsões aceitáveis" em cerca de 88% a 94% dos casos.

A figura 3.35 apresenta a distribuição de $\widehat{DTC}$ correctos e incorrectos, condicionada por cada uma das classes de DTC, para a "rede óptima" com o menor $MSE_{min_{NE_{DTC}}}$, isto é, para a rede óptima da arquitectura $4-5-1$.

Da análise da figura 3.35 verificamos que 83% das previsões feitas para os alunos

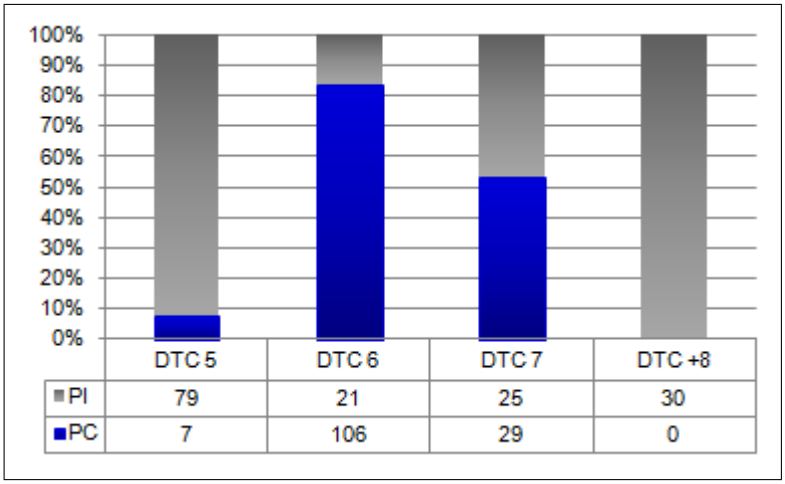


Figura 3.35: Resultados Cenário 1 "rede óptima" com o menor $MSE_{min_{NE_{DTC}}}$

que terminaram o curso em seis anos estão correctas. Tal não é de estranhar se tivermos em conta que essa é a classe modal.

Este comportamento, foi igualmente verificado para as restantes "redes óptimas", pode indicar que na presença de uma amostra de maior dimensão, possamos obter uma melhor performance da rede. Para consulta dos resultados para as restantes "redes óptimas" ver tabela A.12 em apêndice.

Para o Cenário 1, o melhor caso em termos de $\overline{MSE_{te\,DTC}}$, apresenta uma redução de 27% em relação ao $MSE0_{NE_{DTC}}$. Em termos de previsões correctas obtivemos no máximo 48%, no entanto verificamos que mais de 40% das previsões erram por ± 1 ano, nos cinco ciclos de experiências. Constatamos ainda que, quando analisamos a distribuição de PC condicionada pela classe de DTC, em particular para a "rede óptima" com o menor $MSE_{min_{NE_{DTC}}}$, a classe modal que tem cerca de 43% dos dados totais da amostra, é a que apresenta a maior frequência de PC, aproximadamente 83%. Este resultado leva-nos a crer que, na presença de uma amostra de maior dimensão, seria provavelmente possível obter resultados substancialmente melhores, nomeadamente com uma maior capacidade de previsão correcta de DTC's com menor frequência (por exemplo DTC igual a cinco ou, maior ou igual a oito).

3.4.2 Cenário 1A

Na expectativa de obter melhores resultados, procurou-se numa fase seguinte, cenários alternativos que verificassem melhorarias na capacidade preditiva, face ao Cenário 1, com o aumento de *input*, nomeadamente com a introdução de outras variáveis.

No Cenário 1A introduz-se mais uma variável de *input* o AING, já descrita na secção 3.2. A introdução da variável AING no Cenário 1 como *input* visa modelar o efeito ou tendência de uma eventual mudança na preparação dos alunos ao longo dos anos, associada às alterações no ensino secundário e consequentemente na *performance* global dos alunos, ou seja nas variáveis DTC e NFC. Em contrapartida, a introdução de mais uma variável significa um aumento no número de parâmetros a estimar na rede o que pode acabar por implicar uma diminuição da *performance* da rede, uma vez que quantos mais parâmetros a ajustar, mais complexo é o treino da rede e maior o risco de *overfitting*.

Por outro lado, como já foi referido na secção 3.2.3, na avaliação da correlação entre DTC e AING, e na secção 3.3.1, aquando da comparação da divisão de dados em *block* ou *rand*, as expectativas de melhorias ao introduzir esta variável não são muito elevadas uma vez que não se verificou uma evidência óbvia na variável DTC ao longo do período dos vinte anos estudados.

A figura 3.36 apresenta um diagrama com a estrutura da RNA no contexto do Cenário 1A. As variáveis de *input* são, as já descritas na secção anterior, MAT, FIS, QUIM, OUT e AING, a de *output* DTC e o número de neurónios da camada escondida n variável, entre um e cinco.

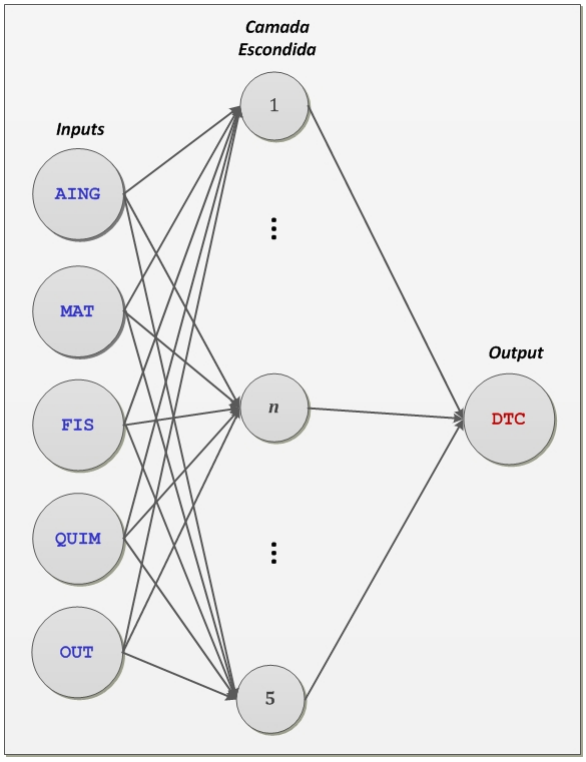


Figura 3.36: Diagrama da RNA para o Cenário 1A

Foram realizados cinco ciclos de experiências, cada um com um número n de neurónios na camada escondida que varia de um a cinco, de forma semelhante aos descritos na secção 3.4.1, para o Cenário 1.

A figura 3.37 representa um resumo dos resultados obtidos para os cinco ciclos de experiências de forma semelhante à apresentada na secção anterior para o Cenário 1.

Os resultados estatísticos integrais do Cenário 1A podem ser consultados na tabela A.6 em apêndice.

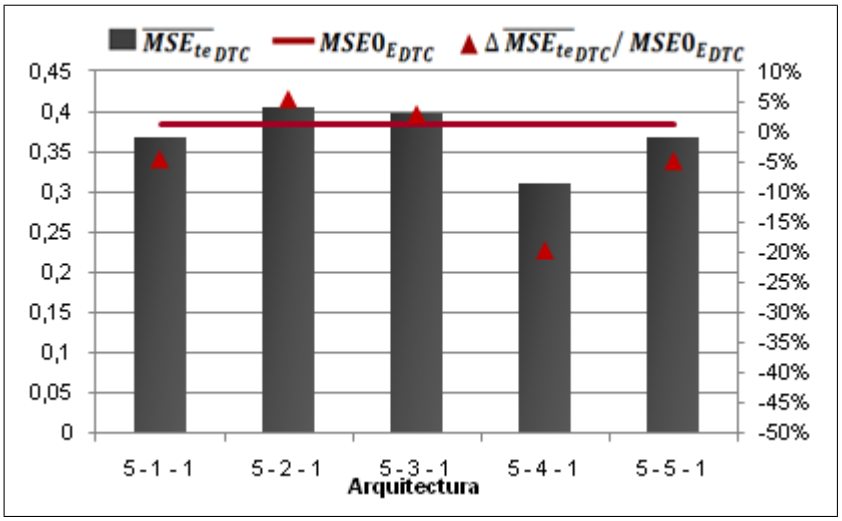


Figura 3.37: Resultados Cenário 1A

Como podemos verificar na figura 3.37, os resultados obtidos não foram melhores do que os obtidos no Cenário 1 (figura 3.33), em alguns dos casos houve mesmo uma pioria do $\overline{MSE_{teDTC}}$. A arquitectura com o menor $\overline{MSE_{teDTC}}$ foi a $5-4-1$ com uma redução de aproximadamente 20% face ao $MSE0_{E_{DTC}}$.

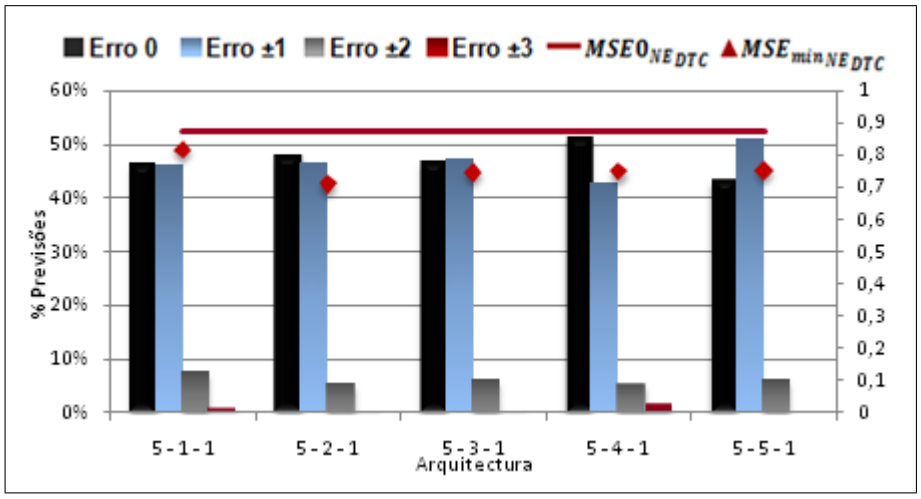


Figura 3.38: Resultados Cenário 1A para “redes óptimas”

A figura 3.38 apresenta a distribuição, para cada uma das RNA óptimas dos cinco ciclos (ou cinco arquitecturas), dos erros absolutos (Erro 0, Erro ± 1, ± 2 ou ± 3 anos), descritos na secção anterior à semelhança da figura 3.34.

Analisando a figura 3.38 verificamos que a "rede óptima" com o menor $MSE_{min_{NE_{DTC}}}$ é a da arquitectura $5-2-1$, representando uma redução em relação ao $MSE0_{NE_{DTC}}$de 19%. A arquitectura com maior percentagem de PC, aproximadamente 52%, é a $5-4-1$, que em contrapartida, apresenta também a maior percentagem de PI por ± 3 anos, o que contribui para o aumento do $MSE_{min_{NE_{DTC}}}$. Tal como constatamos para o Cenário 1, a maioria de $\widehat{DTC}$ incorrectos diferem de um ano de DTC real, com todas as arquitecturas com mais de 42% de PI a um ano. Para o Cenário 1A verificamos que caso pretendamos prever DTC com uma margem de erro máxima de um ano obtemos cerca de 93% a 95% de "previsões aceitáveis" o que, em comparação ao Cenário 1, não difere muito, não havendo melhorias.

Podemos concluir que a introdução da variável AING não contribui para a melhoria da *performance* da rede em geral, tendo em alguns dos casos representado uma pioria.

3.4.3 Cenário 2

Esta segunda fase considera os dois primeiros anos frequentados, visando assim melhorar a qualidade da previsão de DTC, face ao Cenário 1.

A figura 3.39 apresenta um diagrama com a estrutura da RNA no contexto do Cenário 2. As variáveis de *input* são, as já descritas na secção anterior, MAT2, FIS2, QUIM2 e OUT2, a de *output* DTC e o número de neurónios da camada escondida n variável, entre um e cinco.

De forma semelhante à já descrita nas secções anteriores, foram realizados cinco ciclos de experiências, cada um com um número n de neurónios na camada escondida que varia de um a cinco.

A figura 3.40 representa um resumo dos resultados obtidos para os cinco conjuntos de experiências para o Cenário 2, de forma semelhante à apresentada na secção 3.4.1 e 3.4.2

Os resultados estatísticos integrais do Cenário 2 podem ser consultados na tabela A.7 em apêndice.

Analisando a figura 3.40 verificamos que, em geral, os resultados foram melhores do que os obtidos no Cenário 1. Tal como se observa no Cenário 1, a rede com a melhor *performance*, em termos de $\overline{MSE_{te_{DTC}}}$, foi a de arquitectura mais simples, com apenas

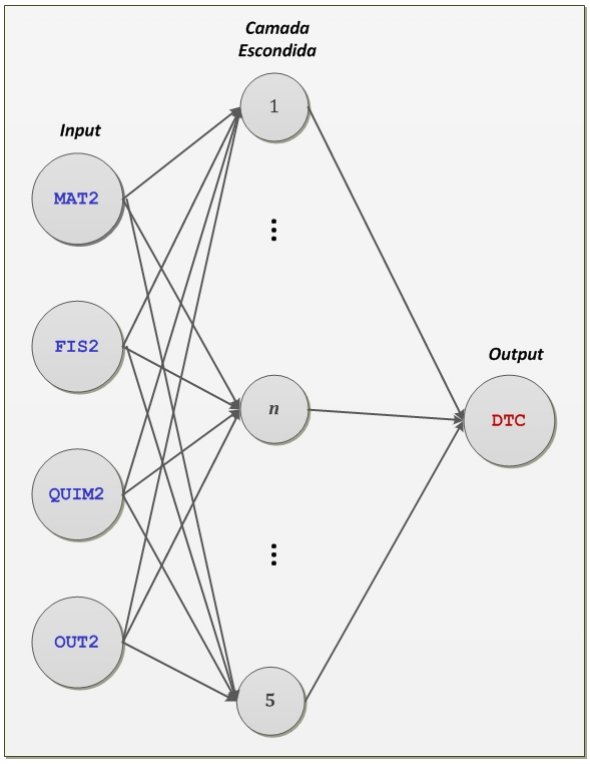


Figura 3.39: Diagrama da RNA para o Cenário 2

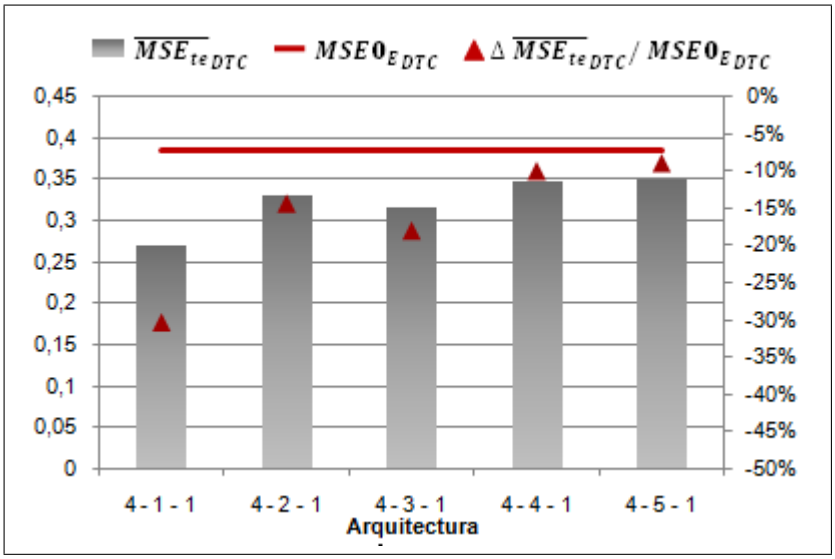


Figura 3.40: Resultados Cenário 2

um neurónio na camada escondida $(4-1-1)$. O melhor resultado obtido representa uma redução do $\overline{MSE_{teDTC}}$ face ao $MSE0_{E_{DTC}}$, de aproximadamente 30%, o que, em comparação com a redução obtida no Cenário 1 de 27%, representa apenas uma melhoria de 3%.

Em geral, tal como se verificou para o Cenário 1, o aumento do número de neurónios

na camada escondida implica um aumento do $\overline{MSE_{teDTC}}$.

Se pensarmos numa futura utilização destes modelos para efeitos de gestão universitária na previsão do desempenho da população estudantil, temos que concluir que não houve um grande acréscimo na qualidade das previsões (em termos de $\overline{MSE_{teDTC}}$) quando passamos do Cenário 1 para o Cenário 2, isto é, quando incorporamos a informação relativa ao segundo ano frequentado. No entanto, quando se analisa a percentagem de PC (figura 3.41) a situação já não é a mesma.

A figura 3.41 apresenta a distribuição, para cada uma das RNA óptimas dos cinco conjuntos de experiências do Cenário 2, dos erros absolutos (Erro 0, Erro ±1, ±2 ou ±3 anos), descritos na secção 3.4.1 para a figura 3.34.

