Universidade de Coimbra

Faculdade de Ciências e Tecnologia

Computação Neuronal e Sistemas Difusos

Trabalho 2

OCR
Optical Character Recognition

João Duarte
Ricardo Martins

1. Objectivo

Pretende-se através da realização deste trabalho prático desenvolver redes neuronais capazes de fazer o reconhecimento dos caracteres numéricos de 0 a 9. Serão comparadas redes neuronais com funções de activação diferentes (linear, sigmóidal, perceptrão) e dois tipos de arquitecturas diferentes: (memória associativa + classificador) e classificador apenas. Pretende-se também estudar a influência que o número e variabilidade dos dados com que é efectuado o treino dessas redes têm na performance destas.

2. Implementação

De forma a implementar mecanismos de treino para qualquer uma das arquitecturas consideradas e para poder avaliar a influência que o conteúdo dos dados de treino tem no desempenho posterior das redes, criaram-se dois conjuntos de dados. Um deles (*P_50.dat*) com 50 caracteres (5 de cada dígito) e outro (*P_500.dat*) com 500 caracteres (50 de cada dígito). O primeiro destes conjuntos contém apenas dígitos desenhados de forma cuidada, enquanto que o conjunto de 500 caracteres contém dígitos desenhados de forma cuidada e desenhados de forma mais imperfeita e variada.

2.1 Treino Memória Associativa + Classificador

De forma a poder efectuar o treino desta arquitectura foi necessário definir um conjunto objectivo de 10 caracteres, obtidos através da função mpaper. Estes caracteres pretendem representar os caracteres perfeitos que se pretende que sejam reconhecidos por esta arquitectura.

A memória associativa foi treinada através método da pseudo inversa, tendo-se utilizado como dados de treino os conjuntos de dados de 50 e 500 caracteres referidos anteriormente e como caracteres objectivo os 10 caracteres referidos neste ponto. Estes repetiram-se o número de vezes suficiente para definir uma matriz objectivo da mesma dimensão da matriz de dados utilizados nas duas situações de treino possíveis (*T_MA_50.dat* e *T_MA_500.dat*).

Implementaram-se e treinaram-se classificadores com 3 funções de activação diferentes:

-sigmoidal
-perceptrão (hardlim)
-linear

Para o caso de utilização das funções de activação sigmoidal e linear utilizaram-se como métodos de aprendizagem: regra do gradiente, regra de Hebb e regra de Hebb com decaimento dos pesos. Para o caso de utilização da função de activação perceptrão utilizou-se o método de aprendizagem do perceptrão. Os parâmetros de treino de cada um dos tipos de classificador utilizados encontram-se descritos nas tabelas I a IV.

Para qualquer dos casos, o classificador foi treinado utilizando como entrada a saída da memória associativa depois de treinada, quando esta tem à entrada o conjunto de 50 ou 500 caracteres, e como objectivo uma matriz definida para o efeito e que tem em cada coluna um 1 na posição correspondente ao algarismo que se pretende reconhecer (posição 10 para o algarismo 0)(*T_CL_50.dat* e *T_CL_500.dat*).

2.2 Classificador

Implementaram-se e treinaram-se classificadores com características semelhantes aos definidos para a arquitectura memória associativa + classificador.

O classificador foi treinado utilizando como entrada o conjunto de 50 ou 500 caracteres e como objectivo uma matriz definida de forma idêntica à referida para o caso memória associativa + classificador. Os parâmetros de treino de cada um dos tipos de classificador utilizados encontram-se descritos nas tabelas I a IV.

2.3 Implementação da função classify

Um dos objectivos deste trabalho consiste na construção de uma função ***classify*** para cada um dos tipos de classificador implementados. Essa função é invocada no código da função ***ocr_fun***, que por sua vez é invocada no código da função ***mpaper***, quando esta é utilizada para desenhar caracteres a serem classificados de seguida e automaticamente.

