INSTITUTO POLITÉCNICO DE LISBOA
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

SysPlan

Ricardo Candeias n.° 42087, e-mail:

a42087@alunos.isel.pt

Orientador: Prof. Nuno Leite

Projeto e Seminário

Relatório Inicial

junho 2021

Abstract

Com o avanço na área de *Inteligência Artificial* (IA) diversos serviços podem ser prestados por robôs que previamente apenas eram proporcionados por humanos. Nomeadamente serviços que envolvem robôs de locomoção local por exemplo, em edifícios. Um robô que ministra o serviço de *check in*, num aeroporto, é uma das inspirações para o projeto. Este tem como objetivo o desenvolvimento do conhecimento e das competências na área de IA dos participantes. Pretende-se, portanto, produzir dois componentes de software principais com esse mesmo intuito. Uma *Rede Neuronal* (RN) para a leitura de caracteres, que representa a leitura de um bilhete para a obtenção do terminal destino. Um algoritmo de pesquisa para o planeamento de um caminho para este mesmo destino, que simula o transporte de malas até ao terminal. Para visualização do funcionamento destes componentes, pretende-se também desenvolver um ambiente gráfico. Como resultado espera-se uma prova de conceito funcional que iniciou e expandiu o domínio de IA dos participantes.

Palavras Chave: Rede Neuronal, Algoritmo de Pesquisa, Caminho, Leitura.

List of Acronyms

Contents

List of Figures

Chapter 1

Introduction

Com o crescente desenvolvimento da *Inteligência Artificial* (IA), regista-se atualmente uma grande tendência para a automação de tarefas, que são agora realizadas por sistemas dotados de IA, em vez de serem realizadas por humanos. Existem diversas tarefas que máquinas executam de forma mais eficiente e consistente do que humanos. Uma das tarefas que máquinas irão dominar no futuro á o transporte de cargas e pessoas *Staff (2018)*.

Chapter 2

State of the art

Hoje em dia existem uma multitude de serviços que robôs conseguem providenciar, tais como rececionistas, robôs de limpeza, arrumador de malas em hotéis, entre outros. Existem empresas que comercializam robôs deste tipo *Chang (2015); Servicerobots*. Existem também serviços mais delicados como cuidadores em lares. Diversos indicadores demonstram a necessidade destes robôs tornarem-se uma realidade comum, devido ao envelhecimento da população *Walker*. Mais relevante devido à inspiração que prestou para este projeto é um robô de transporte de malas em aeroportos. Como por exemplo o robô '*Leo*', que tem funcionalidades como dar ,check in, imprimir etiquetas de malas, transporta-las, bem como a capacidade de evitar obstáculos e manobrar numa área de elevada densidade de pessoas *Loughran (2016)*.

Chapter 3

Requirements

Contents

3.1 Formulação do problema

O sistema desenvolvido neste projeto, foca-se na obtenção de um destino e no cálculo da rota para o mesmo, tendo por base conceptual um sistema semelhante ao Robô '*Leo*'. Ou seja, desenvolver componentes de software para um robô, por exemplo, de transporte de malas num aeroporto que inclui:

- Leitura e reconhecimento de caracteres manuscritos/bilhete indicando o destino, o terminal de *check in*
- Sistema de planeamento de caminho até ao terminal do aeroporto
- Interface Gráfica

3.1.1 Requisitos obrigatórios

The mandatory functional requirements are the following:

- Desenvolver uma *Rede Neuronal* (RN) de raiz, com a função de ler caracteres manuscritos, programada em *C#*;
- Implementar o algoritmo *A** em *Prolog* (Pl) para o cálculo da rota a tomar;
- Integração *C#* e Pl para a comunicação entre os componentes;
- Interface gráfica para interação e demonstração do sistema;

3.1.2 Requisitos Opcionais

- Capacidade da rede neuronal ler mais caracteres e códigos mais complexos;
- Cálculo de rotas adaptável a obstáculos;

3.2 Funcionalidade

O sistema permite escolher a sequência de caracteres que representa o código do destino e procede então ao cálculo da rota correspondente, eventualmente com adaptação a obstáculos.

