Relatório -Trabalho Prático 2

OCR – Optical Character Recognition

Realizado por:

2009109673 – Daniel Frutuoso

2009111924 – Igor Cruz

Grupo 4

Faculdade de Ciências e Tecnologias da UC

Departamento de Engenharia Informática

Coimbra, 24 de Outubro de 2012

Introdução

É-nos proposto, neste trabalho, desenvolver uma aplicação Optical Character Recognition que identifique os números 1, 2, 3, 4, 5, 6, 7, 8, 9 e 0 recorrendo a duas arquitecturas de redes neuronais:

- Classificador
- Memória Associativa + Classificador

Para além disto, temos também que testar as redes com diferentes conjuntos de funções de transferência e funções de aprendizagem e descobrir qual a melhor combinação. Mais à frente serão apresentados os resultados.

Classificador

O classificador implementado é uma rede neuronal que tem como função classificar o input com uma determinada classe. Por exemplo, se o input corresponder ao número 1 o classificador deve classifica-lo como pertencente à classe 1. Como é dito no enunciado, esta rede tem a característica de ser uma rede neuronal com apenas uma camada (10 neurónios), com uma certa função de activação e com o *bias*.

Para além disto, tem as seguintes propriedades:

```
net.performParam.ratio = 0.5;   % learning rate
net.trainParam.epochs = 1000;   % maximum epochs
net.trainParam.show = 35;       % show
net.trainParam.goal = 1e-6;     % goal=objective
net.performFcn = 'sse';         % criterion
```

O treino desta rede consiste em passar à função *train* casos feitos por nós, onde se encontram repetidas vezes a sequência de 1 a 0, e um target.

Para testar a rede neuronal foi criado um conjunto de números de teste, o qual passamos para a função *myClassifier.*

Memória Associativa + Comparador

A memória associativa tem por objectivo pegar no input que passarmos à função *myClassifier* e, através de pesos, aperfeiçoar os números que estão no input. Os pesos são calculados através do método da pseudo-inversa ou pela regra de Hebb. No caso de ser a pseudo-inversa, os pesos são calculados com base no produto entre a matriz Manual*Perfect* (matriz que contém os números feitos manualmente o mais perfeitos possível) e o *pinv* da matriz de casos de treino. No caso de ser a regra de Hebb, é feito o produto entre a matriz ManualPerfect e a transposta da matriz dos casos de treino.

Após isto, é realizado um último produto entre os pesos e o caso de teste. O objectivo é, como foi dito anteriormente, melhorar o desenho dos números feito pelo usuário para que assim seja mais fácil o comparador classificar o input. O resultado do produto é passado ao classificador que vai, por fim, tentar reconhecer os números desenhados.

Resultados

Para testar todo o programa foi dado ao programa como input o seguinte conjunto de números:

Figure 1 - Input

Quando testamos o apenas o classificador com o input anterior, obtemos o seguinte resultado:

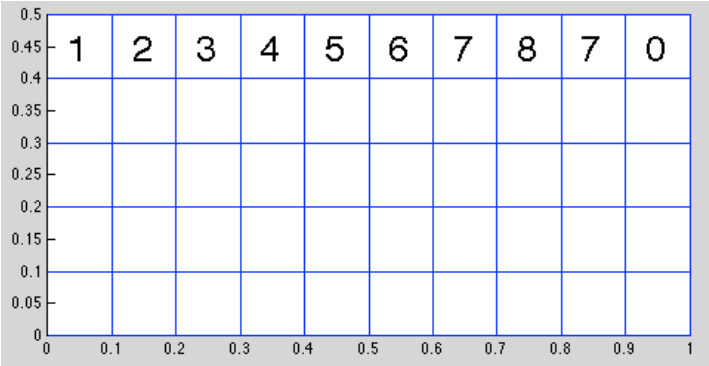

Figure 2 - Resultado do Classificador

Já quando testamos, o mesmo input, na rede neuronal da Memória associativa + Classificador, o resultado é:

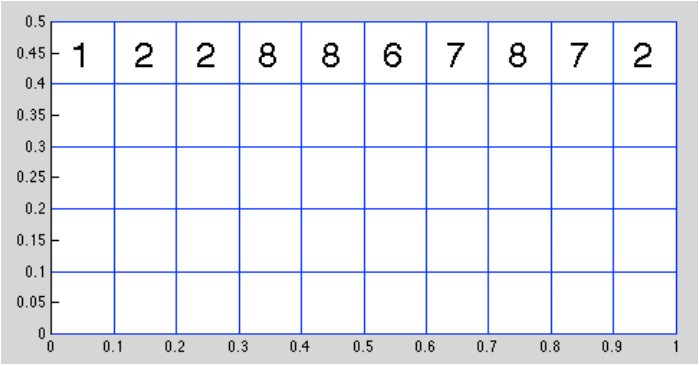

Figure 3 - Resultado da Memória Associativa + Classificador

Como é possível ver através das figuras, o Classificador teve melhor desempenho que a Memória Associativa + Classificador. A taxa de sucesso, no caso de apenas o Classificador correr é de 90% enquanto que a taxa de sucesso da Memória Associativa + Classificador ficou-se apenas pelos 50%. De salientar, que os casos de treino foram, na sua maioria, feitos por um dos autores enquanto que o caso de teste foi feito pelo outro. Isto revela que mesmo que a caligrafia não seja exactamente a mesma, o programa não deixa de reconhecer os números.