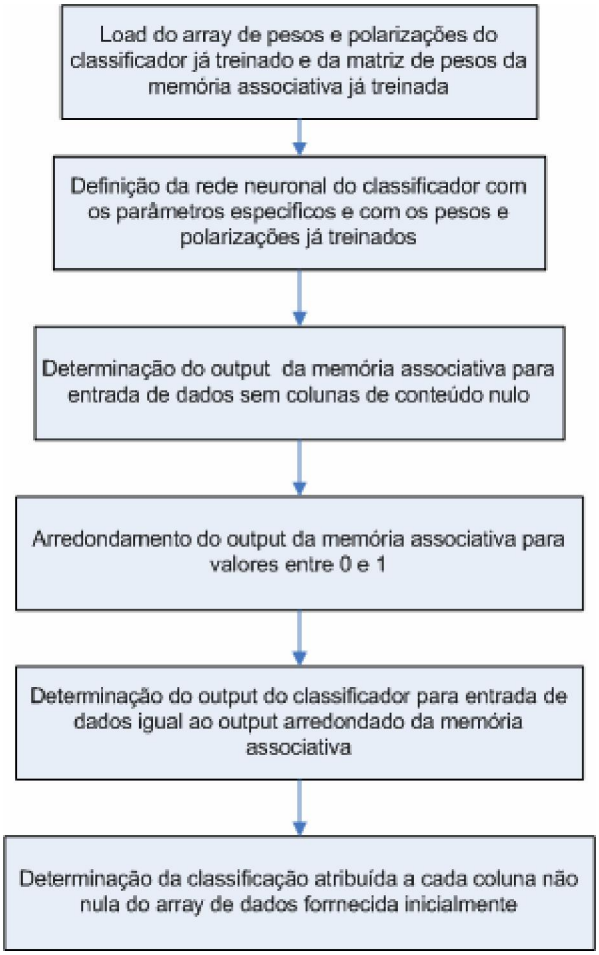
Através da análise do contexto em que a função ***classify*** era invocada na função ***ocr_fun***, verificou-se que esta teria de ter as seguintes características:

- Teria de receber como parâmetros um array de (256 x 50) correspondente aos dados provenientes da função ***mpaper***. Teria ainda de receber um vector linha de dimensão igual ao número de colunas de conteúdo não nulo referido no parâmetro anterior. Este vector armazena os índices das colunas da matriz cujo conteúdo é não nulo.

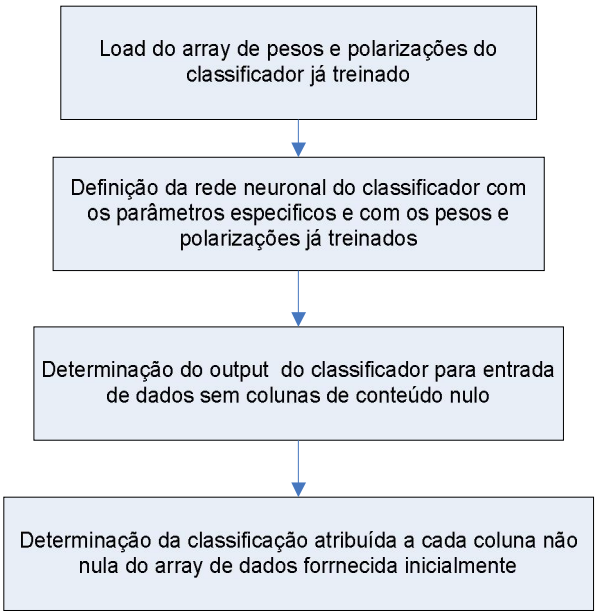
Na função ocr_fun o resultado da função classify é utilizado para preencher as posições de um vector y (incialiazado com todos os elementos a -1) correspondentes às posições das colunas de conteúdo não nulo, com o valor que o algoritmo de classificação implementado atribui a cada uma dessas colunas. Para o digito 0, a função classify deve devolver o valor 10, dado que avaliando a variável ocr_labels o digito 0 aparece na posição 10. Resumindo a função classify deverá devolver um vector linha de dimensão idêntica à dimensão do vector linha dado como argumento e em que cada um dos elementos deste vector devolvido é o valor que o classificador atribui como estando representado na coluna respectiva da matriz (256 x 50) dada como argumento.