3.3 Arquitetura da solução

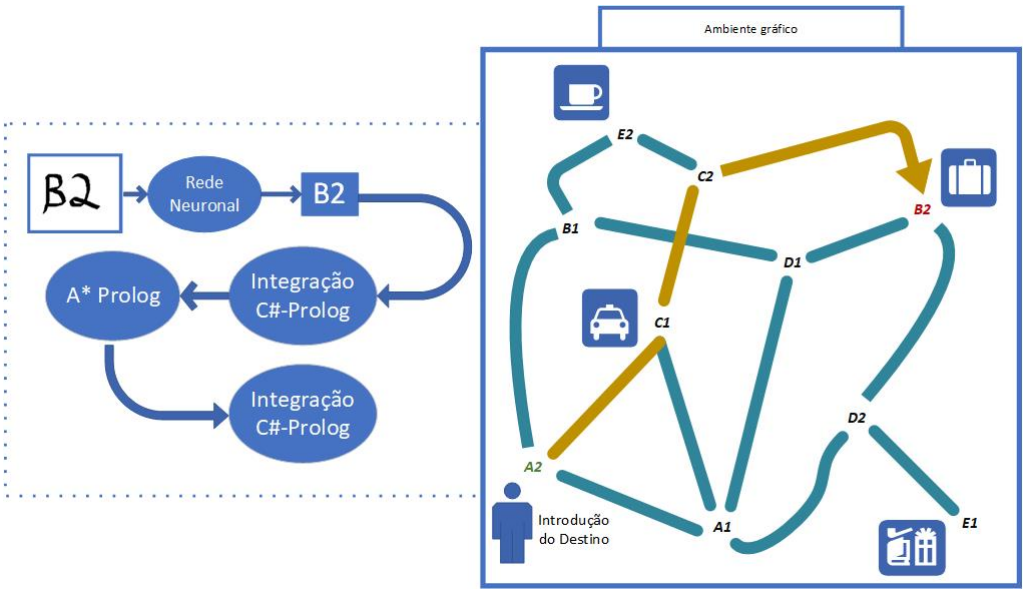

Figure 3.1: Esquema do funcionamento do sistema.

Chapter 4

Development support

Contents

4.1 Implementação

4.1.1 Ferramentas usadas

São usados o Visual studio para o desenvolvimento da rede neuronal e da interface gráfica. É usado também o *swi-prolog* para o desenvolvimento do algoritmo *A**.

4.1.2 *Data Set*

A escolha do *data set* foi feita com base na reputação e disponibilidade, sendo o *data set* denominado *Modified National Institute of Standards and Technology database* (MNIST) disponivel online *leCun*, o escolhido. Este *data set* é a versão curada pelo Professor Yann LeCun do Instituto de ciências matemáticas da universidade de Nova York, onde é feito um reprocessamento das imagens para centra-las pelo centro de massa numa janela maior. O MNIST é usado por diversos engenheiros de *Rede Neuronal* (RN) como uma espécie de '*hello world*' para verificar novas técnicas de aprendizagem e reconhecimento de padrões. Para a manuseamento do *data set* seguiu-se o exemplo de leitura das imagens do MNIST presente num artigo online **?**. No MNIST as imagens estão formatadas com um tamanho de 28x28 pixeis e guardadas num formato próprio que consiste num ficheiro binário para guardar matrizes, onde se encontram a lista das imagens, e um segundo ficheiro com o mesmo número de entradas mas com a informação de *label*, ou seja, a que dígito corresponde a imagem do ficheiro das imagens.

4.1.3 Rede Neuronal

Para a implementação da rede neuronal estudou-se diversos livros e material de referência sobre o tema, sendo dois livros os escolhidos como base da implementação *Kim; Rashid*.

A base conceptual de uma RN é o neurónio. Cada entrada tem associada um valor denominado por 'peso', que é simplesmente um fator multiplicativo que vai definir a 'importância' dessa entrada para o resultado que o neurónio irá produzir.

Tendo como exemplo o neurónio da figura 4.1, a resolução das entradas é dada pela seguinte expressão.

$$v = (w_1 * x_1) + (w_2 * x_2) + (w_3 * x_3) + b$$

Com o intuito de facilitar os cálculos e a leitura das expressões são usadas matrizes, tendo então a expressão anterior.