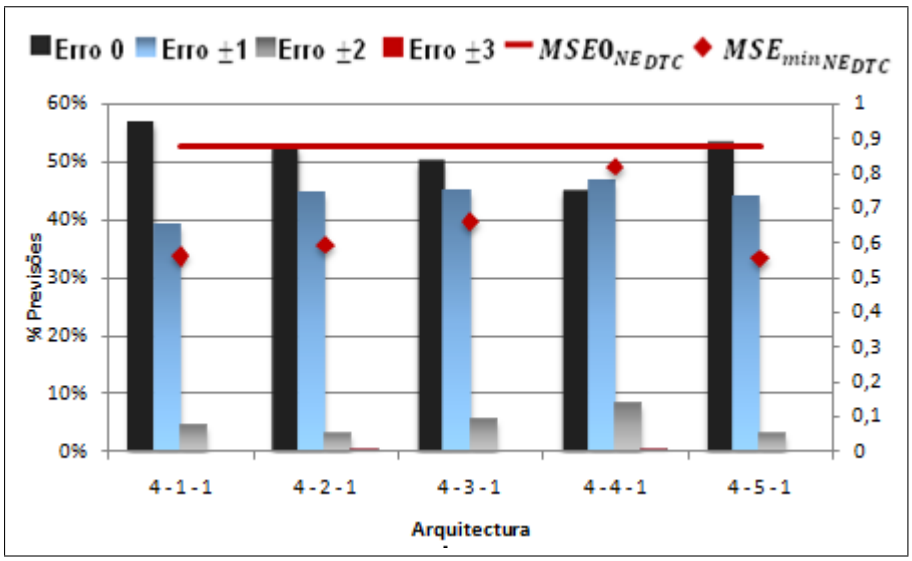


Figura 3.41: Resultados Cenário 2 para "redes óptimas"

Analisando a figura 3.41 constatamos que a "rede óptima" com o menor $MSE_{min_{NE_{DTC}}}$ é a da arquitectura $4-5-1$, representando uma redução em relação ao $MSE0_{NE_{DTC}}$ de 36%, superior à anteriormente obtida para o Cenário 1 de 20%. No entanto, a arquitectura com maior percentagem de PC, aproximadamente 57%, é a $4-1-1$ que, devido a ter uma maior percentagem de PI por ±2 tem um $MSE_{min_{NE_{DTC}}}$ superior à arquitectura $4-5-1$.

Em geral, nas cinco arquitecturas, a maioria de $\widehat{DTC}$ incorrectos diferem de um ano de DTC real. Para as cinco "redes óptima", mais de 39% das previsões erram por um ano, sendo que praticamente não existem PI por três anos.

Comparando o Cenário 1 (figura 3.34) com o Cenário 2, verificamos que há um aumento das percentagens de erro zero (aumento das barras cinza escuro) em detrimento

de uma diminuição das previsões de erro um (diminuição das barras azuis), com a excepção da arquitectura $4-4-1$. Para as previsões correctas verificamos uma percentagem entre 57% e 45%, que comparando com o Cenário 1, significa um aumento de precisão nas previsões correctas de aproximadamente 6%. Se considerarmos a previsão de DTC com uma margem de erro máxima de um ano "acertamos" em cerca de 92% a 97% dos casos, o que comparado com o Cenário 1 representa um acréscimo de precisão de cerca de 3% a 4%. Isto significa que se for aceitável prever DTC com, no máximo, um ano de margem de erro, não há grandes diferenças entre o Cenário 1 e 2.

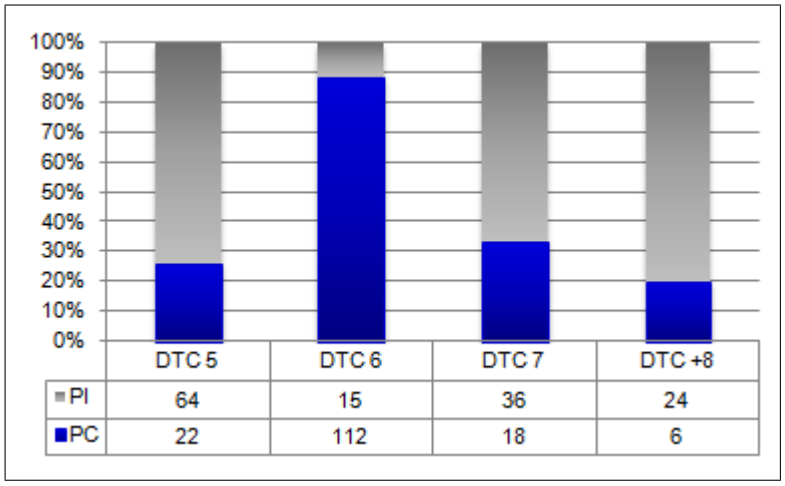


Figura 3.42: Resultados Cenário 2 "rede óptima" com o menor $MSE_{min_{NE_{DTC}}}$

A figura 3.42 apresenta a distribuição de $\widehat{DTC}$ correctos e incorrectos, condicionada por cada uma das classes de DTC, para a "rede óptima" com o menor $MSE_{min_{NE_{DTC}}}$, semelhante ao apresentado na secção 3.4.1 para o Cenário 1.

Verificamos, ao observar a figura 3.42, que na classe modal, existe uma maior percentagem e quantidade de PC, em comparação com as restantes classes, o que também tinha sido observado no contexto do Cenário 1. No entanto, para o Cenário 2, existe uma maior percentagem de PC nas restantes classes. Para a classe modal, em comparação ao Cenário 1, a percentagem de PC é aproximadamente 88%, o que representa um aumento de precisão para esta classe de 5%. Nas restantes classes, em geral, o aumento de precisão em relação ao Cenário 1 é mais significativo, em particular para a classe cinco e oito, com um aumento de precisão de 17% e 20%, respectivamente. Isto significa que as melhorias obtidas na *performance* da rede no contexto do Cenário 2 foram obtidas principalmente à custa, do aumento de precisão nas restantes classes, em particular na classes cinco e oito.

Este facto poderá contribuir para a ideia que, na presença de uma amostra de maior dimensão, principalmente nas classes de menor frequência, é provável que obtenhamos

um melhoramento significativo nos resultados. Este comportamento foi igualmente verificado nas restantes redes óptimas. Para uma consulta destes resultados, consultar tabela A.12 em apêndice.

Podemos concluir que, o Cenário 2 apresenta uma redução do $\overline{MSE_{te_{DTC}}}$ de 30% em relação ao $MSE0_E$ e uma melhoria de 3% em relação ao $\overline{MSE}_{te_{DTC}}$ do Cenário 1. Em relação à percentagem de PC para as "redes óptimas" verifica-se um acréscimo na precisão da previsão de também 6%. No entanto, se considerarmos a previsão de DTC com um erro máximo de um ano, obtemos um aumento de previsão face ao Cenário 1, menos significativo, de 3%.

Analisando a distribuição de PC condicionada pela classe de DTC da"rede óptima" com o menor $MSE_{min_{NE_{DTC}}}$ verificamos que, à semelhança do Cenário 1, a classe modal é a que apresenta a maior frequência de PC. Em geral, há um aumento de precisão na previsão nas classes, sendo mais significativo para as classes de menor frequência, em particular as classes cinco e oito (maior ou igual a oito) de 17% e 20%, respectivamente. Podemos desta forma concluir que a introdução da informação adicional relativamente ao segundo ano frequentado na previsão de DTC significou uma melhoria, no entanto menos significativa do que era espectável.

À semelhança do Cenário 1, foram realizados testes incluindo como variável de *input* AING (Cenário 2A). Os resultados obtidos foram igualmente insatisfatórios tal como tinha ocorrido no Cenário 1A, não tendo acrescentado capacidade preditiva ao modelo. Por este motivo, decidimos não detalhar os resultados das experiências realizadas no contexto do Cenário 2A.

Os resultados estatísticos integrais do Cenário 2A podem ser consultados na tabela A.8 em apêndice.

3.4.4 Cenário 3

Perante a ausência de cenários alternativos que pudessem indicar possíveis melhorias na previsão de DTC, e não fazendo sentido considerar dados para além dos dois primeiros anos frequentados, uma vez que estamos a considerar um curso de cinco anos, nesta terceira fase, procuramos prever a variável de *output* NFC.

Para prever NFC utilizamos como ponto de partida o melhor cenário obtido no conjunto de experiências realizadas para a previsão da variável DTC, dada a correlação existente entre estas duas variáveis.

O cenário com os melhores resultados obtidos foi o Cenário 2, quer em termos de PC, quando consideramos a "rede óptima" de cada um dos cinco ciclo de experiências, quer em termos de $\overline{MSE_{te}}$.

A figura 3.43 representa graficamente a estrutura da RNA no contexto do Cenário 3. As variáveis de *input* são, as já descritas na secção anterior, MAT2, FIS2, QUIM2 e OUT2, a de *output* NFC, e o número de neurónios da camada escondida n variável, entre um e cinco.

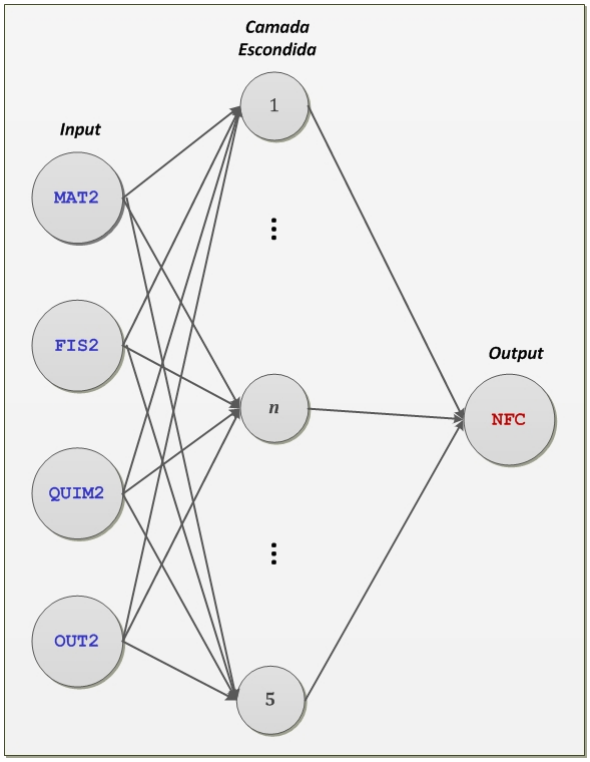


Figura 3.43: Diagrama da RNA para o Cenário 3

De forma semelhante aos cenários anteriores, foram realizados cinco ciclos de experiências, cada um com um número n de neurónios na camada escondida que varia de um a cinco.

A figura 3.44 representa, à semelhança dos cenários anteriores, um resumo dos resultados obtidos para os cinco ciclos de experiências, no contexto do Cenário 3.

Os resultados estatísticos integrais do Cenário 3 podem ser consultados na tabela A.9 em apêndice.

Analisando os resultados obtidos no Cenário 3, verificamos que o comportamento do $\overline{MSE_{teNFC}}$, com o aumento do número de neurónios da camada escondida, é semelhante ao do $\overline{MSE_{teDTC}}$ no contexto dos cenários testados para DTC, isto é, aumenta. A arquitectura de estrutura mais simples $4-1-1$, é que obteve melhores resultados com o

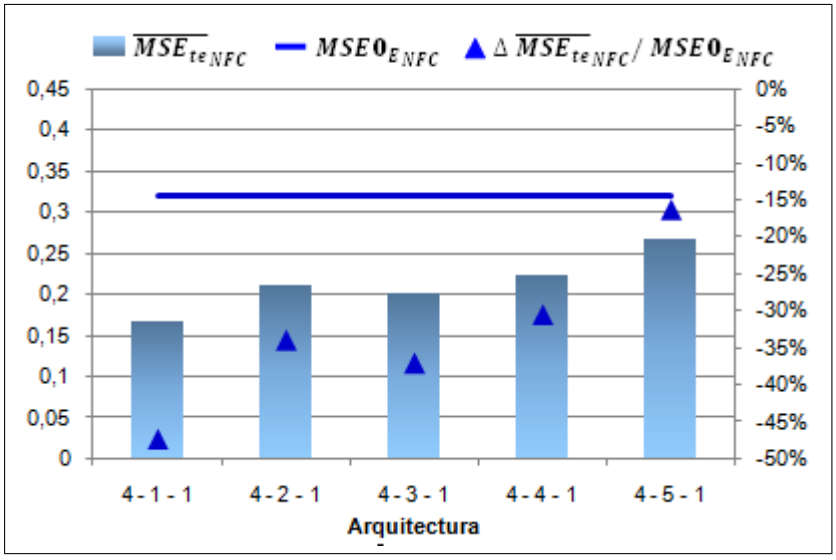


Figura 3.44: Resultados Cenário 3

$\overline{MSE_{teNFC}}$ igual a $0,168$ e uma redução em relação ao $MSE0_{E_{NFC}}$ de 47%. Mais uma vez se constata o benefício de uma arquitectura mais simples em consequência de um menor número de parâmetros a estimar.

À semelhança do que foi feito para os cenários de previsão de DTC, para cada "rede óptima" dos cinco ciclos de experiências, simulamos os valores da NFC, invertemos a mudança de escala destas previsões e arredondamos às unidades $\left(\widehat{NFC}\right)$. A figura 3.45 apresenta a distribuição dos erros absolutos entre NFC real (na escala original) e $\widehat{NFC}$: para previsões correctas erro zero, e para previsões incorrectas erro ± 1, ± 2 ou ± 3 valores. Adicionalmente, apresenta-se MSE obtido de $\widehat{NFC}$ $\left(MSE_{min_{NE_{NFC}}}\right)$ e o valor de referência de MSE na escala original $(MSE0_{NE_{DTC}})$.

Como se pode verificar na figura 3.45, a "rede óptima" com o maior número de neurónios na camada escondida foi a que obteve a maior percentagem de previsões certas, 48%, e o menor $MSE_{min_{NE_{NFC}}}$ com uma redução de 37% em relação ao $MSE0_{NE_{NFC}}$. Em relação a PI, cerca de 45% erram por um valor o que faz com que, se considerarmos previsões de NFC com erro menor ou igual a um, obtemos cerca de 87% a 95% previsões "aceitáveis". Sendo a NFC uma variável de intervalo $[10, 20]$, prever com uma margem de erro de uma unidade, parece ser ainda menos "preocupante" do que seria para DTC.

A figura 3.46 apresenta a distribuição de $\widehat{NFC}$ correctos e incorrectos, condicionada por cada uma das classes da NFC, para a "rede óptima" com o menor $MSE_{min_{NE_{NFC}}}$, semelhante ao apresentado para os cenários anteriores.

Verificamos, ao observar a figura 3.45 que, em geral, à excepção da classe de NFC

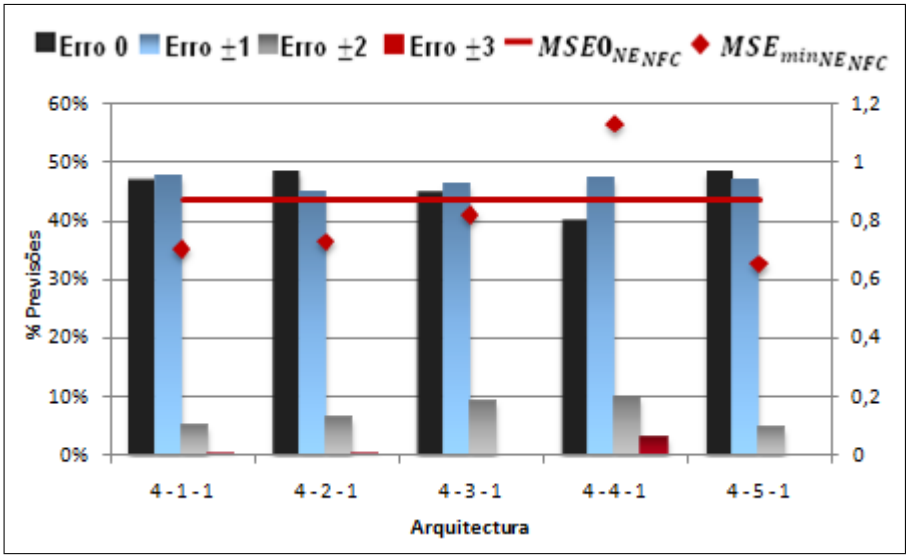


Figura 3.45: Resultados Cenário 3 para "redes óptimas"

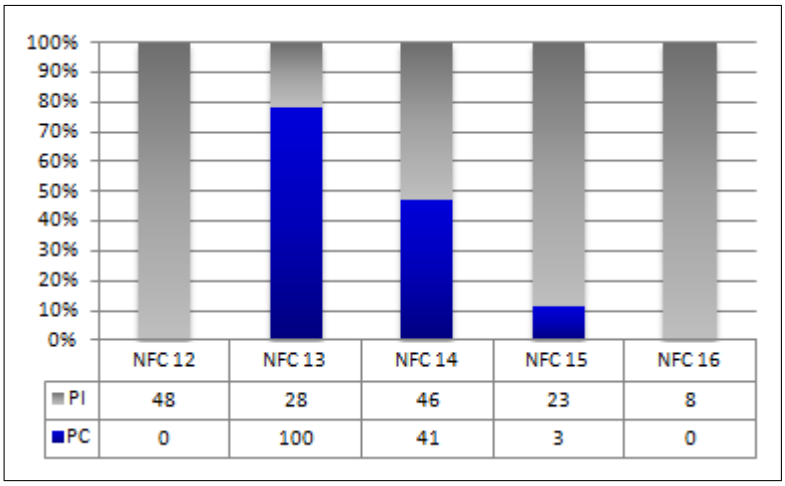


Figura 3.46: Resultados Cenário 3 "rede óptima" com o menor $MSE_{min_{NE_{NFC}}}$

igual a doze, as classes com maior frequência absoluta de NFC são as que têm maior frequência de PC, em particular a classe modal. Esta particularidade já tinha sido verificada quer para o Cenário 1 quer para o Cenário 2, e vem novamente apoiar a hipótese de que, na presença de uma amostra de maior dimensão fosse provável que se obtivesse melhores resultados. Este comportamento foi igualmente verificado nas restantes redes óptimas. Para uma consulta destes resultados pode-se recorrer à tabela A.13 em apêndice.

