Aprendizagem Computacional - Trabalho Prático 2

João Tiago Márcia do Nascimento Fernandes - 2011162899
Joaquim Pedro Bento Gonçalves Pratas Leitão - 2011150072

8 de Outubro de 2014

Índice

1 Introdução

Este trabalho foca-se no reconhecimento de caracteres da numeração árabe, ou seja, os caracteres 0 a 9.

Pretende-se que este reconhecimento seja realizado por uma aplicação desenvolvida em *Matlab*, que faz uso de redes neuronais na sua arquitetura interna, disponíveis na *Nerual Networks Toolbox* do próprio *Matlab*.

A aplicação desenvolvida visa o estudo de duas arquiteturas distintas no reconhecimento dos caracteres:

- Na primeira arquitetura a aplicação será constituída por uma *memória associativa* e um *classificador*
- Na segunda arquitetura a aplicação apenas recorre ao *classificador*

As duas arquiteturas apresentadas estão presentes nas figuras que se seguem:

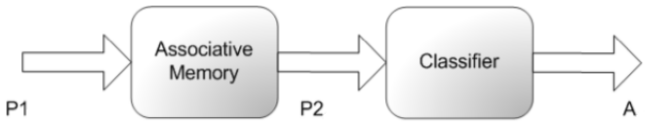

Figura 1: Arquitetura da aplicação com *memória associativa + classificador*

Figura 2: Arquitetura da aplicação apenas com o *classificador*

Através da análise destas figuras podemos determinar um comportamento padrão para a aplicação:

- Numa fase inicial, os caracteres a identificar poderão, ou não, ser fornecidos à *memória associativa*, que está encarregue da sua "filtragem"ou "correção": Se os caracteres fornecidos não forem perfeitos, a memória associativa aproxima-os dos respetivos caracteres perfeitos.
- De seguida os dados, corrigidos ou não, serão fornecidos ao *classificador*, que se encarregará de proceder à identificação dos mesmos.

No presente documento iremos proceder à apresentação em maior detalhe destas duas arquiteturas e das suas implementações, bem como da aplicação *Matlab* desenvolvida, e de como poderá ser utilizada. Pretendemos também fazer uma análise crítica da performance da aplicação, nomeadamente da sua capacidade de classificar corretamente novos caracteres fornecidos.

2 Aplicação Desenvolvida

A aplicação desenvolvida visa identificar corretamente caracteres desenhados pelo utilizador, implementando para isso as duas arquiteturas apresentadas anteriormente: *Memória Associativa + Classificador* e *Classificador*.

Ambas as arquiteturas e os respetivos modos de funcionamento serão apresentados de seguida.

2.1 Memória Associativa + Classificador

Na arquitetura *Memória Associativa + Classificador*, os dados

2.2 Classificador

2.3 Implementação em Matlab

Para o desenvolvimento da aplicação foi-nos fornecido código-fonte, que cria e disponibiliza ao utilizador uma grelha onde este irá desenhar os caracteres a identificar, encarregando-se de toda a lógica interna da aplicação com a exceção da implementação do sistema de classificação dos caracteres e de toda a lógica a ele inerente (implementação das diferentes arquiteturas da aplicação, realização dos treinos das redes neuronais implementadas, etc)

2.3.1 associativeMemory.m

2.3.2 createNetwork.m

2.3.3 myclassify.m

2.3.4 run.m

Este é o ficheiro a executar para que a aplicação possa ser testada e utilizada.

2.4 Execução

Para executar a aplicação o utilizador deverá executar o ficheiro *run.m*. Uma vez iniciado, será pedido ao utilizador que selecione uma de duas possíveis arquiteturas para a aplicação: Utilizando *Memória Associativa* em conjunto com o *Classificador*, e utilizando apenas o *Classificador*.

Caso o utilizador selecione a utilização da *Memória Associativa*, então terá de escolher o tipo de treino a realizar na mesma.

Existem dois tipos de treino distintos: O primeiro recorre à *Pseudo-Inversa*, onde os pesos da rede neuronal que a *Memória Associativa* implementa são calculados pelo produto do *output* esperado pelo valor da aplicação função *pinv* (nativa do *Matlab*) aos dados de *input*. Já o segundo faz uso da *Regra de Hebb* para calcular os pesos: Aqui os pesos resultam do produto do *output* esperado pela seguinte matriz: $input^T \times \left(input \times input^T\right)^{-1}$.

Após esta seleção a *Memória Associativa* é criada, aparecendo de seguida uma grelha, onde o utilizador desenhará os caracteres a serem classificados. Assim que os caracteres são desenhados, o utilizador tem de carregar no botão do meio do seu rato, de forma a transitar para a fase de execução seguinte.

Nesta fase, o utilizador terá que escolher algumas características da rede neural que o classificador da aplicação implementa. Caso já exista alguma rede previamente criada com as características especificadas pelo utilizador, essa rede é carregada para memória e utilizada na execução. Caso ainda não exista, uma rede neuronal com as características desejadas é criada e treinada, sendo também guardada para posteriores execuções.

Uma vez obtida a rede a utilizar, esta classifica os dados introduzidos pelo utilizador, que lhe serão apresentados numa grelha semelhante à onde inicialmente desenhou os caracteres.

3 Testes e Resultados

Descrição de como fizémos os casos de teste, dimensões, etc

4 Conclusões

Conclusões, lolol

FIXME: Testar classificação de dígitos perfeitos e de dígitos não perfeitos (alguns não perfeitos são corretamente classificados, e todos os perfeitos são corretamente classificados)

FIXME: Dizer que memória associativa só funciona se preenchermos a tabela toda pela ordem: 1,2,3,4,5,6,7,8,9,0 linha-a-linha

FIXME: Perguntas relatório:

- How does the data set influence the performance of the classification system?
- Which architecture provides better results: only the classifier or the associative memory+classifier?
- Which is the best activation function: hardlim, linear or logsig?
- Does the Hebb rule perform well?
- Is the classification system able to achieve the main objectives (classification of digits)?
- Which is the percentage of well classified digits?
- How is the generalization capacity?
- Is the classification system robust enough to give correct outputs when new inputs are not perfect?
- Which is the percentage of well classified new inputs?