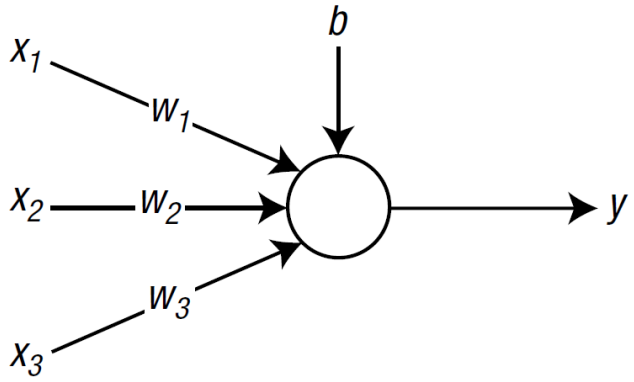

Figure 4.1: Um neurónio com três entradas.

$$v = wx + b$$

Onde *w* é uma matriz *(1,3)* com os pesos do neurónio e *x* uma matriz *(3,1)* com as entradas do mesmo. Para realmente obter o resultado do neurónio, que consiste na ativação ou não do mesmo, passa-se o valor *v* por uma função de ativação.

$$y = \varphi(v)$$

Existem diversas funções de ativação, mas aquela que é usada, é das mais utilizadas para o efeito, corresponde à função *sigmoid* com a a famosa curvatura em forma de *S*. Um exemplo comum da função *sigmoid* é a função logística (*logistic function*), que se encontra na figura 4.2.

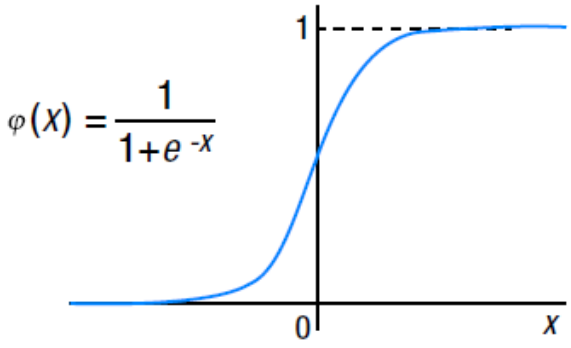

Figure 4.2: A função *sigmoid* definida, função logística.

A RN desenvolvida corresponde àquilo que é chamado de *shallow Multi-layer Neural Network*, representada pela figura 4.3, o termo *shallow* é usado para distinguir RN

multi-camada com apenas uma camada escondida de RN com mais do que uma camada escondida.

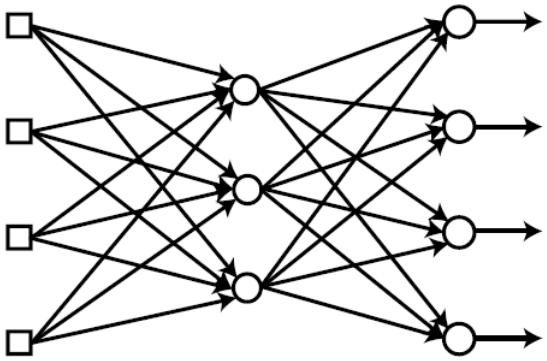

Figure 4.3: RN Multi-camada (*shallow*)

Como regra de aprendizagem da RN usa-se a *Delta Rule*. O algoritmo de aprendizagem denomina-se por algoritmo de *back propagation* que consiste em, após a produção de um resultado, pegar na resposta esperada e calcular o erro. Depois propaga-se este erro para o resto das camadas da RN usando a regra de aprendizagem.

A figura 4.4 demonstra a forma de obtenção do erro na ultima camada, a camada de *output*.

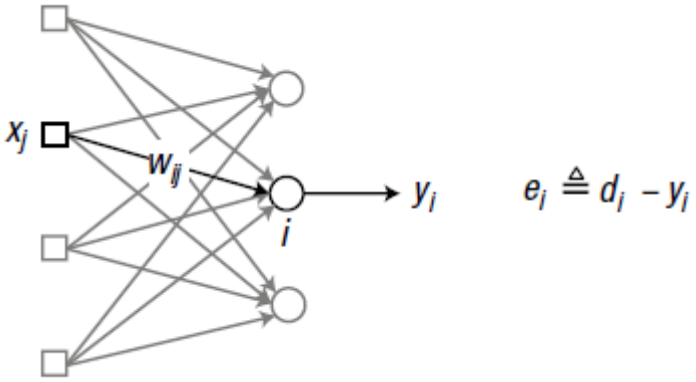

Figure 4.4: Obtenção do erro onde, d_i é o resultado esperado e y_i é o resultado obtido