3.4.5 Cenário 3A

À semelhança do que foi efectuado no Cenário 1A e 2A foi testado uma alternativa ao Cenário 3 com a introdução da variável AING. Assim o Cenário 3A, como o designaremos, é constituído pelas variáveis de *input* MAT2, FIS2, QUIM2, OUT2 e AING, a de output NFC, e o número de neurónios da camada escondida n variável, entre um e cinco.

A figura 3.47 representa graficamente a estrutura da RNA no contexto do Cenário 3A.

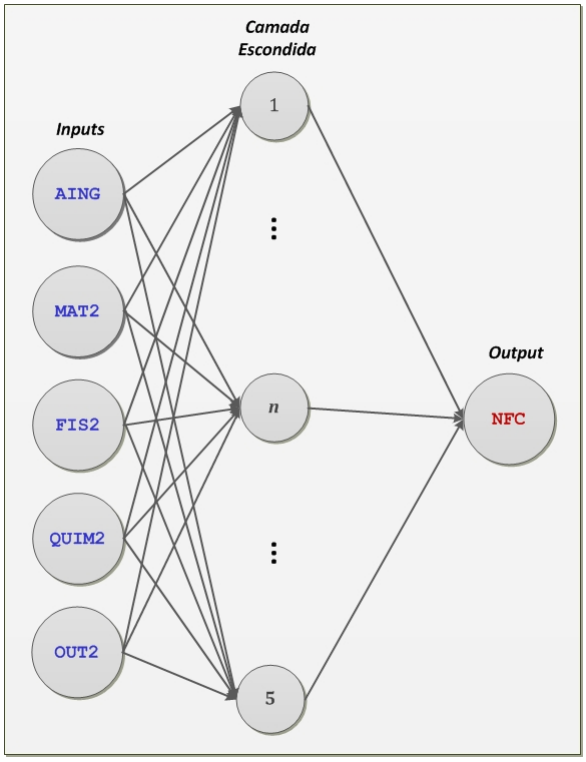


Figura 3.47: Diagrama da RNA para o Cenário 3A

De forma semelhante aos cenários anteriores, foram realizados cinco ciclos de experiências, cada um com um número n de neurónios na camada escondida que varia de um a cinco.

A figura 3.48 representa, à semelhança dos cenários anteriores, um resumo dos resultados obtidos para os cinco ciclos de experiências, no contexto do Cenário 3A e, para comparação, do Cenário 3. Pretendemos comparar os resultados obtidos das duas alternativas testadas para prever a NFC.

Os resultados estatísticos integrais do Cenário 3A podem ser consultados na tabela A.10 em apêndice.

Ao contrário do que estaríamos à espera os resultados, em geral, não foram melhores, havendo em alguns casos pioria na precisão. O facto de existir uma correlação mais forte

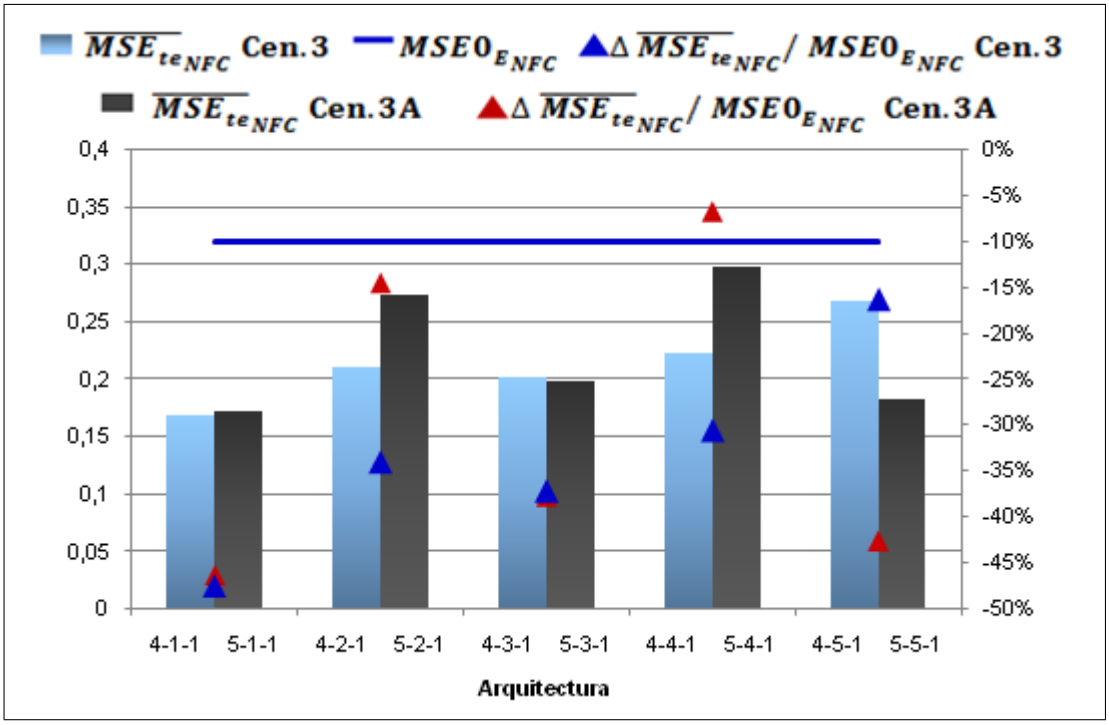


Figura 3.48: Resultados Cenário 3A vs Cenário 3

da variável de AING com NFC do que a DTC levou-nos a esperar que neste caso, ao contrário dos Cenário 1A e 2A, o Cenário 3A apresentasse vantagens em relação ao Cenário 3. Como se pode verificar na figura 3.48 tal não se verifica, havendo mesmo situações de perda da capacidade preditiva.

Como foi constatado, em geral, em todos os cenários testados, o aumento de complexidade do cenário, quer com a introdução de mais variáveis de *input* (Cenário 1A e 2A) quer com a introdução de mais neurónios na camada escondida, não apresentou vantagens face aos cenários mais simples, isto é, com menos neurónios da camada de *input* ou da camada escondida. Este facto levanta a hipótese de a justificação pelo qual o Cenário 3A não apresentar vantagens face ao Cenário 3, estar relacionado com o aumento de parâmetros a estimar na rede (pesos), em consequência do acréscimo de uma variável de *input* no modelo e da dimensão "reduzida" da amostra.

Podemos concluir deste cenário que apesar da existência de uma aparente influência entre NFC e DTC, em particular na figura 3.21, da alteração da NFC ao longo do período dos vinte anos de ingresso, com um claro aumento das classificações baixas acompanhado com uma diminuição das classificações altas, o Cenário 3A não verifica vantagens face ao Cenário 3.

3.4.6 Cenário 4

Imaginemos que tínhamos um modelo ou cenário capaz de prever DTC com elevado grau de precisão. Nestas circunstâncias, uma alternativa ao Cenário 3 seria tomar como variável de entrada no cenário para prever a NFC o $\widehat{DTC}$, obtido nesse suposto modelo.

Consideremos então o Cenário 4, como o designaremos, constituído pelas variáveis de *input* MAT2, FIS2, QUIM2, OUT2 e DTC, de *output* NFC, e o número de neurónios da camada escondida n variável, entre um e cinco.

A figura 3.49 representa graficamente a estrutura da RNA no contexto do Cenário 4.

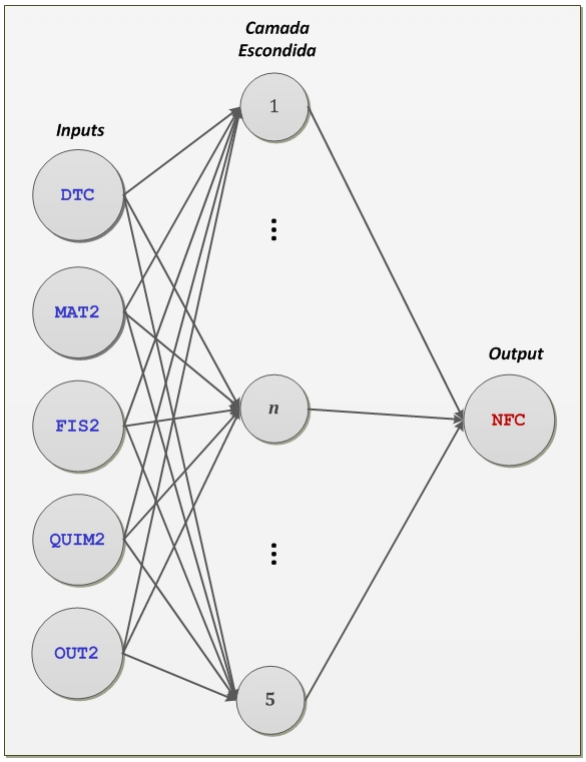


Figura 3.49: Diagrama da RNA para o Cenário 4

De forma semelhante aos cenários anteriores, foram realizados cinco ciclos de experiências, cada um com um número n de neurónios na camada escondida que varia de um a cinco.

A figura 3.50 representa, à semelhança dos cenários anteriores, um resumo dos resultados obtidos para os cinco ciclos de experiências, no contexto do Cenário 4 e, para comparação, do Cenário 3. Pretendemos comparar os resultados obtidos das duas alternativas testadas para prever a NFC.

Os resultados estatísticos integrais do Cenário 4 podem ser consultados na tabela A.11 em apêndice.

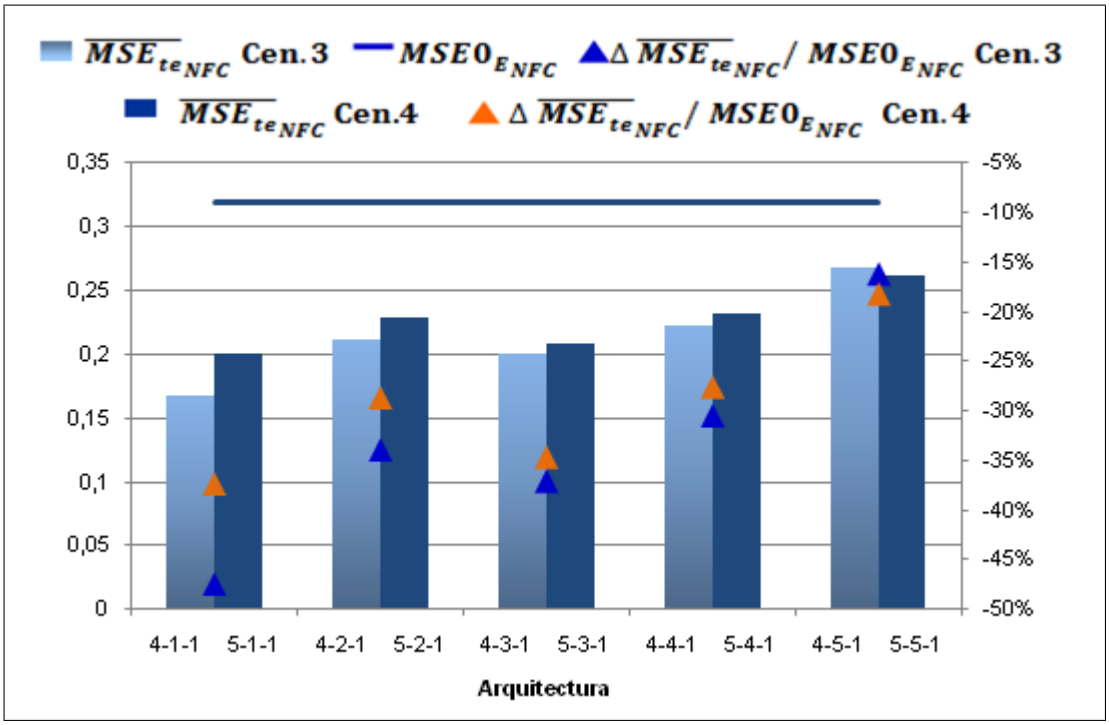


Figura 3.50: Resultados Cenário 3 vs Cenário 4

O facto de existir uma correlação negativa moderada entre DTC e NFC levou-nos a esperar que o Cenário 4 apresentasse vantagens relativamente ao Cenário 3. No entanto, tal não se verificou, como se pode constatar pelos resultados apresentados na figura 3.50.

Novamente, questiona-se se que o acréscimo de uma variável de *input* no modelo, com a consequente necessidade de estimação de mais parâmetros (pesos da rede), face à dimensão "reduzida" da amostra explica este "insucesso"? Eis um tópico que, certamente, justifica uma análise adicional num contexto de "ampliação" da amostra utilizada.

3.5 Conclusão

Na primeira parte deste capítulo foram descritas e analisadas todas as variáveis de *input* e *output* para o modelo de RNA que nos propusemos criar, ou seja, para uma RNA capaz de modelar as variáveis DTC e NFC. Na segunda parte deste Capítulo definimos as principais características da RNA e na terceira apresentados sete cenários possíveis para a previsão de DTC e/ou NFC, com base na informação obtida da analise das variáveis de *input* e *output* e utilizando RNA's com os parâmetros definidos na segunda parte deste capítulo. Foram apresentados quatro cenários para a previsão da variável DTC e três

cenários para a variável NFC.

Para DTC os melhores resultados obtidos foram os obtidos no Cenário 2. Neste cenário, a percentagem de previsões correctas varia entre 45% e 57% e a menor redução do $\overline{MSE_{teDTC}}$ relação ao $MSE0_{E_{DTC}}$ é de 30%. Em termos de previsões incorrectas, mais de 39% apresentam erro de um ano e menos de 16% erram por mais de um ano. Se considerarmos tolerável prever DTC com uma margem de erro máxima de um ano, obtemos uma percentagem de previsões "aceitáveis" entre 92% a 97%. No entanto, quando comparado com o Cenário 1, o Cenário 2 apresenta uma melhoria na precisão, quer da percentagem de previsões "aceitáveis", quer em termos de redução do $\overline{MSE_{teDTC}}$ em relação ao $MSE0_{E_{DTC}}$, de apenas 3%.

Para todos os cenários de previsão de DTC, nas previsões incorrectas de $\widehat{NFC}$ verificou-se uma maior percentagem de previsões com um ano de erro absoluto, sendo as previsões incorrectas por mais de um ano sempre inferiores a 12% da amostra.

Para a NFC o melhor cenário testado foi o Cenário 3, tendo-se verificado que a inclusão de novas variáveis, quer da variável de AING quer da DTC, não contribuiu para melhorar a previsão da mesma. Para o Cenário 3 obtivemos uma redução máxima do $\overline{MSE_{teNFC}}$ em relação ao $MSE0_{E_{NFC}}$ de 47%. Em relação às previsões de NFC, o Cenário 3 apresenta, para todas as "redes óptimas" dos cinco ciclos de experiências, percentagens de previsões correctas superiores a 40% e de previsões incorrectas por um valor, superiores a 45%. Se considerarmos previsões de NFC com uma margem de erro máxima de um valor obtemos uma percentagem de previsões "aceitáveis" superior a 95%, para o melhor caso.

Em geral, verificamos que os melhores resultados obtidos foram para RNA's com estruturas mais simples, isto é, com quatro variáveis de *input* e um neurónio na camada escondida. Há uma tendência clara para piorar os resultados, quer com a introdução de mais uma variável de *input*, AING (Cenário 1A, 2A e 3A) ou DTC (no caso do Cenário 4), quer com o aumento do número de neurónios da camada escondida. Este facto poderá eventualmente ser justificado pelo acréscimo do número de parâmetros a estimar em consequência do aumento de número de neurónios (de *input* e da camada escondida) face à reduzida dimensão da "amostra".

Analisando a distribuição de $\widehat{DTC}$ correctas e incorrectas, condicionada por cada uma das classes de DTC, para as "redes óptimas" de cada um dos cinco ciclos de ex-

periências, verificou-se em geral, que a classe modal de DTC é a que apresenta maior frequência de previsões correctas, tendo sido sempre superior a 80%. Comparando cenários, verificou-se que o aumento de precisão (previsões correctas) do Cenário 2 face ao Cenário 1 foi essencialmente devido ao aumento de previsões correctas nas classes de DTC com menor frequência, em particular na classe de DTC igual a cinco ou oito (maior ou igual a oito). Este facto leva-nos a esperar que num cenário de "ampliação" da amostra possa ocorrer uma melhoria na precisão do Cenário 1 e/ou Cenário 2 na previsão de DTC.

Este comportamento foi igualmente verificado para a distribuição de $\widehat{NFC}$ correctas e incorrectas, condicionada por cada uma das classes de NFC.