Na seguinte tabela encontram-se os restantes resultados:

Função de aprendizagem (Memória Associativa)	Função de transferência (Classificador)	Função de Aprendizagem (Classificador)	Percentagem de números correctos
Pinv	logsig	learnp	50%
	purelin	learngd	30%
	hardlim	learnp	20%
Regra de Hebb	logsig	learnp	10%
	purelin	learngd	10%
	hardlim	learnp	10%
Sem memória associativa	logsig	learnp	90%
	purelin	learngd	40%
	hardlim	learnp	80%

Nota: Durante o teste do programa, mais especificamente, da parte da memória associativa + classificador, chegamos a obter uma desempenho de casos correctos à volta dos 70%. A arquitectura que consiste apenas no classificador chegou a obter resultados na ordem dos 95%.

Questões

Como é que o *data set* influencia a performance do sistema classificador?

Para um desempenho bom do sistema classificador, o *data set* deve ter um grande número de casos de treino e esses casos devem ser diversificados, possibilitando que a rede neuronal faça generalizações e seja assim capaz de classificar diferentes tipos de caracteres, mais especificamente números, desenhados manualmente. É importante variar o tipo de escrita e o tamanho dos caracteres.

Em termos de treino é importante variar o tamanho e tipo de caligrafia facilitando a generalização. No entanto se o treino tiver um número elevado de casos em que a caligrafia é a mesma, bem como os tamanhos dos caracteres, então a aprendizagem é mais específica e mais perspicaz para essa caligrafia.

Qual a estrutura que demonstra melhores resultados: apenas o classificador ou o classificador+memória associativa?

Após implementação e teste de resultados a implementação que demonstrou maior taxa de sucesso foi a utilização de apenas um classificador para identificar os algarismos.

Qual a melhor função de activação: *hardlim*, *linear* ou *logsig*?

Para a activação da rede neuronal responsável pela memória associativa a função que funciona melhor é a *hardlim*, que possibilita a eliminação de ruído de caracteres que não correspondem ao caracter perfeito.

Já para o classificador, após diversos teste, constatamos que a melhor função é a *logsig*, no entanto a *hardlim* também se revelou eficiente.

O desempenho da regra de Hebb é bom?

Não. A regra de Hebb, aplicada à memória associativa revela-se extremamente ineficiente. Isto ocorre pois os padrões de *input* e treino não são ortogonais levando à ocorrência de interferência e sobreposição entre as associações e consecutivo ruído nas saídas da memória associativa.

O sistema classificador é capaz de atingir os objectivos principais (classificação de dígitos)?

Sim, sem sobra de dúvidas. O nosso classificador, apresentou taxas de sucesso na ordem dos 90% a 95%, quando treinado com 350 casos de treino.

Qual a percentagem de dígitos bem classificados?

Tal como dito acima, a taxa de sucesso de dígitos bem classificadores recorrendo apenas ao classificador foi de cerca de 95%.

Quanto à arquitectura que recorre à memória associativa e ao classificador, a taxa de sucesso foi de cerca de 50% a 70%.

Como é a capacidade de generalização? O sistema classificador é suficientemente robusto (para dar saídas correctas quando as entradas não são perfeitas)? Qual a percentagem de novos dígitos bem classificados?

O sistema classificador é capaz de generalizar, no entanto os *inputs* a classificar devem ser razoavelmente legíveis. No caso de se inventarem novos tipos de letra e representações estranhas para os dígitos a percentagem de números bem classificados cai para os 40%.

Conclusão

Através da realização de trabalho prático foi-nos possível ter contacto com diversos métodos de construção e optimização de redes neuronais e perceber melhor o funcionamento de memórias associativas.

Ao início esperávamos que a arquitectura composta por um classificador auxiliado por uma memória associativa fosse ter melhor desempenho do que a arquitectura composta apenas pelo classificador. No entanto após vários treinos e testes, foi-nos possível verificar que o classificador sozinho era capaz de acertar em 95% dos casos de avaliação e o conjunto memória associativa ficava-se pelos 50%70%.

Esta redução da taxa de sucesso deve-se, provavelmente, ao facto de os casos de treino efectuados não serem muito bons para a rede associativa.

Em relação ao treino efectuado, tentámos diversificar nos 350 caracteres desenhados, para que a rede fosse capaz de generalizar e reconhecer caracteres minimamente diferentes dos apresentados nos padrões de treino e, de facto, o classificador foi capaz de identificar caracteres ligeiramente diferentes daqueles treinados.

Em termos de escolha e utilização de funções de activação e treino, foi importante interpretar e escolher as funções que melhores resultados apresentam. Por exemplo, quanto a escolha da função *logsig* para o classificador, foi engraçado verificar que passado da função *hardlim*, para esta, a taxa de sucesso na classificação aumentou cerca de 10%-15%.

Para identificação de caracteres perfeitos começamos por utilizar o ArialPerfect como target, no entanto apercebemo-nos que o nosso classificador estava a responder melhor aos estímulos criados por dígitos escritos manualmente, pelo que decidimos definir nós um target manual como perfeito e vimos que os resultados melhoravam consideravelmente.