Embora cada uma das redes definidas neste trabalho tenha uma função classify específica, podem definir-se duas implementações genéricas: função classify num contexto MA + CL e função classify num contexto CL:

- MA + CL


Load do array de pesos e polarizações do classificador já treinado e da matriz de pesos da memória associativa já treinada
Definição da rede neuronal do classificador com os parâmetros especificos e com os pesos e polarizações já treinados
Determinação do output da memória associativa para entrada de dados sem colunas de conteúdo nulo
Arredondamento do output da memória associativa para valores entre 0 e 1
Determinação do output do classificador para entrada de dados igual ao output arredondado da memória associativa
Determinação da classificação atribuída a cada coluna não nula do array de dados fornecida inicialmente

- CL


Load do array de pesos e polarizações do classificador já treinado
Definição da rede neuronal do classificador com os parâmetros especificos e com os pesos e polarizações já treinados
Determinação do output do classificador para entrada de dados sem colunas de conteúdo nulo
Determinação da classificação atribuída a cada coluna não nula do array de dados fornecida inicialmente

Quer no caso CL + MA, quer CL, a determinação da classificação atribuída a cada coluna não nula é feita através do vector coluna (10 x 1) que lhe corresponde à saída do classificador. Este analisado e a classificação atribuída corresponde à posição do elemento do vector coluna cujo valor é maior que o valor dos restantes elementos.

2.4 Resultados do treino

Os ficheiros correspondentes aos arrays que resultam do treino de cada rede estão guardados na pasta respectiva da rede, como descrito em 2.5.

- 50 digitos MA + CL

Tabela I

Parâmetros de treino e resultado do treino para a situação de treino com 50 dígitos MA + CL

Função de Activação	**Regra de Aprendizagem**	**Learning Rate**	**Performance pretendida**	**Performance Atingida**	**Número de épocas limite**	**Número de épocas atingido**	**Método cálculo erro**
Logsig	Gradient Rule	0.02	0.002	0.00199998	50000	11734	sum squared error
	Hebb rule	0.02	0.003	0.0029998	65000	7842	sum squared error
	Hebb rule with decayng weight	0.02	0.002	0.00199988	50000	11738	sum squared error
Perceptron			0.002	0	50000	2	sum squared error
Purelin	Gradient Rule	0.0002	0.002	0.00192524	50000	143	sum squared error
	Hebb rule	0.0002	0.002	0.0019591	50000	141	sum squared error
	Hebb rule with decayng weight	0.0002	0.002	0.00193362	50000	137	sum squared error

- 50 dígitos CL

Tabela II

Parâmetros de treino e resultado do treino para a situação de treino com 50 dígitos CL

Função de Activação	Regra de Aprendizagem	Learning Rate	Performance pretendida	Performance Atingida	Número de épocas limite	Número de épocas atingido	Método cálculo erro
Logsig	Gradient Rule	0.02	0.002	0.00199997	50000	40598	sum squared error
	Hebb rule	0.02	0.003	0.00299992	65000	27127	sum squared error
	Hebb rule with decayng weight	0.02	0.002	0.00199999	50000	40596	sum squared error
Perceptron			0.002	0	50000	5	sum squared error
Purelin	Gradient Rule	0.0002	0.002	0.00199729	50000	2487	sum squared error
	Hebb rule	0.0002	0.002	0.00199623	50000	2372	sum squared error
	Hebb rule with decayng weight	0.0002	0.002	0.00199984	50000	2335	sum squared error

- 500 dígitos MA + CL

Tabela III

Parâmetros de treino e resultado do treino para a situação de treino com 500 dígitos MA + CL

Função de Activação	Regra de Aprendizagem	Learning Rate	Performance pretendida	Performance Atingida	Número de épocas limite	Número de épocas atingido	Método cálculo erro
Logsig	Gradient Rule	0.005	0.003	0.00819704	80000	80000	sum squared error
	Hebb rule	0.005	0.003	0.00819485	80000	80000	sum squared error
	Hebb rule with decayng weight	0.005	0.003	0.00819804	80000	80000	sum squared error
Perceptron			0.003	0	80000	7	sum squared error
Purelin	Gradient Rule	0.00008	0.003	6,33083	40000	40000	sum squared error
	Hebb rule	0.00008	0.003	6,33073	40000	40000	sum squared error
	Hebb rule with decayng weight	0.00008	0.003	6,33078	40000	40000	sum squared error

- 500 dígitos CL

Tabela IV

Parâmetros de treino e resultado do treino para a situação de treino com 500 dígitos CL

Função de Activação	Regra de Aprendizagem	Learning Rate	Performance pretendida	Performance Atingida	Número de épocas limite	Número de épocas atingido	Método cálculo erro
Logsig	Gradient Rule	0.005	0.003	3,07185	80000	80000	sum squared error
	Hebb rule	0.005	0.003	3,07185	80000	80000	sum squared error
	Hebb rule with decayng weight	0.005	0.003	3,07185	80000	80000	sum squared error
Perceptron			0.003	0	80000	22	sum squared error
Purelin	Gradient Rule	0.001	0.003	101.656	40000	40000	sum squared error
	Hebb rule	0.001	0.002	101.659	40000	40000	sum squared error
	Hebb rule with decayng weight	0.001	0.002	101.673	40000	40000	sum squared error