Como já foi referido no Capítulo 1, os dados utilizados nas experiências foram recolhidos e analisados por Nunes [Nunes, 2007] no contexto da sua dissertação de Mestrado de Matemática Aplicada - Investigação Operacional. Na sua dissertação, Nunes procura prever DTC e NFC com recurso a técnicas de simulação após uma caracterização estatística. Os resultados resumidos deste estudo são apresentados na tabela 3.4.

A tabela 3.5 apresenta os resultados resumidos do caso em estudo, para os melhores cenários para a previsão de DTC e NFC, ou seja, Cenário 2 e Cenário 3, respectivamente. Para cada uma das variáveis, DTC e NFC, são apresentados os intervalos das frequências relativas obtidas para os erros absolutos.

Erro na Previsão	DTC	NFC
Erro 0	11%	31%
Erro 1	54%	43%
Erro 2	23%	20%
Erro 3	11%	6%

Tabela 3.4: Frequência Relativa do Erro para abordagem de [Nunes, 2007]

Erro na Previsão	DTC	NFC
Erro 0	[45, 57] %	[40, 48] %
Erro 1	[39, 47] %	[45, 47] %
Erro 2	[3, 8] %	[5, 10] %
Erro 3	[0, 1],%	[0, 3],%

Tabela 3.5: Frequência Relativa do Erro para RNA's óptimas do Cenário 2 e 3

Comparando as duas abordagens, podemos verificar que o modelo com RNA traduziu-se num acréscimo de cerca de 40% de previsões de DTC correctas relativamente ao modelo de simulação apresentado por Nunes [Nunes, 2007]. Na variável NFC, o acréscimo de previsões correctas é de 15%.

Note-se ainda que quando se considera as previsões com um erro máximo de uma unidade, a modelação com RNA traduz-se num percentagem de previsões "aceitáveis" de aproximadamente 94% na variável DTC e 91% em NFC. Estes valores apresentam um acréscimo de precisão de cerca de 29% na previsão de DTC e 17% na previsão de NFC.

Assim podemos concluir que, a análise agora levada a cabo veio representar um claro acréscimo de qualidade nas previsões, face ao trabalho anteriormente elaborado com base na mesma amostra.

Capítulo 4

Conclusão

Neste capítulo relembramos o contexto em que surgiu este trabalho, assim como os objectivos a que nos propusemos. Iremos também referir as principais conclusões das experiências realizadas, bem como algumas limitações encontradas no contexto destas experiências e indicar possíveis abordagens futuras, na perspectiva de continuidade do trabalho aqui iniciado.

4.1 Síntese da Abordagem Efectuada

A previsão do desempenho académico de alunos universitários, baseada no desempenho académico no primeiro ano frequentado (ou nos dois primeiros anos frequentados) pode ser um instrumento de maior importância na gestão universitária.

Com efeito, este instrumento possibilitaria a previsão da evolução da população universitária com as óbvias implicações na previsão de receitas da instituição, da gestão do corpo docente e dos recursos materiais. Por outro lado, se utilizado em contexto de apoio pedagógico, a previsão do desempenho académico no final do primeiro ano frequentado poderia possibilitar a sinalização de alunos com dificuldades, que justificariam eventuais medidas de apoio.

É neste contexto que surge o caso em estudo. Através do uso de Redes Neuronais Artificiais propusemo-nos construir um modelo capaz de prever o desempenho académico global. Mais especificamente, procuramos um modelo de RNA capaz de prever duas características fundamentais do desempenho académico global, a Duração Total de Curso (DTC) e a Nota Final do Curso (NFC).

Utilizamos no caso em estudo, uma amostra de 297 alunos de um curso de engenharia da FCT-UNL, de cinco anos (pré-Bolonha), relativos a um período de vinte anos, que

havia sido recolhido no contexto de uma dissertação para obtenção do grau de mestre em Matemática Aplicada – Investigação Operacional por Nunes [Nunes, 2007].

No sentido de obter informação relevante que nos pudesse ajudar na construção do modelo, os dados foram analisados e tratados antes de serem alimentados à rede, utilizando para isso o EXCEL. Todos os testes efectuados na busca da melhor RNA capaz de prever DTC e NFC, foram efectuados recorrendo ao software MATLAB.

Os dados da amostra inicial foram agrupados de forma a simplificar e diminuir o número de variáveis de *input* de acordo com a natureza científica das disciplinas: Matemática (MAT), Física (FIS), Química (QUIM) e Outras (OUT).

As variáveis foram divididas em dois grupos: um com a informação referente ao primeiro ano frequentado (MAT, FIS, QUIM e OUT) e outro com informação dos dois primeiros anos frequentados (MAT2, FIS2, QUIM2 e OUT2). Para além das referidas variáveis, dispunha-se ainda do ano de ingresso (AING).

Numa primeira fase, pretendeu-se prever a variável DTC e numa segunda fase a variável NFC, sendo esta última para nós menos relevante, uma vez que para efeitos de gestão universitária a primeira é de maior importância.

Na primeira fase, foram testadas duas abordagens: a primeira abordagem consistiu em utilizar apenas a informação referente ao primeiro ano frequentado (Cenário 1) e a segunda abordagem os resultados obtidos nos dois primeiros anos frequentados. Nesta fase, foram também testados modelos alternativos ao Cenário 1 e Cenário 2, na perspectiva de melhorar a previsão de DTC, que consistiu na inclusão da variável AING como *input*.

Na segunda fase, partindo do melhor cenário obtido para DTC, procurou-se prever NFC (Cenário 3). Foram testados modelos alternativos ao do Cenário 3, com a inclusão adicional de variáveis, nomeadamente DTC (real), Cenário 4, e AING, Cenário 3A.

4.2 Principais Resultados

Na fase inicial de tratamento e análise de dados foi possível verificar a existência em geral de uma correlação moderada negativa de DTC, e positiva de NFC, com as variáveis de *input* associadas ao desempenho académico do primeiro ou dos dois primeiros anos. Esta correlação revelou-se mais significativa para as variáveis das áreas científicas de Matemática e Química, o que pode eventualmente ser explicado pela forte componente destas duas áreas científicas, em particular a de Matemática, no curso em estudo. Em

geral, os alunos com um melhor desempenho nas componentes de Matemática e Química, ou seja, valores mais elevados das respectivas variáveis, apresentam um melhor desempenho global, isto é, valores mais elevados de NFC e mais baixos de DTC.

No estudo da correlação entre as variáveis, verificou-se também a existência de uma correlação moderada negativa entre DTC e NFC, isto é, uma maior probabilidade de valores mais baixo de NFC estarem associados a valores mais elevados de DTC, e vice-versa. A verificação desta característica deu origem a um cenário alternativo ao Cenário 3, o Cenário 4, onde se considerou uma variável de *input* adicional, a variável DTC (real).

No estudo da correlação entre a variável AING e as variáveis de *output*, na expectativa da existência de uma eventual tendência de mudança do desempenho académico global dos alunos ao longo do período dos vinte anos da amostra, constatou-se a existência de uma clara variação NFC. Ao longo dos vinte anos de ingresso verificou-se um aumento da frequência das classificações mais baixas em detrimento de uma diminuição da frequência das classificações mais altas. A verificação desta tendência poderá eventualmente indicar uma menor preparação dos alunos mais recentes em comparação aos mais antigos. Esta característica contribuiu para a procura de melhores resultados nos modelos alternativos ao Cenário 1, Cenário 2 e Cenário 3, com a inclusão da variável AING como *input*, especialmente motivante para o Cenário 3A na previsão de NFC, para o qual a correlação e tendência se verificaram mais evidentes.

A informação recolhida e analisada proveniente das variáveis de *input* e *output* serviram de base para a construção dos cenários de previsão de DTC e NFC. Foram testados sete cenários diferente, quatro para prever a variável DTC e três para a variável NFC.

Na previsão de DTC, os melhores resultados foram os obtidos com base no desempenho académico dos dois primeiros anos frequentados (Cenário 2). Este cenário permitiu prever correctamente DTC em cerca de 45% a 57% dos casos, o que comparado com os resultados anteriormente conseguidos na abordagem utilizada por Nunes com recurso a técnicas de simulação traduz-se num aumento de precisão de 40%. Comparando ainda as duas abordagens, se considerarmos aceitável a previsão de DTC com um erro absoluto máximo de um ano, este cenário produz entre 92% e 97% de previsões “aceitáveis”, no conjunto das 297 observações de DTC, o que representa um aumento aproximado de 30% face aos resultados de Nunes [Nunes, 2007].

No entanto, considerando que é relevante prever com a maior antecedência possível, e se, consequentemente, pretendermos utilizar apenas o desempenho académico do primeiro ano frequentado (Cenário 1), os resultados continuam a ser bastante satisfatórios.

O Cenário 1 comparado com o Cenário 2 apresenta apenas uma perda na precisão de aproximadamente 6% nas previsões correctas, e 3% nas previsões "aceitáveis", isto é, nas previsões com um erro absoluto máximo de um ano. Note-se que ainda assim, em comparação aos resultados obtidos para a técnica de simulação de Nunes, os resultados continuam a ser bastante superiores, com um aumento de precisão nas previsões correctas de aproximadamente 34% e nas previsões "aceitáveis" de aproximadamente 27%.

Para NFC o melhor cenário foi o Cenário 3, com uma percentagem de previsões correctas entre 40% e 48% e de previsões incorrectas com erro igual a uma unidade entre 45% e 47%. Considerando previsões de NFC com uma margem de erro máxima de uma unidade obtém-se uma percentagem de previsões "aceitáveis" entre 87% e 95%. Novamente comparando abordagens, RNA e a simulação por Nunes [Nunes, 2007], verificamos que o uso de RNA traduziu-se num acréscimo de previsões correctas de 13% e de previsões "aceitáveis", com uma margem de erro máxima de uma unidade, de 17%, o que continua a ser bastante satisfatório.

Verificou-se ainda, que, de uma forma geral, os melhores resultados obtidos foram para RNA's com estruturas mais simples, isto é, com um menor número de neurónios na camada escondida e na camada de *input*, ou seja, com apenas quatro variáveis de *input*. Podemos eventualmente explicar este facto pelo aumento do número de parâmetros a ajustar em consequência do aumento de neurónios, da camada de *input* ou na camada escondida, face à "reduzida" dimensão da amostra.

Conclui-se também que, de forma geral, a frequência de previsões correctas, em relação às previsões incorrectas, condicionada pela classe de DTC ou NFC, é mais elevada para as classes de maior frequência absoluta de DTC ou NFC, respectivamente, em particular para a classe modal. Comparados os Cenário 1 e 2, verificou-se que o aumento da precisão no Cenário 2 foi mais significativo para as classes de DTC de menor frequência, em particular classe de DTC igual cinco e oito (maior ou igual a oito), onde a percentagem de previsões correctas aumentou de 8% para 17%, ou seja, quase que duplicou, e de 0% para 20%, respectivamente.

O facto de se ter verificado uma diminuição na capacidade preditiva da rede com o aumento de variáveis e de, em geral, se verificar uma maior frequência de previsões correctas nas classes de maior frequência absoluta, leva-nos a esperar com alguma expectativa e entusiasmo, uma melhoria relevante nos resultados num eventual contexto de ampliação da amostra.

Por todos os motivos aqui apresentados, considerando os resultados obtidos no caso em estudo e os objectivos a que nos propusemos, concluímos que os resultados, e as perspectivas futuras da sua aplicação, são bastante satisfatórios.

4.3 Limitações da Abordagem Efectuada

Apesar de considerarmos os resultados obtidos bastante satisfatórios, consideramos relevante mencionar as principais limitações encontradas ao longo do desenvolvimento deste trabalho.

A principal limitação na abordagem efectuada consistiu no volume de dados utilizados para a realização das experiências. A "reduzida" dimensão da amostra significou claramente uma limitação, quer nos resultados obtidos, quer na possibilidade de testar diferentes cenários. A dimensão da amostra limitou, por um lado, numa fase inicial a definição da arquitectura da rede a uma camada escondida com cinco neurónios no máximo, inviabilizando arquitecturas mais complexas com mais camadas escondidas, e por outro lado, nos resultados dos testes realizados com a inclusão de mais variáveis *input*, como se constatou nos cenários alternativos com a variável de AING ou DTC (Cenário 4), onde se esperava uma melhoria na capacidade preditiva da rede, o que não se verificou.

Podemos ainda referir como uma limitação o horizonte temporal do período dos vinte anos considerado na amostra, durante o qual ocorreram reestruturações curriculares quer ao nível do ensino secundário quer no próprio curso em estudo. Tais reformulações curriculares podem implicar, directamente ou indirectamente, alterações no desempenho global dos alunos, em particular nas duas variáveis que se procurou modelar (DTC e/ou NFC), e desta forma, perturbar a qualidade da análise "global" dos dados recolhidos. Estas reestruturações podem eventualmente justificar, por exemplo, a "diminuição" observada em NFC ao longo do período dos vinte anos.

Para além das reformulações curriculares do curso em estudo, ocorreram também reformulações no ensino pré-universitário que alteraram as condições de acesso ao ensino superior. Isto implica diferentes formações prévias de alunos antes do ingresso ao ensino universitário o que certamente influenciou o desempenho académico dos alunos nos primeiros anos frequentados. Sendo o objectivo a que nos propusemos modelar o desempenho académico global universitário com base no desempenho nos primeiros anos frequentados, que é seguramente dependente do desempenho pré-universitário, o facto de não existir uma medida uniforme consistente que avalie o aluno antes do ano de

ingresso, pode limitar a aplicação e generalização do modelo desenvolvido.

Os testes realizados no contexto dos Cenário 1A, 2A e 3A, foram uma tentativa de modelar estas situações de tendências de alteração ao longo do período dos vinte anos, no entanto a "reduzida" dimensão da amostra não nos permitiu confirmar, ou não, esta tendência.

Outra limitação da abordagem adaptada foi o elevado tempo despendido para o ajuste dos parâmetros de uma RNA. A utilização de RNA's implica um trabalho acrescentado e demorado, quer para o ajuste dos seus principais parâmetros quer para a definição da sua arquitectura, o que na perspectiva da sua aplicação em desenvolvimentos futuros pode ser uma desvantagem clara.

Ainda na perspectiva da aplicação do modelo desenvolvido, podemos ainda considerar como limitação o facto da amostra utilizada considerar alunos diplomados até 2000 e, consequentemente, já terem decorridos onze anos desde então. Neste período de onze anos podem ter existido factores importantes, nomeadamente novas reestruturações curriculares, que influenciem e alterem o comportamento do desempenho global académico, o que certamente limita a aplicação deste modelo a dados mais recentes.

Por último podemos referir como uma limitação o facto de só terem sido utilizados dados referentes a um curso, o que limitará a aplicação deste modelo na generalização a outros cursos. Cremos que não será adequado aplicar "cegamente" este modelo a outros cursos.

4.4 Potencialidades da Abordagem Efectuada e Desenvolvimentos Futuros

Acreditando vivamente que a capacidade de previsão de evolução da população estudantil numa instituição de ensino superior é um instrumento importante na gestão universitária, apresentamos aqui algumas sugestões para futuros desenvolvimentos na abordagem desenvolvida.

Considera-se que as RNA's aplicadas neste contexto pela sua sofisticação podem ser uma metodologia interessante para a modelação do desempenho académico dos estudantes universitários. A este propósito, comparando os resultados que obtivemos com a abordagem de Nunes [Nunes, 2007] podemos constatar um melhoramento global de

aproximadamente 40% nas previsões correctas de DTC e 17% nas previsões correctas de NFC, utilizando RNA.

Tendo sido a dimensão da amostra uma das principais limitações encontradas ao longo das experiências realizadas seria relevante refazer os testes aqui apresentados com uma amostra de maior dimensão, ampliando a amostra no mesmo curso com alunos diplomados até 2011. O aumento de cerca de 50% em termos de horizonte temporal significaria certamente um aumento muito superior em termos de número de alunos. Com este aumento da dimensão da amostra antevê-se um aumento da precisão nas previsões efectuadas.

O desempenho académico do aluno no primeiro ano (ou nos dois primeiros anos) universitário(s) frequentado(s), pode ser fortemente influenciado pela sua formação prévia no ensino secundário. Por este motivo, seria relevante introduzir no modelo desenvolvido informação de *input* referente à formação prévia do aluno. Uma possibilidade seria a inclusão da sua nota de acesso ao ensino universitário. No entanto, as frequentes reestruturações curriculares, nomeadamente na fórmula de cálculo da nota de acesso, inviabilizariam esta possibilidade. Neste caso, para ultrapassar esta dificuldade poderíamos, eventualmente definir os grupos de alunos um, dois, três e quatro, considerando as quatro classes da distribuição total das notas de acesso, definidas pelas fronteiras correspondentes ao primeiro, segundo e terceiro quartil, respectivamente.

Na perspectiva da gestão universitária será certamente interessante a generalização do modelo aqui desenvolvido aos vários cursos da FCT-UNL.