A *Delta Rule*, aplicada como exemplo à figura 4.4, pode ser resumida da seguinte maneira:

"Se um nó (neurónio) de *input* contribui para o erro do nó de *output*, o peso entre os dois nós é ajustado em proporção do valor de *input*, x_j e o erro de *output*, e_i"

Com isto em mente as expressões que categorizam a *Delta Rule* são as seguintes:

$$w_{ij} = w_{ij} + \Delta w_{ij}$$

$$\Delta w_{ij} = \alpha \delta_i x_j$$

$$\delta_i = \varphi'(v_i) e_i$$

A derivada da função logística pode ser expressada por $\varphi'(v) = \varphi(v)(1 - \varphi(v))$ podendo assim a função de ajuste dos pesos ser descrita da seguinte maneira:

$$w_{ij} = w_{ij} + \alpha \varphi(v_i)(1 - \varphi(v_i)) e_i x_j$$

Estas expressões funcionam bem para a camada de output pois o erro é dado por $e_i = d_i - y_i$, o problema é que, para as camadas escondidas este erro tem de ser calculado de outra forma, é aqui que entra o algoritmo de *back propagation*. Exemplificando o algoritmo tem-se a RN da figura 4.5 onde o *delta* é calculado da mesma forma como já explicado.

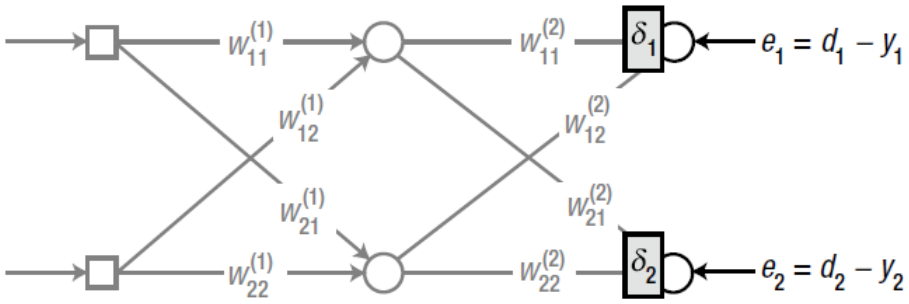

Figure 4.5: Treinar a RN usando o algoritmo de *back propagation*.

Como se tem o delta de cada nó de output, prossegue-se com o mesmo para a próxima camada à esquerda.

Resolvendo o problema de como calcular o erro das camadas escondidas, usando o algoritmo de *back propagation*, o erro do nó é definido como a soma dos deltas multiplicados pelos pesos da camada á direita imediata. Este processo pode ser expresso por:

$$e_1^{(1)} = w_{11}^{(2)} \delta_1 + w_{21}^{(2)} \delta_2$$

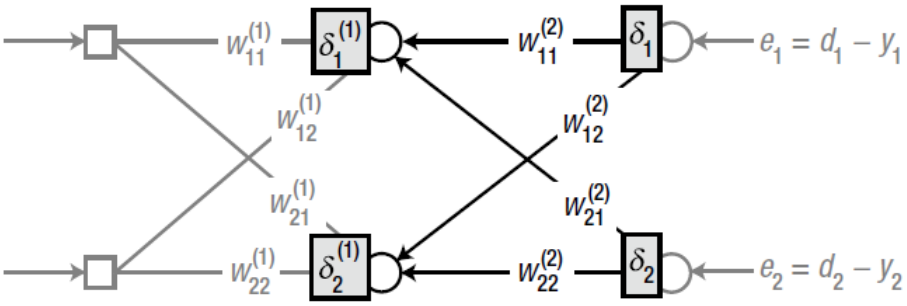

Figure 4.6: Prosseguir para a esquerda, para a os nós escondidos, e calcular o delta.

$$\delta_1^{(1)} = \varphi'(v_1^{(1)})e_1^{(1)}$$

$$e_2^{(1)} = w_{12}^{(2)}\delta_1 + w_{22}^{(2)}\delta_2$$

$$\delta_2^{(1)} = \varphi'(v_2^{(1)})e_2^{(1)}$$

Onde v_1^1 e v_2^1 são as somas das multiplicações dos sinais de entrada com os respetivos pesos, no sentido *foward*. Assim a única diferença entre o algoritmo de correção dos pesos entre a camada de *output* e a as camadas consequentes é o cálculo do erro.

