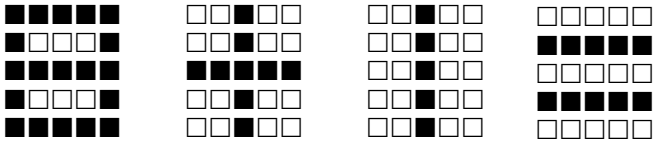
Trabalho 1

Máquina de Calcular



Computação Adaptativa

Aulas práticas

Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

Jorge Henriques

Outubro de 2013

Índice

1. Reconhecimento de Dígitos

1.1 Definição do problema

Pretende-se neste primeiro trabalho prático usar uma rede neuronal (memória associativa e/ou Perceptrão), de forma a reconhecer automaticamente os resultados de uma operação matemática. Os dígitos a reconhecer são:

{1, 2, 3, 4, 5, 6, 7, 8, 9, 0}

E as operações:

{+, -, =}

Deve, por exemplo, tentar calcular automaticamente a seguinte operação

7 + 2 =

Para tal basta reconhecer os dígitos e a operação envolvida !

1.2 Abordagem a seguir

De forma a implementar o processo assume-se que existe um processo de digitalização que transforma o texto manual numa matriz de pixeis. A cada dígito corresponde uma ***matriz de dimensão 5x5***, de elementos 0 (□) e 1 (■).

Uma possível definição para os dígitos a identificar é a seguinte:

0

■■■■■
■□□□■
■□□□■
■□□□■
■■■■■

1

□□■□□
□□■□□
□□■□□
□□■□□
□□■□□

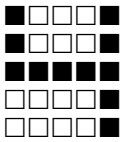
2

■■■■■
□□□□■
■■■■■
■□□□□
■■■■■

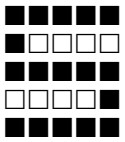
3

■■■■■
□□□□■
■■■■■
□□□□■
■■■■■

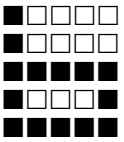
4



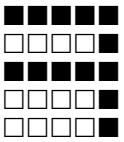
5



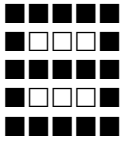
6



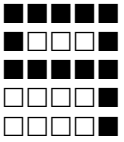
7



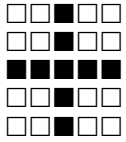
8



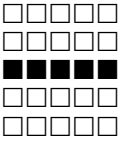
9



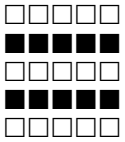
+



-



=



Nota:

Os digitos atrás referidos encontram-se no ficheiro **def_digito.**

No presente trabalho, utiliza-se a aplicação **mpaper(tam)** que permite converter caracteres desenhados pelo utilizador num digito (25x1) de elementos binários (0/1).

Tem também disponíveis as funções **ver_digito(tam,digito)** e **ver_matriz(tam,matriz)** para visualizar, respectivamente, um digito e um conjunto de dígitos.

2. Arquitectura da rede

O classificador a implementar consiste nume rede (memória associativa ou perceptrão), que permite identificar o dígito escrito pelo utilizador.
Assim, a entrada da rede é o dígito escritos pelo utilizador (25×1), devendo a saída do classificador (**A**) permitir reconhecer o dígito em questão.

2.1. Número de Entrada

Dimensão do digito, logo 25x1.

2.2. Número de Saídas

Existem várias soluções para a arquitectura a utilizar, em particular no número de perceptrões utilizado. Uma vez que temos 13 saídas distintas (13 dígitos), necessitamos de **pelo menos 4 perceptrões** (2^4=16 classes).

Neste caso **podemos assumir 13 perceptrões**, cada um encarregue de classificar cada um dos dígitos.

Assim, para os dígitos a classificar

{ '1' '2' '3' '4' '5' '6' '7' '8' '9' '0' '+' '-' '='},

são considerados os inteiros

{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 ,13}.

Ou seja, o digito '1' será representado pela seguinte saída desejada (A).

$$\mathbf{A = [\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,]^T}$$

O digito '+' será representado pela seguinte saída desejada (A).

$$\mathbf{A = [\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,]^T}$$

2.3. Arquitectura/função activação

Em termos de arquitectura o classificador:

- Tem uma única camada.
- Função de activação linear (purelin) ou binária=perceptrão (hardlim)
- Pode ter desvio

Assim sendo, em termos matemáticos

$$A = f(W_N \times P_1 + b)$$

Em que a matriz $\boldsymbol{W_N}$ *tem dimensão (13,*25) e **b** é um vector de dimensão (13,1). A entrada (**P1**) é um vector de dimensões (25,1). A saída (**A**) tem dimensão (13,1).