Neste contexto, poderíamos numa fase inicial introduzir como variável de *input* por exemplo a natureza científica do curso, ou numa análise prévia, classificar ou agrupar os cursos por classes (utilizando para isso RNA numa aplicação de *clustering*) e utilizar essa classificação como *input* adicional no modelo aqui desenvolvido. Neste caso, seria também relevante agrupar os cursos de acordo com as diferentes durações, isto é, cursos de três anos (primeiro ciclo), dois anos (segundo ciclo) e cinco anos (mestrados integrados). Note-se que para cursos de duração de dois ou três anos seria impensável considerar o modelo do Cenário 2, isto é utilizar informação para além do primeiro ano frequentado. Parece relevante ainda referir que para cursos de dois anos, seria pertinente contemplar a hipótese de incluir como informação adicional o desempenho académico global referente ao primeiro ciclo, mas neste caso, teríamos de ter em conta mais uma vez as diferentes formações prévias, isto é, as diferentes origens dos alunos antes do ingresso no segundo

ciclo.

* * *

Considera-se relevante referir que, a título pessoal, as Redes Neuronais Artificiais não haviam sido objecto de estudo em qualquer unidade curricular do primeiro ciclo em Matemática, ou do segundo ciclo em Matemática e Aplicações, tendo assim representado um importante momento de aprendizagem e a sua utilização constituído um grande desafio pessoal. Analogamente, a utilização do MATLAB foi novamente uma "nova" aprendizagem e um novo desafio pessoal.

A terminar, gostaria de fazer algumas breves reflexões pessoais sobre a minha experiência enquanto aluna do mestrado de Matemática e Aplicações e o que tal representou em termos académicos e profissionais.

A frequência do Mestrado em Matemática e Aplicações surgiu na expectativa de aprofundar conhecimentos em áreas científicas menos aprofundadas durante o primeiro ciclo e a sua aplicação prática. A vontade de desenvolver novos conhecimentos e técnicas que pudessem ser aplicadas em experiência profissionais futuras levou à frequência e escolha deste Mestrado.

O facto de já me encontrar integrada no mercado de trabalho dificultou a frequência e desempenho em algumas unidades curriculares, apesar de considerar ter sido privilegiada com alguma compreensão por parte da entidade empregadora. Simultaneamente, esta situação, deu-me uma nova perspectiva e compreensão, o que representou para mim uma mais-valia.

Não estando a área profissional onde me encontro actualmente a desempenhar funções directamente ligada à área cientifica em causa nesta dissertação, Redes Neuronais Artificiais, o trabalho aqui desenvolvido permitiu desenvolver capacidades de aprendizagem e autonomia que serão uma vantagem na minha carreira profissional futura. Para além da aprendizagem de RNA e da utilização de uma nova ferramenta de trabalho, MATLAB, a elaboração desta dissertação permitiu-me desenvolver capacidades de pesquisa, analise e escrita, que são relevantes no âmbito de qualquer actividade profissional.

Bibliografia

[Alves, 2002] Alves, V. (2002). *Resolução de Problemas em Ambientes Distribuídos:Uma Contribuição nas Áreas da Inteligência Artificial e da Saúde*. Dissertação de doutoramento, Universidade do Minho, Escola de Engenharia Departamento de Informática, Portugal.

[Ambrósio, 2002] Ambrósio, P. (2002). *Redes Neurais Arituficiais no apoio ao diagnóstico diferencial de lesões intersticiais pulmunares*. Dissertação de mestrado, Universidade de São Paulo, Ribeirão Preto, Brasil.

[Caudill and Butler, 1992] Caudill, M. and Butler, C. (1992). *Understanding neural networks : computer explorations*. The MIT Press, Cambridge, US.

[Cichocki and Unbehauen, 1993] Cichocki, A. and Unbehauen, R. (1993). *Neural networks for optimization and signal processing*. John Wiley, New York.

[Corrêa and Portugal, 1998] Corrêa, W. and Portugal, M. (1998). Previsão de séries de tempo na presença de mudança estrutural:redes neurais artificiais e modelos estruturais.

[Diederich, 1990] Diederich, J. (1990). *Artificial neural networks : concept learning*. IEEE Computer Society Press, Washington.

[Domany et al., 1996] Domany, E., van Hemmen, J. L., and Sculten, K. (1996). *Models of neural networks : vol. III : association, generalization and representation*. Springer, New York, US.

[Edelman et al., 1999] Edelman, B., Valentin, D., and Abdi, H. (1999). *Neural networks*. New Delhi : SAGE Publications, Newbury Park, US ; London ;.

[Ellacott et al., 1997] Ellacott, S., Mason, J., and Anderson, J. (1997). *Mathematics of neural networks: models, algoritms and applications*. Kluwer Academic Publishers, Boston,US.

[Filho et al.,] Filho, E., Carvalho, A., and Matias, A. Utilização de redes neurais artificiais na análise de risco de crédito a pessoas físicas.

[Freeman, 1994] Freeman, J. (1994). *Simulating neural netwoks with Matematica*. Addison-Wesley, Reeding, US.

[Grigoletti, 2006] Grigoletti, P. (2006). Utilizando o neural network toolbox...

[Haykin, 1999] Haykin, S. (1999). *Neural networks : a comprehensive foundation*. Upper Saddle River, NJ : Prentice Hall, Upper Saddle River, NJ.

[Horst, 1996] Horst, B. (1996). *Pyramidal neural networks*. Lawrence Erlbaum Associates Publisherss, Mahwah, US.

[Howard et al., 1992] Howard, D., Mark, B., and Martin, H. (1992). *Neural Network Toolbox 6: User's Guide*. The MathWorks, Inc., Natick, MA.

[Kartalopoulos, 1996] Kartalopoulos, S. (1996). *Understanding neural networks and fuzzy logic : basic concepts and applications*. IEEE Press, New York, US.

[Mehra and Benjamin, 1992] Mehra, P. and Benjamin, W. (1992). *Artificial neural networks : concepts and theory*. IEEE Computer Society Press, Los Alamitos, US.

[Menezes et al.,] Menezes, F., Esquerre, K., Kalid, R., Kiperstok, A., Matos, M., and Moreira, R. Redes neurais artificiais aplicadas ao processo de coagulação.

[Michael, 2003] Michael, A. (2003). *The Handbook of barin theory and neural networks*. The Mit Press, London, England ; Cambridge, Massachusetts.

[Moreira, 1997] Moreira, M. (1997). Introducao redes neuronais.

[Murteira et al., 2010] Murteira, B., Ribeiro, C., Andrade e Silva, J., and Pimenta, C. (2010). *Introdução à Estatística*. Escolar Editora, Lisboa.

[Neves, 1997] Neves, J. e Cortez, P. (1997). *Algoritmos Genéricos e Redes Neuronais na Previsão de Séries Temporais*. Dissertação de mestrado, Universidade do Minho, Departamento de Informática, Portugal.

[Neves, 2011] Neves, J. e Cortez, P. (2011). Bases biológicas do comportamento humano.

[Norgaard, 2000] Norgaard, M. (2000). *Neural networks for modelling and control of dynamic systems : a practitioner's handbook*. Springer-Verlag, London.

[Nunes, 2007] Nunes, P. (2007). *Modelação do Desempenho Académico de Alunos de Licenciatura da FCT-UNL*. Dissertação de mestrado, Universidade Nova de Lisboa, Faculdade de Ciências e Técnologias, Portugal.

[Petterson, 1996] Petterson, D. (1996). *Artificial Neural Networks – Theory and Applications*. Prentice Hall, Singapore.

[Rojas, 1996] Rojas, R. (1996). *Neural networks : a systematic introduction*. Springer, Berlin.

[Silva, 1998] Silva, L. (1998). *Análise e Síntese de Estratégias de Aprendizado para Redes Neurais Artificiais*. Dissertação de mestrado, Universidade de Campinas, Faculdade de Engenharia Eléctica e de Computação, Campinas, Brasil.

[Swingler, 1996] Swingler, K. (1996). *Applying neural networks : a practical guide*. Academic Press, London.

[Taylor, 1996] Taylor, J. (1996). *Neural networksNeural networks and their applications*. John Wiley, Chichester, UK.

[Universidade de Évora, 2011] Universidade de Évora, Barata, N. (2011). Bases biológicas do comportamento humano.

[UNL FCT, 2000] UNL FCT, D. d. Q. (2000). Redes neuronais artificiais:apontamentos de apoio à disciplina de sistemas inteligentes.

[Vemuri, 1999] Vemuri, V. (1999). *Artificial neural networks : concepts and control applications*. IEEE Computer Society Press, Los Alamitos, US.

Apêndice A

Tabelas

Neste apêndice apresentam-se algumas tabelas referidas ao longo deste trabalho, designadamente as tabelas referentes ao testes realizados para a definição do modelo da RNA e dos Cenários.

Data Division	Conjunto	$\overline{MSE}$	σ_{MSE}	MSE_{Min}	MSE_{Q1}	MSE_{Q2}	MSE_{Q3}	MSE_{Max}
60%-20%-20%	TRE	0,31727	0,06432	0,24404	0,26731	0,29870	0,33719	0,44784
	VAL	0,36944	0,11939	0,23716	0,26856	0,32628	0,49755	0,53770
	TES	0,35711	0,07560	0,25363	0,30809	0,36000	0,40795	0,50843
70%-15%-15%	TRE	0,46173	0,10686	0,30647	0,32831	0,49779	0,54810	0,58253
	VAL	0,49540	0,20927	0,21726	0,33764	0,48492	0,69315	0,82889
	TES	0,44035	0,15910	0,22539	0,33223	0,41067	0,61300	0,65753
80%-10%-10%	TRE	0,35632	0,10212	0,26461	0,29097	0,32522	0,39164	0,61172
	VAL	0,28485	0,08773	0,16043	0,20194	0,28163	0,34930	0,44906
	TES	0,37372	0,15511	0,21265	0,27183	0,35780	0,43219	0,73894
60%-40%-0%	TRE	0,31586	0,06227	0,24188	0,26855	0,30243	0,36814	0,43708
	VAL	0,31812	0,04374	0,23489	0,29336	0,31873	0,34285	0,38906
	TES	NaN	NaN	NaN	NaN	NaN	NaN	NaN
70%-30%-0%	TRE	0,32814	0,09598	0,25376	0,27707	0,29914	0,32669	0,58397
	VAL	0,35548	0,07200	0,27189	0,30526	0,32645	0,40763	0,49123
	TES	NaN	NaN	NaN	NaN	NaN	NaN	NaN
80%-20%-0%	TRE	0,30758	0,04874	0,25432	0,27285	0,29219	0,33939	0,41622
	VAL	0,35168	0,10710	0,19066	0,29519	0,33996	0,45340	0,49003
	TES	NaN	NaN	NaN	NaN	NaN	NaN	NaN
90%-10%-0%	TRE	0,30721	0,04722	0,26471	0,27886	0,29070	0,31979	0,42529
	VAL	0,29982	0,07159	0,18067	0,25992	0,29167	0,33452	0,43678
	TES	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Tabela A.1: Testes Data Division *Rand*

Data Division	Conjunto	$\overline{MSE}$	σ_{MSE}	MSE_{Min}	MSE_{Q1}	MSE_{Q2}	MSE_{Q3}	MSE_{Max}
60%-20%-20%	TRE	0,37480	0,12458	0,28577	0,28739	0,33525	0,41476	0,68661
	VAL	0,40118	0,16782	0,26547	0,29452	0,34984	0,43385	0,82802
	TES	0,44679	0,35544	0,25802	0,26528	0,29483	0,46850	1,41802
70%-15%-15%	TRE	0,35074	0,06726	0,26777	0,27553	0,35130	0,41458	0,45302
	VAL	0,46397	0,10796	0,35821	0,38874	0,40522	0,54390	0,66601
	TES	0,28113	0,07942	0,18463	0,23874	0,26037	0,31513	0,46160
80%-10%-10%	TRE	0,42000	0,18389	0,27891	0,29426	0,35035	0,43584	0,83116
	VAL	0,45426	0,30724	0,20965	0,26663	0,28944	0,61699	1,05215
	TES	0,37323	0,27604	0,18702	0,22107	0,27568	0,34493	1,08671
60%-40%-0%	TRE	0,29925	0,01370	0,28455	0,29003	0,29454	0,30392	0,32496
	VAL	0,28557	0,02823	0,25218	0,25762	0,28170	0,30615	0,33408
	TES	NaN	NaN	NaN	NaN	NaN	NaN	NaN
70%-30%-0%	TRE	0,32979	0,13289	0,27545	0,28354	0,28726	0,29043	0,70660
	VAL	0,37346	0,25509	0,25258	0,27177	0,29786	0,31323	1,09413
	TES	NaN	NaN	NaN	NaN	NaN	NaN	NaN
80%-20%-0%	TRE	0,40001	0,15150	0,28157	0,29186	0,39326	0,41677	0,79552
	VAL	0,43234	0,30988	0,21222	0,22661	0,32353	0,52682	1,22985
	TES	NaN	NaN	NaN	NaN	NaN	NaN	NaN
90%-10%-0%	TRE	0,33240	0,06089	0,27804	0,28409	0,31197	0,36907	0,46714
	VAL	0,26185	0,10980	0,18922	0,19390	0,22148	0,27508	0,55598
	TES	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Tabela A.2: Testes Data Division *Block*

α	Conjunto	$\overline{MSE}$	σ_{MSE}	MSE_{Min}	MSE_{Q1}	MSE_{Q2}	MSE_{Q3}	MSE_{Max}
0,9750	TRE	0,33428	0,10183	0,24094	0,27488	0,29320	0,38192	0,58140
	VAL	0,40807	0,11229	0,27840	0,31524	0,39255	0,48021	0,65191
	TES	0,34647	0,12599	0,24962	0,25340	0,28948	0,40760	0,64401
0,850	TRE	0,46030	0,11816	0,31525	0,33306	0,46147	0,54962	0,62824
	VAL	0,43553	0,13338	0,29836	0,34132	0,39572	0,48365	0,74229
	TES	0,48957	0,16308	0,30817	0,37032	0,42101	0,63441	0,77185
0,750	TRE	0,37179	0,12874	0,26669	0,27693	0,31989	0,41944	0,65646
	VAL	0,39469	0,12369	0,26014	0,30660	0,35920	0,47060	0,64572
	TES	0,37068	0,11583	0,23195	0,26621	0,35886	0,41992	0,58080
0,600	TRE	0,39163	0,14786	0,26958	0,28955	0,32648	0,48776	0,67230
	VAL	0,41928	0,13811	0,23107	0,31748	0,42312	0,47583	0,66435
	TES	0,38385	0,10360	0,24171	0,29747	0,37942	0,48257	0,56674
0,400	TRE	0,32495	0,04318	0,25543	0,29083	0,31628	0,36740	0,37970
	VAL	0,32650	0,08996	0,18516	0,26464	0,33014	0,37090	0,51420
	TES	0,33312	0,05025	0,24452	0,29887	0,33514	0,36939	0,39633
0,250	TRE	0,33732	0,07890	0,25080	0,26859	0,33414	0,38029	0,50227
	VAL	0,37242	0,09212	0,27003	0,30175	0,35254	0,43509	0,56410
	TES	0,34114	0,08569	0,25765	0,30204	0,32141	0,34595	0,57112
0,100	TRE	0,36214	0,15825	0,24298	0,26669	0,30420	0,34866	0,72721
	VAL	0,38171	0,16074	0,28199	0,28887	0,31421	0,39001	0,79990
	TES	0,40306	0,20963	0,26320	0,28877	0,31788	0,45059	0,95531
0,050	TRE	0,32008	0,06812	0,25066	0,26904	0,30606	0,34744	0,47497
	VAL	0,31254	0,09765	0,19242	0,23791	0,27856	0,43900	0,44925
	TES	0,33988	0,05949	0,24776	0,29639	0,34288	0,39562	0,40705
0,025	TRE	0,37454	0,14107	0,24455	0,27686	0,31562	0,39427	0,65179
	VAL	0,36560	0,14123	0,26717	0,27914	0,28630	0,37769	0,68190
	TES	0,39585	0,09153	0,29779	0,32378	0,35481	0,47802	0,55888

Tabela A.3: Testes Taxa de *Momentum*

NR	Conjunto	Duração do Treino	$\overline{MSE}$	σ_{MSE}	MSE_{Min}	MSE_{Q1}	MSE_{Q2}	MSE_{Q3}	MSE_{Max}
1	Treino	00:00:02	0,33719	Na	0,33719	0,33719	0,33719	0,33719	0,33719
	Validação		0,51200	Na	0,51200	0,51200	0,51200	0,51200	0,51200
	Teste		0,30809	Na	0,30809	0,30809	0,30809	0,30809	0,30809
10	Treino	00:00:07	0,36933	0,09705	0,28605	0,29850	0,32403	0,44631	0,53147
	Validação		0,37874	0,12995	0,21492	0,29674	0,34530	0,41036	0,63358
	Teste		0,36057	0,09176	0,26362	0,27263	0,32962	0,43467	0,51322
100	Treino	00:01:49	0,33546	0,08944	0,22254	0,27567	0,30169	0,37187	0,69610
	Validação		0,35440	0,10691	0,18348	0,27029	0,33690	0,40123	0,74985
	Teste		0,35839	0,09782	0,18975	0,29313	0,34233	0,42342	0,61945
1000	Treino	00:16:16	0,34171	0,09668	0,20832	0,27769	0,30572	0,37563	0,82263
	Validação		0,35548	0,11120	0,16569	0,27905	0,33088	0,40198	0,86869
	Teste		0,35630	0,10764	0,16204	0,28386	0,33539	0,41134	0,95882