Este processo na forma de cálculo matricial corresponde a:

$$\begin{bmatrix} e_1^1 \\ e_2^1 \end{bmatrix} = W_2^T \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix}$$

Para os cálculos matriciais em C# foi usada a biblioteca '*MathNet.Numerics.4.15.0*' que permite efetuar todos os cálculos até aqui demonstrados.

4.1.4 Algoritmo *A**

A implementação do algoritmo $A*$ foi baseada na implementação descrita no livro de *Prolog* (Pl) *Bratko* no capítulo 12. Este algoritmo é também denominado de '*best first*', e consiste na minimização da função de custo do próximo nó escolhendo assim o melhor nó possível dada uma heurística adequada. Isto é demonstrado na figura 4.7.

A heurística utilizada para a resolução de um problema como o aqui abordado, a pesquisa do caminho mais curto até ao destino, coincide com a distância euclidiana, em linha reta, do nó a considerar até ao destino.

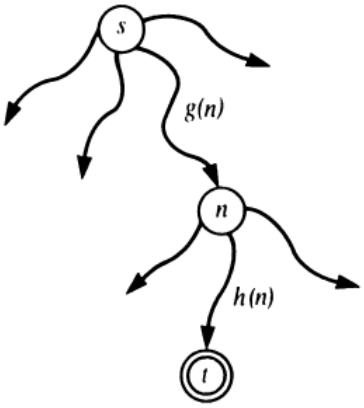

Figure 4.7: Construção da estimação heuristica *f(n)* do custo do caminho mais acessível de *s* a *t* via n: $f(n) = g(n) + h(n)$.

4.1.5 Integração *Prolog-C#*

4.1.6 Interface gráfica

Chapter 5

Conclusions and Future Work

Contents

5.1 Sobre a Rede Neuronal

Em relação à rede neuronal, foi omitido o valor de *bias* pelo facto de seguir-se o livro *Rashid*. A *Rede Neuronal* (RN) ficou então com uma precisão de 97.22% sendo que a adição do valor de *bias* poderia melhorar a mesma. Não se treinou a RN para ler caracteres para além de dígitos, até ao momento.

5.2 Sobre o algoritmo A*

Não se teve em conta a prevenção de obstáculos de momento. Os nós são pontos no espaço estáticos podendo-se apenas escolher o inicio e o destino.

5.3 Trabalho Futuro

Pretendia-se melhorar a RN para ler também letras, e acrescentar um componente de software para o reconhecimento de imagens, de modo a permitir ler o código do destino a partir de uma fotografia. Tornar possível a obtenção de nós do caminho algo dinâmico, seria proveitoso para uma mais fácil adaptação para um sistema realista que estaria a calcular os nós alcançáveis em tempo real.

Appendices

References

Bratko, I. (). *Prolog programming for artificial intelligence by Ivan Bratko*. (4th ed.).

Chang, A. (2015). NY midtown robots allow for conversation-free hotel service. URL: https://www.cnbc.com/2015/05/13/ny-midtown-robots-allow-for-conversation-free-hotel-service.html.

Kim, P. (). *MATLAB Deep Learning*.

leCun, Y. (). The mnist data base. URL: http://yann.lecun.com/exdb/mnist/.

Loughran, J. (2016). Luggage robot leo autonomously checks in airport baggage. URL: https://eandt.theiet.org/content/articles/2016/06/luggage-robot-leo-autonomously-checks-in-airport-baggage/.

Rashid, T. (). *Make your own Neural Network*.

Servicerobots (). Service robots. URL: https://www.servicerobots.com/.

Staff, R. (2018). How automated transportation will change our lives. URL: https://www.roboticsbusinessreview.com/supply-chain/how-automated-transportation-will-change-our-lives/.

Walker, J. (). Does our future depend on elder care robots? URL: https://waypointrobotics.com/blog/elder-care-robots/.