2.5 Organização dos ficheiros e directorias

De forma a organizar os ficheiros deste trabalho criaram-se uma série de directorias organizadas consoante a série de dados utilizados no treino de cada um dos tipos de rede, função de activação e algoritmo de aprendizagem. Cada pasta contém os ficheiros de dados utilizados para o treino dessa rede, o ficheiro de código onde se implementou esse treino, os ficheiros onde se guardam os resultados do treino, a função classify especifica dessa rede, a função mpaper e ocr_fun, uma imagem onde se mostra o resultado gráfico da evolução do treino e uma subpasta que contém o resultado obtido através dessa rede de uma série de testes descritos mais a frente.

3. Resultados

De forma a testar as redes treinadas, construiu-se um conjunto de dados de teste de 200 dígitos. Esses dados de teste estão guardados na pasta *dados teste* (*teste1.dat*, *teste2.dat*, *teste3.dat*, *teste4.dat*). Os dígitos presentes em teste1 e teste3 foram desenhados de forma mais cuidada pretendendo testar as capacidades básicas de reconhecimento de caracteres das redes. Os dígitos de teste2 e teste4 foram desenhados de forma mais grosseira de forma a avaliar se as redes possuíam a capacidade de identificar dígitos desenhados de forma menos perfeita e regular.

Todas as redes foram testadas com este conjunto de dados, tendo-se criado 4 variáveis estrutura com um campo X e em cada uma delas foi guardado um array de dados (256 x 50) correspondente a cada um dos conjunto de dados a testar. Depois essas variáveis estrutura foram dadas uma a uma como argumento da função *ocr_fun* contida na pasta da rede em teste, tendo-se registado e contabilizado o número de reconhecimentos correctos. O registo desses resultados encontra-se na pasta *resultados teste* da pasta respectiva da rede e a contabilização dos mesmos é apresentada nas tabelas seguintes.

- 50 digitos MA + CL

Tabela V

Resultados do testes efectuados para as redes MA + CL treinadas com 50 dígitos

Função de Activação	Regra de Aprendizagem		Respostas Correctas	Percentagem Parcial Respostas correctas	Percentagem Total Respostas Corectas
Logsig	Gradient Rule	teste 1	30	53	47
		teste 3	23		
		teste 2	19	31	
		teste 4	22		
	Hebb rule	teste 1	30	54	48,5
		teste 3	24		
		teste 2	21	43	
		teste 4	22		
	Hebb rule with decayng weight	teste 1	29	52	48,5
		teste 3	23		
		teste 2	22	45	
		teste 4	23		
Perceptron		teste 1	22	41	34,5
		teste 3	19		
		teste 2	14	28	
		teste 4	14		
Purelin	Gradient Rule	teste 1	32	55	47,5
		teste 3	23		
		teste 2	19	40	
		teste 4	21		
	Hebb rule	teste 1	29	53	44,5
		teste 3	24		
		teste 2	16	36	
		teste 4	20		
	Hebb rule with decayng weight	teste 1	29	53	44
		teste 3	24		
		teste 2	16	36	
		teste 4	19		

- 50 digitos CL

Tabela VI

Resultados do testes efectuados para as redes CL treinadas com 50 dígitos

Função de Activação	Regra de Aprendizagem		Respostas Correctas	Percentagem Parcial Respostas correctas	Percentagem Total Respostas Correctas
Logsig	Gradient Rule	teste 1	25	58	47,5
		teste 3	23		
		teste 2	20	37	
		teste 4	17		
	Hebb rule	teste 1	34	57	47,5
		teste 3	23		
		teste 2	20	38	
		teste 4	18		
	Hebb rule with decaiyng weight	teste 1	34	58	47,5
		teste 3	24		
		teste 2	20	37	
		teste 4	17		
Perceptron		teste 1	24	37	32
		teste 3	13		
		teste 2	12	27	
		teste 4	15		
Purelin	Gradient Rule	teste 1	33	53	45
		teste 3	20		
		teste 2	18	37	
		teste 4	19		
	Hebb rule	teste 1	31	53	45,5
		teste 3	22		
		teste 2	18	38	
		teste 4	20		
	Hebb rule with decaiyng weight	teste 1	26	47	43
		teste 3	21		
		teste 2	18	39	
		teste 4	21		