Tabela A.4: Testes Número de Repetições

Arquitectura	Conjunto Dados	$\overline{MSE}$	σ_{MSE}	MSE_{Min}	MSE_{Q1}	MSE_{Q2}	MSE_{Q3}	MSE_{Max}
4-1-1	Treino	0,32232	0,05501	0,26131	0,28075	0,29194	0,39236	0,40402
	Validação	0,28498	0,08235	0,19185	0,23811	0,26223	0,32334	0,44303
	Teste	0,28173	0,10145	0,16029	0,21556	0,25754	0,33383	0,50324
4-2-1	Treino	0,37148	0,03891	0,29496	0,34429	0,37225	0,40928	0,41595
	Validação	0,33871	0,12581	0,17065	0,23662	0,32146	0,39700	0,55246
	Teste	0,36048	0,09841	0,13741	0,33413	0,37582	0,43256	0,47202
4-3-1	Treino	0,36057	0,07461	0,26271	0,31084	0,34949	0,36858	0,50893
	Validação	0,36005	0,12620	0,19689	0,28951	0,33918	0,42118	0,58901
	Teste	0,36744	0,09668	0,16881	0,29841	0,38328	0,45261	0,46148
4-4-1	Treino	0,33804	0,04147	0,27611	0,30346	0,33803	0,36532	0,40560
	Validação	0,32679	0,08217	0,22946	0,28473	0,31466	0,35562	0,51537
	Teste	0,36132	0,10933	0,22021	0,24778	0,35365	0,46462	0,53514
4-5-1	Treino	0,36286	0,07916	0,28671	0,29336	0,34044	0,41970	0,52635
	Validação	0,33760	0,08777	0,21664	0,24393	0,34287	0,39149	0,50026
	Teste	0,38087	0,16412	0,19542	0,30449	0,33485	0,45045	0,77918

Tabela A.5: Testes Cenário 1

Arquitectura	Conjunto Dados	$\overline{MSE}$	σ_{MSE}	MSE_{Min}	MSE_{Q1}	MSE_{Q2}	MSE_{Q3}	MSE_{Max}
4-1-1	Treino	0,34619	0,07244	0,26572	0,28180	0,33462	0,39991	0,45949
	Validação	0,34150	0,07700	0,24803	0,28028	0,33814	0,39617	0,46568
	Teste	0,36799	0,12344	0,22073	0,27403	0,33923	0,43304	0,58356
	Total	0,35832	0,08322	0,27127	0,28485	0,34887	0,38669	0,51817
4-2-1	Treino	0,37308	0,05579	0,26325	0,36267	0,38278	0,41480	0,43335
	Validação	0,39606	0,10292	0,24499	0,31063	0,38063	0,48528	0,55520
	Teste	0,40613	0,12105	0,25994	0,29820	0,36719	0,52559	0,61225
	Total	0,37516	0,05649	0,27234	0,35336	0,38722	0,39854	0,47361
4-3-1	Treino	0,33681	0,14158	0,25139	0,26605	0,28831	0,32938	0,72706
	Validação	0,31247	0,11679	0,21099	0,24986	0,28020	0,35155	0,61371
	Teste	0,39658	0,17352	0,25034	0,26452	0,37375	0,41818	0,85248
	Total	0,35892	0,18378	0,26420	0,26854	0,29179	0,34183	0,86893
4-4-1	Treino	0,30457	0,05869	0,22882	0,26017	0,28892	0,35528	0,40459
	Validação	0,31456	0,07629	0,21458	0,27462	0,31267	0,33282	0,47600
	Teste	0,30995	0,10651	0,14374	0,22806	0,31739	0,41770	0,44127
	Total	0,31416	0,05402	0,26569	0,27335	0,28971	0,35184	0,41738
4-5-1	Treino	0,34794	0,12135	0,24073	0,28850	0,30659	0,36649	0,67021
	Validação	0,37886	0,09899	0,24214	0,27790	0,37421	0,47159	0,51368
	Teste	0,36679	0,10167	0,26290	0,29710	0,34966	0,40978	0,61923
	Total	0,35741	0,09446	0,26299	0,30076	0,32019	0,41692	0,57234

Tabela A.6: Testes Cenário 1A

Arquitectura	Conjunto Dados	$\overline{MSE}$	σ_{MSE}	MSE_{Min}	MSE_{Q1}	MSE_{Q2}	MSE_{Q3}	MSE_{Max}
4-1-1	Treino	0,25546	0,05322	0,19712	0,22340	0,23130	0,28119	0,36540
	Validação	0,23261	0,10373	0,10817	0,17224	0,18094	0,29130	0,39961
	Teste	0,26862	0,07541	0,17482	0,21374	0,25135	0,33982	0,38516
4-2-1	Treino	0,36579	0,07740	0,21143	0,28658	0,39507	0,41615	0,43666
	Validação	0,34183	0,06405	0,23175	0,27748	0,37137	0,38274	0,43222
	Teste	0,32970	0,06075	0,24039	0,28861	0,33017	0,37448	0,42237
4-3-1	Treino	0,29458	0,10725	0,20057	0,21917	0,25890	0,34018	0,55324
	Validação	0,36055	0,13350	0,19646	0,23034	0,34135	0,43927	0,60952
	Teste	0,31567	0,05957	0,23765	0,27205	0,30417	0,37444	0,40796
4-4-1	Treino	0,35905	0,16934	0,21422	0,25685	0,33956	0,35148	0,81147
	Validação	0,37482	0,18892	0,21332	0,27461	0,33942	0,38832	0,86778
	Teste	0,34648	0,08712	0,18857	0,30870	0,36406	0,40315	0,45847
4-5-1	Treino	0,29367	0,09479	0,21672	0,22623	0,25938	0,31806	0,46561
	Validação	0,28721	0,13103	0,16744	0,17182	0,23813	0,39472	0,54348
	Teste	0,35045	0,19887	0,14826	0,17752	0,31882	0,44482	0,80469

Tabela A.7: Testes Cenário 2

Arquitectura	Conjunto Dados	$\overline{MSE}$	σ_{MSE}	MSE_{Min}	MSE_{Q1}	MSE_{Q2}	MSE_{Q3}	MSE_{Max}
4-1-1	Treino	0,30682	0,10105	0,20320	0,20922	0,29511	0,40012	0,43316
	Validação	0,31820	0,07927	0,18685	0,26266	0,30678	0,38891	0,44497
	Teste	0,34430	0,13347	0,19632	0,24177	0,31823	0,40689	0,64144
4-2-1	Treino	0,33201	0,06750	0,22174	0,30775	0,33121	0,38168	0,45098
	Validação	0,35559	0,10696	0,19399	0,30057	0,32793	0,40675	0,55567
	Teste	0,34724	0,10497	0,22220	0,24658	0,32981	0,39454	0,55158
4-3-1	Treino	0,30165	0,14796	0,20774	0,22413	0,23653	0,34515	0,69541
	Validação	0,30509	0,13210	0,19097	0,22632	0,26593	0,34066	0,64803
	Teste	0,34170	0,18992	0,18622	0,23861	0,27410	0,38048	0,84683
4-4-1	Treino	0,25343	0,06035	0,18877	0,21112	0,22906	0,30253	0,36591
	Validação	0,26124	0,05472	0,16707	0,22689	0,26138	0,30575	0,34833
	Teste	0,30095	0,08362	0,14220	0,27401	0,31487	0,33404	0,41966
4-5-1	Treino	0,26304	0,04939	0,21110	0,21985	0,24954	0,28838	0,35986
	Validação	0,29074	0,10115	0,20038	0,24367	0,26952	0,28179	0,56885
	Teste	0,29897	0,09585	0,20159	0,21940	0,26448	0,35246	0,48906

Tabela A.8: Testes Cenário 2A

Arquitectura	Conjunto Dados	$\overline{MSE}$	σ_{MSE}	MSE_{Min}	MSE_{Q1}	MSE_{Q2}	MSE_{Q3}	MSE_{Max}
4-1-1	Treino	0,18731	0,02370	0,14585	0,17288	0,18777	0,20483	0,22489
	Validação	0,18917	0,04222	0,13938	0,15490	0,19007	0,20137	0,28715
	Teste	0,16820	0,03643	0,10549	0,15368	0,16194	0,19114	0,24249
4-2-1	Treino	0,21762	0,04168	0,16817	0,17482	0,21028	0,24767	0,28355
	Validação	0,22176	0,09401	0,10284	0,14847	0,21078	0,27416	0,39042
	Teste	0,21125	0,10860	0,09859	0,12565	0,19238	0,26657	0,43685
4-3-1	Treino	0,18434	0,03897	0,13945	0,15303	0,17469	0,22621	0,25358
	Validação	0,21797	0,10815	0,10254	0,16189	0,18540	0,22662	0,45878
	Teste	0,20116	0,04721	0,14243	0,15977	0,19721	0,24463	0,27192
4-4-1	Treino	0,24466	0,09510	0,13395	0,15773	0,25079	0,28460	0,44310
	Validação	0,25180	0,06717	0,14978	0,20500	0,25679	0,29017	0,35629
	Teste	0,22221	0,11342	0,12220	0,14002	0,17003	0,29704	0,45175
4-5-1	Treino	0,29284	0,17264	0,14845	0,20022	0,26337	0,30798	0,74673
	Validação	0,27101	0,15000	0,15381	0,18851	0,23341	0,27511	0,67314
	Teste	0,26772	0,18133	0,07167	0,15965	0,19698	0,37810	0,69398

Tabela A.9: Testes Cenário 3

Arquitectura	Conjunto Dados	$\overline{MSE}$	σ_{MSE}	MSE_{Min}	MSE_{Q1}	MSE_{Q2}	MSE_{Q3}	MSE_{Max}
4-1-1	Treino	0,18170	0,04511	0,14292	0,14632	0,16520	0,19684	0,28144
	Validação	0,20311	0,04746	0,14264	0,17049	0,19282	0,24019	0,30098
	Teste	0,17135	0,04329	0,12643	0,13787	0,16404	0,19844	0,26343
4-2-1	Treino	0,22840	0,06881	0,13931	0,19483	0,22313	0,24588	0,38568
	Validação	0,24558	0,05840	0,15988	0,19181	0,26067	0,28780	0,32192
	Teste	0,27333	0,11606	0,16839	0,20766	0,23641	0,31071	0,56012
4-3-1	Treino	0,18427	0,04396	0,13459	0,14187	0,18196	0,21493	0,26126
	Validação	0,22069	0,09279	0,14058	0,14317	0,18761	0,29051	0,39548
	Teste	0,19882	0,04401	0,12669	0,18372	0,18995	0,22485	0,27299
4-4-1	Treino	0,24780	0,09136	0,13872	0,14953	0,26510	0,33470	0,37117
	Validação	0,24492	0,13453	0,10932	0,14557	0,22152	0,24869	0,52971
	Teste	0,29825	0,14361	0,13101	0,16157	0,30745	0,39087	0,52972
4-5-1	Treino	0,17480	0,04636	0,13271	0,14136	0,16229	0,19080	0,27740
	Validação	0,19268	0,07018	0,08195	0,13636	0,19649	0,25553	0,29177
	Teste	0,18332	0,03662	0,12301	0,15795	0,18532	0,21978	0,23360

Tabela A.10: Testes Cenário 3A

Arquitectura	Conjunto Dados	$\overline{MSE}$	σ_{MSE}	MSE_{Min}	MSE_{Q1}	MSE_{Q2}	MSE_{Q3}	MSE_{Max}
5-1-1	Treino	0,18254	0,03709	0,14502	0,15150	0,17155	0,19229	0,25905
	Validação	0,19544	0,02730	0,16297	0,17291	0,18963	0,21638	0,24088
	Teste	0,20027	0,05063	0,15308	0,16519	0,19695	0,21274	0,32760
5-2-1	Treino	0,22158	0,06055	0,15598	0,17885	0,22300	0,23371	0,37241
	Validação	0,23306	0,10815	0,10726	0,13972	0,19476	0,32073	0,43492
	Teste	0,22805	0,09081	0,09884	0,12362	0,23901	0,28996	0,37720
5-3-1	Treino	0,19041	0,04403	0,13611	0,16166	0,17790	0,21262	0,28872
	Validação	0,21233	0,04816	0,15003	0,16676	0,21422	0,26478	0,26897
	Teste	0,20893	0,06955	0,10564	0,15472	0,20699	0,27829	0,30053
5-4-1	Treino	0,21788	0,11789	0,13322	0,15887	0,17627	0,24101	0,53612
	Validação	0,19647	0,11069	0,06202	0,13696	0,16779	0,19356	0,44035
	Teste	0,23151	0,19804	0,10736	0,11028	0,19585	0,24500	0,76957
5-5-1	Treino	0,28249	0,14539	0,14541	0,17439	0,22828	0,31449	0,57800
	Validação	0,28572	0,17116	0,09511	0,16314	0,20910	0,40856	0,58270
	Teste	0,26149	0,16938	0,07664	0,15175	0,24796	0,29238	0,67395

Tabela A.11: Cenário 4

Cenário	Arquitectura	MSE_{NE}	Erro 0	Erro ±1	Erro ±2	Erro ±3	DTC = 5		DTC = 6		DTC = 7		DTC = 8	
							PI	PC	PI	PC	PI	PC	PI	PC
Cenário 1	4-1-1	0,72391	139	139	19	0	86	0	20	107	22	32	30	0
	4-2-1	0,74747	129	150	18	0	86	0	26	101	29	25	27	3
	4-3-1	0,75421	138	139	19	1	77	9	31	96	21	33	30	0
	4-4-1	0,95623	141	120	32	4	48	38	39	88	39	15	30	0
	4-5-1	0,70034	142	139	15	1	79	7	21	106	25	29	30	0
Cenário 1A	4-1-1	0,81481	137	136	22	2	81	5	16	111	33	21	30	0
	4-2-1	0,71044	142	138	16	1	58	28	44	83	23	31	30	0
	4-3-1	0,74411	138	140	18	1	74	12	28	99	27	27	30	0
	4-4-1	0,74747	152	126	15	4	63	23	28	99	24	30	30	0
	4-5-1	0,75084	128	151	18	0	86	0	18	109	35	19	30	0
Cenário 2	4-1-1	0,56566	168	116	13	0	49	37	27	100	29	25	24	6
	4-2-1	0,59596	155	132	9	1	61	25	17	110	34	20	30	0
	4-3-1	0,66330	148	133	16	0	66	20	27	100	26	28	30	0
	4-4-1	0,82155	133	139	24	1	86	0	1	126	47	7	30	0
	4-5-1	0,55892	158	130	9	0	64	22	15	112	36	18	24	6
Cenário 2A	4-1-1	0,55892	170	114	13	0	40	46	32	95	32	22	23	7
	4-2-1	0,55892	167	118	12	0	47	39	27	100	34	20	22	8
	4-3-1	0,51178	180	107	9	1	38	48	23	104	29	25	27	3
	4-4-1	0,65320	139	146	12	0	83	3	22	105	33	21	20	10
	4-5-1	0,64310	151	131	15	0	51	35	47	80	21	33	27	3

Tabela A.12: Resultado Simulação com “rede óptima”

Cenário	Arquitectura	MSE_{NE}	Erro 0	Erro ±1	Erro ±2	Erro ±3	NFC = 12		NFC = 13		NFC = 14		NFC = 15		NFC = 16	
							PI	PC	PI	PC	PI	PC	PI	PC	PI	PC
Cenário 3	4-1-1	0,70707	140	141	15	1	48	0	28	100	51	36	22	4	8	0
	4-2-1	0,73401	144	133	19	1	48	0	4	124	67	20	26	0	8	0
	4-3-1	0,82492	133	137	27	0	48	0	51	77	31	56	26	0	8	0
	4-4-1	1,13468	119	140	29	9	48	0	21	107	76	11	25	1	8	0
	4-5-1	0,65657	144	139	14	0	48	0	28	100	46	41	23	3	8	0
Cenário 4	5-1-1	0,82155	130	143	23	1	48	0	46	82	39	48	26	0	8	0
	5-2-1	0,92929	131	136	26	4	48	0	2	126	82	5	26	0	8	0
	5-3-1	0,72391	138	142	16	1	48	0	24	104	57	30	22	4	8	0
	5-4-1	0,79125	139	134	23	1	47	1	36	92	41	46	26	0	8	0
	5-5-1	1,00000	128	138	24	7	31	17	59	69	46	41	25	1	8	0

Tabela A.13: Resultado Simulação com “rede óptima”