Relativamente à função de activação podem-se considerar duas alternativas possíveis: binária (Perceptrão) e linear (memória associativa).

2.3.1 Perceptrão

Neste caso a função de activação é binária

$$A = \text{hardim}(W_N \times P_1 + b)$$

2.3.2 Memória Associativa

Neste caso a função de activação é Linear

$$A = W_N \times P_1 + b$$

$$A = \text{round}(A)$$

$$A = \max(0, \min(A,1))$$

Nota:

No último caso, função de activação linear, é necessário converter a saída contínua (função de activação linear) para um valor binário (0/1) (por questões óbvias de classificação)

3. Implementação em Matlab

3.1 Funções disponibilizadas

São disponibilizadas as seguintes funções

Mpaper(tam)

- Permite escrever um conjunto de dígitos Q (Q<50).
- Converte o resultado numa matriz de dimensões (25,Q) e guarda-a no ***ficheiro P.dat***.

A estrutura **tam** define a dimensão de cada dígito. Neste caso:

- **tam.width = 5** (número de colunas)
- **tam.height = 5** (número de linhas)

ver_digito(tam, dig1, dig2, dig3)

- Permite visualizar até 3 dígitos. Um dígito é um vector de dimensão (25,1)

ver_matriz(tam, M)

- Permite visualizar um conjunto de Q dígitos, de dimensão (25,Q).

def_digitos

- Definição dos dígitos perrfeitos

tp1_treinar

- Exemplo de treino uma rede para os digitos {0,1} e operações {+,-}

tp1_testar

- Exemplo para testar a rede com base nos dados do ficheiro P.dat

3.2 Treino

Dados de treino

A definição do conjunto de dados de treino deve ser por si introduzindo usando a aplicação **mpaper(tam)**.

Note que estes dados devem ter como entradas exemplos dos dígitos escritos a reconhecer e como saídas os respectivos dígitos correctos. Cabe à rede "aprender" a reconhecer as imperfeições de escrita.

Assim sendo, teoricamente, quanto mais rico e de "qualidade" for o conjunto de treino melhor!

1. Treino dos pesos da memória associativa

Para tal recorra à função **pinv**

```
WA=T*pinv(P)
```

2. Treino da rede perceptrão

Assume-se, neste caso, uma rede perceptrão, a ser criada pela função (**newp**), sendo a função de activação **σ(.)**, (**harlim**).

Nota: Para mais esclarecimentos execute o comando

```
» help newp
```

Definição

```
W = newp(P,T);
```

Treino

Uma vez criado/definido um Perceptrão ele deve ser treinado com

```
W = train(W,P,T);
```

Por exemplo, o vector de entrada P tem dimensão (25,Q) e T, de saída, tem dimensão (13,Q), em que Q é o numero de padrões (exemplos de treino)

3.3 Validar / testar

1. Memória associativa

Para validar o classificador (memória associativa), isto é calcular a saída da rede, use

```
a = WA*Pt
```

WA é a matriz de pessos, Pt é um digito de teste (25,1).

2. Perceptão

Para validar o classificador (perceptrão), isto é calcular a saída da rede, use a função **sim**. Pt é um digito de teste (25,1)

```
a = sim(W,Pt)
```

4. Conclusões

4.1 Conclusões

Em função dos resultados que obtiver, justifique sucintamente e tire conclusões, relativamente a:

- **1. Dados para o treino da rede**
  - Cuidados a ter com os dados escolhidos a usar no treino de uma rede;
  - Considere os exemplos de treino que julgar necessários tendo em consideração que a rede deve ser capaz de dar respostas adequadas não só em presença de dígitos "perfeitos" mas também quando lhe forem apresentados dígitos corrompidos (evidentemente até um certo limite).
- **2. Estrutura da rede**
  - Função de activação (binária/linear), ou seja, perceptrão / linear
- **3. Resultados**
  - Capacidade da rede em resolver o problema definido;
  - Capacidade de generalização, isto é, robustez da rede para dados não usados durante a fase de treino e dados corrompidos;
- **4. Outros aspectos que achar convenientes**

4.2 Entrega

O trabalho e respectivo relatório deverão ser entregues na aula de ***06 de Novembro 2013***. Deve ser entregue (**em submissão de trabalhos**):

- **1. Em papel**: Relatório **sucinto** relativo às conclusões principais.

- **2. Ficheiro** zipado com todos os ficheiros necessários
  - Nome: AnacletoCaroço_AlípioAbelha.zip (nomes dos elementos do grupo)
  - Deve haver um ficheiro principal (executável) de nome **tp1.m**
  - Este deve permitir executar o programa de classificação de caracteres assumindo que os dados a testar (dígitos e operação) estão armazenados no ficheiro **P.dat.**

Bom Trabalho !