- 500 dígitos MA + CL

Tabela VII

Resultados do testes efectuados para as redes MA + CL treinadas com 500 dígitos

Função de Activação	Regra de Aprendizagem		Respostas Correctas	Percentagem Parcial Respostas correctas	Percentagem Total Respostas Correctas
Logsig	Gradient Rule	teste 1	46	87	72,5
		teste 3	41		
		teste 2	31	58	
		teste 4	27		
	Hebb rule	teste 1	45	86	72
		teste 3	41		
		teste 2	31	58	
		teste 4	27		
	Hebb rule with decayng weight	teste 1	45	86	72
		teste 3	41		
		teste 2	31	58	
		teste 4	27		
Perceptron		teste 1	44	80	69
		teste 3	36		
		teste 2	32	58	
		teste 4	26		
Purelin	Gradient Rule	teste 1	46	84	71,5
		teste 3	38		
		teste 2	32	59	
		teste 4	27		
	Hebb rule	teste 1	46	89	71,5
		teste 3	38		
		teste 2	32	59	
		teste 4	27		
	Hebb rule with decayng weight	teste 1	47	85	72
		teste 3	38		
		teste 2	32	59	
		teste 4	27		

-500 dígitos CL

Tabela VIII

Resultados do testes efectuados para as redes CL treinadas com 50 dígitos

Função de Activação	Regra de Aprendizagem		Respostas Correctas	Percentagem Parcial Respostas correctas	Percentagem Total Respostas Correctas
Logsig	Gradient Rule	teste 1	50	99	84
		teste 3	49		
		teste 2	35	69	
		teste 4	34		
	Hebb rule	teste 1	50	99	84
		teste 3	49		
		teste 2	34	69	
		teste 4	35		
	Hebb rule with decayng weight	teste 1	50	98	84
		teste 3	48		
		teste 2	35	70	
		teste 4	35		
Perceptron		teste 1	48	91	75,5
		teste 3	43		
		teste 2	32	60	
		teste 4	28		
Purelin	Gradient Rule	teste 1	48	90	76,5
		teste 3	42		
		teste 2	35	63	
		teste 4	28		
	Hebb rule	teste 1	47	90	76,5
		teste 3	43		
		teste 2	35	63	
		teste 4	28		
	Hebb rule with decayng weight	teste 1	48	90	76,5
		teste 3	42		
		teste 2	35	63	
		teste 4	28		

4. Discussão dos Resultados

Após a análise dos resultados obtidos verifica-se que a dimensão e conteúdo dos dados utilizados no treino das redes influenciou o desempenho de todas as redes, mesmo aquelas com desempenhos fracos. A introdução de dígitos bastante imperfeitos e com caligrafias variadas no conjunto de dados de 500 dígitos, fez com que as redes adquirissem uma maior capacidade de generalização e maior percentagem de sucesso.

Nos testes efectuados às redes treinadas com o conjunto de dados de 50 dígitos a arquitectura com memória associativa obteve globalmente resultados ligeiramente melhores, enquanto que nas redes treinadas com o conjunto de dados de 500 dígitos a arquitectura apenas com classificador obteve melhores resultados. Esperava-se que a arquitectura com memória associativa e classificador obtivesse melhores resultados, pois a memória associativa serviria de filtro dos dados colocados à entrada, eliminando certas imperfeições que estes poderiam apresentar, facilitando a posterior classificação por parte do classificador.

Quanto às funções de activação verifica-se que é nas redes que integram um classificador com função de activação sigmoidal que se apresentam os melhores resultados, pois esta função de activação tem um output que varia de forma continua entre 0 e 1, o que proporciona treinos efectuados com mais sucesso visto que os objectivos considerados nos treinos efectuados têm gamas de valores entre 0 e 1.

Com esta função de activação em redes treinadas com o conjunto de dados com 500 dígitos, as redes apresentaram desempenhos elevados.

Mesmo para as redes com os desempenhos mais elevado que aqui se conseguiram obter, esse mesmo desempenho pudesse ser melhorado através da análise dos dígitos que estas redes não identificam correctamente e a inserção destes em conjunto de dados de treinos com as quais seriam treinados.