Apêndice B

Código MATLAB

Nota: Exemplo para um ciclo com data division 60%, 20% e 20% para conjunto de treino, validação e teste, respectivamente (*rand*).

```
tic
 NR=10;
 Rnd_60_20_20=zeros(4,NR);
 clear NETOpt_Rnd_60_20_20;
 clear TROpt_Rnd_60_20_20;
 NETOpt_Rnd_60_20_20=newff(inputs1,targets1,5);
 NETOpt_Rnd_60_20_20.trainFcn='traingdx';
 NETOpt_Rnd_60_20_20.divideFcn ='dividerand';
 NETOpt_Rnd_60_20_20.divideParam.trainRatio=0.6;
 NETOpt_Rnd_60_20_20.divideParam.valRatio=0.20;
 NETOpt_Rnd_60_20_20.divideParam.testRatio=0.20;
 NETOpt_Rnd_60_20_20.trainParam.showWindow=false;
 NETOpt_Rnd_60_20_20.trainParam.showCommandLine=false;
 prev_aux=sim(NETOpt_Rnd_60_20_20,inputs1);
 erro_aux=targets1-prev_aux;
 Rnd_60_20_20(1,1)=TROpt_Rnd_60_20_20.perf(TROpt_Rnd_60_20_20.num_epochs+1);
 Rnd_60_20_20(2,1)=TROpt_Rnd_60_20_20.vperf(TROpt_Rnd_60_20_20.num_epochs+1);
 Rnd_60_20_20(3,1)=TROpt_Rnd_60_20_20.tperf(TROpt_Rnd_60_20_20.num_epochs+1);
 Rnd_60_20_20(4,1)=mse(erro_aux);
 i=2;
 for i=2:NR
            clear net_aux;
            clear tr_aux;
            net_aux=newff(inputs1,targets1,5);
            net_aux.trainFcn='traingdx';
            net_aux.divideFcn ='dividerand';
            net_aux.divideParam.trainRatio=0.6;
             net_aux.divideParam.valRatio=0.20;
             net_aux.divideParam.testRatio=0.20;
             net_aux.trainParam.showWindow=false;
             net_aux.trainParam.showCommandLine=false;
             [net_aux,tr_aux]=train(net_aux,inputs1,targets1);
             if
            (tr_aux.tperf(tr_aux.num_epochs+1)<TROpt_Rnd_60_20_20.tperf(TROpt_Rnd_60_20_20.num_epochs+1)),
            NETOpt_Rnd_60_20_20=net_aux; TROpt_Rnd_60_20_20=tr_aux; end;
             prev_aux=sim(net_aux,inputs1);
             erro_aux=targets1-prev_aux;
             Rnd_60_20_20(1,i)=tr_aux.perf(tr_aux.num_epochs+1);
             Rnd_60_20_20(2,i)=tr_aux.vperf(tr_aux.num_epochs+1);
             Rnd_60_20_20(3,i)=tr_aux.tperf(tr_aux.num_epochs+1);
             Rnd_60_20_20(4,i)=mse(erro_aux);
 end;
 MSE_medio_Rnd_60_20_20=mean(Rnd_60_20_20');
 MSE_DesvPad_Rnd_60_20_20=std(Rnd_60_20_20');
 Quartis_Rnd_60_20_20=quantile(Rnd_60_20_20',[0 0.25 0.50 0.75 1]);
 Estatistica_Rnd_60_20_frm-e0=[MSE_medio_Rnd_60_20_20;MSE_DesvPad_Rnd_60_20_20;Quartis_Rnd_60_20_20];
 Dur_Rnd_60_20_20=toc;
 warning('Fim de Rnd_60_20_20.')
```

Figura B.1: Código MATLAB –Teste para Data Division

Nota: Exemplo para um ciclo com data division 60%, 20% e 20% para conjunto de treino, validação e teste, respectivamente (*block*)

```
tic
 NR=10;
 Blk_60_20_20=zeros(4,NR);
 clear NETOpt_Blk_60_20_20;
 clear TROpt_Blk_60_20_20;
 NETOpt_Blk_60_20_20=newff(inputs1,targets1,5);
 NETOpt_Blk_60_20_20.trainFcn='traingdx';
 NETOpt_Blk_60_20_20.divideFcn ='divideblock';
 NETOpt_Blk_60_20_20.divideParam.trainRatio=0.6;
 NETOpt_Blk_60_20_20.divideParam.valRatio=0.20;
 NETOpt_Blk_60_20_20.divideParam.testRatio=0.20;
 NETOpt_Blk_60_20_20.trainParam.showWindow=false;
 NETOpt_Blk_60_20_20.trainParam.showCommandLine=false;
 [NETOpt_Blk_60_20_20, TROpt_Blk_60_20_20]=train(NETOpt_Blk_60_20_20,inputs1,targets1);
 prev_aux=sim(NETOpt_Blk_60_20_20,inputs1);
 erro_aux=targets1-prev_aux;
 Blk_60_20_20(1,1)=TROpt_Blk_60_20_20.perf(TROpt_Blk_60_20_20.num_epochs+1);
 Blk_60_20_20(2,1)=TROpt_Blk_60_20_20.vperf(TROpt_Blk_60_20_20.num_epochs+1);
 Blk_60_20_20(3,1)=TROpt_Blk_60_20_20.tperf(TROpt_Blk_60_20_20.num_epochs+1);
 Blk_60_20_20(4,1)=mse(erro_aux);
 i=2;
 for i=2:NR
            clear net_aux;
            clear tr_aux;
            net_aux=newff(inputs1,targets1,5);
            net_aux.trainFcn='traingdx';
            net_aux.divideFcn ='divideblock';
            net_aux.divideParam.trainRatio=0.6;
            net_aux.divideParam.valRatio=0.20;
            net_aux.divideParam.testRatio=0.20;
            net_aux.trainParam.showWindow=false;
            net_aux.trainParam.showCommandLine=false;
            [net_aux,tr_aux]=train(net_aux,inputs1,targets1);
            if
(tr_aux.tperf(tr_aux.num_epochs+1)<TROpt_Blk_60_20_20.tperf(TROpt_Blk_60_20_20.num_epochs+1)),
             NETOpt_Blk_60_20_20=net_aux; TROpt_Blk_60_20_20=tr_aux;
 end;
            prev_aux=sim(net_aux,inputs1);
            erro_aux=targets1-prev_aux;
            Blk_60_20_20(1,i)=tr_aux.perf(tr_aux.num_epochs+1);
            Blk_60_20_20(2,i)=tr_aux.vperf(tr_aux.num_epochs+1);
            Blk_60_20_20(3,i)=tr_aux.tperf(tr_aux.num_epochs+1);
             Blk_60_20_20(4,i)=mse(erro_aux);
 end;
 MSE_medio_Blk_60_20_20=mean(Blk_60_20_20');
 MSE_DesvPad_Blk_60_20_20=std(Blk_60_20_20');
 Quartis_Blk_60_20_20=quantile(Blk_60_20_20',[0 0.25 0.50 0.75 1]);
 Estatistica_Blk_60_20_20=[MSE_medio_Blk_60_20_20;MSE_DesvPad_Blk_60_20_20;Quartis_Blk_60_20_20];
 Dur_Blk_60_20_20=toc;
 warning('Fim de Blk_60_20_20.')
```

Figura B.2: Código MATLAB –Teste para Data Division

Nota: Exemplo para um ciclo com $\alpha = 0.05$

```
tic
 NR=10;
 MU_0025=zeros(4,NR);
 clear NETOpt_MU_0025;
 clear TROpt_MU_0025;
 NETOpt_MU_0025=newff(inputs1,targets1,5);
 NETOpt_MU_0025.trainFcn='traingdx';
 NETOpt_MU_0025.trainParam.mu=0.025;
 NETOpt_MU_0025.trainParam.showWindow=false;
 NETOpt_MU_0025.trainParam.showCommandLine=false;
 [NETOpt_MU_0025, TROpt_MU_0025]=train(NETOpt_MU_0025,inputs1,targets1);
 prev_aux=sim(NETOpt_MU_0025,inputs1);
 erro_aux=targets1-prev_aux;
 MU_0025(1,1)=TROpt_MU_0025.perf(TROpt_MU_0025.num_epochs+1);
 MU_0025(2,1)=TROpt_MU_0025.vperf(TROpt_MU_0025.num_epochs+1);
 MU_0025(3,1)=TROpt_MU_0025.tperf(TROpt_MU_0025.num_epochs+1);
 MU_0025(4,1)=mse(erro_aux);
 i=2;
 for i=2:NR
            clear net_aux;
            clear tr_aux;
            net_aux=newff(inputs1,targets1,5);
            net_aux.trainFcn='traingdx';
             net_aux.trainParam.mu=0.025;
             net_aux.trainParam.showWindow=false;
             net_aux.trainParam.showCommandLine=false;
            [net_aux,tr_aux]=train(net_aux,inputs1,targets1);
                    if    (tr_aux.tperf(tr_aux.num_epochs+1)<TROpt_MU_0025.tperf(TROpt_MU_0025.num_epochs+1)),    NE-
TOpt_MU_0025=net_aux;
            TROpt_MU_0025=tr_aux; end;
             prev_aux=sim(net_aux,inputs1);
            erro_aux=targets1-prev_aux;
             MU_0025(1,i)=tr_aux.perf(tr_aux.num_epochs+1);
            MU_0025(2,i)=tr_aux.vperf(tr_aux.num_epochs+1);
            MU_0025(3,i)=tr_aux.tperf(tr_aux.num_epochs+1);
             MU_0025(4,i)=mse(erro_aux);
 end;
 MSE_medio_MU_0025=mean(MU_0025');
 MSE_DesvPad_MU_0025=std(MU_0025');
 Quartis_MU_0025=quantile(MU_0025',[0 0.25 0.50 0.75 1])
 Estatistica_MU_0025=[MSE_medio_MU_0025;MSE_DesvPad_MU_0025;Quartis_MU_0025]
 Dur_MU_0025=toc
 warning('Fim de MU_0025.')
```

Figura B.3: Código MATLAB – Teste para Taxa de *Momentum*

Nota: Exemplo para um ciclo com NR=10

```
tic
 NR=10;
 NR_10=zeros(4,NR);
 clear NETOpt_NR_10;
 clear TROpt_NR_10;
 NETOpt_NR_10=newff(inputs1,targets1,5);
 NETOpt_NR_10.trainFcn='traingdx';
 NETOpt_NR_10.trainParam.showWindow=false;
 NETOpt_NR_10.trainParam.showCommandLine=false;
 [NETOpt_NR_10, TROpt_NR_10]=train(NETOpt_NR_10,inputs1,targets1);
 prev_aux=sim(NETOpt_NR_10,inputs1);
 erro_aux=targets1-prev_aux;
 NR_10(1,1)=TROpt_NR_10.perf(TROpt_NR_10.num_epochs+1);
 NR_10(2,1)=TROpt_NR_10.vperf(TROpt_NR_10.num_epochs+1);
 NR_10(3,1)=TROpt_NR_10.tperf(TROpt_NR_10.num_epochs+1);
 NR_10(4,1)=mse(erro_aux);
 i=2;
 for i=2:NR
            clear net_aux;
            clear tr_aux;
            net_aux=newff(inputs1,targets1,5);
             net_aux.trainFcn='traingdx';
             net_aux.trainParam.showWindow=false;
             net_aux.trainParam.showCommandLine=false;
             [net_aux,tr_aux]=train(net_aux,inputs1,targets1);
             if (tr_aux.tperf(tr_aux.num_epochs+1)<TROpt_NR_10.tperf(TROpt_NR_10.num_epochs+1)),
             NETOpt_NR_10=net_aux; TROpt_NR_10=tr_aux;
 end;
             prev_aux=sim(net_aux,inputs1);
             erro_aux=targets1-prev_aux;
            NR_10(1,i)=tr_aux.perf(tr_aux.num_epochs+1);
             NR_10(2,i)=tr_aux.vperf(tr_aux.num_epochs+1);
             NR_10(3,i)=tr_aux.tperf(tr_aux.num_epochs+1);
             NR_10(4,i)=mse(erro_aux);
 end;
 MSE_medio_NR_10=mean(NR_10');
 MSE_DesvPad_NR_10=std(NR_10');
 Quartis_NR_10=quantile(NR_10',[0 0.25 0.50 0.75 1]);
 Estatistica_NR_10=[MSE_medio_NR_10;

MSE_DesvPad_NR_10;

Quartis_NR_10'];
 Dur_NR_10=toc;
 warning('Fim de NR=10.')
```

Figura B.4: Código MATLAB – Teste para o NR

Nota: Exemplo para um ciclo da arquitectura $4-1-1$

```
tic
 NR=10;
 Cen1_1=zeros(4,NR);
 clear NETOpt_Cen1_1;
 clear TROpt_Cen1_1;
 NETOpt_Cen1_1=newff(inputs1,targets1,1);
 NETOpt_Cen1_1.trainFcn='traingdx';
 NETOpt_Cen1_1.divideFcn ='dividerand';
 NETOpt_Cen1_1.divideParam.trainRatio=0.8;
 NETOpt_Cen1_1.divideParam.valRatio=0.10;
 NETOpt_Cen1_1.divideParam.testRatio=0.10;
 NETOpt_Cen1_1.trainParam.showWindow=false;
 NETOpt_Cen1_1.trainParam.showCommandLine=false;
 [NETOpt_Cen1_1, TROpt_Cen1_1]=train(NETOpt_Cen1_1,inputs1,targets1);
 prev_aux=sim(NETOpt_Cen1_1,inputs1);
 erro_aux=targets1-prev_aux;
 Cen1_1(1,1)=TROpt_Cen1_1.perf(TROpt_Cen1_1.num_epochs+1);
 Cen1_1(2,1)=TROpt_Cen1_1.vperf(TROpt_Cen1_1.num_epochs+1);
 Cen1_1(3,1)=TROpt_Cen1_1.tperf(TROpt_Cen1_1.num_epochs+1);
 Cen1_1(4,1)=mse(erro_aux);
 i=2;
 for i=2:NR
              clear net_aux;
             clear tr_aux;
             net_aux=newff(inputs1,targets1,1);
              net_aux.trainFcn='traingdx';
              net_aux.divideFcn ='dividerand';
              net_aux.divideParam.trainRatio=0.8;
             net_aux.divideParam.valRatio=0.10;
              net_aux.divideParam.testRatio=0.10;
              net_aux.trainParam.showWindow=false;
              net_aux.trainParam.showCommandLine=false;
              [net_aux,tr_aux]=train(net_aux,inputs1,targets1);
              if (tr_aux.tperf(tr_aux.num_epochs+1)<TROpt_Cen1_1.tperf(TROpt_Cen1_1.num_epochs+1)),
              NETOpt_Cen1_1=net_aux; TROpt_Cen1_1=tr_aux; end;
              prev_aux=sim(net_aux,inputs1);
             erro_aux=targets1-prev_aux;
              Cen1_1(1,i)=tr_aux.perf(tr_aux.num_epochs+1);
              Cen1_1(2,i)=tr_aux.vperf(tr_aux.num_epochs+1);
              Cen1_1(3,i)=tr_aux.tperf(tr_aux.num_epochs+1);
              Cen1_1(4,i)=mse(erro_aux);
 end;
 MSE_medio_Cen1_1=mean(Cen1_1');
 MSE_DesvPad_Cen1_1=std(Cen1_1');
 Quartis_Cen1_1=quantile(Cen1_1',[0 0.25 0.50 0.75 1]);
 Estatistica_Cen1_1=[MSE_medio_Cen1_1;MSE_DesvPad_Cen1_1;Quartis_Cen1_1];
 Dur_Cen1_1=toc;
 warning('Fim de Cen1_1.')
```

Figura B.5: Código MATLAB – Cenário 1

Nota: Exemplo para um ciclo da arquitectura $5-1-1$

```
tic
 NR=10;
 Cen1A_1=zeros(4,NR);
 clear NETOpt_Cen1A_1;
 clear TROpt_Cen1A_1;
 NETOpt_Cen1A_1=newff(inputs1A,targets1,1);
 NETOpt_Cen1A_1.trainFcn='traingdx';
 NETOpt_Cen1A_1.divideFcn ='dividerand';
 NETOpt_Cen1A_1.divideParam.trainRatio=0.8;
 NETOpt_Cen1A_1.divideParam.valRatio=0.10;
 NETOpt_Cen1A_1.divideParam.testRatio=0.10;
 NETOpt_Cen1A_1.trainParam.showWindow=false;
 NETOpt_Cen1A_1.trainParam.showCommandLine=false;
 [NETOpt_Cen1A_1, TROpt_Cen1A_1]=train(NETOpt_Cen1A_1,inputs1A,targets1);
 prev_aux=sim(NETOpt_Cen1A_1,inputs1A);
 erro_aux=targets1-prev_aux;
 Cen1A_1(1,1)=TROpt_Cen1A_1.perf(TROpt_Cen1A_1.num_epochs+1);
 Cen1A_1(2,1)=TROpt_Cen1A_1.vperf(TROpt_Cen1A_1.num_epochs+1);
 Cen1A_1(3,1)=TROpt_Cen1A_1.tperf(TROpt_Cen1A_1.num_epochs+1);
 Cen1A_1(4,1)=mse(erro_aux);
 i=2;
 for i=2:NR
clear net_aux;
              clear tr_aux;
              net_aux=newff(inputs1A,targets1,1);
              net_aux.trainFcn='traingdx';
              net_aux.divideFcn ='dividerand';
              net_aux.divideParam.trainRatio=0.8;
              net_aux.divideParam.valRatio=0.10;
              net_aux.divideParam.testRatio=0.10;
              net_aux.trainParam.showWindow=false;
              net_aux.trainParam.showCommandLine=false;
              [net_aux,tr_aux]=train(net_aux,inputs1A,targets1);
              if (tr_aux.tperf(tr_aux.num_epochs+1)<TROpt_Cen1A_1.tperf(TROpt_Cen1A_1.num_epochs+1)),
              NETOpt_Cen1A_1=net_aux; TROpt_Cen1A_1=tr_aux; end;
              prev_aux=sim(net_aux,inputs1A);
              erro_aux=targets1-prev_aux;
              Cen1A_1(1,i)=tr_aux.perf(tr_aux.num_epochs+1);
              Cen1A_1(2,i)=tr_aux.vperf(tr_aux.num_epochs+1);
              Cen1A_1(3,i)=tr_aux.tperf(tr_aux.num_epochs+1);
             Cen1A_1(4,i)=mse(erro_aux);
 end;
 MSE_medio_Cen1A_1=mean(Cen1A_1');
 MSE_DesvPad_Cen1A_1=std(Cen1A_1');
 Quartis_Cen1A_1=quantile(Cen1A_1',[0 0.25 0.50 0.75 1]);
 Estatistica_Cen1A_1=[MSE_medio_Cen1A_1;MSE_DesvPad_Cen1A_1;Quartis_Cen1A_1];
 Dur_Cen1A_1=toc;
 warning('Fim de Cen1A_1.')
```

Figura B.6: Código MATLAB – Cenário 1A

Nota: Exemplo para um ciclo da arquitectura $4-1-1$

```
tic
 NR=10;
 Cen2_1=zeros(4,NR);
 clear NETOpt_Cen2_1;
 clear TROpt_Cen2_1;
 NETOpt_Cen2_1=newff(inputs2,targets1,1);
 NETOpt_Cen2_1.trainFcn='traingdx';
 NETOpt_Cen2_1.divideFcn ='dividerand';
 NETOpt_Cen2_1.divideParam.trainRatio=0.8;
 NETOpt_Cen2_1.divideParam.valRatio=0.10;
 NETOpt_Cen2_1.divideParam.testRatio=0.10;
 NETOpt_Cen2_1.trainParam.showWindow=false;
 NETOpt_Cen2_1.trainParam.showCommandLine=false;
 [NETOpt_Cen2_1, TROpt_Cen2_1]=train(NETOpt_Cen2_1,inputs2,targets1);
 prev_aux=sim(NETOpt_Cen2_1,inputs2);
 erro_aux=targets1-prev_aux;
 Cen2_1(1,1)=TROpt_Cen2_1.perf(TROpt_Cen2_1.num_epochs+1);
 Cen2_1(2,1)=TROpt_Cen2_1.vperf(TROpt_Cen2_1.num_epochs+1);
 Cen2_1(3,1)=TROpt_Cen2_1.tperf(TROpt_Cen2_1.num_epochs+1);
 Cen2_1(4,1)=mse(erro_aux);
 i=2;
 for i=2:NR
              clear net_aux;
              clear tr_aux;
              net_aux=newff(inputs2,targets1,1);
             net_aux.trainFcn='traingdx';
              net_aux.divideFcn ='dividerand';
              net_aux.divideParam.trainRatio=0.8;
              net_aux.divideParam.valRatio=0.10;
              net_aux.divideParam.testRatio=0.10;
              net_aux.trainParam.showWindow=false;
              net_aux.trainParam.showCommandLine=false;
              [net_aux,tr_aux]=train(net_aux,inputs2,targets1);
              if (tr_aux.tperf(tr_aux.num_epochs+1)<TROpt_Cen2_1.tperf(TROpt_Cen2_1.num_epochs+1)),
              NETOpt_Cen2_1=net_aux; TROpt_Cen2_1=tr_aux; end;
              prev_aux=sim(net_aux,inputs2);
              erro_aux=targets1-prev_aux;
              Cen2_1(1,i)=tr_aux.perf(tr_aux.num_epochs+1);
              Cen2_1(2,i)=tr_aux.vperf(tr_aux.num_epochs+1);
              Cen2_1(3,i)=tr_aux.tperf(tr_aux.num_epochs+1);
              Cen2_1(4,i)=mse(erro_aux);
 end;
 MSE_medio_Cen2_1=mean(Cen2_1');
 MSE_DesvPad_Cen2_1=std(Cen2_1');
 Quartis_Cen2_1=quantile(Cen2_1',[0 0.25 0.50 0.75 1]);
 Estatistica_Cen2_1=[MSE_medio_Cen2_1;MSE_DesvPad_Cen2_1;Quartis_Cen2_1];
 Dur_Cen2_1=toc;
 warning('Fim de Cen2_1.')
```

Figura B.7: Código MATLAB – Cenário 2

Nota: Exemplo para um ciclo da arquitectura $5-1-1$

```
tic
 NR=10;
 Cen2A_1=zeros(4,NR);
 clear NETOpt_Cen2A_1;
 clear TROpt_Cen2A_1;
 NETOpt_Cen2A_1=newff(inputs2A,targets1,1);
 NETOpt_Cen2A_1.trainFcn='traingdx';
 NETOpt_Cen2A_1.divideFcn ='dividerand';
 NETOpt_Cen2A_1.divideParam.trainRatio=0.8;
 NETOpt_Cen2A_1.divideParam.valRatio=0.10;
 NETOpt_Cen2A_1.divideParam.testRatio=0.10;
 NETOpt_Cen2A_1.trainParam.showWindow=false;
 NETOpt_Cen2A_1.trainParam.showCommandLine=false;
 [NETOpt_Cen2A_1, TROpt_Cen2A_1]=train(NETOpt_Cen2A_1,inputs2A,targets1);
 prev_aux=sim(NETOpt_Cen2A_1,inputs2A);
 erro_aux=targets1-prev_aux;
 Cen2A_1(1,1)=TROpt_Cen2A_1.perf(TROpt_Cen2A_1.num_epochs+1);
 Cen2A_1(2,1)=TROpt_Cen2A_1.vperf(TROpt_Cen2A_1.num_epochs+1);
 Cen2A_1(3,1)=TROpt_Cen2A_1.tperf(TROpt_Cen2A_1.num_epochs+1);
 Cen2A_1(4,1)=mse(erro_aux);
 i=2;
 for i=2:NR
            clear net_aux;
            clear tr_aux;
            net_aux=newff(inputs2A,targets1,1);
            net_aux.trainFcn='traingdx';
             net_aux.divideFcn ='dividerand';
             net_aux.divideParam.trainRatio=0.8;
            net_aux.divideParam.valRatio=0.10;
            net_aux.divideParam.testRatio=0.10;
            net_aux.trainParam.showWindow=false;
            net_aux.trainParam.showCommandLine=false;
             [net_aux,tr_aux]=train(net_aux,inputs2A,targets1);
            if (tr_aux.tperf(tr_aux.num_epochs+1)<TROpt_Cen2A_1.tperf(TROpt_Cen2A_1.num_epochs+1)),
             NETOpt_Cen2A_1=net_aux; TROpt_Cen2A_1=tr_aux; end;
             prev_aux=sim(net_aux,inputs2A);
             erro_aux=targets1-prev_aux;
             Cen2A_1(1,i)=tr_aux.perf(tr_aux.num_epochs+1);
             Cen2A_1(2,i)=tr_aux.vperf(tr_aux.num_epochs+1);
            Cen2A_1(3,i)=tr_aux.tperf(tr_aux.num_epochs+1);
             Cen2A_1(4,i)=mse(erro_aux);
 end;
 MSE_medio_Cen2A_1=mean(Cen2A_1');
 MSE_DesvPad_Cen2A_1=std(Cen2A_1');
 Quartis_Cen2A_1=quantile(Cen2A_1',[0 0.25 0.50 0.75 1]);
 Estatistica_Cen2A_1=[MSE_medio_Cen2A_1;
MSE_DesvPad_Cen2A_1;Quartis_Cen2A_1];
 Dur_Cen2A_1=toc;
 warning('Fim de Cen2A_1.')
```

Figura B.8: Código MATLAB – Cenário 2A

Nota: Exemplo para um ciclo da arquitectura $4-1-1$

```
tic
 NR=10;
 Cen3_1=zeros(4,NR);
 clear NETOpt_Cen3_1;
 clear TROpt_Cen3_1;
 NETOpt_Cen3_1=newff(inputs2,targets2,1);
 NETOpt_Cen3_1.trainFcn='traingdx';
 NETOpt_Cen3_1.divideFcn ='dividerand';
 NETOpt_Cen3_1.divideParam.trainRatio=0.8;
 NETOpt_Cen3_1.divideParam.valRatio=0.10;
 NETOpt_Cen3_1.divideParam.testRatio=0.10;
 NETOpt_Cen3_1.trainParam.showWindow=false;
 NETOpt_Cen3_1.trainParam.showCommandLine=false;
 [NETOpt_Cen3_1, TROpt_Cen3_1]=train(NETOpt_Cen3_1,inputs2,targets2);
 prev_aux=sim(NETOpt_Cen3_1,inputs2);
 erro_aux=targets2-prev_aux;
 Cen3_1(1,1)=TROpt_Cen3_1.perf(TROpt_Cen3_1.num_epochs+1);
 Cen3_1(2,1)=TROpt_Cen3_1.vperf(TROpt_Cen3_1.num_epochs+1);
 Cen3_1(3,1)=TROpt_Cen3_1.tperf(TROpt_Cen3_1.num_epochs+1);
 Cen3_1(4,1)=mse(erro_aux);
 i=2;
 for i=2:NR
              clear net_aux;
              clear tr_aux;
             net_aux=newff(inputs2,targets2,1);
              net_aux.trainFcn='traingdx';
              net_aux.divideFcn ='dividerand';
              net_aux.divideParam.trainRatio=0.8;
              net_aux.divideParam.valRatio=0.10;
             net_aux.divideParam.testRatio=0.10;
              net_aux.trainParam.showWindow=false;
             net_aux.trainParam.showCommandLine=false;
              [net_aux,tr_aux]=train(net_aux,inputs2,targets2);
              if (tr_aux.tperf(tr_aux.num_epochs+1)<TROpt_Cen3_1.tperf(TROpt_Cen3_1.num_epochs+1)),
             NETOpt_Cen3_1=net_aux; TROpt_Cen3_1=tr_aux; end;
              prev_aux=sim(net_aux,inputs2);
              erro_aux=targets2-prev_aux;
             Cen3_1(1,i)=tr_aux.perf(tr_aux.num_epochs+1);
              Cen3_1(2,i)=tr_aux.vperf(tr_aux.num_epochs+1);
              Cen3_1(3,i)=tr_aux.tperf(tr_aux.num_epochs+1);
              Cen3_1(4,i)=mse(erro_aux);
 end;
 MSE_medio_Cen3_1=mean(Cen3_1');
 MSE_DesvPad_Cen3_1=std(Cen3_1');
 Quartis_Cen3_1=quantile(Cen3_1',[0 0.25 0.50 0.75 1]);
 Estatistica_Cen3_1=[MSE_medio_Cen3_1;MSE_DesvPad_Cen3_1;Quartis_Cen3_1];
 Dur_Cen3_1=toc;
 warning('Fim de Cen3_1.')
```

Figura B.9: Código MATLAB – Cenário 3

Nota: Exemplo para um ciclo da arquitectura $5-1-1$

```
tic
 NR=10;
 Cen3A_1=zeros(4,NR);
 clear NETOpt_Cen3A_1;
 clear TROpt_Cen3A_1;
 NETOpt_Cen3A_1=newff(inputs2A,targets2,1);
 NETOpt_Cen3A_1.trainFcn='traingdx';
 NETOpt_Cen3A_1.divideFcn ='dividerand';
 NETOpt_Cen3A_1.divideParam.trainRatio=0.8;
 NETOpt_Cen3A_1.divideParam.valRatio=0.10;
 NETOpt_Cen3A_1.divideParam.testRatio=0.10;
 NETOpt_Cen3A_1.trainParam.showWindow=false;
 NETOpt_Cen3A_1.trainParam.showCommandLine=false;
 [NETOpt_Cen3A_1, TROpt_Cen3A_1]=train(NETOpt_Cen3A_1,inputs2A,targets2);
 prev_aux=sim(NETOpt_Cen3A_1,inputs2A);
 erro_aux=targets2-prev_aux;
 Cen3A_1(1,1)=TROpt_Cen3A_1.perf(TROpt_Cen3A_1.num_epochs+1);
 Cen3A_1(2,1)=TROpt_Cen3A_1.vperf(TROpt_Cen3A_1.num_epochs+1);
 Cen3A_1(3,1)=TROpt_Cen3A_1.tperf(TROpt_Cen3A_1.num_epochs+1);
 Cen3A_1(4,1)=mse(erro_aux);
 i=2;
 for i=2:NR
             clear net_aux;
             clear tr_aux;
            net_aux=newff(inputs2A,targets2,1);
            net_aux.trainFcn='traingdx';
             net_aux.divideFcn ='dividerand';
             net_aux.divideParam.trainRatio=0.8;
             net_aux.divideParam.valRatio=0.10;
             net_aux.divideParam.testRatio=0.10;
            net_aux.trainParam.showWindow=false;
             net_aux.trainParam.showCommandLine=false;
             [net_aux,tr_aux]=train(net_aux,inputs2A,targets2);
             if (tr_aux.tperf(tr_aux.num_epochs+1)<TROpt_Cen3A_1.tperf(TROpt_Cen3A_1.num_epochs+1)),
             NETOpt_Cen3A_1=net_aux; TROpt_Cen3A_1=tr_aux; end;
            prev_aux=sim(net_aux,inputs2A);
            erro_aux=targets2-prev_aux;
            Cen3A_1(1,i)=tr_aux.perf(tr_aux.num_epochs+1);
            Cen3A_1(2,i)=tr_aux.vperf(tr_aux.num_epochs+1);
             Cen3A_1(3,i)=tr_aux.tperf(tr_aux.num_epochs+1);
             Cen3A_1(4,i)=mse(erro_aux);
 end;
 MSE_medio_Cen3A_1=mean(Cen3A_1');
 MSE_DesvPad_Cen3A_1=std(Cen3A_1');
 Quartis_Cen3A_1=quantile(Cen3A_1',[0 0.25 0.50 0.75 1]);
 Estatistica_Cen3A_1=[MSE_medio_Cen3A_1;MSE_DesvPad_Cen3A_1;Quartis_Cen3A_1];
 Dur_Cen3A_1=toc;
 warning('Fim de Cen3A_1.')
```

Figura B.10: Código MATLAB – Cenário 3A

Nota: Exemplo para um ciclo da arquitectura $5-1-1$

```
tic
 NR=10;
 Cen4_1=zeros(4,NR);
 clear NETOpt_Cen4_1;
 clear TROpt_Cen4_1;
 NETOpt_Cen4_1=newff(inputs3,targets2,1);
 NETOpt_Cen4_1.trainFcn='traingdx';
 NETOpt_Cen4_1.divideFcn ='dividerand';
 NETOpt_Cen4_1.divideParam.trainRatio=0.8;
 NETOpt_Cen4_1.divideParam.valRatio=0.10;
 NETOpt_Cen4_1.divideParam.testRatio=0.10;
 NETOpt_Cen4_1.trainParam.showWindow=false;
 NETOpt_Cen4_1.trainParam.showCommandLine=false;
 [NETOpt_Cen4_1, TROpt_Cen4_1]=train(NETOpt_Cen4_1,inputs3,targets2);
 prev_aux=sim(NETOpt_Cen4_1,inputs3);
 erro_aux=targets2-prev_aux;
 Cen4_1(1,1)=TROpt_Cen4_1.perf(TROpt_Cen4_1.num_epochs+1);
 Cen4_1(2,1)=TROpt_Cen4_1.vperf(TROpt_Cen4_1.num_epochs+1);
 Cen4_1(3,1)=TROpt_Cen4_1.tperf(TROpt_Cen4_1.num_epochs+1);
 Cen4_1(4,1)=mse(erro_aux);
 i=2;
 for i=2:NR
             clear net_aux;
             clear tr_aux;
             net_aux=newff(inputs3,targets2,1);
             net_aux.trainFcn='traingdx';
            net_aux.divideFcn ='dividerand';
            net_aux.divideParam.trainRatio=0.8;
            net_aux.divideParam.valRatio=0.10;
             net_aux.divideParam.testRatio=0.10;
             net_aux.trainParam.showWindow=false;
            net_aux.trainParam.showCommandLine=false;
            [net_aux,tr_aux]=train(net_aux,inputs3,targets2);
            if (tr_aux.tperf(tr_aux.num_epochs+1)<TROpt_Cen4_1.tperf(TROpt_Cen4_1.num_epochs+1)),
            NETOpt_Cen4_1=net_aux; TROpt_Cen4_1=tr_aux; end;
             prev_aux=sim(net_aux,inputs3);
             erro_aux=targets2-prev_aux;
             Cen4_1(1,i)=tr_aux.perf(tr_aux.num_epochs+1);
            Cen4_1(2,i)=tr_aux.vperf(tr_aux.num_epochs+1);
             Cen4_1(3,i)=tr_aux.tperf(tr_aux.num_epochs+1);
             Cen4_1(4,i)=mse(erro_aux);
 end;
 MSE_medio_Cen4_1=mean(Cen4_1');
 MSE_DesvPad_Cen4_1=std(Cen4_1');
 Quartis_Cen4_1=quantile(Cen4_1',[0 0.25 0.50 0.75 1]);
 Estatistica_Cen4_1=[MSE_medio_Cen4_1;MSE_DesvPad_Cen4_1;Quartis_Cen4_1];
 Dur_Cen4_1=toc;
 warning('Fim de Cen4_1.')
```

Figura B.11: Código MATLAB – Cenário 4